INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Natural Statistical Models for Automatic Speech Recognition

Jeffrey Adam Bilmes

TR-99-016

October 1999

Abstract

The performance of state-of-the-art speech recognition systems is still far worse than that of humans. This is partly caused by the use of poor statistical models. In a general statistical pattern classification task, the probabilistic models should represent the statistical structure unique to and distinguishing those objects to be classified. In many cases, however, model families are selected without verification of their ability to represent vital discriminative properties. For example, Hidden Markov Models (HMMs) are frequently used in automatic speech recognition systems even though they possess conditional independence properties that might cause inaccuracies when modeling and classifying speech signals.

In this work, a new method for automatic speech recognition is developed where the natural statistical properties of speech are used to determine the probabilistic model. Starting from an HMM, new models are created by adding dependencies only if they are not already well captured by the HMM, and only if they increase the model's ability to distinguish one object from another. Based on conditional mutual information, a new measure is developed and used for dependency selection. If dependencies are selected to maximize this measure, then the class posterior probability is better approximated leading to a lower Bayes classification error. The method can be seen as a general discriminative structure-learning procedure for Bayesian networks. In a large-vocabulary isolated-word speech recognition task, test results have shown that the new models can result in an appreciable word-error reduction relative to comparable HMM systems.

This thesis is dedicated to my parents, my sister, and all my nephews.

Contents

List of Figures

List of Tables

Acknowledgments

I would first like to thank my advisor and committee chairperson Nelson Morgan who has been highly supportive during my stay at U.C. Berkeley and the International Computer Science Institute (ICSI). His sincerity, integrity, intelligence, and latitude were always extremely valuable.

The other members of my committee included Steven Greenberg, Michael I. Jordan, and David Wessel. Many of the core ideas of this thesis originated during lively discussions with Steve and in Steve's ear-group meetings. The group's fecundity was due in no small part to Steve's creativity and intelligence. I would also like to thank Mike Jordan for being a truly inspirational teacher and researcher, and for being on my committee for two different theses at two different schools at two different times. I would also like to thank David Wessel whose knowledge, interest, and curiosity I have always admired.

It has been a pleasure working in the speech group at ICSI. I would like to thank both past and current members: Takayuki Arai, Herve Bourlard, Barry Chen, Dan Ellis, Eric Fosler-Lussier, Daniel Gildea, Ben Gold, Hynek Hermansky, Adam Janin, Dan Jurafsky, Brian Kingsbury, Katrin Kirchhoff, Yochai Koenig, John Lazzaro, Nikki Mirghafori, Michael Shire, Andreas Stolcke, Warner Warren, Gethin Williams, Chuck Wooters, and Su-Lin Wu. Special mention goes to Geoff Zweig with whom many interesting and productive discussions occurred.

The ICSI community and others at U.C. Berkeley and beyond have created an overall environment that I found both very interesting and enjoyable. I would like to acknowledge and thank Jim Beck, Chris Bregler, Sarah Coleman, Kathryn Crabtree, James Demmel, Jane Edwards, Jerry Feldman, Nir Friedman, Erv Hafter, Joy Hollenback, Vijay Iyer, David Johnson, Jitendra Malik, Kevin Murphy, Steve Omohundro, and Elizabeth Weinstein. They have all influenced me in positive ways. I would also like to acknowledge and thank my good friends Krste Asanović and Rob Lonsdale, and the many musician friends I have made while pursuing graduate study at U.C. Berkeley. A special thanks goes to Katrin Kirchhoff, both for her vast knowledge of speech, linguistic, and phonetic theory, and for her invaluable emotional and inspirational support. Finally, I also must acknowledge the parking lots where late at night much of this thesis was written, and last but not least, the `pmake` utility.

Chapter 1

Introduction

Contents

A human being perceives a sight, sound, touch, or taste and can immediately and appropriately react to such information from the environment. How is it that a person can categorize, differentiate, and gather information from the limitless collection of objects existing in the natural world? And based solely on past ontogenic and phylogenic experiences, a person can extract information from natural objects and natural scenes never before encountered.

Our world consists of natural objects existing within natural scenes. Consider, for example, trees against a landscape, a downtown city block, a sunset, or musical instruments performing within an orchestra. Contrast such scenes with the noise and static of a television set tuned to a non-existent station. One important and undeniable property of natural objects is their non-randomness. Natural objects have structure and redundancy — one can with high accuracy predict features that will exist in future instances of an object or predict one part of an object using information gained from other parts. Structure allows objects to stand out conspicuously from their background and allows them to be differentiated from each other. Furthermore, the structure is consistent — properties of objects are consistent throughout different instances of a particular object type. Consistency allows one to identify an object as what it is.

Perhaps humans use these structurally consistent patterns to recognize and categorize objects. To identify an object, perhaps we first identify an object's distinguishing features and then categorize it if the features match some stored representation well enough. These features, however, need not exist just at a high level (Tversky 1977). They could exist at a "subconscious" level and consist of complex patterns of correlation between multiple low-level deterministic feature detectors or receptive fields.

How can one build a machine to recognize natural objects? One way is to build a model of each object that represents its distinguishing features. The principle of parsimony requires a model to explain only what is necessary, and no more. If only the most identifying and distinguishing features of an object (those that are most consistent among different object instances) are represented, and the remaining attributes (those with greater variability) are ignored, then the resulting model will be both accurate and simple, satisfying the principle of parsimony.

This thesis is about automatically building computer models of natural objects using statistical properties of those objects measured from a corpus of data. In this thesis, however, the natural objects are human speech utterances, the models are statistical, and the task is automatic speech recognition. Like the visual, tactile, and olfactory scenes, the auditory scene (Bregman 1990) is filled with natural objects. Human speech sounds are one type of natural object in the auditory scene. Object recognition therefore becomes the production of a textual transcription of the spoken utterance, or more generally the identification of the intended meaning of the speaker. And like any natural object, if only the important and distinguishing properties of speech are represented by a model, perhaps good automatic speech recognition performance can be achieved with minimal complexity.

Statistical pattern recognition (Duda & Hart 1973; Fukunaga 1990; Bishop 1995), the method used in this thesis, provides one general framework within which such models may be designed. In this case, given a particular object class identifier M and a set of signals or low-level deterministic features describing a natural auditory scene X, the goal is to produce a model $p(X|M)$ that provides a high probability score for instances of the class.

This general approach of automatically developing models that accurately represent the statistical properties of natural objects has a number of attractions.

- **Data-Discovery**

  The automatic identification of models could produce winning yet unanticipated structures that might go unidentified if the models are selected only by hand. This is especially true if the objects are represented in a very high dimensional feature space.

- **Language or Domain Specificity**

  Each task might benefit from its own customized set of models. This meta-level step of first automatically deducing a model could allow different problem domains (such as different spoken languages, different types of speech such as isolated words, read speech, conversational speech, etc.) to use a set of models that provide a domain-specific performance increase.

- **Mimics the Neural System's Encoding of Natural Scenes**

  In the neurobiology community, it has been known for some time that natural information sources rarely if ever produce signals that are purely random. Their messages, instead, are encoded within signals that contain significant statistical structure and redundancy and obey certain physical constraints (Dusenbery 1992). More interestingly, it has been hypothesized that the neural system has evolved to accurately represent the statistical patterns found in natural scenes (Field 1987; Linsker 1990;

Atick 1992; Barinaga 1998; Rieke *et al.* 1997; Baddeley *et al.* 1997; Attias & Schreiner 1997). In these references, two general hypotheses have been advanced. First, it has been found that neural response patterns are significantly stronger for stimuli (usually visual) that have statistical properties similar to natural objects. Stimuli that have random properties (or properties that do not match those of natural objects) tend to have only weak neural responses. Furthermore, those stimuli that possess statistical properties resembling objects important for an organism's survival and reproduction (e.g., predators or potential mates) evoke the strongest neural responses. Second, the neural code in some instances has been accurately predicted based on information-theoretic codings of natural scene stimuli. In this case, the neural system can be seen as performing an entropy compression that is used to efficiently encode objects from natural scenes. Another interesting property of the human language system is that high-level syllabic statistical structure of language can play an important role in infant language learning (Saffran *et al.* 1996).

The above phenomena require some representation of the statistical properties of natural scenes. In other words, they require some form of joint probability distribution. This is because it is necessary to know what signals are statistically natural and, in those signals, where the patterns of redundancy typically exist.

To increase their chances of survival, natural organisms must make decisions about natural signals very quickly. In other words, the organism must perform some form of real-time causal signal encoding, and must have quick access to their "joint probability distributions" with which to make encoding decisions. It would therefore be to the organism's advantage if its model is simple and is optimized to represent only the important statistical properties of natural objects. It does not seem unreasonable, therefore, to predict (as is described in the above references) neural processing by studying statistical properties of natural scenes.

But if neural processing can be predicted from statistical properties of natural scenes, perhaps the statistical structure of probabilistic models can also be "predicted" in a similar way. In other words, perhaps the underlying goal of statistical model selection is similar to the process of evolutionary neurobiology.

- **Noise Filters**

  It is probable that the human auditory system is highly tuned to statistics of speech analogous to how the bat auditory system is tuned to its auditory environment (Suga 1988; Suga 1996). Such a specialization can help filter out noise sources whose statistical properties do not match that of speech. This is one of the approaches advocated by (Greenberg 1996) where spectral properties of sub-band modulation envelopes are band-pass filtered, retaining only those modulation frequencies that are found to be crucial for speech intelligibility. Generalizing on this approach, the hypothesis can be made that when noise is presented simultaneously with speech, the degree of speech intelligibility loss will be a function not just of the signal-to-noise ratio but also of the similarity between the statistical properties of the noise and speech.

  Moreover, if the properties only of speech are represented by a model, that model could act as a filter, essentially ignoring aspects of a signal containing non-speech-like statistical properties. In other words, such a model might possess noise robustness properties.

- **Parsimony and Computation**

  As mentioned above, if models represent only the important and distinguishing statistical properties of objects, then accurate models can be produced with minimal complexity.

- **Generalization**

  A good statistical model should accurately describe objects that it has not yet encountered. If the important statistical properties of a class of objects are accurately identified, and if a model is designed to accurately and parsimoniously represent those properties, the model should generalize better than if the model represented the less important properties.

  As is typical, there is a danger of "over-training" or "over-fitting" (Bishop 1995; Burnham & Anderson 1998) a model to the corpus of data used to identify the statistical properties. Overtraining is a general problem that can occur anytime there are too few constraints imposed on a system that has been trained using too small an amount of training data. Therefore, one must be careful to avoid this phenomenon.

The overall goal of this thesis is to automatically produce statistical models that represent those properties of speech that help to reduce recognition errors in automatic speech recognition systems. This is done by exploiting statistical structure of speech as measured using information theoretic constructs. As will be seen, this data-driven model induction procedure is not dissimilar to the model selection methods found in the field of statistics (Linhart & Zucchini 1986; Burnham & Anderson 1998).

Before getting into specifics, a brief overview is provided of automatic speech recognition and of some basic concepts needed from information theory.

1.1 Automatic Speech Recognition

Figure 1.1 illustrates the general method used by most speech recognition systems. The goal of a speech recognition system is to translate an acoustic spoken utterance into a sequence of words. The general approach is based on statistical pattern recognition (Duda & Hart 1973; Fukunaga 1990; Bishop 1995). This section will briefly describe the main components of a speech recognition system.

A speech signal is first digitized, and then converted by a deterministic feature extraction transformation into a representation that is more amenable to later processing stages. The goal of feature extraction is partially to modify the statistical properties of the speech signal's representation to match the statistical model (see below) while preserving the underlying message in the signal. There are standard types of "speech feature vectors" that are commonly used including mel-frequency cepstral coefficients (MFCCs) (Davis & Mermelstein 1980), RASTA-PLP features (Hermansky 1990; Hermansky & Morgan 1994),

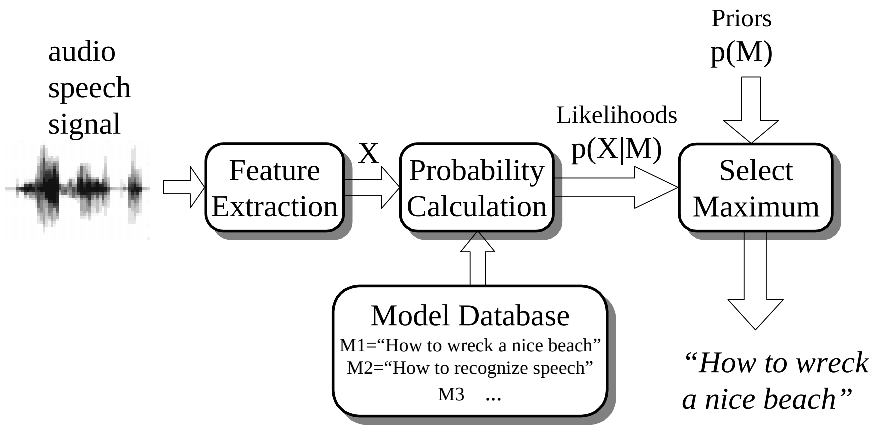

Figure 1.1: The general automatic speech recognition method.

and linear-predictive coefficients (Atal 1974; Makhoul 1975). An overview of the feature-extraction process can be found in Deller *et al.* (1993). In this thesis, the set of features will be referred to as X, which can be considered a random variable.

The next speech recognition stage involves the use of a database of models M each one representing a possible speech utterance. The models are trained using a corpus of labeled (i.e., transcribed, often at the word level) speech data. For each model M, a probability score $p(X|M)$ is produced for an unknown speech utterance. The quantity $p(X|M)$ is also called the likelihood of the model M given the data X. These likelihoods are combined with prior probabilities of each utterance $p(M)$ and the maximum is selected and used for the hypothesized speech utterance.

Because in general there are many utterances possible, it is infeasible to produce a distinct model for each one. Therefore, the speech models are hierarchically decomposed into different levels, the acoustic, phonetic, and linguistic. With this decomposition, models of entire utterances can share sub-models. For example, sentence models can share word models, and word models can share sub-word models. There are a variety of types of sub-word model that are used including phoneme, bi-phone, syllable, or demi-syllable models (Clark & Yallop 1995; Deller *et al.* 1993).

The acoustic models describe the structure of sound for each sub-word unit. In other words, each sub-word unit has an associated probability distribution over a varying length sequence of feature vectors. Acoustic models typically used to represent the distribution of each feature vector include parametric mixture densities (Titterington *et al.* 1985), neural networks (Bishop 1995), and discrete distributions over vector quantized (Gray & Gersho 1991) elements.

The phonetic (or word pronunciation) models describe the statistical structure of words. For each possible word in the system, a probability distribution is defined on a set of sequences of sub-word units (i.e., pronunciations) that can make up the word. Decision trees (Breiman *et al.* 1984) are often used to produce word pronunciation models.

The linguistic models define probability distributions over sequences of words in an attempt to model language. First and second order Markov chains (referred to as bi- and tri-grams) are often used to describe sequences of words but more complex models can also be used (Jelinek 1997).

The models at each of the above levels are combined and used to produce composite models $p(X|M)$, one for each possible utterance. The composite models are represented using hidden Markov models (HMMs) which will be extensively discussed in Chapter 3.

These are the bare essentials of modern speech recognition systems. Many more details will be described in later chapters of this work. How well does such a system do on modern speech recognition tasks? At the time of this writing (May 1999), there are many available commercial products that successfully recognize certain types of speech. On the other hand, at the most recent DARPA Broadcast News workshop (DARPA Broadcast News Workshop 1999), the best individual system achieved word-error performance[1] of about 12 or 13 percent. Moreover, at the most recent LVCSR (LVCSR Workshop 1998) workshop, the best results on the Switchboard corpus (Godfrey *et al.* 1992) was about 28 percent and on the Call-Home (LVCSR Workshop 1998) corpus about 40 percent. Humans, on the other hand, achieve about 3 percent on such corpora. Clearly, modern speech recognition systems must be improved.

In a recent study (McAllaster *et al.* 1998), it was surmised that a large portion of errors on such corpora can be accounted for by statistical modeling problems at the acoustic and phonetic level. In their paper, they evaluate a speech recognition system on synthesized as well as real speech and study how using perfect acoustic models can influence word error. Perfect acoustic models are obtained by synthesizing speech data using previously trained acoustic models which are then also used by the recognition system. Pronunciation models are also varied either by 1) recognizing synthesized speech data using a pronunciation dictionary that has also been used to generate the speech data, or 2) including hand-transcribed test-data pronunciations in the test dictionary.

They found that if both the acoustic and pronunciation models of the speech recognition system are perfect (synthesized data using the acoustic models and the dictionary pronunciations), then recognition word error falls by an order of magnitude. If only the acoustic models are perfect (i.e., synthesized data using the acoustic models and the hand-transcribed pronunciations) and the pronunciation dictionary has been augmented with the hand-transcribed test pronunciations, performance gets better but not by nearly as much. And if only the pronunciations are improved by including the hand-transcribed test pronunciations in the test dictionary (i.e., imperfect acoustic models since the real acoustic data is used) then performance actually gets worse.

A conclusion of this paper is that, one way to significantly improve speech recognition performance is to simultaneously improve both the acoustic and the pronunciation models. Generalizing their result somewhat, an additional hypothesis that can be made is that, to significantly improve a speech recognition system, components at all levels (feature extraction and acoustic, pronunciation, and language modeling) of a speech recognition system should be simultaneously improved (also see Section 7.2). Typically this is not practical because of the combinatorially large set of possibilities implied by simultaneous improvements — accordingly, the test results provided in this thesis (Chapter 6) evaluate improvements applied only to the acoustic modeling component of an automatic speech recognition system. This is done using a new technique to relax the HMM conditional in-

[1]Speech recognition systems are typically judged by their word-error performance which is defined as the percentage of words incorrectly recognized by a system.

dependence assumptions in a principled and data-driven way. As will be seen, however, the general approach can be used to adapt the acoustic models to any type of improvements performed simultaneously at other levels.

1.2 Essentials of Information Theory

Elements of information theory will be used extensively throughout this thesis. This section provides a very brief overview of the information theoretic concepts relevant to this work. More detailed coverage is provided in a number of excellent texts (Cover & Thomas 1991; Gallager 1968; Kullback 1968).

Let X be a random variable, and $\{X = x\}$ be the particular event that the random variable X takes on the value x. This event has a probability $p(X = x)$ or more concisely just $p(x)$. For notational convenience, the event $\{X = x\}$ might simply be written as x. In general, the smaller the probability $p(x)$ is, the event $\{X = x\}$ is viewed as being more surprising. Each event can be thought of as conveying an amount of information that is related to its surprise. One way of representing the amount of information provided by an event, therefore, is using its probability. Improbable events provide much information since they are unexpected, and probable events provide little information since they are expected. The inverse of $p(x)$ can be thought of as conveying the amount of information provided by the event $\{X = x\}$ as can the quantity $\log(1/p(x))$. The amount of information provided by an event is therefore defined as follows:

$$\text{Information provided by the event } \{X = x\} \text{ is equal to } -\log p(X = x)$$

Note that $-\log p(X)$ can be seen as a random variable which has an expected value. The average of this random variable is the average information provided by X:

$$\text{Average information provided by } X \text{ is equal to } H(X) \triangleq -\sum_x p(x) \log p(X = x)$$

The quantity $H(X)$ is called the entropy of the random variable X. Entropy can be similarly defined for continuous random variables.

The random variable X can be thought of as an information source providing messages in some alphabet where each message has a certain probability. The entropy $H(X)$ is the minimum number of bits per message on average that have to be used to losslessly represent messages generated from this source.

Let Y also be a random variable that might be randomly related to X. If it is known that the event $\{Y = y\}$ occurred, this might affect the probability of the event $\{X = x\}$ as is reflected by the conditional probability distribution $p(x|y)$. Following an analogous reasoning to the above, $-\log p(x|y)$ can be seen as the information provided by the event x given y. Averaging over all x and y produces the conditional entropy:

$$H(X|Y) = -\sum_{xy} p(x, y) \log p(x|y)$$

The quantity

$$\log \frac{1}{p(x)} - \log \frac{1}{p(x|y)} = \log \frac{p(x|y)}{p(x)} = \log \frac{p(x, y)}{p(x)p(y)}$$

can be seen as the difference between the information provided by the event x and the information provided by the event x given y. The average of this quantity over all x and y is called mutual information and is defined as:

$$I(X;Y) = \sum_{xy} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

If X is thought of as an information source, and Y a receiver or a version of X after being sent through some transmission channel, $I(X;Y)$ is the amount of information (in bits) on average that is transmitted between X and Y through the channel. The quantity $I(X;Y)$ is sometimes referred to as the information transition rate between X and Y, i.e., it is the number of bits per symbol transmitted on average.

Finally, if Z is a random variable, the quantity

$$\log \frac{1}{p(x|z)} - \log \frac{1}{p(x|y,z)} = \log \frac{p(x,y|z)}{p(x|z)p(y|z)}$$

can be seen as the difference between the information provided by the event x given z and the event x given y and z. Taking averages in this case defines the conditional mutual information:

$$I(X;Y|Z) = \sum_{xyz} p(x,y,z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)}$$

The conditional mutual information may also be represented as:

$$I(X;Y|Z) = \sum_{z} I(X;Y|Z=z)p(Z=z)$$

where

$$I(X;Y|Z=z) = \sum_{xy} p(x,y|z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)}$$

is the conditional mutual information between X and Y under the event $\{Z=z\}$.

Entropy, conditional entropy, mutual information, and conditional mutual information can be depicted using the Venn diagrams given in Figure 1.2. A variety of relationships exist between these quantities (Cover & Thomas 1991), many of which can be easily "read off" from the Venn diagrams. For example, it is easy to see relationships such as

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X),$$

$$I(X;Y,Z) = I(X;Y) + I(X;Z|Y),$$

and

$$I(X;Y;Z) = I(X;Y) - I(X;Y|Z)$$

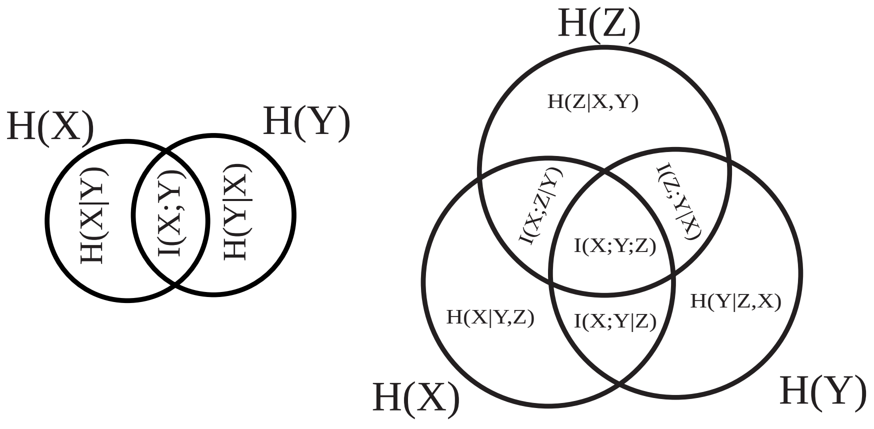

Figure 1.2: Venn diagrams for entropy, conditional entropy, and mutual information (left) and conditional mutual information (right).

where $I(X;Y;Z)$ can be thought of as the mutual information common among the three variables X, Y, and Z.

Another important quantity is the KL-distance (Kullback 1968) between two probability distributions $p(x)$ and $q(x)$ which is defined as:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

This distance is not a true metric since, for example, $D(p||q)$ is not in general equal to $D(q||p)$, but it has the important property that it is always greater than or equal to zero and that it is zero if and only if $p(x)$ and $q(x)$ are identical. Note that mutual information can be defined as the KL-distance between a joint probability distribution and the product of marginal distributions:

$$I(X;Y) = D\big(p(x,y)||p(x)p(y)\big).$$

1.3 Notation

This section briefly reviews the notation used in this work. Capital letters such as X and Y refer to random variables. Lower case letters such as x and y refer to possible values of these random variables. If X is a random variable according to some probability distribution p, it will be written $X \sim p(X)$. Similarly, if x is a sample from that distribution, it will be written $x \sim p(X)$. For notational simplicity, $p(X)$ will at different times represent either a continuous probability density or a discrete probability mass function. The distinction will be clear from the context. The letter Q will typically indicate a discrete random variable taking values in the set $\mathcal{Q}$ of size $|\mathcal{Q}|$. Q will often be used to represent one variable in a Markov chain. The quantities X and x might represent either scalars or vectors. The quantity X_t is the variable at time t. The quantity $X_{r:t}$ is the set of variables $\{X_r, X_{r+1}, \ldots, X_t\}$. The quantity $X_{<t}$ is the set $\{X_1, X_2, \ldots, X_{t-1}\}$. And $X_{\neg t}$ is the

set of variables $\{X_1, X_2, \ldots, X_{t-1}, X_{t+1}, \ldots, X_T\}$ where an implicit length parameter T is assumed.

1.4 Thesis Overview

This thesis is organized as follows. Chapter 2 presents an overview of graphical models and Bayesian networks. Graphical models allow one to reason about the structure of a statistical model without needing to consider any parameterization. Chapter 3 first briefly reviews stochastic processes and Markov chain theory. It then defines hidden Markov models (HMMs) and argues that HMMs have more flexibility then it is sometimes suggested. At the end of this chapter, the principle of parsimony is used as a guideline to suggest ways to produce models that could be better than HMMs. Chapter 4 describes a new type of extension to the HMM, called the buried Markov model (BMM). In this chapter, it will be shown how BMMs are derived by adapting the model's statistical structure to match the natural properties of speech. In Chapter 5 an implementation of BMMs is described and in Chapter 6 this implementation is tested on several isolated-word speech corpora. It is shown that BMMs can produce a lower word-error rate than certain HMMs. Finally, Chapter 7 summarizes and then outlines a variety of future projects suggested by this thesis.

Chapter 2

Graphical Models

Contents

In the previous chapter, it was argued that when choosing models to represent objects originating from natural scenes, the models should somehow represent the distinguishing properties of the objects. If the objects are represented by a collection of features, and if the models are probabilistic, then the models must somehow represent patterns of redundancy inherent in the objects' feature representation.

There are three separate components to a probabilistic model: the structure, the implementation, and the parameterization. The structure of a model corresponds to the set of dependency relationships that a model is inherently able to represent. Bayesian networks are discussed in this chapter. As will be seen, with a Bayesian network one may reason about the structure of a probabilistic model as determined by the model's set of conditional independence properties. The implementation of a model corresponds to the way in which two random variables related by direct dependency can affect each other. For example, given two random variables A and B that are directly related, their relationship could be modeled by conditioning the mean of B as either a linear or a non-linear function of A — the choice is determined by the implementation. Finally, the parameterization of a model corresponds to the actual parameter values of a specific implementation of a particular structure. In the above example, the parameterization of the conditional mean of A might be such that $E[A] = 4B$, where the number 4 is the parameter.

In general, the properties represented by a probabilistic model are determined by its structure, implementation, and parameterization. For example, consider the task of representing dependencies between elements of a multi-dimensional random vector. One implementation choice is the set of multi-dimensional Gaussian distributions. Intra-vector dependence is not represented if the off-diagonal elements of the covariance matrix are zero so the properties of the model are governed by the model's parameters. On the other hand, a Gaussian distribution can not accurately represent a multi-modal distribution regardless of the parameters. To represent such a distribution, a different structure must be used. A model space can be seen as the union of different model structures, implementations, and parameterizations. In this work it will be beneficial to consider the selection of model structure separately.

This chapter consists of a brief overview of graphical models and Bayesian networks. Among other advantages, a graphical model provides intuitive ways to reason about the structure of statistical models without needing to consider either a particular implementation or a parameterization. While the graphical model literature is rich with mathematical rigor (Lauritzen 1996; Pearl 1988), this chapter provides only the basic machinery necessary to discuss the structures considered in the later chapters of this work.

2.1 Graphical Models for Density Estimation

In general, the task of density estimation can be seen as choosing the best $\hat{p}_{\mathcal{I}\Theta} \in \mathcal{P}$ where $\mathcal{P}$ is a space of all models under consideration, and $\hat{p}_{\mathcal{I}\Theta}$ is a model with a particular structure, implementation $\mathcal{I}$ and parameterization Θ. This choice is typically made empirically using a size T set of training samples consisting of N-dimensional data-vectors or features $D = \{x_1, x_2, \ldots, x_T\}$. The random samples are presumably drawn i.i.d. (see Definition 3.1) from some true underlying source probability distribution p for the random vector X. The "true" p is typically unknown and does not necessarily live within $\mathcal{P}$. The choice of model structure $\hat{p}$ and the particular implementation and parameterization can be viewed as distinct aspects of the problem.

The simplest approach to this problem chooses a model that allows for statistical dependencies to exist between all possible subsets of the elements of X. An N-dimensional histogram is an example if the elements of X are discrete — choosing parameter values then becomes a problem of counting. A naively chosen model such as this, however, will quickly lead to estimation inaccuracies, computational intractabilities, and exorbitant demands on the training set size as the dimensionality of X increases.

In general, the amount of training data needed to produce an accurate estimate of a source grows exponentially with N. If the training data set is too small and the model is too complex, overfitting will occur in which case the model represents the idiosyncrasies of the particular training data sample rather than the actual properties of the underlying source object. This data growth problem is often referred to as the curse of dimensionality (Duda & Hart 1973; Bishop 1995).

In many cases, certain elements of X might be statistically independent of each other and representing direct dependencies between these elements is irrelevant to the task of producing an accurate model. An approximation that ignores those dependencies will to some extent mollify the problems mentioned above. One way to ignore dependencies is to change the structure of the model. A graphical model is one way of specifying a subset of dependencies used for a density-estimation task. More precisely, a graphical model explicitly specifies the conditional independence properties of a distribution over a collection of random vectors.

It is said that a random variable X is conditionally independent of random variable Y given random variable Z under a given probability distribution $p(\cdot)$, if the following relation holds:

$$p(X = x, Y = y | Z = z) = p(X = x | Z = z) p(Y = y | Z = z)$$

for all x, y, and z. This is written $X \perp\!\!\!\perp Y | Z$. The conditional independence of X and Y given Z has the following intuitive interpretation: if one has knowledge of Z, then knowledge of Y does not increase one's knowledge of X and vice versa. Conditional independence does not imply unconditional (or equivalently marginal) independence — X and Y can be either conditionally dependent or independent given Z irrespective of X and Y's marginal independence. Many properties of conditional independence are provided in (Lauritzen 1996; Pearl 1988).

A graphical model for a collection of random variables is a graph $\mathcal{G} = (V, E)$ where V is a set of vertices and the set of edges E is a subset of the set $V \times V$. The vertex set V is in one-to-one correspondence with the collection of random variables. One vertex can be use for each scalar random variable or a vertex can correspond to an entire vector of random variables. In the latter case, the vertex implicitly corresponds to a sub graphical model over the individual elements of the vector. In general, the set V may refer both to the set of vertices of the graph and to the set of corresponding random variables. Associated with a graphical model is a probability distribution $p(V)$ over the set of random variables. The edge set E of the model in one way or another specifies a set of conditional independence properties of the associated probability distribution.

A graphical model may posses one or more properties, called Markov properties. A Markov property on a graph is a set of conditional independence assumptions about collections of the graph's vertices. The collections of vertices for which a particular property holds are determined using graph operations (such as subset selection, adjacency, closure, disjointness, etc.). The graph's associated probability distribution may or may not obey different Markov properties with respect to the graph. If a certain Markov property does hold for a graph and for a particular probability distribution, then the graph allows one to reason about conditional independencies among the vertices without having to refer directly to the distribution or its specific parameterization. To be complete, one should specify both a graphical model and its Markov property. Typically, however, the exact Markov property is not stated explicitly and a particular one is assumed by convention for each type of graph.

Different types of graphical models may have different Markov properties. A probability distribution obeying one Markov property with respect to one graphical model may or may not obey a different Markov property with respect to a different graph — in general, it depends on both on properties of the distribution and on the graph. A formal study of these relationships is given in (Lauritzen 1996).

Because a graphical model is dissociated from its probability distribution, a graphical model shows neither the implementation of the dependencies nor the particular parameterization. For example, two vertices connected by an edge could correspond to a variety of implementations such as a conditional histogram or a Gaussian with a conditional mean. Also, the values of the table (or the mean in the dependency matrix) are not specified by a graphical model. An edge simply says that the two variables are somehow directly related. A graphical model therefore provides an easy way to view the inherent structure that a probability distribution imposes on a collection of random variables. Via the graph's edges, a graphical model shows which variables are sufficient for determining the statistical properties of other variables.

Graphical models provide a useful tool to reason about the statistical properties of natural objects. For example, one may designate that one feature element should have a strong affect on another element. In other words, one may first design a graphical model that is known to adequately describe the statistical patterns in features representing a type of natural object, and then afterwards specify a probability distribution that obeys the conditional independence properties of that graph. The distribution will then have the ability to represent the structure and redundancy of the object without using many extraneous parameters. This can significantly reduce the parameter estimation complexity and cause only a small decrease in model accuracy.

For the purposes of this thesis, it suffices to quickly summarize four different types of graphical model, and then explore one of them in more detail. In each case, the model will correspond to conditional independence assumptions over a collection of random variables under some assumed distribution. Each graphical model will consists of a type of graph and a particular Markov property.

Markov Random Field (MRF) Markov random fields (Derin & Kelley 1989; Dubes & Jain 1989; Chellappa 1985; Kashyap 1981; Pearl 1988) are also called undirected models (Lauritzen 1996) since the corresponding graphs have undirected edges. Markov random fields satisfy what is known as the global Markov property, which states that a collection of variables are independent of all other variables given all neighbors (or the boundary) of that collection. It can be shown that the corresponding distribution over a collection of random variables $X_{1:N}$ factorizes as follows:

$$p(X_{1:T}) = \frac{1}{Z} \prod_{c=C} \phi_c(X_c)$$

where C is the set of all cliques in the graph, where $\phi_c(\cdot)$ are a set of clique potential functions, and where Z is a global normalization constant. A clique is a set of nodes in a graph that are fully connected.

An example MRF is given in Figure 2.1. In the figure, the cliques are {A,B}, {A,C}, {B,D}, {C, D}, and { D, E, F}. Some of the conditional independence relationships implied by this graph include: $\{E, F\} \perp\!\!\!\perp \{A, B, C\} | \{D\}$, $\{A\} \perp\!\!\!\perp \{D, E, F\} | \{B, C\}$, etc. The joint probability distribution according to this graph can be represented as:

$$\begin{aligned} & p(A = a, B = b, C = c, D = d, E = e, F = f) \\ = \quad & \phi_{A,B}(a, b)\phi_{A,C}(a, c)\phi_{B,D}(b, d)\phi_{C,D}(c, d)\phi_{D,E,F}(d, e, f) \end{aligned}$$

Computational efficiency is one important consequence of the factorization property. In general, one desires MRFs with small clique sizes because MRF complexity increases exponentially with clique size. One example of a MRF is a Gibbs distribution (Derin & Kelley 1989) where the overall joint distribution is a member of the exponential family. The main difficulty with MRFs is the calculation of the global normalization constant.

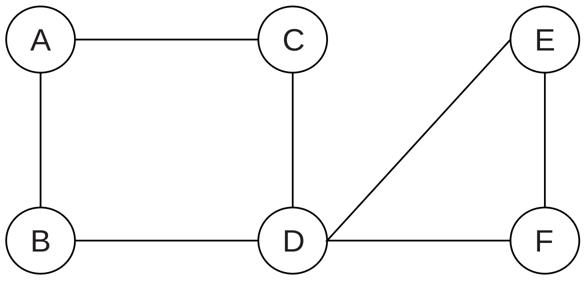

Figure 2.1: A simple Markov random field

Bayesian Networks Bayesian networks are also called directed graphical models because they use only directed edges that form directed acyclic graphs (DAGs). Bayesian networks will be discussed in detail below.

Decomposable (triangulated) Models

The conditional independence properties of certain probability distributions are not perfectly representable by both a MRF and a Bayesian network. Those distributions that are perfectly representable by both can be represented using decomposable models. Their corresponding undirected graphs are necessarily triangulated which means that any cycle of length greater than three must have a chord between two non-adjacent vertices along the cycle. Decomposable models, therefore, comprise the intersection of MRFs and Bayesian networks.

In decomposable models, the cliques of the graph can form a tree called a Junction tree. Between two cliques along the tree are separator sets which consist of the nodes in the intersection of the two adjacent cliques. One of the advantages of decomposable models is that the joint distribution over all variables can be factorized as a product of clique marginals over a product of separator marginals, i.e.:

$$p(X_{1:N}) = \frac{\prod_{c=C} \phi_c(X_c)}{\prod_{s=S} \phi_s(X_s)}$$

where C is the set of cliques and S is the set of separators.

This product representation has important computational implications since global probabilistic inference can be performed by manipulating only local quantities. Specifically, the marginal probability of the nodes in a clique can be specified using just the clique potential function. It suffices to say that all models considered in this work will be decomposable. More detail on decomposability is described in (Pearl 1988; Lauritzen 1996; Jensen 1996).

Chain Graphs Chain graphs are the most general form of graphical model. Their edges can be either directed or undirected. Chain graphs and their possible Markov properties are described in (Lauritzen 1996) but are not discussed further in this work.

2.2 Bayesian Networks

Bayesian networks (or belief networks) are perhaps the most common type of graphical model. Because Bayesian networks encompass both hidden Markov models (Chapter 3) which are widely used for automatic speech recognition and the HMM extensions proposed in Chapter 4, they are discussed extensively in this section.

In a Bayesian network, it is said that if there is an edge going from node A to node B, then A is a parent of B and B is a child of A. These notions are extended so that one may talk of the ancestors, descendants, etc. of a node. The collection of nodes and directed edges in a Bayesian network form a directed acyclic graph (DAG). Directed cycles are not allowed in Bayesian networks since such cycles would suggest conditional independence properties that only a more general graphical model could represent.

Directed edges are often used to depict causal reasoning. In this case, each node represents some event which can, with some probability, affect the outcome of some other event. To this end, Bayesian networks are used in a variety of real-world applications (Heckerman *et al.* 1995) where causal reasoning, combined with a degree of uncertainty, can be useful. Directed edges can also be used to depict "relevance," where the variables most relevant for influencing a certain variable are given by the variable's neighborhood.

Like any graphical model, the directed edges in a Bayesian network represent conditional independence properties over the corresponding variables. In this case, the conditional independence properties of a network depend on the direction of the edges. Figure 2.2 shows two Bayesian networks with the property that A is conditionally independent of C given B. On the left side, the variables form a standard three-variable first-order Markov chain $A \rightarrow B \rightarrow C$ (also see Section 3.1.1). On the right side, the same conditional independence property is represented although one of the arrows is pointing in the opposite direction.

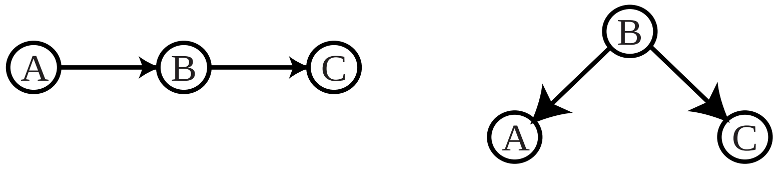

Figure 2.2: A is conditionally independent of C given B

Figure 2.3 depicts the case where variables A and C are marginally independent but given B they no longer are independent. This can be seen by noting that:

$$p(A, B, C) = p(B|A, C)p(A)p(C)$$

so

$$p(A, C) = \sum_b p(A, B, b) = p(A)p(C) \sum_b p(b|A, C) = p(A)p(C)$$

which means that the variables A and C are marginally independent. On the other hand, the quantity $p(A, C|B)$ can not similarly be represented as a product of two factors. This is the notion of "explaining away" described in Pearl's book (Pearl 1988) — suppose the random variables are binary and suppose, as listed in Figure 2.3, that A probabilistically implies B, and C implies B. If it is found that B is true, then B's cause could either be A or C. Therefore, the probability of both A and C increases. If we then find that C is true, this in general makes A less probable, so A is said to have been explained away. If we find that C is false, then we know A is true with high probability since it is the only remaining explanation. The notion of explaining away extends to continuous random variables as well.

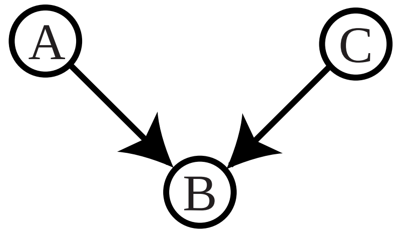

Figure 2.3: A is conditionally *dependent* with C given B

Each variable in a Bayesian network is independent of its non-descendants in the graph given its parents. This is called the local Markov property which, for all distributions considered herein, is equivalent to the global Markov property (Lauritzen 1996) on a corresponding undirected graph. Both of these properties are equivalent to the concept of *d-separation* defined as follows: a group of variables A is conditionally independent of another group of variables B given a third group of variables S if the set S d-separates A from B (Jensen 1996). Two sets of variables A and B in a network are d-separated by a third set of variables S if and only if all paths that connect any node in A and any other node in B have the following property: there is a node v in the path such that either:

- $v \in S$ and the arrows along the path do not converge at v
- $v \notin S$, any descendant of v is not in S, and the arrows along the path converge at v

In other words, for any such path, there must be a node v along the path with a "YES" in the following table:

	$v \in S$	$\Big(v \notin S\Big) \wedge \Big(\text{de}(v) \cap S \neq \emptyset\Big)$	$\Big(v \notin S\Big) \wedge \Big(\text{de}(v) \cap S = \emptyset\Big)$
Arrows Converge at v	NO	NO	YES
Arrows do Not Converge at v	YES	NO	NO

where $de(v)$ is the set of all descendants of v.

Variables in a Bayesian network can be either *hidden*, which means they have an unknown value and represent a true random variable, or they can be *observed*, which means that the values are known. When the question "is $A \perp\!\!\!\perp B|C$?" is asked, it is implicitly assumed that A and B are hidden and C is observed. In general, if the value is known (or if "evidence" has been provided) for a particular node, then it is considered observed — otherwise, it is considered hidden. Probabilistic inference using a network with hidden variables must somehow "marginalize away" the hidden variables to produce the resulting probability of the observed variables. A Bayesian network does not, however, require a variable to always be either hidden or observed. Rather, a variable is either hidden or observed depending on the question that is asked of a Bayesian network. For example, if one asks "what is the probability $p(C = c|A = a)$?" for the graph in Figure 2.2, then B is hidden and A is considered observed. If one asks "what is the probability $p(C = c|B = b)$

or $p(A = a|B = b)$?" then B is considered observed. In this way, a variable in a Bayesian network can be either an input (or observed) variable, a hidden variable, or an output (the object of a query) variable.

There is a relatively easy way, known as the Bayes ball algorithm (Shachter 1998), to determine if two variables are independent given some other set of variables. First, shade in all the nodes of the graph that are observed (or conditioned against). Call this set S. To answer the question "is $A \perp\!\!\!\perp B|S$?," start a ball in A and bounce it along the nodes according to the rules illustrated in Figure 2.4. If it is possible to reach B, then the answer to the question is negative.

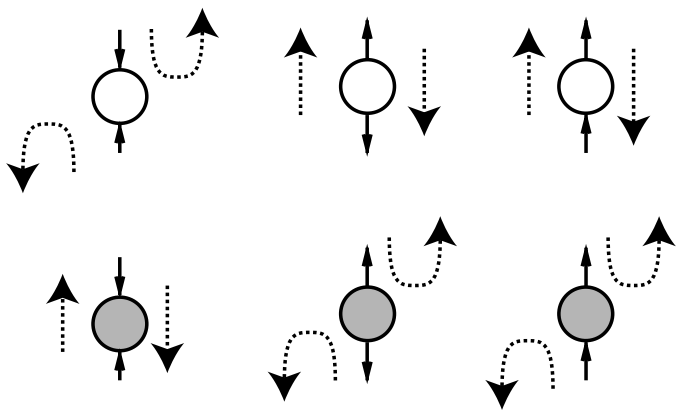

Figure 2.4: The Bayes ball algorithm. A hypothetical ball starting from one node may bounce over the graph using the rules illustrated above. Dashed arrows indicate that the ball may either bounce through or bounce off the given node along the directed edges as shown. Hidden nodes are unshaded, and observed nodes are shaded. The ball may pass through a hidden node if there are any diverging arrows, and may pass through an observed node only if the arrows are convergent.

For a given collection of random variables, one of the most important problems is the computation of the probability distribution of one subset given values of some other subset. This is called probabilistic inference. Probabilistic inference is essential both to make predictions based on the network, and to learn the network parameters using, for example, the EM algorithm (Dempster *et al.* 1977). One of the reasons Bayesian networks are useful is because they permit a more efficient inference procedure than would be obtained by simply marginalizing away all unneeded or hidden variables ignoring conditional independence properties.

There are two types of inference, exact and approximate. Exact inference procedures are useful when the networks are not too complex because in the general case inference is NP-Hard (Cooper & Herskovits 1990). The most popular exact inference method is the junction tree algorithm (Jensen 1996). Essentially, a Bayesian network is converted into a decomposable model via moralization (a process that adds links between the unconnected parents of a node) and triangularization (a process that adds edges to cycles of length greater than three that do not possess a chord). The resulting decomposable model represents a subset of the original conditional independence properties, but since it is decomposable it has the desirable properties described above. For a network consisting of only discrete valued nodes, the complexity of this algorithm is $O(\sum_{c \in C} \prod_{v \in c} |v|)$ where C is the set of cliques in the junction tree, c is the set of variables contained within a clique, and $|v|$ is the number of states (i.e., possible values) of variable v. It can be seen that this algorithm is exponential in the size of the cliques, so when performing moralization and triangulation, minimizing the resulting clique sizes is desirable. It turns out that the commonly used Forward-Backward algorithm (Rabiner & Juang 1993; Huang *et al.* 1990) used to perform inference on and train HMMs is a special case of the junction tree algorithm (Smyth *et al.* 1996).

Approximate inference procedures are used when the clique state space size is too large. Several different types of approximation methods exist including mean field and variational techniques (Saul *et al.* 1996; Jaakkola & Jordan 1998; Jordan *et al.* 1998), Monte Carlo sampling methods (MacKay 1998), and loopy belief propagation (Weiss Submitted). Even approximate inference can be NP-Hard however (Dagum & Luby 1993). Therefore, it is crucial to use a model with the smallest possible complexity.

2.3 Examples of Bayesian Networks

The following paragraphs provide various examples of Bayesian networks and describe how they might be used.

Given a collection of random variables $X_{1:N}$, the chain rule of probability says that:

$$p(X_{1:N}) = \prod_n p(X_n | X_{1:n-1})$$

Each factor on the right hand side of this equation can be thought of as saying that X_n depends on the previous variables $X_{1:n-1}$ as shown on the left in Figure 2.5. If it is known that for each n there is some set π_n (the parents of X_n) such that $X_n \perp\!\!\!\perp \{X_{1:n-1} \setminus \pi_n\} | \pi_n$, then the following is also an exact representation of the joint probability distribution:

$$p(X_{1:N}) = \prod_t p(X_n | \pi_n)$$

This equation can be depicted by a Bayesian network, as shown on the right in Figure 2.5. It turns out that this type of "factorization" is a general property of Bayesian networks and is one of their main advantages: instead of computing the joint probability as a product of a set of relatively complex factors, the joint probability is factored into the product of much less complex and more pertinent quantities.

A Gaussian mixture model is a probability distribution that is a weighted sum of Gaussians as follows:

$$p(x) = \sum_i c_i p_i(x | \mu_i, \Sigma_i)$$

where $p_i(x) = \mathcal{N}(x; \mu_i, \Sigma_i)$ is a Gaussian distribution with mean μ_i and covariance Σ_i. This distribution can be represented by the network shown in Figure 2.6.

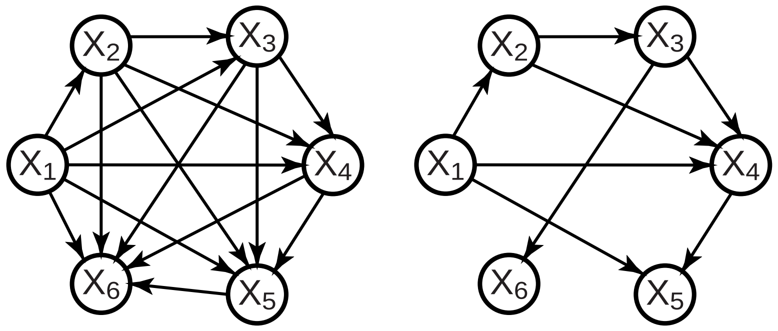

Figure 2.5: On the left is the Bayesian network representation of the chain rule of probability. On the right, conditional independence assumptions have been made (e.g., $X_3 \perp\!\!\!\perp X_1 | X2$, $X_5 \perp\!\!\!\perp \{X_2, X_3\} | \{X_1, X_4\}$, etc.)

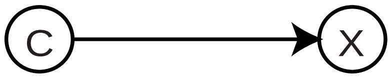

Figure 2.6: A Bayesian network model when $p(X)$ is a mixture distribution and C is a mixture variable.

A hidden Markov model (HMM) (Rabiner & Juang 1993; Huang *et al.* 1990) is a Bayesian network with hidden variables $Q_{1:T}$ and observed variables $X_{1:T}$. The hidden variables form a first order Markov chain and each observed variable is conditionally independent of everything else given its corresponding hidden variable. Figure 2.7 shows the graphical model for an HMM. This is also the structure for the Kalman Filter (Haykin 1989), but in that case, all the variables are continuous single-component Gaussians. In light of the previous discussion on conditional independence and directed models, it can be seen that there are two conditional independence properties associated with an HMM:

$$Q_t \perp\!\!\!\perp \{Q_{1:t-2}, X_{1:t-1}\} | Q_{t-1} \tag{2.1}$$
$$X_t \perp\!\!\!\perp \{Q_{1:t-1}, Q_{t+1:T}, X_{1:t-1}, X_{t+1:T}\} | Q_t \tag{2.2}$$

Inference in HMMs is extremely efficient because of very small clique sizes (i.e., two). HMMs are a subset of a class of models often called Dynamic Bayesian networks (DBNs) (Dean & Kanazawa 1998; Ghahramani & Jordan 1997; Ghahramani 1998) which are essentially collections of identical Bayesian networks strung together with arrows pointing in the direction of time (or space). HMMs, their capabilities, their deficiencies, previously proposed extensions, and their application to automatic speech recognition will all be discussed in detail in Chapter 3. New extensions to HMMs will be introduced in Chapter 4

In a typical statistical pattern classification task, the goal is to identify the object class with the highest probability. That is, find:

$$c^* = \operatorname*{argmax}_c p(c|X) = \operatorname*{argmax}_c p(X|c)p(c)$$

where c identifies the object class and X is a random vector. A Bayesian network can model this as well. The network can consist of the set of feature variables $X_{1:N}$ augmented

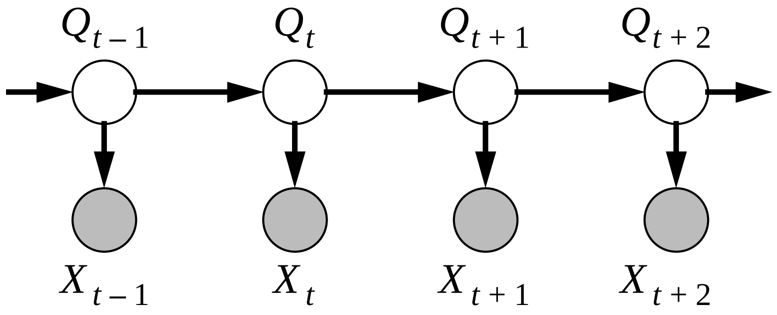

Figure 2.7: A Hidden Markov Model

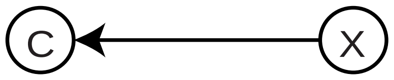

Figure 2.8: A Bayesian network model that could represent a multi-layered perceptron with a softmax output non-linearity.

with a class variable C. Computing the posterior probability $p(c|X)$ for each value of c is a standard probabilistic inference problem that can be solved using the methods mentioned above. Alternatively, $|C|$ distinct networks could be defined with different parameters (and perhaps different structures) for each class c. The decision problem becomes:

$$c^* = \operatorname*{argmax}_c p_c(X)p(c)$$

In the most general form, it can be seen that these approaches are equivalent — in the second case there is an implicit link from a C variable to each class-conditional network.

A multi-layered perceptron (MLP) (Hertz *et al.* 1991; Bishop 1995) with a softmax output nonlinearity can be seen as an implementation of the probability $p(C|X)$ and therefore as an implementation of a particular Bayesian network. An MLP can be represented by the Bayesian network shown in Figure 2.8. An MLP makes no conditional independence assumptions about the elements of its input variables.

The "naive Bayes classifier" (Langley *et al.* 1992) is depicted in Figure 2.9 where the X_i variables are input features and C is a class variable. Data presented at the input X_i determines a probability distribution over C which is used to make a classification decision. This classifier makes the assumption that each variable is independent of other variables given the class variable C, that is:

$$p(C|X_{1:N}) = p(X_{1:N}|C)p(C)/p(X_{1:N}) \propto p(X_1|C)p(X_2|C)\ldots p(X_N|C)p(C),$$

which allows for very efficient inference.

More examples of Bayesian networks may be found in the following references (Machine Learning 1997; Jensen 1996; Pearl 1988; Heckerman *et al.* 1995; Jordan 1998; Frey 1998; Zweig 1998).

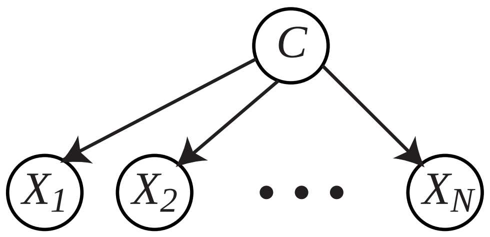

Figure 2.9: The naive Bayes classifier.

2.4 Graphical Model Parameter Learning

Once a given network structure has been specified, the implementation of the local conditional distributions for each node must be selected. There are many choices such as the multinomial, a conditional Gaussian, a mixture of other distributions, etc. Accompanying each choice is a set of parameters.

Letting X be the entire collection of variables in a network, one is typically given a finite sample of data $D = \{x_1, \ldots, x_T\}$ drawn presumably i.i.d. from the corresponding real distribution. From these samples, the goal is to optimally estimate the corresponding real distribution by adjusting the network parameters. This can be formulated as a risk minimization problem (Duda & Hart 1973; Vapnik 1998).

If the samples do not contain missing or hidden values (i.e., full observability) and if the data items are discrete, for certain classes of distributions closed form solutions for the optimal parameter settings may be found (Heckerman 1995). The meaning of optimal depends on the training method. In the most general case, there are three choices: maximum likelihood (ML), maximum a posteriori (MAP), and Bayesian. For ML training, the goal is to find:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\, p(D|\Theta)$$

where Θ^* is the optimal parameter setting and $p(D|\Theta)$ is the probability of the complete data set under the model. Maximum a posteriori estimation assumes the existence of a prior $p(\Theta)$ over the parameters Θ. The goal is to find the optimal posterior of Θ given the data:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\, p(\Theta|D) = \underset{\Theta}{\operatorname{argmax}}\, p(D|\Theta)p(\Theta)$$

Finally, in the Bayesian approach, potentially all values of Θ are considered simultaneously by weighing each one by the posterior of Θ given the data. For example, to compute the probability of some arbitrary variable C given the data and some input feature variable X, the following method would be used:

$$p(C|X, D) = \int p(C|X, \Theta)p(\Theta|D)d\Theta$$

As mentioned above, for certain classes of models, closed form solutions can be obtained for these optimization problems (Heckerman 1995) using “conjugate priors.” For example, a Dirichlet distribution constitutes a conjugate prior for the multinomial distribution. For other densities, closed form solutions may or may not exist. In general, Bayesian network decomposability decreases the parameter optimization complexity since in that case each factor may be optimized independently.

When the data is partially observable (i.e., some elements of the sample are missing) or the network contains intrinsically hidden variables, parameter learning becomes more difficult. In such a case, optimization methods such as the Expectation Maximization (EM) (Dempster *et al.* 1977) algorithm, gradient descent (Bishop 1995; Russel & Norvig 1995), or some other iterative scheme must be used. The EM algorithm is described in Chapter 5.

Bayes decision theory says that to minimize error (risk) one must choose the class with the largest posterior probability $p(c|X)$. Therefore, the ideal parameter optimization procedure should somehow minimize Bayes risk or, alternatively, produce an accurate estimate of the class posterior. The optimization schemes above suggest that one could learn the full joint distribution $p(X, c)$, but this might not be the best procedure for a classification task. For example, consider a ML parameter estimate using a data set D that consists of i.i.d. samples of $p(X, c)$. The i^{th} data sample contains both a feature vector x_i and a corresponding class label c_i, so the data and class-label samples are given by $D = \{D_x, D_c\} = \{(x_1, c_1), (x_2, c_2), \ldots, (x_T, c_T)\}$ and the optimization procedure becomes:

$$\Theta^* = \underset{\Theta}{\text{argmax}} \; \log p(D|\Theta) = \underset{\Theta}{\text{argmax}} \; \log\big(p(D_c|D_x, \Theta)p(D_x|\Theta)\big)$$

To reduce classification error, one need only optimize (maximize) the term $\log p(D_c|D_x, \Theta)$. The other term $\log p(D_x|\Theta)$ might penalize a parameter assignment that would result in an accurate posterior-only approximation. This is especially true when the dimensionality of the feature vector is large (Friedman *et al.* 1997).

Unfortunately, complexity often prevents the estimation of the class posterior probability directly.[1] Instead, one typically optimizes each class-specific likelihood term $p(X|c, \Theta)$ individually. Of course, given a data set D which contains X samples from all of the classes, it is necessary to produce likelihood function estimates $p(X|c, \Theta)$ that provide a high score on samples from the correct class and a low score on samples from other classes. To encourage such behavior, a collection of "discriminative" training methods for likelihood models are often used in the automatic speech recognition community. These techniques either optimize a cost function directly related to the Bayes risk (Juang & Katagiri 1992; Juang *et al.* 1997) or optimize the mutual information between the class variable C and the vector X (Bahl *et al.* 1986; Ephraim & Rabiner 1988; Ephraim *et al.* 1989). Discriminative training will be discussed extensively in Chapter 4.

2.5 Graphical Model Structure Learning

In the Bayesian network literature, the phrase "learning Bayesian networks" often refers both to learning the parameters of a network and learning the structure. The topic

[1]For certain implementations, the class posterior can be estimated fairly easily. For example, a multi-layered perceptron with softmax output non-linearities can be seen as a posterior probability (Bishop 1995) estimator. In this approach, however, it is impossible to marginalize away missing input feature elements as would be possible with a different implementation (Heckerman 1995).

of this section is structure learning. For a given structure, assume that the implementation has been chosen and parameters for each network are chosen optimally according to some procedure (e.g., ML, MAP, etc.) so that benefits arising only from structural differences may be evaluated.

Bayesian network structure learning (Heckerman 1995; Heckerman *et al.* 1994; Buntine 1994; Friedman 1998; Krause 1998) can be crucial to the successful use of such models. If a model is unable to accurately represent important regularities in the data, it will lead to an inaccurate estimation of the probability distribution regardless of the training method and regardless of how well the model is trained.

The most obvious method of structure learning is simple enumeration. In this case, each possible structure is considered, trained, evaluated, and ultimately selected if it results in the best score. Obviously, enumeration is infeasible because the set of possible structures is enormous.

Structure learning is very similar to what statisticians typically call "model selection" (Linhart & Zucchini 1986; Burnham & Anderson 1998). Model selection is a technique where one of a collection of statistical models is selected to best describe some phenomena. For example, in a regression task, one might consider as candidate models a class of polynomials of variable degree. Given a sampled data set D, one might choose the least complex model from that collection which best matches the data according to some specific criterion.

In general, a given training data set can be thought of as containing true information that has been distorted by noise. The goal of model selection is to choose a model that accurately represents the underlying information in the data while ignoring and filtering out any noise. Such a model will be less prone to errors when used for prediction. Often, prior knowledge about a domain can be used to constrain the set of candidate models and essentially "bootstrap" the model selection process (Burnham & Anderson 1998). In many cases, prior knowledge can be crucial to the success of model selection — the alternative is to blindly churn through all available models with the risk of finding one that does not represent the most important properties of the data.

Similar to a parameter estimation task (discussed in Section 5.1), there is a bias-variance trade-off for model selection. The bias-variance trade-off for parameter estimation can be described as follows. It is assumed that there is some true parameter Θ to be estimated from some training data set D. Each data set occurs with a certain probability $p(D|\Theta)$. The parameter estimate produced using that data set is $\Theta^*(D)$. The mean squared error (MSE) is used to evaluate the quality of the parameter estimate. I.e.,

$$\begin{aligned}
\text{MSE} &= \sum_D p(D|\Theta)\Big(\Theta - \Theta^*(D)\Big)^2 \\
&= \sum_D p(D|\Theta)\Big(\Theta - E_{p(D|\Theta)}(\Theta^*(D)) + E_{p(D|\Theta)}(\Theta^*(D)) - \Theta^*(D)\Big)^2 \\
&= \sum_D p(D|\Theta)\Big[\Big(\Theta - E_{p(D|\Theta)}(\Theta^*(D))\Big)^2 + \Big(E_{p(D|\Theta)}(\Theta^*(D)) - \Theta^*(D)\Big)^2 \\
&\quad + \Big(\Theta - E_{p(D|\Theta)}(\Theta^*(D))\Big)\Big(E_{p(D|\Theta)}(\Theta^*(D)) - \Theta^*(D)\Big)\Big] \\
&= \sum_D p(D|\Theta)\Big(\Theta - E_{p(D|\Theta)}(\Theta^*(D))\Big)^2 + \sum_D p(D|\Theta)\Big(E_{p(D|\Theta)}(\Theta^*(D)) - \Theta^*(D)\Big)^2 \\
&= \Big(\Theta - E_{p(D|\Theta)}(\Theta^*(D))\Big)^2 + \sum_D p(D|\Theta)\Big(E_{p(D|\Theta)}(\Theta^*(D)) - \Theta^*(D)\Big)^2 \\
&= (\text{bias})^2 + (\text{variance})
\end{aligned}$$

where

$$E_{p(D|\Theta)}(\Theta^*(D)) = \sum_D \Theta^*(D)p(D|\Theta)$$

Note that in going from the third to the forth step above, the cross-terms sum to zero. The bias reflects the degree to which the typical (or average) estimate is different from the true estimate. The variance reflects the inherent sensitivity of the estimate to variability from different training sets.

Note that Θ need not represent a parameter. It could also represent a true model and $\Theta^*(D)$ could signify some structural estimate that has been selected using a finite sized training set from some model family. In such a case, a similar bias-variance trade-off will exist. If a model family is too simple, the model estimate will have a high bias (because predictions made using the estimate will be inaccurate relative to the true model) but will have a low variance (because the model selection procedure will typically produce the same estimate for different data sets). It the model family is too complex, it will have a low bias (at data-points contained within the training data, the estimate will match the true model), but will have large variance (the estimate will be very sensitive to noise contained in the training data). Like any bias-variance trade-off, it is important to select a class of models which results in a good balance. Sometimes an overly simple model class is used, accepting a high bias, just to explain the data in a simple way. Another desirable property that applies both to a parameter and a structural estimate is the notion of consistency (if a rich enough model class and large enough training set is used, will the resulting estimate ultimately converge to the "right one").

Another aspect of model selection is the notion of "model selection uncertainty" (Burnham & Anderson 1998): If the same data is used both to select the model and then train the resulting model parameters, the resulting estimate of the model's variance must take into account both the variance due to model selection and the variance due to parameter estimation. Otherwise, it might be concluded that the variance is lower than it actually is.

Bayesian network structure learning algorithms may thus be considered model selection procedures. In general, however, learning the structure of a Bayesian network is NP-complete (Chickering 1996). In many cases, domain restriction or approximations can be made resulting in good network structures.

One of the earliest developed structure learning procedures was Chow-tree algorithm (Chow & Liu 1968). For a collection of variables, a tree-dependent distribution is one whose directed edges in a Bayesian network form a tree. Chow presented a method where the best tree-dependent distribution over a set of random variables may be obtained in a KL-distance sense. Chow proved that the best tree-dependent approximation is one obtained using a maximal spanning tree algorithm, where edge weights are determined using pair-wise mutual information between the corresponding random variables. Much more recently, in (Meila 1999) it is shown how to produce a mixture of tree-dependent distributions.

In (Friedman 1998), the EM algorithm was extended to be operable over network structures. Essentially, the EM algorithm's auxiliary function (described in Section 5.2) acquires arguments representing both the parameters and the structure at each EM iteration, i.e., $Q(M, \Theta; M^i, \Theta^i)$ — structure and then parameters are maximized alternatively. The resulting optimization, however, can often be infeasible. A particular difficulty with this approach is that the gradient with respect to a network is not defined so one must resort to a search method, evaluating each candidate along the way.

Several approaches that augment the naive Bayes classifier with intra-feature dependencies have also been proposed. The first is an approximate algorithm (Sahami 1996) that uses mutual information and conditional mutual information between features to choose a good set of intra-feature edges. Similar to the Chow-tree algorithm, Friedman (Friedman *et al.* 1997) also presented a method that uses conditional mutual information to produce the best tree-dependent approximation over the feature variables. It is shown that this optimizes the joint probability of the feature and class variables. Friedman also discusses a method to directly optimize the posterior of the class variable given the feature variables but it was not tested because of computational difficulties.

The most general form of structure learning is the Bayesian approach.[2] In this case, rather than choosing one fixed structure, one uses priors over structures and each structure is used weighted by its prior in a Bayesian setting. To compute the probability of some variable C given the data and input features X, a Bayesian approach over both structures and parameters would be the following:

$$p(C|X, D) = \sum_M p(M|D) \int p(C|X, \Theta_M, M) p(\Theta_M|D, M) d\Theta_M$$

where M is a particular model and Θ_M is a set of parameters for model M. Heckerman (1995) provides a nice tutorial on learning both the parameters and the structure of a Bayesian network in a Bayesian framework. The posterior of model M given the data D is given by:

$$p(M|D) = p(M)p(D|M)/p(D)$$

[2]These are called Bayesian approaches for determining Bayesian networks.

In Heckerman's (1995) paper, $p(D|M)$ is the likelihood of the model M under a particular parameterization Θ determined again by Bayesian methods (MAP or ML estimation could also be used). Of course, one must also define priors over model structure $p(M)$.

An advantage of the Bayesian approach is that it avoids the bias-variance issue. A disadvantage is computational cost – "integrating" over all possible models and all possible parameters is infeasible. An alternative approach is called "selective model averaging," where only a small subset of likely candidate models is used. Selective model averaging is also similar to producing a mixture of models, where each "mixture component" corresponds potentially to a completely different structure. This is also similar to Bayesian multinets as described below. Selecting a single model is also an option (as mentioned above) where either MAP or some penalized likelihood such as BIC or MDL (Burnham & Anderson 1998) can be used to score each model. In Heckerman's (1995) paper, methods are discussed that search over the set of network variants to find one with a high score.

2.6 Bayesian Multinets

Consider a four-variable network A, B, C and a hidden variable Q. For some values of Q, the conditional independencies among A, B, and C might be different than for other values of Q. For example, if Q is binary, and $C {\perp\!\!\!\perp} A|B, Q = 0$ but $C \not\!{\perp\!\!\!\perp} A|B, Q = 1$, the joint probability can be written as:

$$\begin{aligned} & p(A, B, C) \\ = \; & \sum_{q=0}^{1} p(A, B, C, Q = q) \\ = \; & \sum_{q=0}^{1} p(A, B, C|Q = q)p(Q = q) \\ = \; & p(C|B, Q = 0)p(B|A, Q = 0)p(Q = 0) + p(C|B, A, Q = 1)p(B|A, Q = 1)p(Q = 1) \end{aligned}$$

So the conditional dependency *structure* depends on the particular value of Q. Such a scenario could be represented by multiple Bayesian networks but more generally this corresponds to a Bayesian multinet (Geiger & Heckerman 1996). Examples of multinets include a mixture of tree-dependent distributions each with different sets of edges, as described in Meila (1999). Also, Friedman *et al.* (1997) adds a different collection of intra-feature dependency edges for each value of the class variable to the naive Bayes classifier

In general, the statistical dependencies in a multinet could be represented by a regular Bayesian network via specific values of the parameters (e.g., certain parameters could be zero). In practice, however, a multinet can result in a substantial savings in memory, computation, and sample-size complexity relative to a Bayesian network. Furthermore, with fewer parameters for a fixed size data set, a lower estimation variance can result. Bayesian multinets, and dynamic versions thereof, will again be discussed in Chapter 4.

2.7 Discussion

In this chapter, the basics of Graphical models and Bayesian networks were described. In subsequent chapters, Bayesian networks will be used to reason about the conditional independence properties of different models. Many useful procedures have been developed prior to the advent of Bayesian networks (e.g., HMMs, Kalman filters (Haykin 1989), factor analysis (Mardia *et al.* 1979), etc.) that have more recently been shown to possess very simple network structures. While Bayesian networks are not strictly necessary for the discussions in this work, it is this author's belief that their intuitive and easy-to-understand depictions of statistical dependencies justify their use as a tool to help select good model structures.

As will be seen, Chapter 4 presents a procedure that can be viewed as Bayesian multinet structure learning using a criterion related to Bayes error. Prior knowledge of the domain, speech recognition, is obtained by starting (or booting) from models known already to perform quite well on this task.

Chapter 3

Hidden Markov Models

Contents

The statistical model most widely used by automatic speech recognition systems is the Hidden Markov model (HMM). HMMs are commonly used in other applications as well including handwriting recognition, gene sequencing, and even rainfall and Old Faithful geyser characterization (MacDonald & Zucchini 1997). In the speech community, HMMs are used to represent the joint probability distribution of a collection of speech feature vectors.

In this chapter, HMMs, their capabilities, and their limitations are explored in detail. The conclusion of this chapter will be that, in general, there is no theoretical limit to the ability of HMMs. Instead, a particular HMM used by a (speech recognition) system might suffer from the conditional independence properties that provide for tractable algorithms. And while, given enough training data, it is possible to improve an HMM by increasing the number of hidden Markov states and capacity of the observation distributions, an alternative and potentially advantageous approach is to produce an inherently more parsimonious model, one that achieves as good or better performance with comparable or reduced complexity. As will be discussed in Chapter 4, one way of doing this is to examine and extend a particular HMM only where it is found to be deficient.

Before discussing HMMs, basic elements from stochastic processes and discrete Markov chains needed for the HMM discussion will be briefly reviewed.

3.1 Stochastic Processes and Discrete-time Markov Chains

A discrete-time stochastic process is a collection $\{X_t : t \in \mathcal{T}\}$ of random variables ordered by the discrete index t. Although $\mathcal{T}$ could be any set, in this work $\mathcal{T}$ will consist of the set of positive integers — t is considered a time index. The notation X_t will also at times be used to refer to a stochastic process where the index set is implicitly assumed. When the variables themselves take on values only from some discrete space, the process will be designated using the notation $\{Q_t : t \in \mathcal{T}\}$ where $Q_t \in \mathcal{Q}$ and where $\mathcal{Q}$ is a finite set that may be put in one-to-one correspondence with the positive integers. The cardinality of $\mathcal{Q}$ will be denoted $|\mathcal{Q}|$. When variables are continuous or when a distinction between continuous and discrete variables is not necessary, the notation X_t will be used.

In a general stochastic process, the distribution of each of the variables X_t could be arbitrary and different for each t. There also could be arbitrary interdependency relationships between different subsets of variables of the process. Certain types of stochastic processes are encountered particularly often because of their analytical or computational simplicity.

One example is independent and identically distributed processes. A definition follows:

Definition 3.1. Independent and Identically Distributed (i.i.d.) *The stochastic process is said to be i.i.d. (Cover & Thomas 1991; Papoulis 1991) if the following condition holds:*

$$p(X_t = x_t, X_{t+1} = x_{t+1}, \dots, X_{t+h} = x_{t+h}) = \prod_{i=0}^{h} p(X = x_{t+i}) \tag{3.1}$$

for all t, for all $h \geq 0$, for all $x_{t:t+h}$, and for some distribution $p(X = x)$ that is independent of the index t.

An i.i.d. process is therefore composed of an ordered collection of independent random variables each one having the same distribution.

One may also characterize a stochastic process by the degree that statistical properties evolve over time. If the statistical properties of a stochastic process do not change over time, the process is said to be stationary.

Definition 3.2. Stationary Stochastic Process *The stochastic process* $\{X_t : t \geq 1\}$ *is said to be (strongly) stationary (Grimmett & Stirzaker 1991) if the two collections of random variables*

$$\{X_{t_1}, X_{t_2}, \ldots, X_{t_n}\}$$

and

$$\{X_{t_1+h}, X_{t_2+h}, \ldots, X_{t_n+h}\}$$

have the same joint probability distributions for all n *and* h.

In the continuous case, stationarity is equivalent to the condition that $F_{X_{t_{1:n}}}(a) = F_{X_{t_{1:n}+h}}(a)$ for all a where $F(\cdot)$ is a cumulative distribution function for the random variables and a is any constant vector of length n. In the discrete case, stationarity is equivalent to the condition that

$$P(Q_{t_1} = q_1, Q_{t_2} = q_2, \ldots, Q_{t_n} = q_n) = P(Q_{t_1+h} = q_1, Q_{t_2+h} = q_2, \ldots, Q_{t_n+h} = q_n)$$

for all $t_1, t_2, \ldots, t_n$, for all $n > 0$, for all $h > 0$, and for all q_i. All i.i.d. processes are stationary.

It is sometimes stated that two random variables are either correlated or uncorrelated. The covariance between two random vectors X and Y is defined as;

$$\text{cov}(X, Y) = E[(X - EX)(Y - EY)'] = E(XY') - E(X)E(Y)'$$

If X and Y are independent, then their covariance is the zero matrix but zero covariance does not imply independence. It is said that X and Y are uncorrelated if $\text{cov}(X, Y) = \vec{0}$ (equivalently, if $E(XY') = E(X)E(Y)'$) where $\vec{0}$ is again the zero matrix.

3.1.1 Markov Chains

The collection of discrete-valued random variables $\{Q_t : t \geq 1\}$ form a n^{th}-order Markov chain (Grimmett & Stirzaker 1991) if

$$\begin{aligned} &P(Q_t = q_t | Q_{t-1} = q_{t-1}, Q_{t-2} = q_{t-2}, \ldots, Q_1 = q_1) \\ &\quad = \quad P(Q_t = q_t | Q_{t-1} = q_{t-1}, Q_{t-2} = q_{t-2}, \ldots, Q_{t-n} = q_{t-n}) \end{aligned}$$

for all $t \geq 1$, and all $q_1, q_2, \ldots, q_t \in \mathcal{Q}$. In other words, conditioned on the previous n states, the current state is independent of earlier states. One often refers to the quantity $p(Q_t = i)$ as the probability that the Markov chain is "in" state i at time t. In general, a Markov chain may have an infinite number of states, but in the current work Markov chains with only a finite number of states are considered.

An n^{th}-order Markov chain may always be converted to a first-order Markov chain (Jelinek 1997) by the following construction:

$$Q'_t = \{Q_t, Q_{t-1}, \ldots, Q_{t-n}\}$$

where Q_t is an n^{th}-order Markov chain. Then Q'_t is a first-order Markov chain because

$$\begin{aligned}
&P(Q'_t = q'_t | Q'_{t-1} = q'_{t-1}, Q'_{t-2} = q'_{t-2}, \ldots, Q'_1 = q'_1) \\
&= P(Q_{t-n:t} = q_{t-n:t} | Q_{1:t} = q_{1:t}) \\
&= P(Q_{t-n:t} = q_{t-n:t} | Q_{t-n-1:t} = q_{t-n-1:t}) \\
&= P(Q'_t = q'_t | Q'_{t-1} = q'_{t-1})
\end{aligned}$$

Therefore, given a large enough state space, a first-order Markov chain may represent any n^{th}-order Markov chain.[1] In this work, therefore, Markov chains will always be first-order.

The statistical evolution of a Markov chain is determined by the state transition probabilities $a_{ij}(t) \stackrel{\Delta}{=} P(Q_t = j | Q_{t-1} = i)$. In general, the transition probabilities can be a function both of the states at successive time steps and of the current time t. In most cases, it is assumed that there is no such dependence on t. Such a time-independent chain is called time-homogeneous or just homogeneous because $a_{ij}(t) = a_{ij}$ regardless of t. For our purposes, it is sufficient to consider only homogeneous Markov chains.

The transition probabilities in a homogeneous Markov chain are determined by a transition matrix A where $(A)_{ij} = a_{ij}$. The rows of A form potentially different probability mass functions over the states of the chain. For this reason, A is also called a stochastic transition matrix (or just a transition matrix).

A state of a Markov chain may be categorized into one of three distinct categories (Grimmett & Stirzaker 1991). A state i is said to be *transient* if, after visiting the state, it is possible the state will never again be visited. That is, state i is transient if

$$p(Q_n = i \text{ for some } n > t | Q_t = i) < 1$$

A state is said to be *null-recurrent* if it is not transient but the expected return time is infinite. Finally, a state is *positive-recurrent* if

$$p(Q_n = i \text{ for some } n > t | X_t = i) = 1$$

and the expected return time to that state is finite. For a Markov chain with a finite cardinality, a state can only be transient or positive-recurrent.

Like any stochastic process, a homogeneous Markov chain might or might not be a stationary stochastic process. In other words, homogeneity and stationarity are orthogonal properties of the Markov chain. The stationarity condition of a homogeneous Markov chain, however, is determined by both the transition matrix and the current probability distribution over the states. If Q_t is a time-homogeneous stationary Markov chain then:

$$\begin{aligned}
&P(Q_{t_1} = q_1, Q_{t_2} = q_2, \ldots, Q_{t_n} = q_n) \\
&= P(Q_{t_1+h} = q_1, Q_{t_2+h} = q_2, \ldots, Q_{t_n+h} = q_n)
\end{aligned}$$

[1] In speech recognition systems, Markov states have meaning and often correspond to sub-word units. In this case, representing an n^{th}-order Markov chain as a first-order Markov chain would require a re-definition of the meaning of each state. This is not required in the general case where the meanings of the hidden states are not specified.

for all t_i, h, n, and q_i. Using the first order Markov property, the above can be written as:

$$\begin{aligned} & P(Q_{t_n} = q_n | Q_{t_{n-1}} = q_{n-1}) P(Q_{t_{n-1}} = q_{n-1} | Q_{t_{n-2}} = q_{n-2}) \ldots \\ & \quad P(Q_{t_2} = q_2 | Q_{t_1} = q_1) P(Q_{t_1} = q_1) \\ = & \quad P(Q_{t_n+h} = q_n | Q_{t_{n-1}+h} = q_{n-1}) P(Q_{t_{n-1}+h} = q_{n-1} | Q_{t_{n-2}+h} = q_{n-2}) \ldots \\ & \quad P(Q_{t_2+h} = q_2 | Q_{t_1+h} = q_1) P(Q_{t_1+h} = q_1) \end{aligned}$$

Therefore, a Markov chain is stationary only when $P(Q_{t_1} = q) = P(Q_{t_1+h} = q) = P(Q_t = q)$ for all $q \in \mathcal{Q}$. Such a distribution over the states is called a stationary distribution of the Markov chain and is designated by the vector π where $\pi_i = P(Q_t = i)$. According to the definition of the transition matrix, a stationary distribution has the property that $\pi A = \pi$. In other words, π is a left eigenvector of the transition matrix A.

In general, there can be more than one stationary distribution for a given Markov chain. The stationarity of the chain, however, is completely determined by whether or not the chain "admits" a stationary distribution, and if it does, whether the current marginal distribution over the states is one of the stationary distributions. If a chain does admit a stationary distribution π, then $\pi_j = 0$ for all j that are transient and null-recurrent (Grimmett & Stirzaker 1991); i.e., a stationary distribution has positive probability only for positive-recurrent states.

One way of determining if a Markov chain admits a stationary distribution is to use "probability flow." If π is a stationary distribution, then $\pi A = \pi$. This implies that for all i

$$\pi_i = \sum_j \pi_j a_{ji}$$

or equivalently,

$$\pi_i(1 - a_{ii}) = \sum_{j \neq i} \pi_j a_{ji}$$

which is the same as

$$\sum_{j \neq i} \pi_i a_{ij} = \sum_{j \neq i} \pi_j a_{ji}$$

The left side of this equation can be interpreted as the probability flow out of state i and the right side can be interpreted as the flow into state i. A stationary distribution requires that the inflow and outflow in all states cancel each other out. This is described in Figure 3.1.

3.2 Hidden Markov Models

As discussed in Section 2.3, a hidden Markov model is collection of random variables, the "hidden" variables $Q_{1:T}$ and the "observed" variables $X_{1:T}$, along with a set of accompanying conditional independence properties. The hidden variables $Q_{1:T}$ form a discrete-time, discrete-valued, first-order Markov chain — each Q_t may take on one of a set of finite values $Q_t \in \mathcal{Q}$ where $\mathcal{Q}$ is called the state space. The number of states in

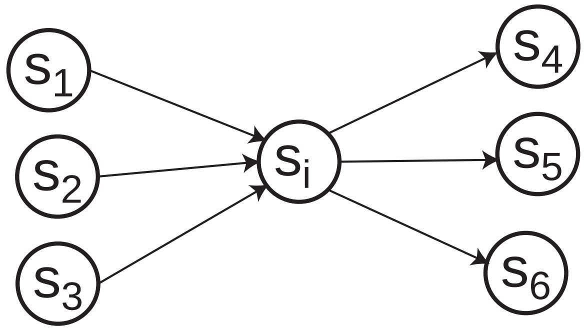

Figure 3.1: Probability flow view of a stationary Markov chain. For the chain to be stationary, the flow into state s_i (equal to $\pi_1 a_{1i} + \pi_2 a_{2i} + \pi_3 a_{3i}$) must equal the flow out of state s_i (equal to $\pi_i a_{i4} + \pi_i a_{i5} + \pi_i a_{i6}$).

this space is indicated by $|\mathcal{Q}|$. The observed variables $X_{1:T}$ form a discrete time stochastic process and can be either discrete or continuous valued. Each observed variable may also be either scalar or vector valued. In this chapter, the observed variables will be considered continuous and vector-valued except where noted. There are two conditional independence assumptions associated with an HMM. The first one states that the hidden variables form a first-order Markov chain:

$$Q_t \perp\!\!\!\perp \{Q_{1:t-2}, X_{1:t-1}\} | Q_{t-1}$$

In other words, given the value of Q_{t-1}, the distribution of Q_t does not depend on any of Q_t's non-descendants (See Section 2.2). The second conditional independence assumption states that:

$$X_t \perp\!\!\!\perp \{Q_{1:t-1}, Q_{t+1:T}, X_{1:t-1}, X_{t+1:T}\} | Q_t$$

which means that given the assignment to Q_t, the distribution of X_t is independent of *all* other variables in the HMM.

Formally, the definition of an HMM will be taken as follows:

Definition 3.3. Hidden Markov Model *A hidden Markov model (HMM) is collection of random variables, the "hidden" variables $Q_{1:T}$ which are discrete and the "observed" variables $X_{1:T}$ which may be either discrete or continuous, that possess the following conditional independence properties.*

$$Q_t \perp\!\!\!\perp \{Q_{1:t-2}, X_{1:t-1}\} | Q_{t-1}$$

$$X_t \perp\!\!\!\perp \{Q_{1:t-1}, Q_{t+1:T}, X_{1:t-1}, X_{t+1:T}\} | Q_t$$

This definition does not specify the number of states in the hidden Markov chain, does not mention if the observation variables are discrete or continuous, does not designate the implementation of the dependencies (e.g., general regression, probability table, neural network, etc.), does not fix the model families used for each of the variables, and does not determine the parameterization or any tying mechanism.

Under an HMM, one observes only the collection of values $X_{1:T} = x_{1:t}$ which have presumably been produced by some unknown assignment to the hidden variables. A given observation assignment with unknown hidden variable values could, with different probabilities, have been produced from one of many different assignments to the hidden variables. To compute the probability of the event $X_{1:t} = x_{1:t}$, one must marginalize (or integrate) away the hidden variables as in the following:

$$\begin{aligned}
p(x_{1:T}) &= \sum_{q_{1:T}} p(x_{1:T}, q_{1:T}) \\
&= \sum_{q_{1:T}} p(x_T|x_{1:T-1}, q_{1:T}) p(x_{1:T-1}, q_{1:T}) \\
&= \sum_{q_{1:T}} p(x_T|q_T) p(x_{1:T-1}, q_{1:T}) \\
&= \sum_{q_{1:T}} p(x_T|q_T) p(x_{T-1}|x_{1:T-2}, q_{1:T}) p(x_{1:T-2}, q_{1:T}) \\
&= \sum_{q_{1:T}} p(x_T|q_T) p(x_{T-1}|q_{T-1}) p(x_{1:T-2}, q_{1:T}) \\
&\quad \ldots \\
&= \sum_{q_{1:T}} \prod_t p(x_t|q_t) p(q_{1:T}) \\
&= \sum_{q_{1:T}} \prod_t p(x_t|q_t) p(q_t|q_{t-1})
\end{aligned}$$

where it is assumed that $p(q_1|q_0) = \pi$ which is some (not necessarily stationary) initial distribution over the hidden Markov chain at the starting time.

The parameters of an HMM can be characterized as follows. First, there is the initial state distribution π which is a vector of length $|\mathcal{Q}|$ and where π_i, the i^{th} element of π, is such that $p(Q_1 = i) = \pi_i$. Second, there are a collection of observation probability distributions $b_j(x) = p(X_t = x|Q_t = j)$ and the associated parameters which depend on the family of probability distribution functions used for $b_j(x)$. Finally, there are the transition probabilities represented by the homogeneous stochastic matrix A where $(A)_{ij} = p(Q_t = j|Q_{t-1} = i)$ for all t.

There are three ways that one can graphically depict an HMM. In the first view, an HMM is seen as a graph with directed edges as shown in Figure 3.2. Each node in the graph corresponds to one of the states in $\mathcal{Q}$, and an edge going from node i to node j indicates that $a_{ij} > 0$. The lack of such an edge indicates that $a_{ij} = 0$. The transition

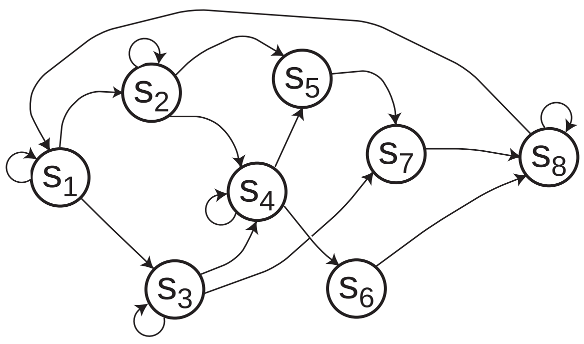

Figure 3.2: Stochastic finite-state automaton view of an HMM. In this case, only the possible (i.e., non-zero probability) hidden Markov chain state transitions are shown.

matrix associated with Figure 3.2 is as follows:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{22} & 0 & a_{24} & a_{25} & 0 & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 & 0 & a_{37} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & a_{46} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{57} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{68} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{78} \\ a_{81} & 0 & 0 & 0 & 0 & 0 & 0 & a_{88} \end{pmatrix}$$

where it is assumed that all explicitly mentioned a_{ij} are non-zero. In this view, an HMM can be seen as a stochastic finite state automaton (FSA). Using such a graph, one can envisage being in a particular state j at a certain time, producing an observation sample from the observation distribution corresponding to that state $b_j(x)$, and then advancing to the next state according to the allowable non-zero transitions. This view of an HMM does not depict either the output distributions or the HMM conditional independence properties. Only the topology, in the form of the non-zero entries of A, of the underlying Markov chain is depicted.

The second way one may view an HMM (shown in Figure 3.3) shows the collection of states at different time steps and the set of possible transitions from states at one time step to states at the next time step. Again, this view depicts only the transition structure of the underlying Markov chain. In this case, however, the transitions that are possible at different slices of time are shown explicitly as the chain evolves. Unlike Figure 3.2, the transition structure of a non-homogeneous Markov chain could be displayed by having different transition edges at each time step.

The third view of an HMM is displayed in Figure 2.7 and is given again in Figure 3.4. This illustration is also a graph with directed edges but it shows the Bayesian network view of an HMM. In this case, the Markov-chain topology is not specified — only the HMM conditional independence properties are shown. Since an HMM is characterized by a set of random variables with conditional independence properties, this third view of an HMM is preferable when discussing the statistical dependencies (or lack thereof) directly

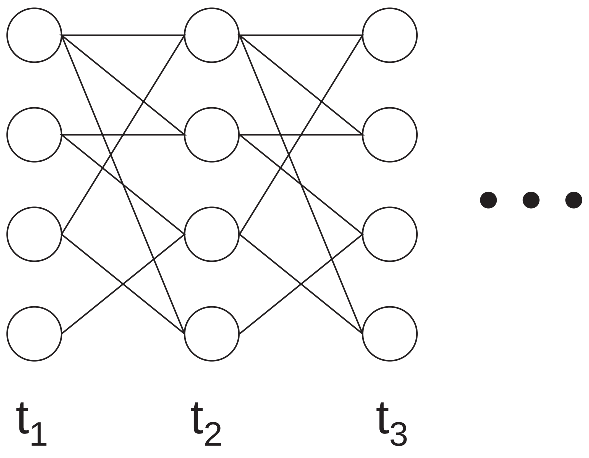

Figure 3.3: Time-slice view of a Hidden Markov Model's state transitions.

represented by an HMM. The stochastic FSA view in Figure 3.2 is useful primarily to analyze the underlying hidden Markov chain topology. At the very least, it should be clear that Figure 3.2 and Figure 3.4 display completely different properties of an HMM.

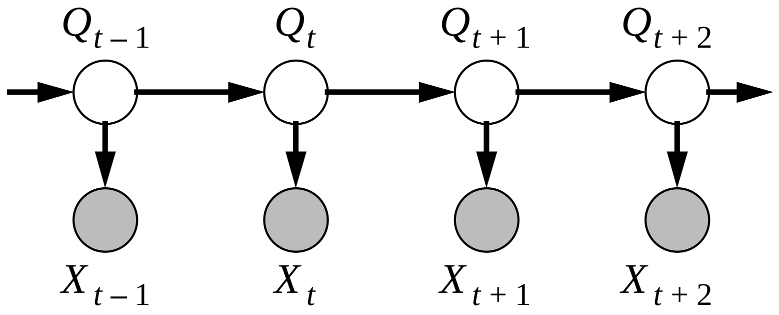

Figure 3.4: A Hidden Markov Model

Samples from an HMM are obtained in the following way:

$$\begin{aligned} q_t &= i \text{ with prob. } p(Q_t = i | q_{t-1}) \\ x_t &\sim b_{q_t}(x) \end{aligned}$$

The first line uses the fact that the hidden variable sample at time t is obtained from a distribution that, given the state assignment at time $t-1$, is conditionally independent of the previous hidden and observation variables. The second line uses the fact that only the hidden state assignment at time t is used to determine the observation distribution at that time.

It can be seen that to sample from an HMM, one may first obtain a complete sample from the hidden Markov chain (i.e., sample from all the random variables $Q_{1:T}$), and then at each time point t produce an output observation sample using the observation distribution corresponding to the hidden state at time t.

One important fact to realize about an HMM is that each new sample of observations requires a completely new sample from the hidden Markov chain. In other words, two different observation samples from an HMM will typically comes from two different underlying state assignments to the hidden chain. Put yet another way, a sample from an HMM is obtained using the marginal distribution $p(X_{1:T}) = \sum_{q_{1:T}} p(X_{1:T}, q_{1:T})$ and not from the conditional distribution $p(X_{1:T}|q_{1:t})$ for some fixed set of hidden variable assignments $q_{1:T}$. As will be seen, this marginal distribution $p(X_{1:T})$ can be quite flexible.

There are many different choices of state-conditioned observation distribution that may be used (MacDonald & Zucchini 1997; Rabiner & Juang 1993). When the observation vectors are discrete, the observation probability distributions $b_j(x)$ form a probability mass function. When the observations are continuous, the observation distributions are often specified using some parametric model family. The most commonly used family for speech recognition are Gaussian mixtures where

$$b_j(x) = \sum_{k=1}^{N_j} c_{jk}\mathcal{N}(x|\mu_{jk}, \Sigma_{jk})$$

and where $\mathcal{N}(x|\mu_{jk}, \Sigma_{jk})$ is a Gaussian distribution with mean vector μ_{jk} and covariance matrix Σ_{jk}. The values c_{jk} are mixing coefficients for hidden state j with $c_{jk} \geq 0$ and $\sum_k c_{jk} = 1$. With such a distribution, an HMM is often referred to as a Gaussian Mixture HMM (GMHMM). A Bayesian network can once again be used to describe HMMs that use mixtures of some component distribution to model the observations (Figure 3.5). Other choices for observation distributions including discrete probability tables (Rabiner & Juang 1993), neural networks (i.e., hybrid systems) (Bourlard & Morgan 1994), auto-regressive distributions (Poritz 1982; Poritz 1988) or mixtures thereof (Juang & Rabiner 1985), the standard set of named distributions (MacDonald & Zucchini 1997), etc.

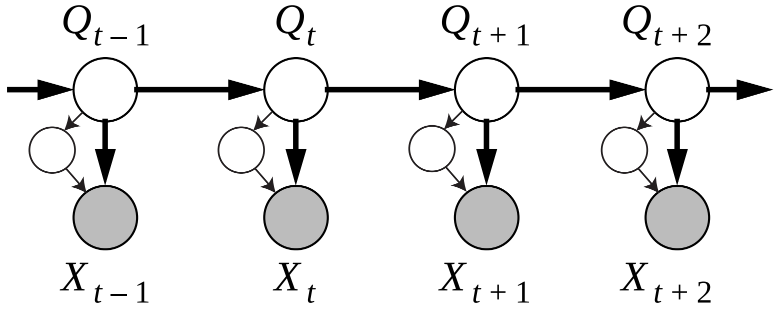

Figure 3.5: A Mixture-Observation Hidden Markov Model

3.3 What HMMs Can Do

For reasons resembling the HMM conditional independence properties, it is sometimes said that HMMs are poor at representing speech signals. The HMM conditional independence properties are sometimes portrayed rather imprecisely, however, and improvements to HMMs are occasionally proposed to correct problems stated as such.

In order to develop a clearer understanding of HMMs, this section compiles a list of HMM deficiencies, as they are sometimes portrayed, and analyzes them in detail. It will be concluded that in general, HMMs are more capable than is sometimes intimated. Furthermore, some HMM extensions might not best correct the problems that ultimately cause errors in a pattern classification or speech recognition system. Chapter 4 will then focus on measuring deficiencies of particular HMMs. That chapter will present a method to automatically derive new potentially more parsimonious models in an attempt to correct only the measured deficiencies.

HMMs have been criticized as a framework to represent speech for the following reasons:

- 3.3.1 they assume the observation variables are i.i.d.
- 3.3.2 they assume the observation variables are i.i.d. conditioned on the state sequence or are "locally" i.i.d.
- 3.3.3 they assume the observation variables are i.i.d. under the most likely hidden variable assignment (i.e., the Viterbi path)
- 3.3.4 they assume the observation variables are uncorrelated over time
- 3.3.5 HMMs do not capture acoustic context
- 3.3.6 HMMs correspond to segmented or piece-wise stationary distributions (this is sometimes called the "beads on a string" problem)
- 3.3.7 when using an HMM, speech is represented as a sequence of feature vectors, or "frames", within which the speech signal is assumed to be stationary
- 3.3.8 when sampling from an HMM, the time distribution of the duration when a particular observation distribution is active corresponds to a geometric probability distribution
- 3.3.9 the first-order hidden Markov assumption is not as good as an n^{th} order chain
- 3.3.10 an HMM represents only $p(X|M)$ (a synthesis model) but to minimize Bayes error, one must represent $p(M|X)$ (a production model)

For reasons that will be enumerated in the following sections, these statements do not point to inherent problems with HMMs in the general case.

3.3.1 i.i.d. observations

Given Definition 3.1 of an i.i.d. process (Section 3.1), it can easily be shown that an HMM is not in general an i.i.d. stochastic process. Under an HMM, the joint probability over the feature vectors $X_{t:t+h}$ is represented as follows:

$$p(X_{t:t+h} = x_{t:t+h}) = \sum_{q_{t:t+h}} \prod_{j=t}^{t+h} p(X_j = x_j | Q_j = q_j) a_{q_j q_{j-1}}.$$

Unless only one state in the hidden Markov chain has non-zero probability for all times in the segment $t : t + h$, this quantity can not in general be represented as the product form $\prod_{j=t}^{t+h} f(x_j)$ for some time-independent distribution $f(\cdot)$ as would be required for an i.i.d. process.

3.3.2 conditionally i.i.d. observations

It has been said that HMMs are i.i.d. conditioned on the state sequence.[2] This is because

$$p(X_{t:t+h} = x_{t:t+h}|Q_{t:t+h} = q_{t:t+h}) = \prod_{\tau=t}^{t+h} p(X_\tau = x_\tau|Q_\tau = q_\tau).$$

and if for $t \leq \tau \leq t + h$, $q_\tau = j$ for some fixed j then

$$p(X_{t:t+h} = x_{t:t+h}|Q_{t:t+h} = q_{t:t+h}) = \prod_{\tau=t}^{t+h} b_j(x_\tau)$$

which is an i.i.d. sequence for this particular state assignment over this time segment $t : t+h$.

While this is true, it was previously noted that each sample from an HMM requires a potentially different assignment to the hidden Markov chain. Unless one and only one state assignment during the segment $t : t + h$ has non-zero probability, the hidden state sequence will change for each HMM sample and there will be no i.i.d. property. To say that an HMM is i.i.d. conditioned on a state sequence is not a property that applies to HMMs as they are actually used. An HMM is used to model the joint distribution of feature vectors $p(X_{1:T})$ which is obtained by marginalizing away (summing over) the hidden variables. HMMs are not in general used to model the joint distribution of feature vectors $p(X_{1:T}|Q_{1:T})$ conditioned on one and only one particular state sequence.

3.3.3 Viterbi i.i.d.

The Viterbi (maximum likelihood) path (Rabiner & Juang 1993; Huang *et al.* 1990) of an HMM is defined as follows:

$$q^*_{1:T} = \underset{q_{1:T}}{\operatorname{argmax}}\, p(X_{1:T} = x_{1:T}, q_{1:T})$$

where $p(X_{1:T} = x_{1:T}, q_{1:T})$ is the joint probability of an observation sequence $x_{1:T}$ and hidden state assignment $q_{1:T}$ for an HMM.

When using an HMM, it is often the case that the joint probability distribution of features is taken according to the Viterbi path:

$$\begin{aligned} & p_{\text{vit}}(X_{1:T} = x_{1:T}) \\ &= cp(X_{1:T} = x_{1:T}, Q_{1:T} = q^*_{1:T}) \\ &= c \max_{q_{1:T}} p(X_{1:T} = x_{1:T}, Q_{1:T} = q_{1:T}) \\ &= c \max_{q_{1:T}} \prod_{t=1}^{T} p(X_t = x_t|Q_t = q_t) p(Q_t = q_t|Q_{t-1} = q_{t-1}) \end{aligned} \tag{3.2}$$

[2]or that they are locally i.i.d., if "locally" means conditioned on the state sequence.

where c is some normalizing constant. This can be different than the complete probability distribution:

$$p(X_{1:T} = x_{1:T}) = \sum_{q_{1:T}} p(X_{1:T} = x_{1:T}, Q_{1:T} = q_{1:T}).$$

Even under a Viterbi approximation, however, the resulting distribution is not necessarily i.i.d. unless the Viterbi paths for all observation assignments are identical. But because the Viterbi path will be different for each observation sequence, and the max operator does not in general commute with the product operator in Equation 3.2, the product representation required for an i.i.d. process is not in general attainable.

3.3.4 uncorrelated observations

Two observations at different times might not be independent, but are they correlated? If X_t and X_{t+h} are uncorrelated, then $E[X_t X'_{t+h}] = E[X_t]E[X_{t+h}]'$. For simplicity, consider an HMM that has single component Gaussian observation distributions, i.e., $b_j(x) \sim \mathcal{N}(x|\mu_j, \Sigma_j)$ for all states j. Also assume that the hidden Markov chain of the HMM is currently a stationary process with some stationary distribution π. For such an HMM, the covariance can be computed explicitly. In this case, the mean value of each observation is a weighted sum of the Gaussian means:

$$\begin{aligned}
E[X_t] &= \int x p(X_t = x) dx \\
&= \int x \sum_i p(X_t = x | Q_t = i) \pi_i dx \\
&= \sum_i E[X_t | Q_t = i] \pi_i \\
&= \sum_i \mu_i \pi_i
\end{aligned}$$

Similarly,

$$\begin{aligned}
E[X_t X'_{t+h}] &= \int x y' p(X_t = x, X_{t+h} = y) dx dy \\
&= \int x y' \sum_{ij} p(X_t = x, X_{t+h} = y | Q_t = i, Q_{t+h} = j) p(Q_{t+h} = j | Q_t = i) \pi_i dx dy \\
&= \sum_{ij} E[X_t X'_{t+h} | Q_t = i, Q_{t+h} = j] (A^h)_{ij} \pi_i dx dy \\
&= \sum_{ij} E[X_t X'_{t+h} | Q_t = i, Q_{t+h} = j] (A^h)_{ij} \pi_i dx dy
\end{aligned}$$

The above equations used the fact that $p(Q_{t+h} = j | Q_t = i) = (A^h)_{ij}$ by the Chapman-Kolmogorov equations (Grimmett & Stirzaker 1991) where $(A^h)_{ij}$ is the i, j^{th} element of the matrix A raised to the h power. Because of the conditional independence properties, it follows that:

$$E[X_t X'_{t+h} | Q_t = i, Q_{t+h} = j] = E[X_t | Q_t = i] E[X'_{t+h} | Q_{t+h} = j] = \mu_i \mu'_j$$

yielding

$$E[X_t X'_{t+h}] = \sum_{ij} \mu_i \mu'_j (A^h)_{ij} \pi_i$$

The covariance between feature vectors may therefore be expressed as:

$$\text{cov}(X_t, X_{t+h}) = \sum_{ij} \mu_i \mu'_j (A^h)_{ij} \pi_i - \left(\sum_i \mu_i \pi_i \right) \left(\sum_i \mu_i \pi_i \right)'$$

It can be seen that this quantity is not in general the zero matrix and therefore HMMs, even with a simple Gaussian observation distribution and a stationary Markov chain, can represent correlation between feature vectors. Similar results for other observation distributions have been derived in MacDonald & Zucchini (1997).

3.3.5 no acoustic context

It is sometimes argued that HMMs do not represent the dependency between a variable X_t and the surrounding acoustic context. The reason given is that the observation vector X_t is independent of the acoustic context given the corresponding hidden variable Q_t. This issue is a variant of the one described in Section 3.3.4. HMMs can represent information in the acoustic context indirectly via the hidden variable. The hidden variable encodes information about the acoustic context and in general as the number of hidden states increases, so does the amount of information that can be encoded. This point will be further explored in Section 3.3.11 and Chapter 4.

3.3.6 piece-wise or segment-wise stationary

The condition of stationarity for an HMM may be discovered by finding explicit conditions that must hold for an HMM to be a stationary process.

According to Definition 3.2, an HMM is stationary when:

$$p(X_{t_1+h} = x_1, \dots, X_{t_n+h} = x_n) = p(X_{t_1} = x_1, \dots, X_{t_n} = x_n)$$

or equivalently when

$$p(X_{t_{1:n}+h} = x_{1:n}) = p(X_{t_{1:n}} = x_{1:n})$$

for all n, h, $t_{1:n}$, and $x_{1:n}$. The quantity $P(X_{t_{1:n}+h} = x_{1:n})$ can be expanded as follows:

$$\begin{aligned} & p(X_{t_{1:n}+h} = x_{1:n}) \\ & = \sum_{q_{1:n}} p(X_{t_{1:n}+h} = x_{1:n}, Q_{t_{1:n}+h} = q_{1:n}) \end{aligned}$$

$$= \sum_{q_{1:n}} p(Q_{t_1+h} = q_1) p(X_{t_1+h} = x_1 | Q_{t_1+h} = q_1)$$

$$\prod_{i=2}^{n} p(X_{t_i+h} = x_i | Q_{t_i+h} = q_i) p(Q_{t_i+h} = q_i | Q_{t_{i-1}+h} = q_{i-1})$$

$$= \sum_{q_1} p(Q_{t_1+h} = q_1) p(X_{t_1+h} = x_1 | Q_{t_1+h} = q_1)$$

$$\sum_{q_{2:T}} \prod_{i=2}^{n} p(X_{t_i+h} = x_i | Q_{t_i+h} = q_i) p(Q_{t_i+h} = q_i | Q_{t_{i-1}+h} = q_{i-1})$$

$$= \sum_{q_1} p(Q_{t_1+h} = q_1) p(X_{t_1+h} = x_1 | Q_{t_1+h} = q_1)$$

$$\sum_{q_{2:T}} \prod_{i=2}^{n} p(X_{t_i} = x_i | Q_{t_i} = q_i) p(Q_{t_i} = q_i | Q_{t_{i-1}} = q_{i-1})$$

$$= \sum_{q_1} p(Q_{t_1+h} = q_1) p(X_{t_1+h} = x_1 | Q_{t_1+h} = q_1) f(x_{2:n}, q_1)$$

where $f(x_{2:n}, q_1)$ is a function that is independent of the variable h. If the condition

$$p(X_{t_{1:n}+h} = x_{1:n}) = p(X_{t_{1:n}} = x_{1:n})$$

is to be true for all h (i.e., for stationarity to hold), then it is required that $p(Q_{t_1+h} = q_1) = p(Q_{t_1} = q_1)$ for all h. This means that the HMM is stationary only when the underlying hidden Markov chain is under a stationary distribution. An HMM therefore does not necessarily correspond to a stationary stochastic process.

The HMMs used for speech recognition systems commonly have left-to-right Markov chain topologies. This means that the transition matrices are upper triangular where $a_{ij} = 0 \;\; \forall j > i$. The Markov graph of such a topology corresponds to a DAG over all the states where each state (node) can also potentially have a self transition (loop). In such graphs, all states with children (i.e., non-zero exit transition probabilities) will have decreasing occupancy probability over time. This can be seen inductively by first considering the start states, the states without any parents. Such states will have decreasing occupancy probability over time because there are no input transitions to create inflow. These states will correspondingly have decreasing outflow over time. Next, consider any state with a parent that has decreasing outflow. Such a state will therefore have decreasing inflow, decreasing occupancy probability, and also decreasing outflow. Only the final states, the states with only parents and with no children, have the potential to retain their occupancy probability over time. Since any stationary distribution must have zero net probability flow through all states, a stationary distribution for a DAG topology must have zero occupancy probability for any states with children.

Another way to see this is to observe that all such states have a less than unity return probability, and therefore may be classified as transient. Any stationary distribution over those states must have a zero probability (Grimmett & Stirzaker 1991). Therefore, any left-to-right HMM (e.g., the HMMs used in speech recognition systems) is not a stationary stochastic process.

One might agree with the above, but still claim that the HMM is "piece-wise" stationary, saying that for the "piece" of time during which an HMM is in a particular state, the observations are i.i.d. and therefore stationary. It was established above that each sample from an HMM results from a potentially different assignment to the hidden Markov chain. Therefore, what could be called a segment (a sequence of identical state assignments to successive hidden variables) in the hidden chain of one HMM sample will not necessarily be a segment in the chain of a different sample. Therefore, there is no corresponding stationary property unless either 1) all samples from an HMM always result in the same hidden assignment for some segment at a fixed time position, or 2) the hidden chain is stationary over that segment. In the general case, however, an HMM does not produce samples from piece-wise stationary segments.

It should be noted that the notions of stationarity and i.i.d. are properties of the random processes, or equivalently, of the complete ensemble of samples from such a random process. The concepts of stationarity and i.i.d. do not apply to a single sample from an HMM. A perhaps more appropriate characteristic of a single sample from a process is that of "steady state," where the short-time spectrum of a signal is constant over a region of time.

It has been known for some time that the information in a speech signal necessary to convey an intelligent message to the listener is contained in the speech sub-band modulation envelopes (Dudley 1939; Drullman *et al.* 1994b; Drullman *et al.* 1994a) and that the spectral energy in the sub-band modulation envelope signals is temporally band-limited. A very liberal estimate of this upper limit is 50 Hz. This fact is deliberately used in speech coding algorithms which achieve significant compression ratios with little or no loss of intelligibility. One way such compression is achieved is by band-pass filtering the sub-band modulation envelopes. Similarly, any stochastic process that represents the information in a speech signal containing the intelligible message need only possess dynamic non-stationary (or non-steady-state) properties at rates no higher than a certain rate.

The Nyquist sampling theorem states that any band-limited signal may be accurately represented by a discrete-time signal sampled at a sufficiently high rate (at least twice the highest frequency in the signal). A signal that describes the statistics of speech as they evolve over time may therefore be accurately represented by a discrete time signal sampled at a suitably high rate.

It has been argued that HMMs are a poor model of speech because samples from an HMM are piece-wise steady-state, whereas real speech does not contain such steady-state segments. In an HMM, however, the hidden Markov chain controls the temporal evolution of the process's statistical properties. Therefore, any band-limited non-stationary or non-steady-state signal can be represented by a Markov chain having a state change at a fast enough rate on average and having enough states to represent all the variability inherent in the signal. It will be argued below that only a finite number of states are necessary for real-world signals.

3.3.7 within-frame stationary

Speech (or any natural signal) is a continuous time signal. A feature extraction process is used to extract frames of speech at some regular time interval (such as 10 ms) where each frame has some window width (such as 20 ms). An HMM is then used to characterize the distribution over the discrete-time set of frames. It has been argued that HMMs are inherently flawed because, within a particular frame, the speech signal might vary but the information provided by that variation is lost via the framing of speech.

Again because the properties of speech that convey the message are band-limited, if the sample rate (rate of hidden state change) is high enough, and if the window-width of measurement is small enough, such a framing of speech will not result in loss of information about the actual message.

3.3.8 geometric state distributions

In a hidden Markov chain, the random state-occupancy duration is determined by a geometric distribution with parameter a_{ii}. It has therefore been said that HMMs are a poor model because their state durations are inherently geometric and that geometric distributions do not have the capability to precisely model the optimal durational distributions.

HMMs do not necessarily suffer from such a problem. Different states of an HMM may share the same observation probability distributions. If a sequence of n states that each use same observation distribution are strung together in series, and each of the states has self transition probability α, then the resulting distribution will be the sum of n geometric distributions. The distribution of a sum of n independent geometric random variables has a negative binomial distribution (Stirzaker 1994). Unlike a geometric distribution, a negative binomial distribution has a mode located at a point greater than zero. In general, a large collection of HMM states may be combined in a variety of series and parallel connections. This can create a very general class of distributions that can characterize the interval of time over which a particular shared observation distribution is used.

3.3.9 first-order hidden Markov assumption

As was demonstrated in Section 3.1.1 and as described in (Jelinek 1997), any n^{th}-order Markov chain may be transformed into a first-order chain. Therefore, assuming the first-order Markov chain possess enough states, there is no inherent accuracy loss when using a first-order as opposed to an n^{th}-order HMM. [3]

3.3.10 synthesis vs. recognition

It is sometimes said that HMMs are impaired because they represent the distribution of feature vectors for a given model, i.e., the likelihood $p(X|M)$. Accordingly, this can viewed as a synthesis model because sampling from this distribution should produce (or synthesize) a set of features representing the object M (e.g., a synthesized speech utterance). To minimize Bayes error, however, one must instead model the posterior probability $p(M|X)$ which is more like a recognition model; given an instance of X, a sample from $p(M|X)$ will produce a source utterance and the true identity of the source utterance is the goal of a recognition system.

[3] Again, in speech recognition systems, any "meanings" of the hidden states might need to change when moving to a higher-order Markov chain.

There are several reasons why the synthesis/recognition division is not a deficiency. First, by Bayes rule, $p(M|X) = p(X|M)p(M)/p(X)$ so if an HMM accurately represents the likelihood $p(X|M)$ and given accurate priors $P(M)$ then the posterior will also be accurate. Maximum-likelihood training will adjust the parameters of a model so that the resulting distribution best matches the training-data empirical distribution. It is said that maximum-likelihood training is asymptotically optimal, so given enough training data and a rich enough model, an accurate estimate of the posterior will be found just by producing an accurate likelihood $p(X|M)$ and prior $p(M)$.

On the other hand, modeling a distribution such as $p(X|M)$ might be a more ambitious task than is necessary to achieve good classification performance. In a classification task, one of a set of different models M_i is chosen as the target class for a given X. In this case, only the decision boundaries, or sub-spaces in the feature space $\{x : p(x|M_i) = p(x|M_j)\}$ for all $i \neq j$, affect classification performance (Duda & Hart 1973). An attempt to model the entire distribution, including the regions between the decision boundaries, is potentially an attempt to solve a more difficult task that needs more training data than necessary to achieve good performance.

There are three reasons why this still is not necessarily a limitation. First, the degree to which boundary information is represented by an HMM (or any model for that mater) depends on the training method. Discriminative training methods have been developed which adjust the parameters of each model to increase not the individual likelihood of each model but rather increase the posterior probability. Methods such as maximum mutual information (MMI) (Bahl *et al.* 1986; Brown 1987), minimum discrimination information (MDI) (Ephraim *et al.* 1989; Ephraim & Rabiner 1990), and minimum classification error (MCE) (Juang & Katagiri 1992; Juang *et al.* 1997) essentially attempt to optimize $p(M|X) = p(X|M)p(M)/p(X)$ by adjusting the parameters of $p(X|M)$ for all M simultaneously. In this case, the parameters of an HMM are adjusted so that the resulting distributions when sampled will not necessarily lead to accurately synthesized speech. Instead, the goal is that the resulting distributions will be accurate only at the decision boundaries.

Second, when training using a maximum likelihood procedure, the degree to which boundary information is represented by an HMM depends on the relative penalty on the likelihood criterion caused by samples close to the boundary region vs. the penalty on the likelihood criterion owing to samples away from the boundary regions. A modified training procedure, for example, could be used which adjusts the penalty on the likelihood of each sample according to the degree that a sample is confusable with other samples of different classes. The degree of confusability of a sample could be determined by its proximity to a decision region (or by how similar a likelihood score the sample gets from different competing models). Admittedly, this becomes a new training procedure, but the complexity in this case is not much worse than the standard maximum likelihood based training procedure.

Third, the degree to which boundary information is represented also depends on each model's ability to represent the probability distribution at the decision boundaries vs. its ability to represent the distribution between the boundaries. This can be thought of as how inherently discriminative the structure of the model is independent of its parameters. Models with such properties could be termed *structurally discriminative.*

This idea can be motivated using a simple example. Consider the two classes of objects shown in Figure 3.6. Objects of class A consist of an annulus with an extruding

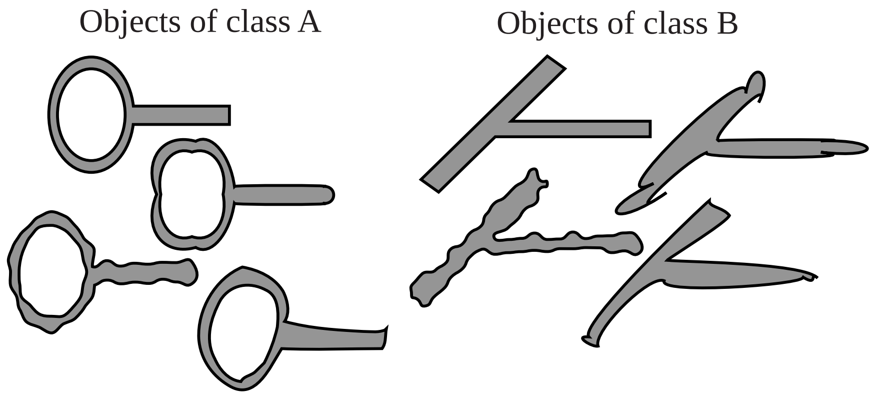

Figure 3.6: Two types of objects that share a common attribute, a horizontal bar on the right of each object. This attribute should not be modeled in a classification task.

horizontal bar on the right. Objects of class B consist of a diagonal bar with an extruding horizontal bar on the right.

Consider a probability distribution family in this space that is only good at representing horizontal bars — the average length, width, smoothness, etc. could for example be parameters that determine a particular distribution. When such a family is used to represent the distribution of objects from each of these classes, the resulting class specific models will not be capable of representing the differences between objects of class A vs. class B. The models will be blind to the differences between the classes regardless of how well the models are trained or even the type of training method used, discriminant or not. These models are structurally indiscriminate.

Consider instead two probability distribution families in this space. The first family accurately represents only annuli of various radii and distortions, and the second family accurately represents only diagonal bars. When each model family is used to represent objects of the corresponding class, the resulting models will easily represent the differences between the two classes. These models are inherently blind to the commonalities between the two classes regardless of the training method used since the resulting models are able to represent only the distinctive features of each class. In other words, even if each model is trained using a maximum likelihood procedure using samples only from its own class, the models will not represent the commonalities between the classes because they are incapable of doing so. The model families are structurally discriminative. Sampling from a model of one class will produce an object containing attributes that only distinguish it from samples of the other class's model. The sample will not necessarily resemble the class of objects its model represents. This, however, is of no consequence to a classification procedure's error rate. This idea, of course, can be generalized to multiple classes each with distinctive attributes.

An HMM is sometimes said to be deficient because it does not synthesize a valid (or even recognizable) spoken utterance. But synthesis is not the goal of a classification task. A valid synthesized speech utterance should correspond to something that could be uttered by an identifiable speaker. When used for speech recognition, HMMs attempt to describe the probability distribution of speech in general, a distribution which corresponds to the average over many different speakers (or at the very least, many different instances of an utterance spoken by the same speaker). Ideally, any idiosyncratic speaker-specific information, which might result in a more accurate synthesis, but not more accurate discrimination, should not be represented by a probabilistic model — such information can only cause a parameter increase without a corresponding classification performance increase. As mentioned above, an HMM should represent the distinctive properties of a particular speech utterance relative to other rival speech utterances. Such a model would not necessarily produce good synthesized speech.

The question then becomes, how structurally discriminative are HMMs when attempting to model the distinctive attributes of speech utterances? In an HMM, a different Markov chain topology is used to model each speech utterance. It could be argued that HMMs are not structurally indiscriminate because, even when trained using a simple maximum likelihood procedure, HMM-based speech recognition systems can perform reasonably well. Sampling from such an HMM might not produce a realistic speech utterance, but the corresponding model distribution might be accurate around the decision boundaries. This topic will arise again in Chapter 4 where a procedure is presented that can automatically produce new more structurally discriminative models.

3.3.11 Necessary Conditions for HMM Accuracy

Suppose that $p(X_{1:T})$ is the "true" underlying distribution of the collection of observation random variables $X_{1:T}$. In this section, it will be shown that if an HMM represents this distribution accurately, necessary conditions on the number of hidden states and the necessary complexity of the observation distributions may be found. Let $p_h(X_{1:T})$ be the joint distribution over the observation variables under an HMM. If the HMM is completely accurate, then the KL-distance between the two distributions will be zero, i.e.:

$$D(p(X_{1:T})||p_h(X_{1:T})) = 0$$

If this condition is true, the mutual information between any subset of variables under each distribution will be equal. That is,

$$I(X_{S_1};X_{S_2}) = I_h(X_{S_1};X_{S_2})$$

where $I(\cdot;\cdot)$ is the mutual information between two random vectors under the true distribution, $I_h(\cdot;\cdot)$ is the mutual information under the HMM, and S_i is any subset of $1:T$.

Consider the two sets of variables X_t, the observation at time t, and $X_{\neg t}$, the collection of observations at all times other than t. The variable X_t may be viewed as the output of a noisy channel that has input $X_{\neg t}$ as shown in Figure 3.7. The information transmission rate between $X_{\neg t}$ and X_t is therefore equal to the mutual information between the two $I(X_{\neg t};X_t)$.

The KL-distance equality condition implies that for an HMM to be an accurate representation of the true distribution $p(X_t|X_{\neg t})$, its corresponding noisy channel representation must have the same transmission rate. Because of the conditional independence properties, an HMM's hidden variable Q_t separates X_t from its context $X_{\neg t}$ and the conditional distribution becomes

$$p_h(X_t|X_{\neg t}) = \sum_q p_h(X_t|Q_t = q)p_h(Q_t = q|X_{\neg t})$$

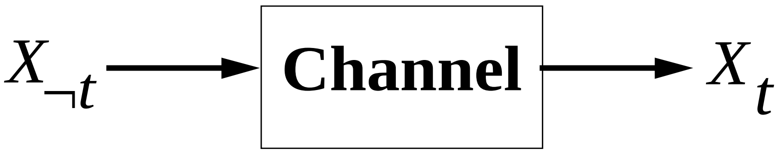

Figure 3.7: The noisy channel view of X_t's dependence on $X_{\neg t}$.

An HMM, therefore, attempts to compress the information about X_t contained in $X_{\neg t}$ into a single discrete variable Q_t. A noisy channel view of an HMM is depicted in Figure 3.8.

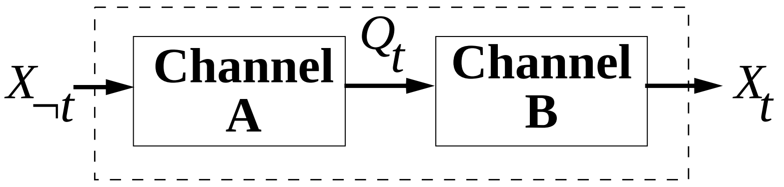

Figure 3.8: A noisy channel view of one of the HMM conditional independence property.

For an accurate HMM representation, the composite channel in Figure 3.8 must have at least the same information transmission rate as that of Figure 3.7. Note that $I_h(X_{\neg t}; Q_t)$ is the information transmission rate between $X_{\neg t}$ and Q_t, and that $I_h(Q_t; X_t)$ is the information transmission rate between Q_t and X_t. The maximum information transmission rate through the HMM composite channel is no greater than to the minimum of $I_h(X_{\neg t}; Q_t)$ and $I_h(Q_t; X_t)$. Intuitively, HMM accuracy requires $I_h(X_{\neg t}; Q_t) \geq I(X_t; X_{\neg t})$ and $I_h(Q_t; X_t) \geq I(X_t; X_{\neg t})$. This is because, if one of these inequalities does not hold for an HMM, then channel A and/or channel B in Figure 3.8 will become a bottle-neck. This would result in restricting the composite channel's transmission rate to be less than the true rate of Figure 3.7. An additional requirement is that the variable Q_t have enough storage capacity (i.e., states) to encode the information typically flowing between the two channels. This last condition should take the form of a lower bound on the number of hidden states. This is formalized by the following theorem.

Theorem 3.1. Necessary conditions for HMM accuracy. *An HMM as defined above (Definition 3.3) with joint observation distribution $p_h(X_{1:T})$ will accurately model the true distribution $p(X_{1:T})$ only if the following three conditions hold:*

- $I_h(X_{\neg t}; Q_t) \geq I(X_t; X_{\neg t})$,
- $I_h(Q_t; X_t) \geq I(X_t; X_{\neg t})$, *and*
- $|\mathcal{Q}| \geq 2^{I(X_t; X_{\neg t})}$

where $I_h(X_{\neg t}; Q_t)$ (resp. $I_h(Q_t; X_t)$) is the information transmission rate between $X_{\neg t}$ and Q_t (resp. Q_t and X_t) under an HMM, and $I(X_t; X_{\neg t})$ is the true information transmission rate between $I(X_t; X_{\neg t})$.

Proof. If an HMM is accurate (i.e., has zero KL-distance from the true distribution), then $I(X_{\neg t}; X_t) = I_h(X_{\neg t}; X_t)$. As for the data-processing inequality (Cover & Thomas 1991), the chain rule of mutual information can be used to expand the quantity $I_h(X_{\neg t}; Q_t, X_t)$ in two ways as follows:

$$\begin{aligned} & I_h(X_{\neg t}; Q_t, X_t) & (3.3)\\ &= I_h(X_{\neg t}; Q_t) + I_h(X_{\neg t}; X_t|Q_t) & (3.4)\\ &= I_h(X_{\neg t}; X_t) + I_h(X_{\neg t}; Q_t|X_t) & (3.5)\\ &= I(X_{\neg t}; X_t) + I_h(X_{\neg t}; Q_t|X_t) & (3.6)\end{aligned}$$

The conditional independence properties of an HMM say that $I_h(X_{\neg t}; X_t|Q_t) = 0$. This implies that

$$I_h(X_{\neg t}; Q_t) = I(X_{\neg t}; X_t) + I_h(X_{\neg t}; Q_t|X_t)$$

or that

$$I_h(X_{\neg t}; Q_t) \geq I(X_{\neg t}; X_t)$$

since $I_h(X_{\neg t}; Q_t|X_t) \geq 0$. This is the first condition. Similarly, the quantity $I_h(X_t; Q_t, X_{\neg t})$ may be expanded as follows:

$$\begin{aligned} & I_h(X_t; Q_t, X_{\neg t}) & (3.7)\\ &= I_h(X_t; Q_t) + I_h(X_t; X_{\neg t}|Q_t) & (3.8)\\ &= I(X_t; X_{\neg t}) + I_h(X_t; Q_t|X_{\neg t}) & (3.9)\end{aligned}$$

By the same reasoning as above, this leads to

$$I_h(X_t; Q_t) \geq I(X_t; X_{\neg t})$$

the second condition. The following sequence of inequalities establishes the third condition:

$$\log|\mathcal{Q}| \geq H(Q_t) \geq H(Q_t) - H(Q_t|X_t) = I_h(Q_t; X_t) \geq I(X_t; X_{\neg t})$$

so $|\mathcal{Q}| \geq 2^{I(X_t; X_{\neg t})}$. □

There are two implications of this theorem. First, it says that an insufficient number of hidden states can lead to an inaccurate model. This has been known for some time in the speech recognition community, but a lower bound on the required number of states has not been established. With an HMM, the information about X_t contained in $X_{<t}$ is squeezed through the hidden state variable Q_t. Depending on the number of hidden states, this can overburden Q_t and result in an inaccurate probabilistic model. But if there are enough states, and if the information in the acoustic context is appropriately encoded in the hidden states, the required information about the surrounding context of an observation may be compressed and represented by Q_t. An appropriate encoding of the contextual information is essential since just adding states does not guarantee accuracy will increase.

To achieve accuracy, it is likely that only a finite number of states is required for any real task since signals representing natural objects will have only finite mutual information. Recall that the first order Markov assumption in the hidden Markov chain is not necessarily a problem since a first-order chain may represent an n^{th} order chain (see Section 3.1.1 and (Jelinek 1997)).

The second implication of this theorem is that each of the two channels in Figure 3.8 must be sufficiently powerful. HMM inaccuracy can result from using a poor observation distribution family which corresponds to using a channel with too small a capacity. The power of an observation distribution is, for example, controlled by the number of Gaussian components in a Gaussian mixture HMM (Young 1996), or the number of hidden units in an HMM with MLP observation distributions (Bourlard & Morgan 1994).

In any event, just increasing the number of components in a Gaussian mixture system or increasing the number of hidden units in an MLP system will not necessarily improve HMM accuracy because the bottle-neck ultimately becomes the fixed number of hidden states (i.e., value of $|\mathcal{Q}|$). Correspondingly, just increasing the number of hidden states in an HMM might not increase accuracy if too poor an observation model is used. Of course, any increase in the number of parameters of a model should be accompanied with a corresponding increase in the amount of training data so that reliable low-variance parameter estimates may be found.

Can sufficient conditions for HMM accuracy be found? Assume for the moment that X_t is a discrete random variable with finite cardinality. Recall that $X_{<t} \triangleq X_{1:t-1}$. Suppose that $H_h(Q_t|X_{<t}) = 0$ for all t (a weak condition on an HMM for this property is that every observation sequence have its own unique Markov chain state assignment). This implies that Q_t is a deterministic function of $X_{<t}$ (i.e., $Q_t = f(X_{<t})$ for some $f(\cdot)$). Consider the HMM approximation:

$$p_h(x_t|x_{<t}) = \sum_{q_t} p_h(x_t|q_t)p_h(q_t|x_{<t}) \tag{3.10}$$

but because $H(Q_t|X_{<t}) = 0$, the approximation becomes

$$p_h(x_t|x_{<t}) = p_h(x_t|q_{x_{<t}})$$

where $q_{x_{<t}} = f(x_{<t})$ since all other terms in the sum in Equation 3.10 are zero. The variable X_t is discrete, so for each value of x_t and for each hidden state assignment $q_{x_{<t}}$, the distribution $p_h(X_t = x_t|q_{x_{<t}})$ can be set as follows:

$$p_h(X_t = x_t|q_{x_{<t}}) = p(X_t = x_t|X_{<t} = x_{<t})$$

This last condition assumes that the number of hidden states might be as big as the cardinality of the entire discrete observation space, i.e., $|X_{1:T}|$ which can be very large. In any event, it follows that for all t:

$$\begin{aligned}
&D(p(X_t|X_{<t})||p_h(X_t|X_{<t})) \\
&= \sum_{x_{1:t}} p(x_{1:t}) \log \frac{p(x_t|x_{<t})}{p_h(x_t|x_{<t})} \\
&= \sum_{x_{1:t}} p(x_{1:t}) \log \frac{p(x_t|x_{<t})}{\sum_{q_t} p_h(x_t|q_t) p_h(q_t|x_{<t})} \\
&= \sum_{x_{1:t}} p(x_{1:t}) \log \frac{p(x_t|x_{<t})}{p_h(x_t|q_{x_{<t}})} \\
&= \sum_{x_{1:t}} p(x_{1:t}) \log \frac{p(x_t|x_{<t})}{p(x_t|x_{<t})} \\
&= 0
\end{aligned}$$

It then follows, using the above equation, that:

$$\begin{aligned}
0 &= \sum_t D(p(X_t|X_{<t})||p_h(X_t|X_{<t})) \\
&= \sum_t \sum_{x_{1:t}} p(x_{1:t}) \log \frac{p(x_t|x_{<t})}{p_h(x_t|x_{<t})} \\
&= \sum_t \sum_{x_{1:T}} p(x_{1:T}) \log \frac{p(x_t|x_{<t})}{p_h(x_t|x_{<t})} \\
&= \sum_{x_{1:T}} p(x_{1:T}) \log \frac{\prod_t p(x_t|x_{<t})}{\prod_t p_h(x_t|x_{<t})} \\
&= \sum_{x_{1:T}} p(x_{1:T}) \log \frac{p(x_{1:T})}{p_h(x_{1:T})} \\
&= D(p(X_{1:T})||p_h(X_{1:T}))
\end{aligned}$$

In other words, the HMM is a perfect representation of the true distribution. This proves the following theorem.

Theorem 3.2. Sufficient conditions for HMM accuracy. *An HMM as defined above (Definition 3.3) with a joint discrete distribution $p_h(X_{1:T})$ will accurately model a true discrete distribution $p(X_{1:T})$ if the following conditions hold for all t:*

- $H(Q_t|X_{<t}) = 0$
- $p_h(X_t = x_t|q_{x_{<t}}) = p(X_t = x_t|X_{<t} = x_{<t})$.

It remains to be seen if simultaneously necessary and sufficient conditions can be derived to achieve HMM accuracy and if it is possible to derive sufficient conditions for continuous observation vector HMMs under some reasonable conditions (e.g., finite power, etc.).

3.4 What HMMs Can't Do

Given the results of the previous section, is there anything that an HMM is unable to do? If under the true probability distribution two random variables possess infinite mutual information, an HMM approximation will fail because an infinite number of states would be required. Infinite mutual information is unlikely, however, from distributions representing objects contained in a natural scene.

The main problems with HMMs lie in how they are used; the conditional independence properties become problematic when there are too few hidden states, or when the observation distributions are too weak. Another potential problem is that a demonstration of the generality of HMMs says nothing about other inherently more parsimonious models which might perform as well or better. This is explored in the next section.

3.5 How to Improve an HMM

One of the conceptually easiest ways to increase an HMM's accuracy is by increasing the number of hidden states and perhaps also the capacity of the observation distributions. This approach seems to be very effective. In speech recognition systems, it is not uncommon to have multiple states per phoneme and to use collections of states corresponding to tri-phones, quad-phones, or even penta-phones. State-of-the-art speech recognition systems have achieved their performance on difficult speech corpora partially increasing the number of hidden states. For example, in the 1999 DARPA Broadcast News Workshop (DARPA Broadcast News Workshop 1999), the best performing systems used penta-phone (a state representing a phoneme in the context of two preceding and two following phonemes) and multiple hidden states for each penta-phone. At the time of this writing, some advanced systems condition on both the preceding and following five phonemes leading to what could be called "unodeca-phones." Given limits of training data size, such systems must use methods to reduce what otherwise would be an enormous number of parameters — this is done by automatically tying parameters of different states together.

How many hidden states are needed? From the previous section, the conditions for HMM accuracy might require a very large number of states. HMM computations are efficient because of small clique sizes (i.e., size two). The cost of probabilistic inference in HMMs grows as $O(TN^2)$ where T is the number of time steps and N is the number of states, so increasing the number of states quadratically increases computational cost.

In general, given enough hidden states and a sufficiently rich class of observation distributions, an HMM can accurately model any real-world probability distribution. HMMs therefore constitute a very powerful class of probabilistic model families. In theory, at least, there is no limit to their ability to model a distribution over signals representing natural scenes.

Any attempt, therefore, to correct problems with HMMs should start by asking the following question: *is there a class of models that inherently leads to a more parsimonious representation (i.e., fewer parameters, lower complexity, or both) of the relevant aspects of speech, and that also provides the same or better speech recognition (or more generally, classification) performance?*

Many alternatives have been proposed for use in speech recognition systems. Some of them are discussed in the following paragraphs.

One HMM alternative very similar to adding more hidden states factors the hidden state representation into multiple independent Markov chains. This type of representation is shown in Figure 3.9. Factored hidden state representations have been called HMM decomposition (Varga & Moore 1990; Varga & Moore 1991), and factorial HMMs (Ghahramani & Jordan 1997). A related method that estimates the parameters of a composite HMM given a collection of separate, independent, and already trained HMMs is called parallel model combination (Gales & Young 1995). A factorial HMM can represent the combination of multiple signals produced independently, the characteristics of each described by a distinct Markov chain. For example, one chain might represent speech and another could represent some dynamic noise source (Kadirkamanathan & Varga 1991). Alternatively, one chain could represent the speech to be recognized and the other chain could represent confounding background speech (Varga & Moore 1991), or the two chains might each represent two underlying concurrent and independent sub-processes governing the realization of the observation vectors (Logan & Moreno 1998). A modified form of factorial HMMs used for speech recognition couples each Markov chain using a cross-chain dependency at each time step (Zweig 1998). In this case, the first chain represents the typical phonetic constituents of speech and the second chain is encouraged to represent articulatory attributes of the speaker (e.g., the voicing condition).

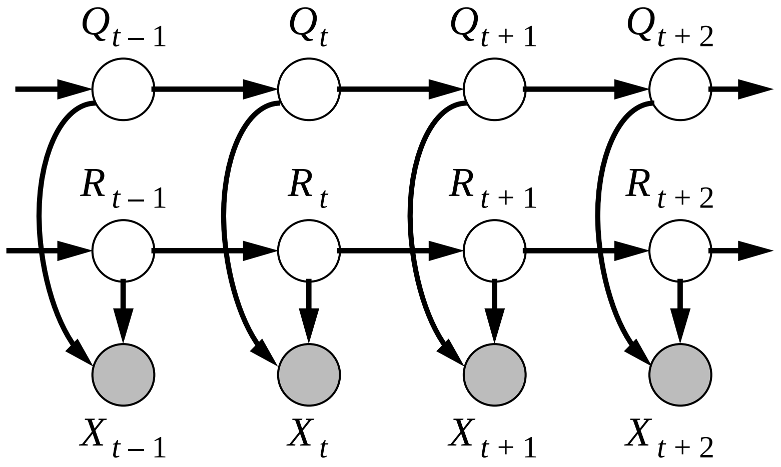

Figure 3.9: A factorial HMM with two underlying Markov chains Q_t and R_t governing the temporal evolution of the statistics of the observation vectors X_t.

The factorial HMMs described above are all special cases of HMMs. That is, they are HMMs with tied parameters and state transition restrictions made according to the factorization. Starting with a factorial HMM consisting of two hidden chains Q_t and R_t, an equivalent HMM may be constructed by using $|\mathcal{Q}||\mathcal{R}|$ states and by restricting the set of state transitions and parameter assignments to be those only allowed by the factorial model. A factorial HMM using M hidden Markov chains each with K states that all span over T time steps will have time complexity $O(TMK^{M+1})$ (Ghahramani & Jordan 1997). If one translates the factorial HMM into an HMM having K^M states, the complexity becomes $O(TK^{2M})$. The underlying complexity of an factorial HMM therefore is significantly smaller than that of an equivalent HMM. An unrestricted HMM with K^M states will, however, have more expressive power than a factorial HMM with M chains each with K states because in the HMM there are no required state transition restrictions and correlation may be represented between the separate chains.

More generally, dynamic Bayesian networks (DBNs) are Bayesian networks consisting of a sequence of Bayesian sub-networks strung together with arrows pointing in the direction of time (or space). Factorial HMMs are an example of DBNs. DBNs were investigated for speech recognition by Zweig (1998). The general class of DBNs that are not reducible to an HMM representation, however, have not been systematically explored for speech recognition. The models introduced in Chapter 4 are also examples of DBNs, as are many of the HMM extensions described in the following paragraphs.

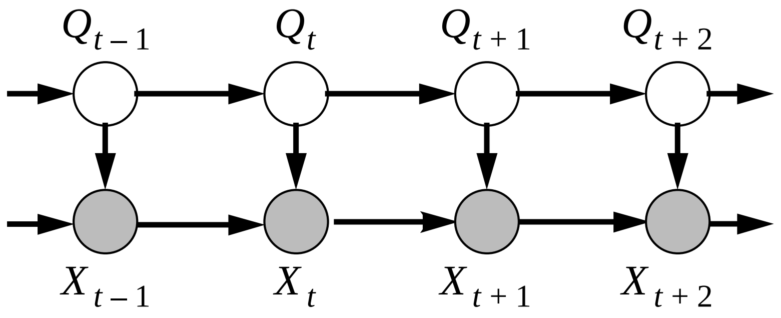

Figure 3.10: An HMM augmented with direct dependencies between neighboring observations.

Another type of HMM extension uses a neural network as a discriminatively trained phonetic posterior probability estimator (Bourlard & Morgan 1994; Morgan & Bourlard 1995). The posterior probabilities $p(q|x)$ are converted to scaled likelihoods via a division by the priors $p(q)$ (the resulting scaled likelihoods are written as $p(x|q)/p(x)$). The scaled likelihoods are then used for the HMM's observation distributions. Typically, a multi-layered perceptron (MLP) (Bishop 1995) is used to produce the posterior probabilities. The size of the hidden-layer of the network controls the capacity of the observation distributions. The input layer of the network typically spans a number of temporal frames both into the past and into the future.

A remark that can be made about an HMM is that additional information might exist about an observation X_t in an adjacent frame (say X_{t-1}) that is not provided by the hidden variable Q_t. One may correspondingly define correlation (Wellekens 1987) or conditionally Gaussian (Ostendorf *et al.* 1996) HMMs, where an additional dependence is added to the HMM Bayesian network graph between adjacent observation vectors. In general, the variable X_t might have as a parent not only the variable Q_t but also the variables X_{t-l} for $l = 1, 2, \ldots, K$ for some K. The case where $K = 1$ is shown in Figure 3.10.

A K^{th}-order Gaussian vector auto-regressive process (Grimmett & Stirzaker 1991) may be represented as:

$$x_t = \sum_{k=1}^{K} A_k x_{t-k} + \epsilon$$

where A_k is a matrix that controls the dependence of x_t on the k^{th} previous observation, and ϵ is a Gaussian random variable with some mean and variance. As described in Section 3.2, a Gaussian mixture HMM may also be described using similar notation. Using this scheme, a general K^{th} order conditionally mixture Gaussian HMM may be described as follows:

$$\begin{aligned} q_t &= i \text{ with prob. } p(Q_t = i | q_{t-1}) \\ x_t &\sim \sum_{k=1}^{K} A_k^{q_t n} x_{t-k} + \mathcal{N}(\mu_{q_t n}, \Sigma_{q_t n}) \text{ with prob. } c_{q_t n} \text{ for } i = \{1, 2, \ldots, N\} \end{aligned}$$

where K is the auto-regression order, A_k^{in} is the regression matrix and c_{in} is the mixture coefficient for state i and mixture n (with $\sum_n c_{in} = 1$ for all i), and N is the number of mixture components per state. In this case, the mean of the variable X_t is determined using previous observations and the mean of the randomly chosen Gaussian component $\mu_{q_t n}$.

Note that although these models are sometimes called vector-valued auto-regressive HMMs, they are not to be confused with auto-regressive, linear predictive, or hidden filter HMMs (Poritz 1982; Poritz 1988; Juang & Rabiner 1985; Rabiner & Juang 1993) which are HMMs that, inspired from the use of linear-predictive coefficients for speech (Rabiner & Juang 1993), use the observation distribution that arises from coloring a random source with a hidden-state conditioned AR filter.

Gaussian vector auto-regressive processes have been considered for speech recognition with $K = 1$ and $N = 1$. The first investigation was by (Wellekens 1987) who only provided EM update equations for maximum-likelihood parameter estimation. Results on a speech task were not presented in that work, although an implementation apparently was tested and it was found not to improve on the case without the additional dependencies (Bourlard 1999). Both Brown (1987) and Kenny *et al.* (1990) tested implementations of such models with mixed success. Namely, improvements were found only when "delta features" (to be described shortly) were not used. Similar results were found by Digalakis *et al.* (1989) but for segment models (also described below). In (Paliwal 1993), the dependency structure in Figure 3.10 was used with discrete rather than with Gaussian observation densities. And in (Noda & Shirazi 1994), a parallel algorithm was presented that can efficiently perform inference with such models.

The use of dynamic or delta features (Elenius & Blomberg 1982; Furui 1981; Furui 1986a; Furui 1986b) has become standard in state-of-the-art speech recognition systems. Dynamic features provide information similar to what is provided by conditionally Gaussian HMMs and are obtained by computing an estimate of the time derivative of each feature $\frac{d}{dt}X_t = \dot{X}_t$ and then augmenting the feature stream with those estimates, i.e., $X_t' = \{X_t, \frac{d}{dt}X_t\}$. Acceleration, or delta-delta, features may similarly be defined and are sometimes found to be additionally beneficial (Wilpon *et al.* 1991; Lee *et al.* 1991).

Most often, estimates of the feature derivative are obtained (Rabiner & Juang

1993) using linear regression, i.e.,

$$\dot{x}_t = \frac{\sum\limits_{k=-K}^{K} k x_t}{\sum\limits_{k=-K}^{K} k^2}$$

where K in this case is the number of points used to fit the regression. Delta (or delta-delta) features are therefore similar to auto-regression, but where the regression is over samples not just from the past but also from the future. That is, consider a process defined by

$$x_t = \sum_{k=-K}^{K} a_k x_{t-k} + \epsilon$$

where the fixed regression coefficients a_k are defined by $a_k = k/\sum_{l=-K}^{K} l^2$ for $k \neq 0$ and $a_0 = 1$. This is equivalent to

$$x_t - \sum_{k=-K}^{K} a_k x_{t-k} = \frac{\sum_{k=-K}^{K} k x_{t-k}}{\sum_{l=-K}^{K} l^2} = \epsilon$$

which is the same as modeling delta features with a single Gaussian component.

The addition of delta features to a feature stream is therefore similar to additionally using a separate conditionally Gaussian observation model. Observing the HMM Bayesian network (Figure 3.4), delta features add dependencies between observation nodes and their neighbors from both the past and the future (the maximum range determined by K). Of course, this would create a directed cycle in a Bayesian network. To be theoretically accurate, one would have to perform a global re-normalization as is done by a Markov random field. Nevertheless, it can be seen that the use of delta features corresponds in some sense to a relaxation of the conditional independence properties of an HMM.

As mentioned above, conditionally Gaussian HMMs often do not provide an improvement when delta features are included in the feature stream. Improvements were reported with the use of delta features in Woodland (1992) where discriminative output distributions (Woodland 1991) were used. In (Levin 1990; Levin 1992), successful results were obtained using delta features but where the conditional mean, rather than being linear, was non-linear and was implemented using a neural network. Also, in (Takahashi *et al.* 1993), benefits were obtained using mixture of discrete distributions. As will be seen in Chapter 6, improvements when using delta features will be reported for models that are similar to the mixture Gaussian auto-regressive case with $N > 1$ and $K > 1$, but where the dependencies are sparse, data-derived, and hidden variable dependent.

In general, one can consider the model

$$\begin{aligned} q_t &= i \text{ with prob. } p(Q_t = i | q_{t-1}) \\ x_t &= F_t(x_{t-1}, x_{t-1}, \ldots, x_{t-k}) \end{aligned}$$

where F_t is an arbitrary random function of the previous k observations. In (Deng *et al.* 1994; Deng & Rathinavelu 1995), a model of the form

$$x_t = \sum_{k=1}^{K} \phi_{q_t,t,k} x_{t-k} + g_{q_t,t} + \epsilon_{q_t}$$

was used where $\phi_{i,t,k}$ is a dependency matrix for state i and time lag k and is a polynomial function of t, $g_{i,t}$ is a fixed mean for state i and time t, and ϵ_i is a state dependent Gaussian. Improvements using this model were also found with feature streams that included delta features.

Another general class of models that extend HMMs are called segment or trajectory models (Ostendorf *et al.* 1996). In a segment model, the underlying hidden Markov chain governs the evolution not of the statistics of individual observation vectors. Instead, the Markov chain governs the evolution of sequences (or segments) of observation vectors where each sequence may be described using an arbitrary distribution. More specifically, a segment model uses the joint distribution of a variable length segment of observations conditioned on the hidden state for that segment. In a segment model, the joint distribution of features is described as follows:

$$\begin{aligned} & p(X_{1:T} = x_{1:T}) \qquad (3.11) \\ = & \sum_{\tau} \sum_{q_{1:\tau}} \sum_{\ell_{1:\tau}} \prod_{i=1}^{\tau} p(x_{t(q_{1:\tau},\ell_{1:\tau},i,1)}, x_{t(q_{1:\tau},\ell_{1:\tau},i,2)}, \ldots, x_{t(q_{1:\tau},\ell_{1:\tau},i,\ell_i)}, \ell_i | q_i, \tau) p(q_i | q_{i-1}, \tau) p(\tau) \end{aligned}$$

There are T time frames and τ segments where the i^{th} segment is of a hypothesized length ℓ_i. The collection of lengths are constrained so that $\sum_{i=1}^{\tau} \ell_i = T$. For a general hypothesized segmentation and collection of lengths, the i^{th} segment starts at time frame $t(q_{1:\tau}, \ell_{1:\tau}, i, 1)$ and ends at time frame $t(q_{1:\tau}, \ell_{1:\tau}, i, \ell_i)$. In this general case, the time variable t could be a function of the complete Markov chain assignment $q_{1:\tau}$, the complete set of currently hypothesized segment lengths $\ell_{1:\tau}$, the segment number i, and the frame position within that segment 1 through ℓ_i. It is assumed that $t(q_{1:\tau}, \ell_{1:\tau}, i, \ell_i) = t(q_{1:\tau}, \ell_{1:\tau}, i+1, 1) - 1$ for all values of all quantities.

Renumbering the time sequence for a hypothesized segment starting at one, the joint distribution over the observations of a segment is given by:

$$p(x_1, x_2, \ldots, x_\ell, \ell | q) = p(x_1, x_2, \ldots, x_\ell | \ell, q) p(\ell | q)$$

where $p(x_1, x_2, \ldots, x_\ell | \ell, q)$ is the joint segment probability for length ℓ and for hidden Markov state q, and where $p(\ell | q)$ is the explicit duration model for state q.

A plain HMM may be represented using this framework if $p(\ell|q)$ is a geometric distribution in ℓ and if

$$p(x_1, x_2, \ldots, x_\ell | \ell, q) = \prod_{j=1}^{\ell} p(x_j | q)$$

for a state specific distribution $p(x|q)$. The stochastic segment model (Ostendorf *et al.* 1992) is a generalization which allows observations in a segment to be additionally dependent on a region within a segment

$$p(x_1, x_2, \ldots, x_\ell | \ell, q) = \prod_{j=1}^{\ell} p(x_j | r_j, q)$$

where r_j is one of a set of fixed regions within the segment. A more general model is called a segmental hidden Markov model (Gales & Young 1993)

$$p(x_1, x_2, \ldots, x_\ell | \ell, q) = \int p(\mu | q) \prod_{j=1}^{\ell} p(x_j | \mu, q) d\mu$$

where μ is the multi-dimensional conditional mean of the segment and where the resulting distribution is obtained by integrating over all possible state-conditioned means in a Bayesian setting. More general still, in trended hidden Markov models (Deng *et al.* 1994; Deng & Rathinavelu 1995), the mean trajectory within a segment is described by a polynomial function over time. Equation 3.11 generalizes many models including the conditional Gaussian methods discussed above. A nice summary of segment models and their learning equations, and a complete bibliography is given by Ostendorf *et al.* (1996).

Markov Processes on Curves (Saul & Rahim 1998) is a recently proposed dynamic model used to represent speech at various speaking rates. Certain measures on continuous trajectories are invariant to some transformations, such as monotonic non-linear time warping. The arc-length, for example, of a trajectory $x(t)$ from time t_1 to time t_2 is given by:

$$\ell = \int_{t_1}^{t_2} \left[\dot{x}(t) g(x(t)) \dot{x}(t) \right]^{1/2} dt$$

where $\dot{x}(t) = \frac{d}{dt} x(t)$ is the time derivative of $x(t)$, and $g(x)$ is an arc-length metric. The entire trajectory $x(t)$ is segmented into a collection of discrete segments. Associated with each segment of the trajectory is a particular state of a hidden Markov chain. The probability of staying in each Markov state is controlled by the arc-length of the observation trajectory. The resulting Markov process on curves is set up by defining a differential equation on $p_i(t)$ which is the probability of being in state i at time t. This equation takes the form:

$$\frac{dp_i}{dt} = -\lambda_i p_i \left[\dot{x}(t) g_i(x(t)) \dot{x}(t) \right]^{1/2} + \sum_{j \neq i} \lambda_j p_j a_{ji} \left[\dot{x}(t) g_j(x(t)) \dot{x}(t) \right]^{1/2}$$

where λ_i is the rate at which the probability of staying in state i declines, a_{ji} is the transition probability of the underlying Markov chain, and $g_j(x)$ is the length metric for state j. From this equation, a maximum likelihood update equations and segmentation procedures can be obtained (Saul & Rahim 1998).

Can changes in speaking rate can be modeled by a non-linear warping in time? Faster speech is often associated with missed articulatory targets, referred to as undershoot (Clark & Yallop 1995), in which case not only the time axis is warped but the trajectory itself can be completely different. Nevertheless, continuous-time inspired models such as these have the potential to yield properties very different than an HMM with the same number of parameters.

3.6 Towards Data-driven HMM Extensions

The approaches described above, which extend HMMs, all potentially provide a more attractive model for a fixed number of parameters. Given the generality of an HMM as described in Section 3.3.11, it is not certain, however, that these models provide additional capability. If they do, it is not clear where that extra ability lies or how useful it might be for the speech recognition, or more generally, the pattern classification task.

Since HMMs already work fairly well for many tasks, one beneficial approach might be to examine a *particular* HMM, discover if and where it is deficient for a particular task (namely classification), and then explicitly correct for any deficiency by adding capability only where a deficiency is found. The resulting corrected model will have all the capability of the original model, will have been customized to perform better on the particular task, but will have been augmented in a minimal way. This would be one attempt to reach the ultimate goal, which is to find a model that performs as well or better than an HMM with the same or fewer parameters and using comparable or less computation.

In the next section, a new method to extend an HMM will be proposed. It will be argued that the deficiency of a particular HMM can be measured using conditional mutual information. If the HMM is accurate, then $I(X_t; X_{<t}|Q_t)$ will be zero. The degree to which $I(X_t; X_{<t}|Q_t) > 0$ or more generally $I(X_t; X_{\neg t}|Q_t) > 0$ can be seen as a measure of a particular HMM's loss. Based on training data, this loss measure, and discriminative variations thereof, will be used to produce new statistical models that attempt to minimize the loss.

Chapter 4

Buried Markov Models

Contents

In the analysis of Chapter 3, it was found that HMMs have no inherent theoretical inadequacy for the task of representing the probability distribution of speech. To improve upon an HMM, it was argued, one should somehow seek an inherently more parsimonious representation that also leads to better speech recognition performance.

Like all objects originating from natural auditory scenes, speech has unique statistical properties that distinguish it from random noise and other sounds. Furthermore, examples of a particular speech utterance have statistical properties that distinguish it from examples of different utterances. As discussed in Chapter 1, it is these patterns of redundancy that distinguish signals from each other and from noise. Furthermore, certain (perhaps statistical) properties of natural objects must be consistent from one object instance to another, for otherwise different instances of a class would not be recognizable as being examples of the same class.

Statistical models that represent only the important statistical properties of objects will perhaps provide accurate representations without either a large computational expense or a large number of parameters. The problem then is to automatically identify the inherent statistical structure of natural objects and produce models which accurately reflect such structure. If the models are probabilistic and are evaluated in terms of their conditional independence properties, such a procedure becomes structure learning as described in Chapter 2. The result will hopefully be models that are both parsimonious and accurate.

HMMs are known to work reasonably well for the speech recognition task. While a theoretical inadequacy can not be found with HMMs, perhaps instead deficiencies of a particular HMM can be identified. If deficiencies can be found, perhaps that particular HMM can be adjusted in ways that specifically address only the problems and do not cause a large increase in either complexity or the number of parameters. One way, therefore, to extend HMMs is to start with a working HMM (the boot HMM), measure how and where it is inaccurate according to some structural loss function, and then augment that HMM's statistical dependencies to reduce the loss.

To this end, a quantitative measure can be obtained of the validity of a particular HMM's conditional independence. In any HMM (see Definition 3.3), the assumption is made that $X_t {\perp\!\!\!\perp} X_{\neg t}|Q_t$ for all t. This property is true if and only if the conditional mutual information $I(X_t; X_{\neg t}|Q_t)$ is zero. The degree to which the conditional mutual information for a particular HMM is greater than zero corresponds directly to the degree to which the conditional independence property is false for that HMM.

If the conditional mutual information is greater than zero, than that particular HMM can not accurately represent the underlying probability distribution in the KL-distance sense because information exists about X_t in the acoustic context of X_t that is not provided by Q_t. This is because, regardless of the actual true value of $I(X_t; X_{\neg t}|Q_t)$, an HMM makes the assumption that $X_t {\perp\!\!\!\perp} X_{\neg t}|Q_t$. Again using the chain rule of mutual information (Cover & Thomas 1991), the following expansion may be obtained:

$$\begin{aligned}
&I(X_{\neg t}; Q_t, X_t) && (4.1)\\
&= I(X_{\neg t}; Q_t) + I(X_{\neg t}; X_t|Q_t) && (4.2)\\
&= I(X_{\neg t}; X_t) + I(X_{\neg t}; Q_t|X_t) && (4.3)
\end{aligned}$$

Therefore, the true amount of information transmitted between $X_{\neg t}$ and X_t can be represented as:

$$I(X_{\neg t}; X_t) = I(X_{\neg t}; X_t|Q_t) + I(X_{\neg t}; Q_t) - I(X_{\neg t}; Q_t|X_t)$$

An HMM makes the *assumption* that $I(X_{\neg t}; X_t|Q_t) = 0$, so at best the amount of information transmitted between $X_{\neg t}$ and X_t with an HMM, represented as $I_h(X_{\neg t}; X_t)$, is:

$$I_h(X_{\neg t}; X_t) = I(X_{\neg t}; Q_t) - I(X_{\neg t}; Q_t|X_t)$$

And therefore

$$I_h(X_{\neg t}; X_t) = I(X_{\neg t}; X_t) - I(X_{\neg t}; X_t|Q_t) \tag{4.4}$$

From this equation, if it is true that $I(X_{\neg t}; X_t|Q_t) = 0$, then it is possible for an HMM to be accurate. If on the other hand $I(X_{\neg t}; X_t|Q_t) > 0$, then $I_h(X_{\neg t}; X_t) \neq I(X_{\neg t}; X_t)$ and the HMM can not be KL-distance accurate. In Equation 4.4, $I(X_{\neg t}; X_t|Q_t)$ can be seen as the amount of information lost when transmitting from $X_{\neg t}$ to X_t using an HMM relative to the true transmission rate. Conditional mutual information therefore can be viewed as a measure of a particular HMM's deficiency or its loss.

In this chapter, conditional mutual information, and discriminative variants thereof, is used to methodologically extend the structure of HMMs to reduce loss. In particular, the observation probability model becomes $p(X|Q, \mathcal{Z})$ where $\mathcal{Z}$ is a set of useful additional dependency variables.

4.1 Conditional Mutual Information In Speech

Given a collection of data vectors $X_{1:T}$ representing samples from some underlying distribution, and given an HMM that has been developed and trained on that data sample, does additional information typically exist about the feature vector X_t in the surrounding context $X_{\neg t}$ given the hidden state variable Q_t? If not, then there is no reason to add additional dependency relationships to an HMM, and the reason for any errors made when using an HMM in a classification task must exist elsewhere. The data should of course represent some natural signal evolving over time or space such as speech, audio, handwriting, video, etc. If the data actually originated from an HMM or some i.i.d. random process, the conditional mutual information would be zero.

Figure 4.1 shows a conditional mutual information density plot in bits per unit area (more precisely, bits per an area of size one quarter octave by 12.5 ms) computed from a two-hour portion of randomly selected utterances obtained from the Switchboard conversational-speech corpus (Godfrey *et al.* 1992).[1] The horizontal axis shows the time-lag in milliseconds, and the vertical axis shows frequency difference in octaves.

The plot shows the mutual information between cube-root-compressed spectral sub-band envelopes. More precisely, the speech signal was first processed by a 22-channel constant-Q sub-band filter bank using FIR filters (designed by windowing the ideal impulse response with a Kaiser window). Each sub-band channel was then full-wave rectified and low-pass filtered (again using a Kaiser FIR filter) using a filter cut-off frequency of about 30 Hz. The sub-band channels were then cube-root compressed to reduce the dynamic range and thereby decrease the chance of numerical overflow problems (note that mutual information is unaffected by monotonic functions applied to the random variables arguments, i.e., $I(X;Y) = I(f(X); g(Y))$ where $f()$ and $g()$ are monotonic (Cover & Thomas 1991; Haykin 1999)). Finally, the signals were down-sampled to recover full bandwidth and significantly reduce the otherwise enormous amount of computation required. Spectral sub-band envelopes were used to evaluate the mutual information of speech for several reasons:

[1]At the most recent LVCSR workshop (LVCSR Workshop 1998), the best performing system achieved only about 28% word error rate on this speech corpus.

1) computational tractability, 2) similarity to feature vectors typically used by automatic speech recognition systems, 3) easy and intuitive to visually display, and 4) the resulting signals do not contain fine speech structure, they contain only modulation envelopes that, since the time of Dudley's Vocoder (Dudley 1939), have been known to contain the information necessary for speech intelligibility (Drullman *et al.* 1994b; Drullman *et al.* 1994a; Drullman 1995). The actual computation of Mutual information is described in Appendix C and in Bilmes (1998).

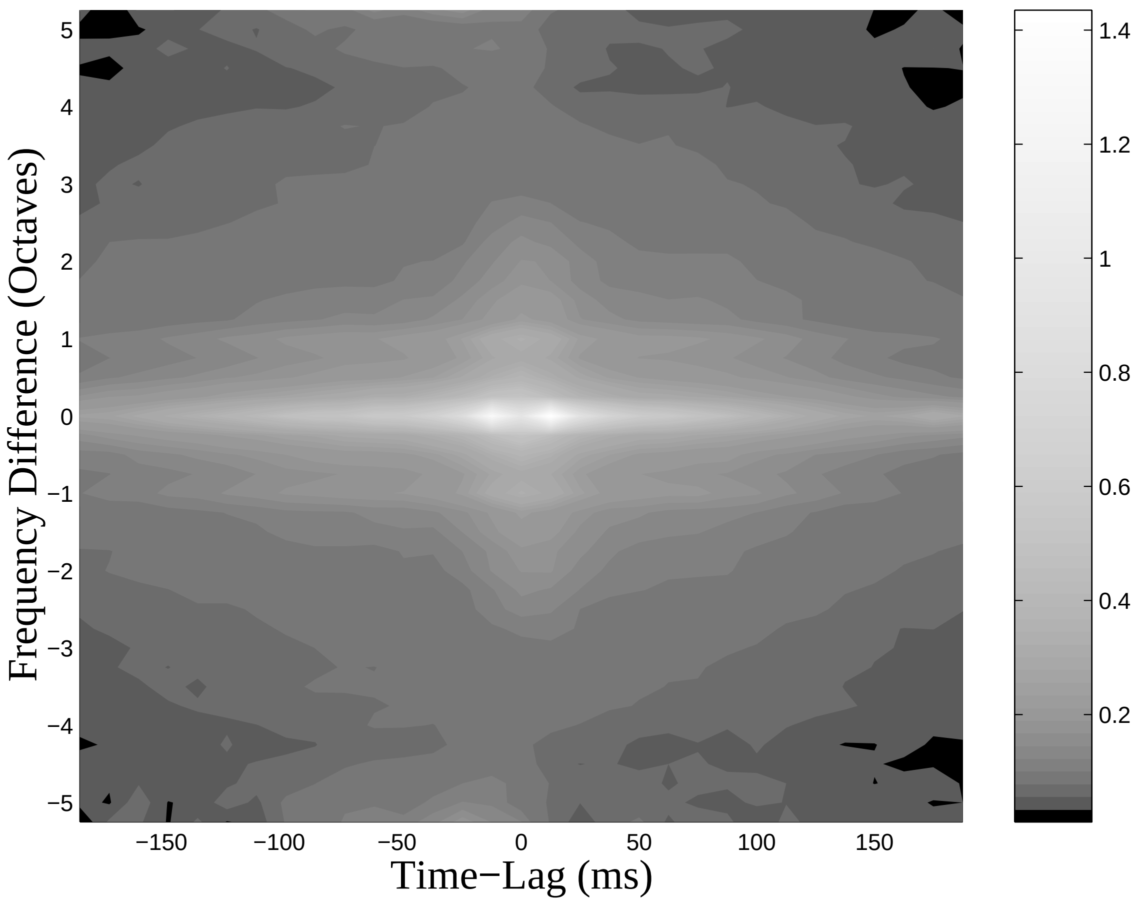

Figure 4.1: The conditional mutual information density of a randomly selected 2-hour section of the Switchboard conversational-speech corpus (in bits per unit area).

In the plot, the hidden variable Q represents decision-tree clustered tri-phones (for these plots, the number of resulting separate clusters and therefore the number of distinct observation distributions is 83). A Viterbi path was used to label each speech frame — the conditional mutual information was estimated using only those appropriately labeled frames for each condition $Q = q$. Also, the plot shows only the conditional mutual information between individual feature elements. That is, let X_{ti} be the i^{th} scalar element of the random feature vector X_t at time t. Then $I(X_{ti}; X_{\tau j}|Q)$ is the mutual information between two variables having a time difference of $\Delta t = t - \tau$ and a frequency difference of $\Delta f = i - j$.[2] The plot shows the average for a constant frequency difference, i.e., $I(\Delta f, \Delta t|Q)$ where:

$$I(\Delta f, \Delta t|Q) = avg_{i-j=\Delta f} I(X_{ti}; X_{t-\Delta t,j}|Q)$$

As can be seen, additional information (or loss) does exist on average throughout the entire acoustic context. A higher luminance in the plot corresponds to a greater amount of information that is lost by the HMM. The information increases significantly as spectro-temporal distance decreases. This plot does not look dissimilar to the spatial co-information plots representing the correlation between two points in a visual scene (Ruderman 1994; Field 1987; Ripley 1981; Baddeley 1997). The plot seems to display a $1/f$-like distribution that is common in many natural occurring signals.

To estimate the integrity of the information displayed in Figure 4.1, the same process (i.e., compressed spectral sub-band envelope processing and mutual information computation) was applied to 1 hour of Gaussian white-noise audio signals. The result is displayed in Figure 4.2. By comparing the two figures, and in particular observing the differing luminance scales, at least two observations can be made. First, the mutual information is on average smaller in the random case. The average value of the information in Figure 4.1 is about 1×10^{-1} bits whereas in the random case it is about 8×10^{-4} bits. This suggests that the mutual information values obtained from the speech signals are significant.[3] Second, there is no apparent structure in the random case other than a small amount of information in the same frequency channel over a small temporal extent. In the speech plot there is a general trend, namely that mutual information decreases with spectro-temporal distance. This suggests that Figure 4.1 is showing actual statistical structure extant in the speech signal itself. Any statistical coloration induced by the feature extraction process, which is seen in the random plot, does not have an appreciable affect on the mutual information shown in the speech plot.

Figure 4.3 shows a similar procedure applied to mel-frequency cepstral coefficients (MFCCs) (Rabiner & Juang 1993) and RASTA-PLP coefficients (Hermansky & Morgan 1994), two common types of features used by speech recognition systems. The MFCC plot shows data obtained from 13 coefficients (c0 through c12) and their deltas. The RASTA-PLP plot shows data from 9 coefficients and their deltas. Both of these coefficient types undergo a final discrete cosine transformation (DCT), a transformation that produces coefficients representing the untransformed vectors with an orthogonal sinusoidal basis. The DCT is sometimes thought to produce uncorrelated features, but this is not true as orthogonality does not imply lack of correlation. Nevertheless, the plots do indicate that the features tend to have relatively small mutual information across feature position, and the temporal extent of information is longer for the same feature than across different features. There is significant mutual information between features and their corresponding deltas

[2]Spectral features were sampled with a 12.5 ms period and span a time range of ± 200 ms, so there are 16 frames in each direction. There are 22 features per frame. The zero-lag frame has a symmetric mutual-information matrix. In total, there are $32 \times 22 \times 22 + 22 * 21/2 = 15,719$ combinations of pairs of features displayed in these plots.

[3]In purely random data (i.e., when the mutual information computation is performed between random vectors not derived from the spectral sub-band envelope process), the same calculation on the 15719 pairs produces mutual information values with quartiles of 0.005, 0.006, and 0.008 and has a maximum value of 0.014 bits which can be used as a significance level with very high confidence.

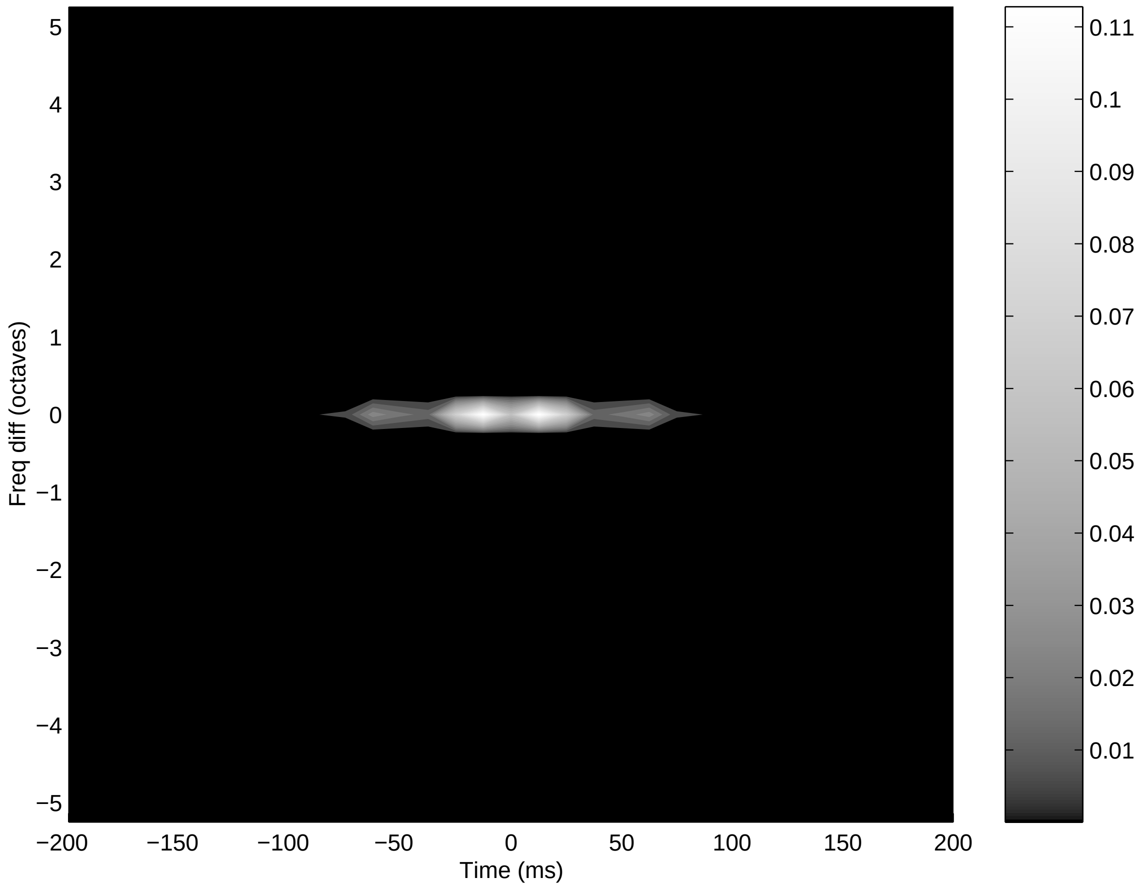

Figure 4.2: The mutual information density of Gaussian white noise audio signals, processed as compressed subband spectral envelopes.

(which suggests that features are informative about their derivatives, and vice versa) as there is for a feature over time.

Note that the mutual information dynamic range for the MFCC features is lower than for the RASTA-PLP or CSSE features. The lower range might indicate that the HMM's hidden variables are better at representing information for the MFCC acoustic environment, at least for this labeling scheme. This could have resulted from the fact that the tri-phone class labels were developed using MFCC coefficients. Alternatively, it could be that MFCCs are particularly well suited to HMMs with Gaussian mixture distributions (a conjecture sometimes made by members of the speech recognition community). The mutual information range, in general, might indicate how well a particular feature set will perform for a particular HMM — the lower the better. Such notions about using the range of conditional mutual information values to measure the match between features and models are not explored further in this thesis.

To obtain an estimate of the significance levels of the MFCC and RASTA-PLP plots, it would be required to perform the MFCC or RASTA-PLP procedure on noise signals.

In other words, the significance level suggested by Figure 4.2 does not apply to the MFCC and RASTA-PLP plots.

All of the speech-based conditional mutual information plots show averages of averages, i.e., they show the prior-weighted average over all conditions (using priors $p(Q = q)$), and for each class the average is taken over all pairs of features with a fixed feature position difference (or frequency difference in the case of Figures 4.1 and 4.2). Therefore, any idiosyncratic and more complex class- and feature- specific patterns are not visible. Since the actual data is four-dimensional (feature position, time-lagged feature position, time-lag, and class), a reduction to two dimensions is an unfortunate necessity for the purpose of visual display.

Results similar to these have also been found using different labeling schemes (e.g., mono-phones and syllables), feature sets such as LPC coefficients (Rabiner & Juang 1993), and speech corpora (Pitrelli *et al.* 1995; Bellcore Digits+ Database 1990's).

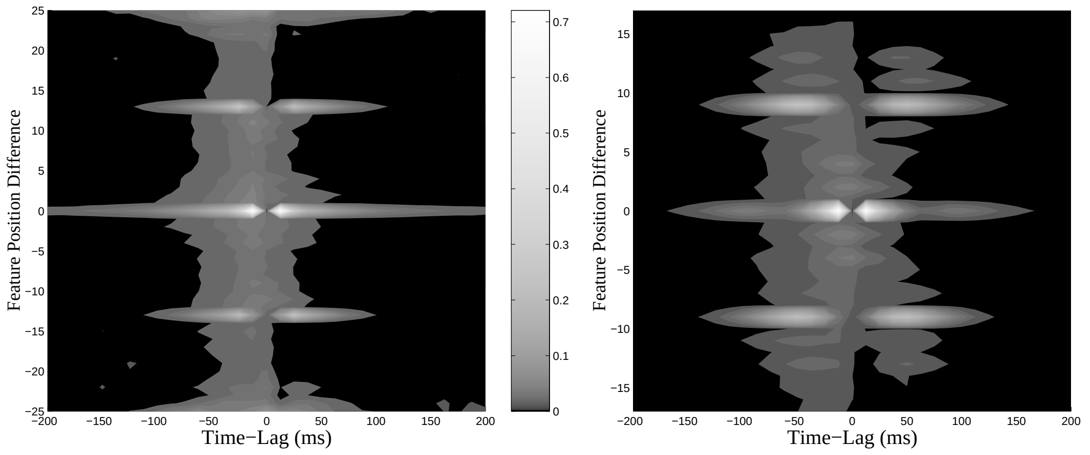

Figure 4.3: The conditional mutual information density of Switchboard using mel-frequency cepstral coefficients (left) and RASTA-PLP coefficients (right).

The above plots confirm the suspicion that the hidden variables in an HMM might not account for all the information available in the acoustic environment, or another words, that the HMM's loss is not necessarily zero. Apparently, using decision-tree clustered tri-phones for Q_t is not sufficient to represent all the information contained in X_t's acoustic environment. It therefore seems safe to assume that more information capable of decreasing X_t's entropy is available. The problem remaining is to decide what part is relevant and non-redundant.

4.2 Likelihood Increasing Dependency Selection

The previous section suggests that using an observation model of the form $p(X_t|Q_t)$ is deficient to the extent that the HMM loss $I(X_t; X_{\neg t}|Q_t)$ is greater than zero. The conditional mutual information $I(X_t; X_{\neg t}|Q_t)$ represents the quantity of additional information $X_{\neg t}$ provides about X_t not already provided by Q_t. Therefore, it might be possible to increase the accuracy of an HMM without increasing the number of hidden states by augmenting the HMM with additional cross-observation dependencies. In this section and those that follow, a series of different rules will be proposed for automatically determining the set of additional cross-observation dependencies using a variety of information theoretic measures.

One suggestion might be to use the quantity $p(X_t|Q_t, X_{\neg t})$ as an acoustic model. The probability of $X_{1:T}$ under an HMM is:

$$\begin{aligned}
p(X_{1:T}) &= \sum_{q_{1:T}} p(X_{1:T}, q_{1:T}) \\
&= \sum_{q_{1:T}} p(X_{1:T}|q_{1:T})p(q_{1:T}) \\
&= \sum_{q_{1:T}} \prod_t p(X_t|X_{1:t-1}, q_t)p(q_t|q_{t-1}) \\
&= \sum_{q_{1:T}} \prod_t p(X_t|q_t)p(q_t|q_{t-1})
\end{aligned}$$

The last equality is obtained using the HMM assumption that X_t is independent of $X_{1:t-1}$ given q_t. Therefore, using just the probability model $p(X_t|X_{<t}, Q_t)$ is sufficient.

There are obvious difficulties with this approach. Including a dependency on all $X_{<t}$ adds an enormous number of parameters to the system. Furthermore, the information contained in $X_{<t}$ is surely redundant and could either be compressed or could be represented by a subset of delegates. In this work, the latter approach is taken.

The problem then is to choose the best fixed-size collection of variables $\mathcal{Z} \subseteq X_{<t}$, to choose the size $|\mathcal{Z}|$, and to define the meaning of "best." At times, $\mathcal{Z}$ might also be chosen from the larger collection $\mathcal{Z} \subseteq X_{\neg t}$ which would potentially lead to a Markov random field. This will be discussed in the context where it is used.

One solution might be to simply pick a particular subset of $X_{<t}$ for $\mathcal{Z}$. For example, one could choose $\mathcal{Z} = X_{t-\ell}$ for some ℓ or for a combination of ℓ's. This is the approach used by correlation models (Wellekens 1987) and vector auto-regressive HMMs (Kenny *et al.* 1990; Woodland 1992). This method, however, chooses dependency variables regardless of their usefulness. Dependency variables should be chosen only if they are relevant, i.e., only if they provide new information not already provided by Q_t or $I(X_t; \mathcal{Z}|Q_t) > 0$.

Another solution might be to choose $\mathcal{Z} = X_{<t}$, train the resulting model, and delete individual entries from $\mathcal{Z}$ that do not show a strong dependence. The strength of a dependence in this case would be determined from the parameters of the model. A parameter close to zero, say, might indicate a lack of dependence. This in theory might work but it is problematic for a variety of reasons. First, the amount of computer memory and/or computation required to train the full system could be prohibitive. Second, the full system will model dependencies between variables that have little or no probabilistic dependence (i.e., that have a purely random relationship). Training a system that contains an overabundance of dependencies (and therefore contains an overabundance of parameters) would require a much larger training corpus. Without sufficient training data, over-training could occur and a random relationship between two variables could be confused with a proper dependency. Finally, with a system that contains many irrelevant parameters, it could be difficult to properly estimate the parameters governing the important dependencies.

A third approach might be to choose a random subset of variables from $X_{<t}$ for $\mathcal{Z}$, train the resulting system, and then prune. While this approach avoids the high memory, computational, and training data costs of the previous approach, the selection of good initial dependencies is not guaranteed.

Another potential drawback of these pruning approaches is that they posses a dependency on the parameter training method. If the training method does not, after training, leave the parameters for the undesirable dependencies in an identifiable state (e.g., close to zero, etc.), then these approaches will fail. Even a discriminative training method is not guaranteed to work — the parameters for two dependency variables might cancel each other out on average but still have large absolute values.

The approach used in this work starts from a simple model, compute the model's loss, and add dependencies that are found to be missing. If the quantity $I(X_t;\mathcal{Z}|Q)$ is maximized for a given size of $\mathcal{Z}$ (i.e., for $|\mathcal{Z}| = k$ for some k), then the uncertainty about X_t reducible from the unused remaining acoustic environment is minimized. This follows from the chain rule of mutual information (Cover & Thomas 1991), which states that:

$$I(X_t;X_{1:t-1}|Q) = I(X_t;\mathcal{Z}|Q) + I(X_t;X_{1:t-1} \setminus \mathcal{Z}|Q,\mathcal{Z})$$

where $\setminus$ is the set difference operator. Choosing $\mathcal{Z}$ to maximize $I(X_t;\mathcal{Z}|Q)$ will therefore minimize $I(X_t;X_{1:t-1} \setminus \mathcal{Z}|Q,\mathcal{Z})$ which can be seen as the remaining loss of the new model.

This leads to the following dependency selection rule:

Selection Rule 4.1. Maximize Conditional Mutual Information. *Choose the size $|\mathcal{Z}|$ set of random variables $\mathcal{Z} \subseteq X_{<t}$ such that the quantity $I(X_t;\mathcal{Z}|Q)$ is maximized.*

The conditional mutual information may be expanded as:

$$I(X_t;\mathcal{Z}|Q_t) = \sum_q I(X_t;\mathcal{Z}|Q_t = q)p(Q_t = q)$$

which is a sum of prior weighted entropy reductions, each for a particular class q. For different values of q, X given q might have strong dependencies only on a particular subset of $\mathcal{Z}$. Another possibility, therefore, is to choose a different set of variables $\mathcal{Z}_q$ for each q, that maximize $I(X_t;\mathcal{Z}_q|Q_t = q)$. Here, $\mathcal{Z}_q$ is defined as a subset of $X_{<t}$ such that X is essentially independent of $X_{<t} \setminus \mathcal{Z}_q$ given $Q = q$ and $\mathcal{Z}_q$. In other words, $\mathcal{Z}_q$ is chosen so that $I(X;X_{<t} \setminus \mathcal{Z}_q|q,\mathcal{Z}_q) \approx 0$. This leads to the following new criterion for the selection of $\mathcal{Z}_q$:

Selection Rule 4.2. Maximize q-specific Conditional Mutual Information. *For each q, choose the size $|\mathcal{Z}_q|$ set of random variables $\mathcal{Z}_q \subseteq X_{<t}$ such that the quantity $I(X_t;\mathcal{Z}_q|Q = q)$ is maximized.*

Compared to rule 4.1, this criterion is desirable to the degree that $\mathcal{Z}_q \neq \mathcal{Z}_r$ for $q \neq r$. It also might be useful because the prior probabilities $p(q)$ are subject to change between training and testing environments. Assuming that the statistical properties of the class conditional observation variables are not subject to change between training and testing environments (e.g., instances of the same phones have similar statistical properties), minimizing a training-data determined prior weighed sum, as in rule 4.1, might not minimize

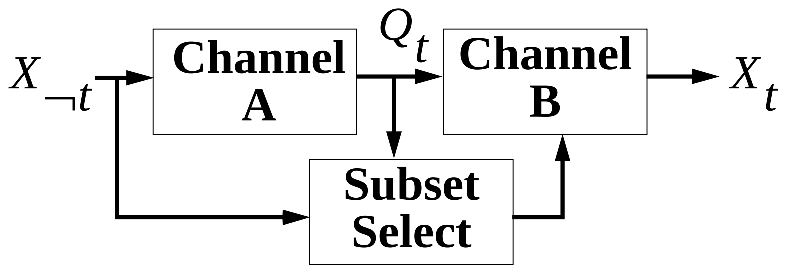

Figure 4.4: Improving an HMM by including additional direct dependencies on the relevant portions of $X_{\neg t}$ depending on the value of Q_t.

a test-data determined prior weighted sum. Rule 4.2 chooses $\mathcal{Z}_q$ individually using only the class conditional statistics without considering the priors.

The selection of q-specific dependency variables is depicted in Figure 4.4. As compared to the corresponding figure for an HMM (Figure 3.8), the potential for information transmission between $X_{<t}$ and X_t has increased. Therefore, less burden is placed on the hidden state to encode all relevant information about the observation environment.

From here on, the t subscript on X_t and other variables will be dropped unless it is needed for clarity. Without t, the problem may be abstracted as follows: given a data random variable X, a class random variable Q, and a set of all possible candidate auxiliary random variables $\mathscr{Z} = \{Z_1, Z_2, \ldots, Z_N\}$ for some N, choose the best fixed size subset of $\mathcal{Z} \subseteq \mathscr{Z}$ such that the probability model $p(X|Q, \mathcal{Z})$ is optimized for a classification task. This problem is depicted in Figure 4.5, where $\mathcal{Z}^{(a)}$ and $\mathcal{Z}^{(b)}$ are (both size four) candidate subsets of $\mathscr{Z}$ and where $\mathcal{Z}^{(a)} = \{Z_1, Z_2, Z_3, Z_4\}$ and $\mathcal{Z}^{(b)} = \{Z_4, Z_5, Z_6, Z_7\}$.

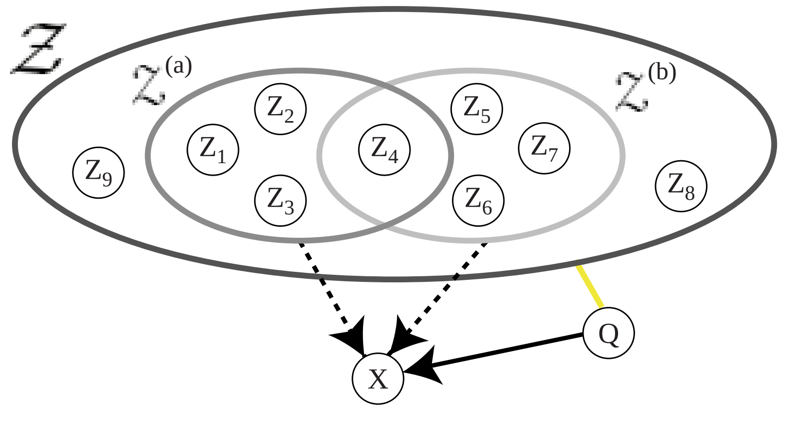

Figure 4.5: Evaluating the set of dependency variables $\mathcal{Z}^{(a)}$ or $\mathcal{Z}^{(b)}$ from the entire set $\mathscr{Z}$.

In this abstracted form, why is conditional mutual information a useful selection criterion? Intuitively, if only one of the two random variables Z_1 and Z_2 could be used as an information source about X (i.e., the model could either be $p(X|Z_1)$ or $p(X|Z_2)$ but not $p(X|Z_{1:2})$), Z_1 is chosen if $I(X; Z_1) > I(X; Z_2)$ because Z_1 provides more information about X. Choosing the variables Z_1 will increase the likelihood of the data under the new model relative to either the unconditional model or to the model that uses Z_2. A higher likelihood results because the typical probability values of the sharper (lower entropy) distribution $p(X|Z_1)$ are larger than those of the duller $p(X)$ or $p(X|Z_2)$. More formally, consider the following theorem.

Theorem 4.1. Mutual Information and Likelihood. *Given a random variable X, and two context variables $\mathcal{Z}^{(a)}$ and $\mathcal{Z}^{(b)}$, where $I(X;\mathcal{Z}^{(a)}) > I(X;\mathcal{Z}^{(b)})$, the likelihood of data sampled from X is higher given data sampled from $\mathcal{Z}^{(a)}$ than given data sampled from $\mathcal{Z}^{(b)}$, for n, the sample size, large enough. That is,*

$$\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(a)}) > \frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(b)})$$

Proof. Suppose

$$I(X;\mathcal{Z}^{(a)}) > I(X;\mathcal{Z}^{(b)})$$

for two different random vectors $\mathcal{Z}^{(a)}$ and $\mathcal{Z}^{(b)}$. It immediately follows that:

$$H(X) - H(X|\mathcal{Z}^{(a)}) > H(X) - H(X|\mathcal{Z}^{(b)})$$

or equivalently

$$H(X|\mathcal{Z}^{(a)}) < H(X|\mathcal{Z}^{(b)}).$$

Expanding into integral form, gives

$$-\int p(x,z^{(a)})\log p(x|z^{(a)})dxdz^{(a)} < -\int p(x,z^{(b)})\log p(x|z^{(b)})dxdz^{(b)}$$

or in limit form,

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(a)}) > \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(b)})$$

where $(x_i, z_i^{(k)}) \sim p(X,\mathcal{Z}^{(k)})$. By the definition of the limit, the equivalence of the two previous inequalities require that $\forall\epsilon > 0$, $\exists\ n_1$ and n_2 such that for $n > \max(n_1, n_2)$,

$$\left|\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(a)}) + H(X|\mathcal{Z}^{(a)})\right| < \epsilon$$

and

$$\left|\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(b)}) + H(X|\mathcal{Z}^{(b)})\right| < \epsilon$$

If we choose $\epsilon < |H(X|\mathcal{Z}^{(a)}) - H(X|\mathcal{Z}^{(b)})|/2$ to get n, this implies:

$$\frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(a)}) > \frac{1}{n}\sum_{i=1}^{n}\log p(x_i|z_i^{(b)})$$

which are the log likelihoods for a size n sampled data set. $\square$

In practice, the actual probability distribution $p(x|z)$ is not available. Instead, an approximation $\hat{p}(x|z,\Theta)$ is used where the parameters Θ are often estimated using a maximum likelihood parameter estimation procedure. A maximum likelihood procedure is, in the limit, equivalent to a minimization of the KL-distance between $\hat{p}$ and p (Bishop 1995). This can be seen by viewing the likelihood function as a cross entropy. In other words,

$$-\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}\log\hat{p}(x_i|z_i,\Theta) = -\int p(x,z)\log\hat{p}(x|z,\Theta)dxdz$$

but

$$-\int p(x,z)\log\hat{p}(x|z,\Theta)dxdz - H(X|\mathcal{Z}) = D(p(x|z)||\hat{p}(x|z,\Theta))$$

Since $H(X|\mathcal{Z})$ is a constant with respect to Θ, a maximum likelihood procedure decreases $D(p(x|z)||\hat{p}(x|z))$ again if n is large enough. If $\hat{p}$ is close enough to p, then the theorem above still holds. For the remainder of this thesis, it is assumed that $\hat{p}$ is close enough to p so that any differences are negligible.

In light of the above theorem, phrases such as "maximize the mutual information between X and $\mathcal{Z}$," "minimize the conditional entropy of X given $\mathcal{Z}$," and "maximize the likelihood of X given $\mathcal{Z}$" will be used interchangeably. Also, other law-of-large-numbers-like equivalences for sufficiently large n will at times be assumed without proof. It should be noted that the phrase "maximize the likelihood" does not in this context mean "adjust parameters to maximize the likelihood." Instead, it means to adjust the statistical structure (as controlled by the set of edges in a probabilistic network) that maximize the potential likelihood of the model when optimally trained using a maximum likelihood procedure.

The above theorem is also true for conditional mutual information (i.e., $I(X;\mathcal{Z}|Q)$ or for a particular value of q, $I(X;\mathcal{Z}|Q=q)$.). In other words, if $I(X;\mathcal{Z}_q^{(a)}|Q=q) > I(X;\mathcal{Z}_q^{(b)}|Q=q)$, then:

$$\frac{1}{n}\sum_{t=1}^{T}\log p(x_t|z_{q_t,t}^{(a)},Q_t=q_t) > \frac{1}{n}\sum_{t=1}^{T}\log p(x_t|z_{q_t,t}^{(b)},Q_t=q_t)$$

These quantities can be viewed as likelihoods of the data given Viterbi paths q_t of modified HMMs. In the left case, the Viterbi path likelihood is higher. Note that using a similar argument as in the theorem, and because $H(X) \geq H(X|\mathcal{Z})$,

$$\frac{1}{n}\sum_{t=1}^{T}\log p(x_t|z_{r_t,t}^{(a)},Q_t=r_t) \geq \frac{1}{n}\sum_{t=1}^{T}\log p(x_t|Q_t=r_t)$$

for some non-Viterbi path r_t and for n large enough. In other words, relative to an HMM, the likelihood of the data for paths other than the Viterbi path do not decrease when adding conditioning variables. It has therefore just been shown that the modified HMM probability will provide a higher likelihood score than the unmodified HMM. The following theorem therefore holds.

Theorem 4.2. An HMM with dependency variables added according to conditional mutual information produces a higher likelihood score on sampled data than the original HMM.

The resulting modified HMMs represent statistical relationships contained in the data that produce higher likelihood models. Increasing the likelihood of a model via parameter estimation is one technique that often leads to better performance (Rabiner & Juang 1993; Bishop 1995). As will be shown in the next section, however, this is not necessarily the best dependency selection criterion.

4.3 Discriminability

The selection methods above are sufficient to increase the descriptive power (i.e., lead to a higher likelihood) of a model but they do not necessarily decrease classification error. To decrease classification error, one must instead minimize Bayes error (equivalently Bayes risk) (Duda & Hart 1973; Vapnik 1998), or optimize the posterior probability of a class given the data. This follows from Bayes decision theory which states that the minimum probability of error is achieved when a class is chosen that has the highest posterior probability.

In general, there are two types of class-conditional density training methods, likelihood-based and discriminative.[4] Likelihood-based training adjusts the parameters of a model for each class using only samples from that class. The primary goal is for the model to report a high probability for typical samples. Samples from classes other than the one the model represents, however, are not used for training. Therefore, if two different classes have similar prominent attributes, the models trained for those classes might each report a high probability for samples from both classes. In the worst case, a competing model could report a higher probability for a sample than the correct model reports. In this case, even with large likelihood scores from each model, discriminability (the ability for the models to distinguish one class from another) decreases.

Let x be a particular data sample. Then $p(x|q)$ is the likelihood of the sample according to the model for class q, and $p(x) = \sum_r p(r)p(x|r)$ is the likelihood of a sample x according to the average over all models. If x is drawn according to the q model, then $p(x|q)$ should report a higher score for x than does $p(x)$. The difference between the two is a measure of discriminability, and so is the log ratio of likelihoods $log[p(x|q)/p(x)]$. The average log ratio over samples drawn according to q is the average discriminability of the q model, i.e.:

$$\text{Discriminability of } q \text{ model} = \int p(x|q) \log \frac{p(x|q)}{p(x)} dx$$

which is equal to the KL-distance $D(p(x|q)||p(x))$. Averaging over all classes, the *discrim-*

[4]The distinction made here is different from the distinction between ML, MAP, and Bayesian parameter estimation procedures. These three methods, described in Section 2.4, all estimate a parameter Θ for a single joint model. The main difference between ML, MAP, and Bayesian parameter estimation is the existence of a prior or a cost over Θ and how that cost is used (multiplied or integrated).

inability of a set of models is informally defined as the following:

$$\text{Discriminability of a set of models} = \sum_q p(q) D(p(x|q)||p(x)).$$

In Section 4.4.1, it is shown how this quantity is related to the posterior probability.

Discriminative training, as does likelihood training, adjusts the model parameters for a class based on samples from that class. Unlike likelihood training, however, it also adjusts the model to produce a low probability score for samples from competing classes. Therefore, if there are similar attributes between classes, the importance of those attributes in contributing to the probability of each of the models will diminish (also see Section 3.3.10). Assuming the models have the ability to represent the unique attributes of each class, a discriminative training method will produce models where only the unique attributes of a class have strong affects on the probability scores,

In the preceding sections, procedures were presented to optimize the structure of an HMM (parameter optimization methods have not yet been specified). Nevertheless, a procedure that optimizes model structure to increase the potential likelihood could suffer the same problems as likelihood-based training. In this case, the dependency variables chosen for one class might also be chosen for a different class, the resulting probabilities of data sampled from both classes could be high for both models, and discriminability could decrease. This problem can occur even when choosing a different $\mathcal{Z}_q$ for each q; it depends on the statistical properties of the data from each class. Therefore, a procedure is needed that chooses the unique entropy reducing dependency variables for each class, or in other words, that increases the model's structural discriminability. The approach taken here is to derive discriminative versions of an HMM's loss.

The conditional mutual information can be seen as reducing X's entropy in a particular context. That is,

$$I(X;\mathcal{Z}|q) = \int p(x,z|q) \left[\log \frac{1}{p(x|q)} - \log \frac{1}{p(x|q,z)}\right] dxdz$$

where $p(x,z|q)$ is the probability of x and z in the q context, and $\left[\log \frac{1}{p(x|q)} - \log \frac{1}{p(x|q,z)}\right]$ is the individual event-wise entropy reduction of x provided by z under the q model. Conditional mutual information can be seen as the entropy reduction of x provided by z averaged in the q context.

Discriminability can also be viewed from an information-theoretic perspective. While the chosen dependency variables $\mathcal{Z}$ might reduce the entropy of X under the q-model in the q-context, $\mathcal{Z}$ might also reduce "entropy" in a different and incorrect context, say r, under the q-model. To increase the discriminability between different classes, dependencies should be chosen that both 1) decrease entropy of a model in its correct context and 2) do not decrease the entropy in other contexts.

A measure of this second concept can be obtained by changing the context where the entropy reduction of a model is evaluated.

Definition 4.1. Cross-Context Conditional Mutual Information (CCCMI). *The* *cross-context conditional mutual information is defined as:*

$$\begin{aligned} I_r(X;\mathcal{Z}|q) &\triangleq \int p(x,z|r)\left[\log\frac{1}{p(x|q)} - \log\frac{1}{p(x|q,z)}\right]dxdz \\ &= \int p(x,z|r)\log\frac{p(x,z|q)}{p(x|q)p(z|q)}dxdz \end{aligned}$$

where r is the context class, and q is the model class.

The CCMI can be viewed as event-wise entropy reductions of the model for q but averaged over the context of the probability model r. When $r = q$, standard conditional mutual information is obtained (i.e., $I_q(X;\mathcal{Z}|q) = I(X;\mathcal{Z}|q)$). When $r \neq q$, it represents the typical situation in a classification when evaluating the q-model in the r context.

It is now apparent that a goal for a chosen $\mathcal{Z}_q$ should be that conditional mutual information is large, and CCCMI is small. The following quantity will be useful when developing approximation algorithms to the above dependency selection rule.

Definition 4.2. Utility. *The utility of a set of commentary variables $\mathcal{Z}_q$ is defined as:*

$$U(X;\mathcal{Z}_q|q) = I(X;\mathcal{Z}_q|q) - \sum_r p(r)I_r(X;\mathcal{Z}_q|q)$$

As defined, utility uses the priors $p(r)$, but this is not necessary — since the priors can change between training and testing conditions, uniform or some other estimate of the test condition priors could be used.

Some sets of classes might be more confusable with each other than other sets of classes. If certain classes are known to be highly confusable (e.g., in speech, vowels with other vowels, or plosives with other plosives), confusability clusters can be used to produce a modified form of utility:

Definition 4.3. Confusability Refined Utility. *The confusability refined utility of a set of commentary variables $\mathcal{Z}_q$ is defined as:*

$$U_c(X;\mathcal{Z}_q|q) = I(X;\mathcal{Z}_q|q) - \frac{1}{K_q}\sum_{r\in C_q} p(r)I_r(X;\mathcal{Z}_q|q)$$

where C_q is the set of classes that are potentially confusable with the class q, and where $K_q = \sum_{r\in C_q} p(r)$ is a normalizing constant.

If C_q contains all classes in definition 4.3, then it is equivalent to definition 4.2.

From Theorem 4.1, it can be seen that the utility of a set of dependency variables $\mathcal{Z}_q$ measures how much $\mathcal{Z}_q$ will increase the q-model's likelihood in the context of q relative to how much $\mathcal{Z}_q$ will increase the q-model's likelihood averaged in other contexts. Given two possible dependency variable sets $\mathcal{Z}_q^{(a)}$ and $\mathcal{Z}_q^{(b)}$, define $\underset{b\to a}{\Delta} L_r(q)$ as the average change in likelihood that occurs using $\mathcal{Z}_q^{(a)}$ instead of $\mathcal{Z}_q^{(b)}$ when evaluating the q model in the context of r-sampled data. If $U(X;\mathcal{Z}_q^{(a)}|q) > U(X;\mathcal{Z}_q^{(b)}|q)$ for all q, then

$$I(X;\mathcal{Z}_q^{(a)}|q) - I(X;\mathcal{Z}_q^{(b)}|q) > \sum_r p(r)I_r(X;\mathcal{Z}_q^{(a)}|q) - \sum_r p(r)I_r(X;\mathcal{Z}_q^{(b)}|q)$$

or equivalently,

$$\underset{b\to a}{\Delta} L_q(q) > \sum_r p(r) \underset{b\to a}{\Delta} L_r(q).$$

This inequality implies that, from using $\mathcal{Z}_q^{(a)}$ instead of $\mathcal{Z}_q^{(b)}$, the likelihood increase of the q model evaluated in the q context is greater than the average likelihood increase of the q model evaluated in all other contexts. In other words, in the context of q-sampled data, the q-model's probability will increase more than it does in the context of other data.

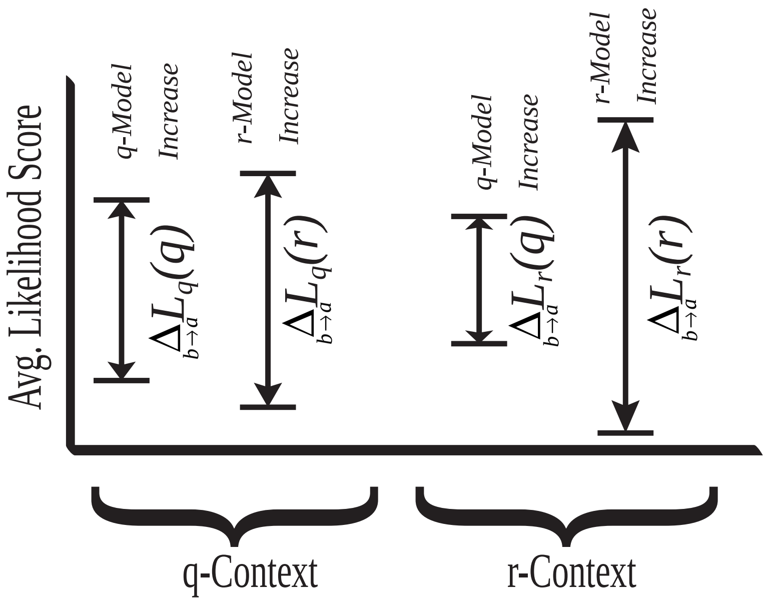
Avg. Likelihood Score
q-Model Increase
r-Model Increase
q-Model Increase
r-Model Increase
$\underset{b\to a}{\Delta} L_q(q)$
$\underset{b\to a}{\Delta} L_q(r)$
$\underset{b\to a}{\Delta} L_r(q)$
$\underset{b\to a}{\Delta} L_r(r)$
q-Context
r-Context

Figure 4.6: Utility is maximized for two classes q and r since $\underset{b\to a}{\Delta} L_q(q) > \underset{b\to a}{\Delta} L_r(q)$ and $\underset{b\to a}{\Delta} L_r(r) > \underset{b\to a}{\Delta} L_q(r)$, but discriminability is decreased in the q context.

This suggests the following dependency selection rule.

Selection Rule 4.3. Maximize Utility. *For each q, choose the size $|\mathcal{Z}_q|$ set of random variables $\mathcal{Z}_q$ such that the utility $U(X;\mathcal{Z}_q|q)$ is maximized.*

If the utility for $\mathcal{Z}_q$ is large and positive, then $\mathcal{Z}_q$ is likely to be a more discriminative set of dependency variables. Unfortunately, large utility does not necessarily imply an increase in discriminability. Figure 4.6 depicts a case where the utility is large but discriminability has not increased. A more desirable condition is that for all q:

$$\underset{b\to a}{\Delta} L_q(q) > \sum_r p(r) \underset{b\to a}{\Delta} L_q(r).$$

In other words, the average likelihood increase of the competing models r evaluated in the context of q should be less than the likelihood increase of the model q evaluated in the q context. This condition is depicted in Figure 4.7 for two classes.

The structural discriminability of a model therefore depends both on its own dependency variables $\mathcal{Z}_q$ and on dependency variables chosen for other models. An appropriate

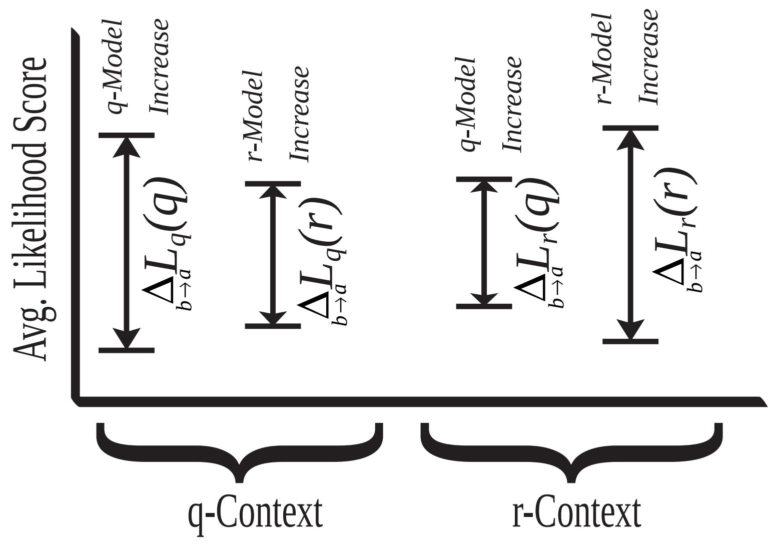
Avg. Likelihood Score
q-Model Increase
$\underset{b\to a}{\Delta} L_q(q)$
r-Model Increase
$\underset{b\to a}{\Delta} L_q(r)$
q-Model Increase
$\underset{b\to a}{\Delta} L_r(q)$
r-Model Increase
$\underset{b\to a}{\Delta} L_r(r)$
q-Context
r-Context

Figure 4.7: Discriminability increase condition for two classes q and r since $\underset{b\to a}{\Delta} L_q(q) > \underset{b\to a}{\Delta} L_q(r)$ and $\underset{b\to a}{\Delta} L_r(r) > \underset{b\to a}{\Delta} L_r(q)$.

loss measure should therefore be defined on $\mathcal{Z}$, where $\mathcal{Z}$ is defined as the entire collection of dependency variables for all q.

Definition 4.4. Discriminative Conditional Mutual Information (DCMI). *The discriminative conditional mutual information of a set of dependency variables* $\mathcal{Z} = \{\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \ldots \cup \mathcal{Z}_N\}$ *is defined as:*

$$S(X;\mathcal{Z}|q) = I(X;\mathcal{Z}_q|q) - \sum_r p(r) I_q(X;\mathcal{Z}_r|r)$$

If $S(X;\mathcal{Z}^{(a)}|q) > S(X;\mathcal{Z}^{(b)}|q)$ for all q, then $\underset{b\to a}{\Delta} L_q(q) > \sum_r p(r) \underset{b\to a}{\Delta} L_q(r)$. This leads directly to the following prior-weighted dependency selection rule:

Selection Rule 4.4. Maximize Average DCMI. *Choose* $\mathcal{Z} = \{\mathcal{Z}_1 \cup \mathcal{Z}_2 \cup \ldots \cup \mathcal{Z}_N\}$ *to maximize the quantity*

$$S(X;\mathcal{Z}|Q) \triangleq \sum_q p(q) S(X;\mathcal{Z}|q)$$

In Section 4.4.2, the relationship between this rule and the posterior probability of a class is given.

Although maximizing utility does not directly yield more discriminative models, it is useful for several reasons. First utility and DCMI are related by:

$$\sum_q p(q) U(X;\mathcal{Z}_q|q) = \sum_q p(q) S(X;\mathcal{Z}|q) = S(X;\mathcal{Z}|Q)$$

so the average utility can be substituted in place of average DCMI in rule 4.4. Second, using utility, it is possible to obtain an estimate of the value of a particular $\mathcal{Z}_q$ for a given

q without considering all of $\mathcal{Z}$. CCMI, on the other hand, is a function not just of $\mathcal{Z}_q$ but of the entire $\mathcal{Z}$ ensemble of variables.

Conditional mutual information $I(X;Z|Q=q)$ and utility[5] are compared, for compressed sub-band spectral envelopes, in Figures 4.8 and 4.9 for various values of q spanning over different phonemes. The utility plots show a type of discriminative loss for the corresponding HMM. In other words, these plots show where in time-frequency additional information exists that could potentially reduce recognition error. The conditional mutual information plots show a similar pattern regardless of the condition. The utility plots, on the other hand, show very different patterns of loss for each condition. In particular, plosives (d, k, b, and p) tend to have more discriminative loss across frequency within a narrow-time region. Fricatives (f, s, th, and v) also show loss primarily over frequency but the loss spans a greater temporal range. Other phonemes such as vowels (ae, ah, ih, and iy) show loss both across time and frequency. For vowels, the least informative spectro-temporal regions, however, are close in time and frequency. Nasals (m and n) and liquids (l and r) show more complex patterns of loss over time and frequency. The patterns shown by these discriminative loss functions are consistent with known spectro-temporal phonemic characteristics (Clark & Yallop 1995).

4.4 Posterior Probability

In the previous section, a dependency variable selection algorithm was presented that uses DCMI. The ultimate form of discriminability criterion, however, is the posterior probability. According to Bayes decision theory, choosing the class with the largest class posterior will minimize the probability of error. Furthermore, posterior-based training algorithms are typically superior to pure likelihood-based training (Bahl *et al.* 1986; Brown 1987; Ephraim *et al.* 1989; Ephraim & Rabiner 1990; Juang & Katagiri 1992; Juang *et al.* 1997; Bourlard & Morgan 1994; Konig 1996). Therefore, a procedure to select dependencies that more accurately approximate the posterior probability is desirable. In this section, it is first shown how mutual information and the average posterior probability are essentially equivalent. Next, it is shown that, under certain assumptions, selection rule 4.2 increases an upper bound on the average posterior probability. Finally, dependency variable selection algorithms are derived that directly increase the average posterior probability.

4.4.1 Mutual Information and Posterior Probability

If Q is a class random variable and X is a data random variable, then $\log p(Q|X)$ is also a random variable. The expected value of $\log p(Q|X)$ is equal to the average log posterior probability. The mutual information between X and Q is directly related to the average posterior probability as is shown in the following theorem.

Theorem 4.3. Equivalence of Mutual Information and Class Posterior. *Increasing the mutual information between a feature random variable X and a class random variable Q correspondingly increases the average posterior probability of Q given X, i.e., $E[\log p(Q|X)]$.*

[5]The figure actually shows an approximation to utility. See Section 4.6.

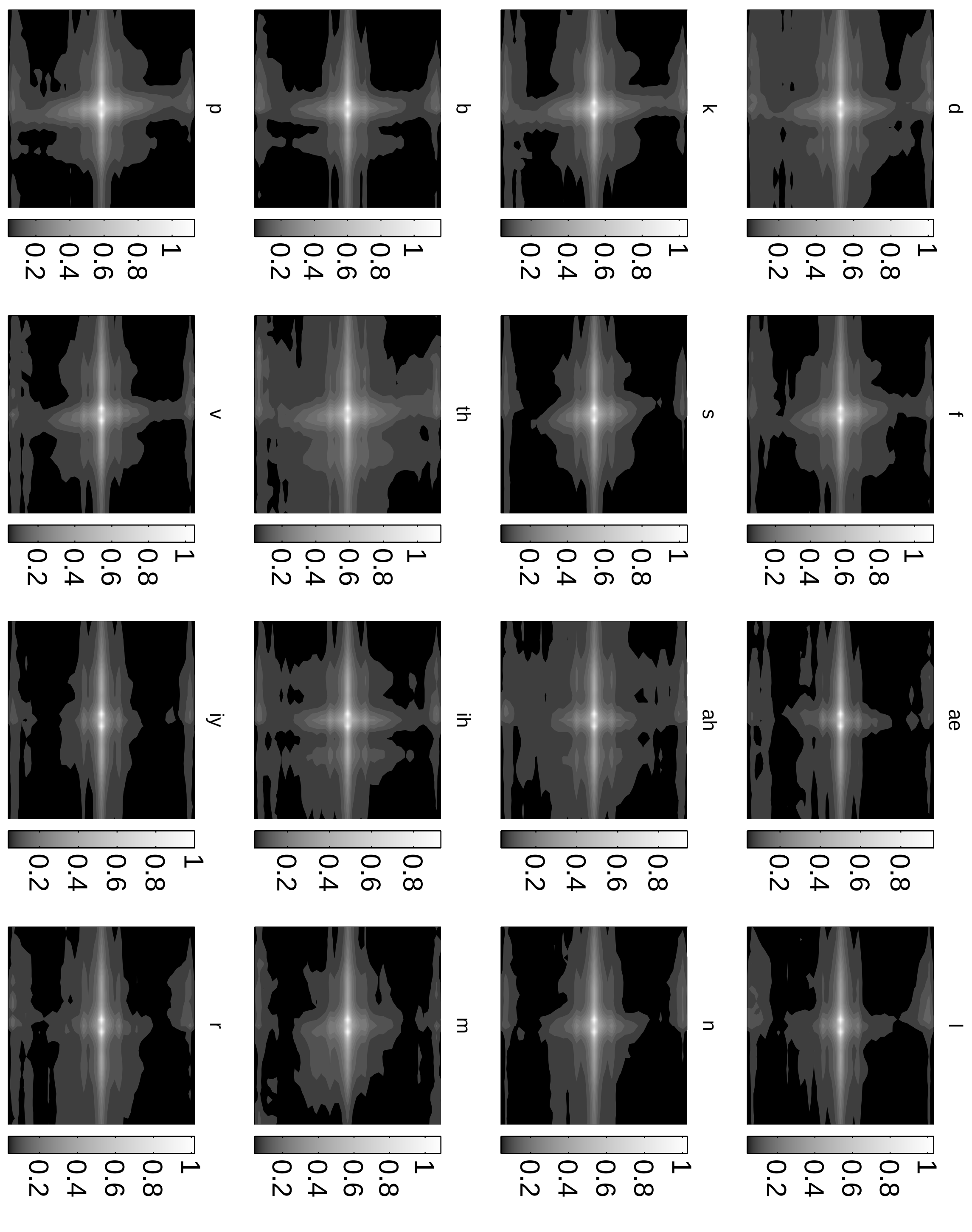

Figure 4.8: Conditional Mutual Information $I(X;Z|Q=q)$ for various q in Switchboard. Plot are provided for plosives (d, k, b, p), fricatives (f, s, th, and v), vowels (ae, ah, ih, and iy), liquids (l and r), and nasals (m and n).

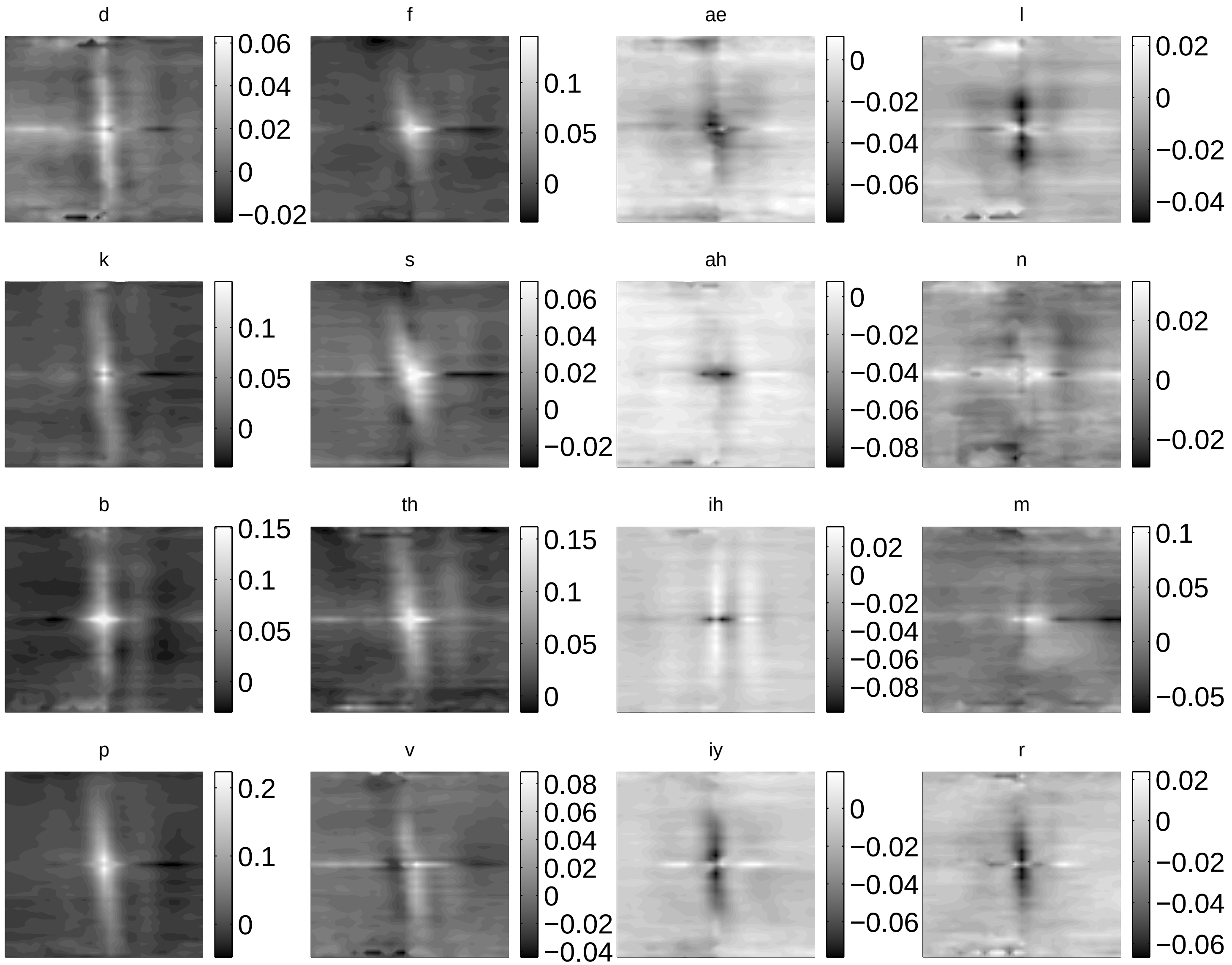

Figure 4.9: The distinctive features of the Switchboard phonemes as shown by utility approximated as described in Section 4.6. The displayed phonemes are the same as in Figure 4.8.

Proof. Consider the average log of the posterior over the data and over all classes.

$$E[\log p(Q|X)] = \sum_q p(q) \int p(x|q) \log p(q|x) dx = \int \log p(q|x) dF(x,q)$$

where $dF(x,q)$ is the joint distribution over x and q (Grimmett & Stirzaker 1991). It can be seen that this quantity is equal to $-H(Q|X)$ implying that:

$$H(Q) + E[\log p(Q|X)] = H(Q) - H(Q|X) = I(Q;X)$$

Therefore, increasing the mutual information between X and Q increases the average posterior and vice versa. Note that $H(Q)$ is unaffected by changes in the probabilistic relationship between X and Q. □

The theorem implies that adjusting the parameters of a classifier by maximizing the mutual information between the input and output will correspondingly increase the average posterior probability. Classifier training methods such as MMI (Bahl *et al.* 1986) and MDI (Ephraim & Rabiner 1990) which maximize mutual information can therefore correctly be called discriminative training methods. Furthermore, posterior-based training methods (such as multi-layered perceptron training with softmax outputs (Bishop 1995)) are essentially increasing the mutual information between the input features and the output class variables.

Looking back at the definition of discriminability given in Section 4.3, it can now be seen that $E[\log p(Q|X)] = \text{Discriminability} - H(Q)$ and that discriminability is equal to $I(Q;X)$. Therefore, increasing discriminability also increases average posterior probability.

4.4.2 DCMI and Posterior Probability

In this section, a comparison is made between DCMI and the posterior probability. It will be shown that, under certain conditions, increasing the DCMI will increase an upper bound on the posterior.

Consider the following form of posterior probability $p(Q|X,Z)$ where X is a feature random variable, Q is the class random variable, and Z are a set of dependency random variables. The variable Z once again denotes the collection of variables used by all models. That is, $Z = \{Z_1 \cup Z_2 \cup \ldots \cup Z_N\} \subseteq \mathcal{Z}$ where $\mathcal{Z}$ is the set of all possible dependency variables.

Consider the following form of the posterior probability $p(q|X,Z)$ expanded, using Bayes rule, as follows

$$p(q|X,Z) = \frac{p(X|q,Z)p(q|Z)}{\sum_r p(X|r,Z)p(r|Z)} = \frac{p(X|q,Z_q)p(q|Z)}{\sum_r p(X|r,Z_r)p(r|Z)}.$$

Note that Z_q is presumably chosen so that $X \perp\!\!\!\perp (\mathcal{Z} \setminus Z_q) | \{Z_q, q\}$ for all q and therefore

$p(X|q,\mathcal{Z}) = p(X|q,\mathcal{Z}_q)$. It follows that:

$$\begin{aligned}
&E[\log p(Q|X,\mathcal{Z})] \\
&= \sum_q p(q) \int p(x,z|q) \log p(q|x,z) dx dz \\
&= \sum_q p(q) \int p(x,z|q) \log \left[\frac{p(x|q,z)p(q|z)}{\sum_r p(x|r,z)p(r|z)} \right] dx dz \\
&= \sum_q p(q) \int p(x,z|q) \log p(x|q,z) dx dz + \sum_q p(q) \int p(z|q) \log p(q|z) dz \\
&\qquad - \sum_q p(q) \int p(x,z|q) \log \left[\sum_r p(x|r,z)p(r|z) \right] dx dz \\
&= -H(X|\mathcal{Z},Q) - H(Q|\mathcal{Z}) - \sum_q p(q) \int p(x,z|q) \log \left[\sum_r p(x|r,z)p(r|z) \right] dx dz
\end{aligned}$$

From Jenson's Inequality, it follows that for any x and z:

$$\log \left[\sum_r p(x|z,r)p(r|z) \right] \geq \sum_r p(r|z) \log p(x|z,r)$$

Therefore,

$$\begin{aligned}
&E[\log p(Q|X,\mathcal{Z})] \\
&\leq -H(X|\mathcal{Z},Q) - H(Q|\mathcal{Z}) - \sum_q p(q) \int p(x,z|q) \sum_r p(r|z) \log p(x|r,z) dx dz \qquad (4.5) \\
&= -H(X|\mathcal{Z},Q) - H(Q|\mathcal{Z}) - \sum_q p(q) \int p(x,z|q) \sum_r p(r|z) \log p(x|r,z_r) dx dz \qquad (4.6)
\end{aligned}$$

Equation (4.5) follows from Jensen's inequality and (4.6) follows since X is independent of $\mathcal{Z} \setminus \mathcal{Z}_q$ given $Q = q$ and $\mathcal{Z}_q$.

Equation (4.6) can be simplified further if the assumption is made that either $\mathcal{Z} \perp\!\!\!\perp Q$ or at least that $\mathcal{Z}$ has only a weak influence on Q (i.e., that $I(\mathcal{Z};Q)$ is sufficiently small). This approximation is true to a greater or lesser extent depending on the definition of $\mathcal{Z}$ (more on this in Section 4.7). Also, since mutual information decreases with increasing spectro-temporal distance in speech (Morris 1992; Morris *et al.* 1993; Yang *et al.* 1999), if the location of $\mathcal{Z}$ is spectro-temporally distant from Q, the assumption becomes more reasonable. This approximation and the definition of $\mathcal{Z}$ is further discussed in section 4.4.4. Making this assumption, $H(Q|\mathcal{Z}) \approx H(Q)$ and $p(r|z) \approx p(r)$ yielding:

$$\begin{aligned}
&E(p(Q|X,\mathcal{Z})) \\
&\leq -H(X|\mathcal{Z},Q) - H(Q) - \sum_q \sum_r p(q)p(r) \int p(x,z_r|q) \log p(x|r,z_r) dx dz \qquad (4.7)
\end{aligned}$$

As in Theorem 4.3, quantities independent of the choice of $\mathcal{Z}$ can be added and subtracted to both sides of this inequality without affecting an optimization procedure with respect to the selection of $\mathcal{Z}$. Therefore, adding the quantity

$$H(X|Q) + H(Q) + \sum_q \sum_r p(r)p(q) \int p(x|q) \log p(x|r) dx$$

to both sides of (4.7) produces:

$$\begin{aligned} & I(X;\mathcal{Z}|Q) - \sum_q \sum_r p(q)p(r) \int p(x, z_r|q) \log \frac{p(x|r, z_r)}{p(x|r)} dx dz \\ = \; & I(X;\mathcal{Z}|Q) - \sum_q \sum_r p(q)p(r) I_q(X;\mathcal{Z}_r|r) \\ = \; & S(X;\mathcal{Z}|Q) \end{aligned}$$

The preceding steps therefore prove the following theorem:

Theorem 4.4. Selection rule 4.4 increases an upper bound on the average posterior probability if $I(Q;\mathcal{Z})$ is small enough.

Increasing an upper bound could potentially reduce Bayes error, but it does not necessarily reduce it. A selection procedure directly related to Bayes error is described in the following section.

4.4.3 Posterior-based Dependency Variable Selection

Dependency variable selection rules may be derived directly starting from the posterior probability. Observe once again at the average log posterior:

$$\begin{aligned} E(\log p(Q|X, \mathscr{Z})) &= \sum_q p(q) \int p(x, z|q) \log p(q|x, z) dx dz \\ &= -H(Q|X, \mathscr{Z}) \end{aligned}$$

Adding $H(Q|X)$ to both sides does not affect the choice of $\mathcal{Z}$ in an optimization procedure

$$E(\log p(Q|X, \mathscr{Z})) + H(Q|X) = I(Q;\mathscr{Z}|X)$$

Therefore, when choosing a collection of dependency variables $\mathcal{Z} \subset \mathscr{Z}$ to maximize the resulting average posterior, the $\mathcal{Z}$ chosen should maximize the conditional mutual information between Q and $\mathcal{Z}$ given X. If Q and $\mathcal{Z}$ are independent given X, then $\mathcal{Z}$ is non-informative, and does not help to increase the average log posterior probability which then degenerates to the negative conditional entropy $-H(Q|X)$. The variable set $\mathcal{Z}$ should be chosen to provide new information about the class variable Q not already provided by X. This is an intuitively satisfying result since adding features containing only redundant information to a classifier provides no new information and should not grant any benefit. Such a criterion, in fact, could be used in a feature selection procedure — features could be added one at a time with each newly chosen feature providing maximal new information.

For the problem of selection $\mathcal{Z}$ for use by the class conditional models $p(X|\mathcal{Z}_q, q)$ for each q, a more useful way to look at the average posterior probability is as follows.

$$\begin{aligned} E(\log p(Q|X, \mathscr{Z})) &= -H(Q|X, \mathscr{Z}) \\ &= -H(X|\mathscr{Z}, Q) - H(Q|\mathscr{Z}) + H(X|\mathscr{Z}) \end{aligned}$$

Again, adding values that do not depend on $\mathscr{Z}$ leads to:

$$\begin{aligned} & E(\log p(Q|X, \mathscr{Z})) + H(X|Q) + H(Q) - H(X) \\ &= H(X|Q) - H(X|\mathscr{Z}, Q) + H(Q) - H(Q|\mathscr{Z}) - \big(H(X) - H(X|\mathscr{Z})\big) \\ &= I(X; \mathscr{Z}|Q) + I(Q; \mathscr{Z}) - I(X; \mathscr{Z}) \end{aligned}$$

In this form, there are three contributions to the average posterior. In the first term, $\mathscr{Z}$ provides new information about X that is not already provided by Q. This is essentially the likelihood criterion described in Section 4.2. In the second term, $\mathscr{Z}$ provides information about Q. Note, however, that even if Q and $\mathscr{Z}$ are marginally independent, $\mathscr{Z}$ can affect the posterior probability of Q given X via the first term (see Section 4.7.2 for an example). In the last term, the degree to which $\mathscr{Z}$ is informative about X in general (i.e., unconditional of any class) decreases the posterior probability. This last term determines if the $\mathscr{Z}$ variables help to distinguish one class from another. Discriminability requires that $\mathscr{Z}$ reduce the entropy of X in the context of Q, but not reduce the entropy of X as much in general.

A posterior-based dependency selection rule should choose the fixed-size subset $\mathcal{Z} \subseteq \mathscr{Z}$ that results in the smallest decrease in average posterior probability. This choice (and class specific choices $\mathcal{Z}_q$ for all q) will affect only the class conditional observation probability model $p(X|\mathcal{Z}, Q)$ (or $p(X|\mathcal{Z}_q, q)$). The observation models, however, are found only in the quantities $I(X; \mathscr{Z}|Q)$ and $I(X; \mathscr{Z})$ and not in the quantity $I(Q; \mathscr{Z})$. This can be seen from the following. First,

$$\begin{aligned} I(X; \mathscr{Z}|Q) &= H(X|Q) - H(X|Q, \mathscr{Z}) \\ &= -\int \log p(x|q) dF(x, q) + \int \log p(x|z, q) dF(x, z, q) \\ &= -\int \log p(x|q) dF(x, q) + \int \log p(x|z_q, q) dF(x, z_q, q) \\ &= I(X; \mathcal{Z}|Q) \end{aligned}$$

where z_q is an instance of $\mathcal{Z}_q$ and where the following property is assumed: $X \perp\!\!\!\perp (\mathscr{Z} \setminus \mathcal{Z}_q)|\{\mathcal{Z}_q, Q = q\}$. Second,

$$\begin{aligned} I(Q; \mathscr{Z}) &= H(Q) - H(Q|\mathscr{Z}) \\ &= -\int \log p(q) dF(q) + \int \log p(q|z) dF(q, z) \end{aligned}$$

where in this case z is an instance of $\mathscr{Z}$. Therefore, the choice of $\mathcal{Z}$ does not affect this quantity. Third,

$$\begin{aligned} I(X;\mathcal{Z}) &= H(X) - H(X|\mathcal{Z}) \\ &= -\int \log p(x) dF(x) + \int \log p(x|z) dF(x,z) \\ &= -\int \log p(x) dF(x) + \int \log \left(\sum_r p(x|r, z_r) p(r|z) \right) dF(x,z) \\ &\triangleq I_{\mathcal{Z}}(X;\mathcal{Z}) \end{aligned}$$

where again z_r is an instance of $\mathcal{Z}_r$ and z an instance of $\mathcal{Z}$. It can be seen, therefore, that this latest quantity is dependent on both the entire set $\mathcal{Z}$ for the quantities $p(r|z)$ but also on $\mathcal{Z}$ for the class conditional observation densities. It is therefore written as $I_{\mathcal{Z}}(X;\mathcal{Z})$.

Using the above, the best choice for $\mathcal{Z}$ is found using:

$$\begin{aligned} \mathcal{Z}^* &= \operatorname*{argmax}_{\mathcal{Z}} I(X;\mathcal{Z}|Q) + I(Q;\mathcal{Z}) - I_{\mathcal{Z}}(X;\mathcal{Z}) \\ &= \operatorname*{argmax}_{\mathcal{Z}} I(X;\mathcal{Z}|Q) - I_{\mathcal{Z}}(X;\mathcal{Z}) \end{aligned}$$

This leads to the following selection rule:

Selection Rule 4.5. Maximize Posterior. *Choose* $\mathcal{Z}$ *to maximize*

$$I(X;\mathcal{Z}|Q) - I_{\mathcal{Z}}(X;\mathcal{Z}).$$

The preceding can be taken as a proof of the following theorem:

Theorem 4.5. Selection rule 4.5 better approximates the average posterior probability and therefore decreases Bayes error.

4.4.4 Relationship to HMM Posterior Probabilities

The previous dependency selection methods have been motivated by optimizing what could called the "local" posterior probability of a hidden state Q given the features, i.e., $p(Q|X;\mathcal{Z})$. This was described in Figure 4.5, where hidden variables at other times were ignored. HMM-based pattern classification, however, is performed by selecting the HMM M^* that has the maximum "global" posterior probability given the data:

$$M^* = \operatorname*{argmax}_{M} p(M|X_{1:T})$$

where M indicates a distinct HMM (representing a word, phrase, sentence, etc.), and $X_{1:T}$ is a length T set of observation vectors. Similar to (Bourlard & Morgan 1994), $p(M|X_{1:T})$ can be expanded as

$$p(M|X_{1:T}) = \sum_{q_{1:T}} p(M, q_{1:T}|X_{1:T}) = \sum_{q_{1:T}} p(q_{1:T}|X_{1:T}) p(M|q_{1:T}, X_{1:T})$$

where $q_{1:T}$ is a particular HMM state sequence. This leads to:

$$\begin{aligned} p(M|X_{1:T}) &= \sum_{q_{1:T}} p(M|q_{1:T}, X_{1:T}) p(q_{1:T}|X_{1:T}) \\ &= \sum_{q_{1:T}} p(M|q_{1:T}, X_{1:T}) \prod_t p(q_t|q_{<t}, X_{1:T}) \end{aligned}$$

Where the last equality uses the chain rule of probability. It is therefore important to estimate the local conditional posterior probabilities $p(q_t|q_{<t}, X_{1:T})$. Proceeding again by taking the expected value of the log posterior probabilities yields:

$$\begin{aligned} &E[\log p(Q_t|Q_{<t}, X_{1:T})] \\ &= -H(Q_t|Q_{<t}, X_{1:T}) \\ &= -H(Q_t|Q_{<t}, X_t, X_{\neg t}) \\ &= -H(X_t|Q_t, Q_{<t}, X_{\neg t}) - H(Q_t|Q_{<t}, X_{\neg t}) + H(X_t|Q_{<t}, X_{\neg t}) \end{aligned}$$

If it is assumed that X_t is independent of $Q_{<t}$ given Q_t, and that X_t is independent of $Q_{<t}$ given the entire remaining observation environment $X_{\neg t}$, then this leads to:

$$\begin{aligned} &E[\log p(Q_t|Q_{<t}, X_{1:T})] \\ &= -H(X_t|Q_t, X_{\neg t}) - H(Q_t|Q_{<t}, X_{\neg t}) + H(X_t|X_{\neg t}) \end{aligned}$$

to which adding $H(X_t|Q_t) + H(Q_t|X_{\neg t}) - H(X_t)$ results in:

$$I(X_t; X_{\neg t}|Q_t) + I(Q_t; Q_{<t}|X_{\neg t}) - I(X_t; X_{\neg t}).$$

Once again, the entropy of the observation probability models given the dependency variables affects only the first and the third terms. The second term may be ignored.

Taking $\mathcal{Z} = X_{\neg t}$, the problem of choosing dependency variables leads to the following:

Selection Rule 4.6. Maximize Global Posterior. *Choose $\mathcal{Z} \subset \mathcal{Z} = X_{\neg t}$, to maximize the quantity:*

$$I(X; \mathcal{Z}|Q) - I_{\mathcal{Z}}(X; \mathcal{Z}).$$

Like rule 4.5, this rule consists of the subtraction of an unconditional mutual information term from a conditional mutual information term. In this rule, however, the dependencies may be chosen from the past or the future, i.e., $\mathcal{Z} = X_{\neg t}$. This seemingly minor difference can cause significant complications because the resulting conditional independence assumptions could cause directed loops in the corresponding dependency graph. An accurate implementation would represent this as a Markov random field requiring a global normalization. This rule will not be explored in this work.[6]

[6]See, however, Chapter 6 which presents results using an implementation that, ignoring global normalizations, uses dependency variables from the future, the present, and the past.

4.5 Summary of Dependency Selection Rules

The preceding sections listed a variety of rules for selecting the variable set $\mathcal{Z}$ for use in the augmented observation model $p(X|Q, \mathcal{Z})$. These rules are summarized in this section. The first two items are included here just for completeness. All the rules assume some fixed upper limit on the number of entries in $\mathcal{Z}$.

- Choose some particular subset: $\mathcal{Z} \subset X_{<t}$.
- Choose a random subset: $\mathcal{Z} \subset X_{<t}$.
- Rule 4.1: Choose $\mathcal{Z}^* = \underset{\mathcal{Z}}{\operatorname{argmax}}\, I(X; \mathcal{Z}|Q)$ for $\mathcal{Z} \subset X_{<t}$.
- Rule 4.2: For each q, choose $\mathcal{Z}_q^* = \underset{\mathcal{Z}_q}{\operatorname{argmax}}\, I(X; \mathcal{Z}_q|Q = q)$ for $\mathcal{Z}_q \subset X_{<t}$.
- Rule 4.3: For each q, choose $\mathcal{Z}_q^* = \underset{\mathcal{Z}_q}{\operatorname{argmax}}\, U(X; \mathcal{Z}_q|q)$ for $\mathcal{Z}_q \subset X_{<t}$.
- Rule 4.4: Choose $\mathcal{Z}^* = \underset{\mathcal{Z}}{\operatorname{argmax}}\, S(X; \mathcal{Z}|Q)$ where $\mathcal{Z} = \{\mathcal{Z}_1 \cup \ldots \cup \mathcal{Z}_n\}$ and $\mathcal{Z}_q \subset X_{<t}$ for each q.
- Rule 4.5: Choose $\mathcal{Z}^* = \underset{\mathcal{Z}}{\operatorname{argmax}}\, I(X, \mathcal{Z}|Q) - I_{\mathcal{Z}}(X; \mathscr{Z})$ for $\mathcal{Z} \subseteq \mathscr{Z} = X_{<t}$.
- Rule 4.6: Choose $\mathcal{Z}^* = \underset{\mathcal{Z}}{\operatorname{argmax}}\, I(X, \mathcal{Z}|Q) - I_{\mathcal{Z}}(X; \mathscr{Z})$ for $\mathcal{Z} \subseteq \mathscr{Z} = X_{\neg t}$.

Some of these rules, along with the approximation algorithm described in the next section, will be evaluated in a complete automatic speech recognition Chapter 6.

4.6 Approximation Algorithms for Dependency Selection

The dependency selection algorithms presented in the previous sections involve the computation of mutual information between multi-dimensional vectors evaluated under multiple probabilistic contexts. Clearly, to compute such quantities directly would be computationally prohibitive. In this section, a tractable heuristic algorithm is developed for dependency selection. The heuristic uses sub-optimal greedy search strategies with restarts and intuitively arguable approximations to upper and lower bounds. Although several dependency rules were presented in Section 4.5, the heuristic is applicable to all of them — the resulting heuristic is such that a certain dependency selection rule is obtained by "plugging in" the appropriate parameters.

To start, dependencies are considered and added individually for each feature element of X. Let X_t^i be the i^{th} element of the vector X_t. For each i, dependency variables are chosen for X_t^i from the set $X_{<t}$ independently of the choices made for X_t^j where $j \neq i$. Assuming that the probability model for X_t includes intra-vector dependence (e.g., a full-covariance Gaussian, or Gaussian Mixture), this approximation at worst allows the addition of redundant variables because the information provided by a variable added for X_t^i might

indirectly have already been included via X_t^j and X_t^j's dependence with X_t^i. The above approximation reduces the problem to one of evaluating quantities such as $I(X_t^i; \mathcal{Z}|q)$ where X_t^i is a scalar random variable and $\mathcal{Z}$ is a vector random variable from X_t^i's context.

A second difficulty stems from evaluating mutual information between vectors rather than scalars. The chain rule of mutual information (Cover & Thomas 1991) can be stated as follows:

$$I(X_t^i; Z_{1:N}|Q) = \sum_j I(X_t^i; Z_j|Z_{1:(j-1)}, Q)$$

This suggests that the quantity $I(X_t^i; Z_{1:N}|Q)$ can be approximated by a greedy algorithm: choose Z_1 so that $I(X_t^i; Z_1|Q)$ is large, choose Z_2 so that $I(X_t^i; Z_2|Z_1, Q)$ is large, choose Z_3 so that $I(X_t^i; Z_3|Z_{1:2}, Q)$ is large, and so on. Because of this approximation, however, dependency selection choices made early will limit the set of possible choices available later — each selection that is chosen greedily based on the current criterion might not lead to the global optimum. Therefore, the selection order must be chosen intelligently (see below) and the heuristic will allow for restarts.

A third difficulty arises from the evaluation of the conditional mutual information $I(X_t^i; Z_j|Z_{1:j-1}, Q)$. This quantity captures the notion that a dependency variable should not be added if it contains only redundant information already provided by previously added variables. More formally, given Q, no variable Z_i, $i < j$ in the already chosen set of dependency variables $Z_{1:j}$ should have a Markov blanket (Pearl 1988) in $Z_{1:j}$ shielding it from X_t^i. To approximate a lower bound of this quantity, $I(X_t^i; Z_j|Z_{1:j-1}, Q)$ is considered large if both $I(X_t^i; Z_j|Q)$ is large and if $I(Z_j; Z_k|Q)$ is small for $k < j$. This is depicted in Figure 4.10 (in the figure, dependence on Q is assumed). In the figure, $Z_{1:3}$ have already been chosen and Z_4 is being evaluated as a candidate dependency variable. Z_4 will be chosen only if $I(X; Z_4|Q)$ is large, and if $I(Z_4; Z_j|Q)$ is small for $j \in \{1, 2, 3\}$. This heuristic can go wrong if the information provided by Z_4 about X is different in nature than the information provided by Z_j, $j < 4$ about X and if the variables Z_4 and Z_j, $j < 4$ are strongly dependent.

It will sometimes be necessary to approximate an upper bound to $I(X_t^i; Z_j|Z_{1:j-1}, Q)$ as well. The approximation is made that conditioning on previously selected dependency variables will not increase mutual information which is of course not true in general. Accordingly, the quantity $I(X_t^i; Z_j|Z_{1:j-1}, Q)$ is considered small if $I(X_t^i; Z_j|Q)$ is small.

Several of the dependency selection rules summarized in Section 4.5 require the computation of the cross-context conditional mutual information. Given the above approximations, this computation could be performed. For each element of X, for each candidate dependency variable, and for each q, this requires computing $|C_q|$ CCCMI univariate entities, where $|C_q|$ is the number of elements in the confusability class for q. This leads to a total of $|X||\mathcal{Z}|\sum_q |C_q|$ entities, where $|X|$ is the dimension of X, and where $|\mathcal{Z}|$ is the total possible number of dependency variables considered. This is not as bad as it sounds if mutual information is computed by first calculating the relevant distributions (requiring the majority of the computation), and then computing the KL-distance between the joint distribution and the product of the marginals (see Appendix C). In such a case, the distributions can be re-used. This requires computing $|X||\mathcal{Z}||Q|$ distributions, where $|Q|$ is the number of classes (this is needed anyway to compute conditional mutual information $I(X_t^i; Z|q)$ for

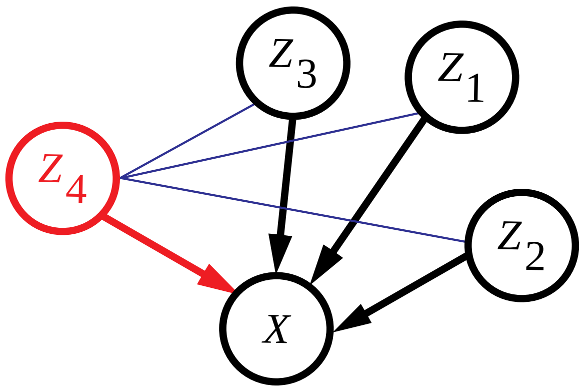

Figure 4.10: Approximating $I(X;Z_4|Z_1,Z_2,Z_3)$ using only $I(X;Z_4)$ and $I(Z_4;Z_j)$ for $j \in \{1,2,3\}$.

each i, $Z \in \mathcal{Z}$, and q). Using the resulting distributions, producing the CCCMI for each case requires computing $|X||\mathcal{Z}|\sum_q |C_q|$ additional univariate KL-distances.

As an alternative, the cross-context conditional mutual information $I_r(X_t^i;Z|q)$ for some scalar Z can be approximated using the conditional mutual information $I(X_t^i;Z|r)$, a "guess" at an upper bound. This can be argued by observing the difference between the two quantities:

$$
\begin{aligned}
&I(X;Z|r) - I_r(X;Z|q) \\
&= \int p(x,z|r)\log\frac{p(x,z|r)}{p(x|r)p(z|r)}dxdz - \int p(x,z|r)\log\frac{p(x,z|q)}{p(x|q)p(z|q)}dxdz \\
&= \int p(x,z|r)\log\frac{p(x,z|r)p(x|q)p(z|q)}{p(x,z|q)p(x|r)p(z|r)}dxdz \\
&= \int p(x,z|r)\log\frac{p(x|z,r)p(x|q)}{p(x|z,q)p(x|r)}dxdz \\
&= \int p(x,z|r)\log\frac{p(x|z,r)}{p(x|z,q)}dxdz - \int p(x,z|r)\log\frac{p(x|r)}{p(x|q)}dxdz \\
&= \int p(x,z|r)\log\frac{p(x|z,r)}{p(x|z,q)}dxdz - \int p(x|r)\log\frac{p(x|r)}{p(x|q)}dx \\
&= D(p(X|Z,r)||p(X|Z,q)) - D(p(X|r)||p(X|q))
\end{aligned}
$$

where $D(p_1||p_2)$ is the KL-distance between distributions p_1 and p_2. While there is no guarantee that this difference is non-negative, intuitively it can be argued that additionally conditioning on Z as in $D(p(X_t^i|Z,r)||p(X_t^i|Z,q))$ is not likely to decrease the KL-distance between $p(X_t^i|r)$ and $p(X_t^i|q)$. This is because, for $r \in C_q$, the quantity $D(p(X_t^i|r)||p(X_t^i|q))$ is small (confusable probability models should typically have a small KL-distance). Also, Z is chosen to highlight rather than suppress the differences between the distribution of X_t^i given q and the distribution of X_t^i given r. It is unlikely that such a chosen Z will result in a lower KL-distance, even if selected using $I(X_t^i;Z|r)$ which evaluates the entropy reduction of a different probabilistic model than does $I_r(X_t^i;Z|q)$. Therefore, the following relation is assumed typical for $r \in C_q$.

$$I(X_t^i; Z|r) \geq I_r(X_t^i; Z|q)$$

Choosing a Z that minimizes the cross-context conditional mutual information $I_r(X_t^i; Z|q)$ thus becomes choosing one that minimizes $I(X_t^i; Z|r)$. Using a liberal estimate for C_q (i.e., $\hat{C}_q \supseteq C_q$ as an estimate of C_q), results in a stronger constraint on the chosen Z. A liberal $\hat{C}_q$ potentially eliminates some useful dependencies, but any remaining dependencies will still be informative and discriminative for the confusable classes. C_q can therefore be approximated with a larger set, perhaps even the entire set of states (sans q). This approximation requires no additional computation beyond the calculation of the original $|X||\mathcal{Z}||Q|$ distributions.

In the preceding paragraphs, phrases such as "large enough" and "small enough" were used. In general, there is no way to tell how large or small a particular parameter should be. In order to achieve control of these parameters in different scenarios, percentiles are used to specify thresholds. For example, suppose there are M dependency variables $Z_k, k = 1 \ldots M$ to choose from. The condition "$I(X; Z_k)$ is large" is met by the condition $|\{j : I(X; Z_k) > I(X; Z_j)\}| > \tau M$ for some $0 < \tau < 1$ specified as an input parameter to the heuristic. The value of τ determines the significance required of a particular quantity. This condition will be denoted by $I(X; Z_k) > \pi_\tau$ for some τ.

```
INPUT: $q$, $i$, $\mathcal{Z}$, $M$, $S(\cdot)$, $\eta_u$, $\tau_u$, $\tau_q$, $\tau_c$, $\tau_g$, $C_q$
OUTPUT: $\mathbb{Z}$
Set $\mathbb{Z} = \emptyset$
Sort $Z_j \in \mathcal{Z}$ into an order decreasing by $S(\cdot)$
Repeat over $j$ until $S(Z_j) < \min(\pi_{\tau_u}, \eta_u)$ or $|\mathbb{Z}| = M$:
    If $Z_j$ satisfies all the following criteria:
        1) $I(X_t^i; Z_j|q) > \pi_{\tau_q}$
        2) For each $Z \in \mathbb{Z}$, $I(Z_j; Z|Q_t) < \tau_g I(Z_j; X_t^i|q)$
        3) $I(X_t^i; Z_j|C_q) < \pi_{\tau_c}$
    then add $Z_j$ to $\mathbb{Z}$.
```

Figure 4.11: Dependency selection heuristic: this algorithm chooses the dependency variables for the i^{th} feature position of X_t and for class q.

These approximations lead to the following algorithm for choosing $\mathbb{Z}_{qi}$ for each q and i shown in Figure 4.11. Again, the parameter $\mathcal{Z}$ is the entire set of possible dependency variables that may be considered. Typically, $\mathcal{Z}$ contains all of the scalars contained within $X_{t-m:t-1}$ for some $m > 1$. In some cases, $\mathcal{Z}$ might contain all the scalars within $\{X_{t-m:t+n}\} \setminus \{X_t^i\}$ for some $m \geq 0$ and $n \geq 0$ (see Section 4.4.4). In either case, m and n may also be considered input parameters that control the maximum temporal extent of the candidate dependency variables. The parameter M is the maximum number of dependency variables allowable for a particular element of X.

The function $S(\cdot)$ controls the greedy selection order for the variables $Z \in \mathcal{Z}$. Typically, $S(Z)$ will be an approximation of a scalar version of the utility (or the confusability refined utility) function. In other words, for a particular i and q,

$$S(Z) = \hat{U}(X_t^i; Z|q) = I(X_t^i; Z|q) - \frac{1}{K_q} \sum_{r \in C_q} p(r) I(X_t^i; Z|r)$$

which uses the CCCMI approximation mentioned above. A modification which uses the true CCCMI is:

$$S(Z) = U(X_t^i; Z|q) = I(X_t^i; Z|q) - \frac{1}{K_q} \sum_{r \in C_q} p(r) I_r(X_t^i; Z|q)$$

For rules 4.5 and 4.6, the selection order function becomes:

$$S(Z) = I(X_t^i; Z|q) - I(X_t^i; Z)$$

which produces the set of q specific dependencies. In any case, $S(\cdot)$ should select the best (i.e., most informative and discriminative) candidate dependency variables sooner rather than later.

The parameters η_u and τ_u place a lower bound on Z according to $S(\cdot)$. The parameter τ_u is a percentile from the set of values $S(Z_j)$ for all j. The parameter η_u is a hard threshold intended to disallow any Z for which $S(Z) < \eta_u$. For example, $\eta_u = 0$ would preclude the addition of any negative S-valued (i.e., anti-discriminative) variables regardless of the percentile threshold.

The candidate Z must also satisfy three criteria. Criterion one ensures that any added dependency provides a significant amount of information (determined by the threshold τ_q) to the current model not already provided by Q_t.

Criterion two is a redundancy check, and puts an upper bound on the amount of information a dependency variable may have about previously added dependency variables using the approximation mentioned above and described in Figure 4.10. The upper bound τ_g is a relative value determined by a fraction of the information between Z_j and X_t^i. Therefore, the upper bound can change for each Z_j.

Criterion three essentially places an upper bound τ_c on the prior-weighted entropy reduction of X_t^i by Z when evaluating the current model q in other potentially confusable contexts C_q. As listed in the figure, criterion three uses the approximation $I_r(X; Z|q) \leq I(X; Z|r)$ in the following:

$$I(X_t^i; Z_j|C_q) \triangleq \frac{\sum_{r \in C_q} p(r) I(X_t^i; Z_j|r)}{\sum_{r \in C_q} p(r)} \geq \frac{\sum_{r \in C_q} p(r) I_r(X_t^i; Z_j|q)}{\sum_{r \in C_q} p(r)}$$

If the true cross-context conditional mutual information is used, criterion three becomes:

$$\frac{\sum_{r \in C_q} p(r) I_r(X_t^i; Z_j|q)}{\sum_r p(r)} < \pi_{\tau_c}$$

The two previous cases approximate utility (Definition 4.2), where Z is chosen such that the q model using Z has only a small entropy reduction when evaluated in the r context, for $r \in C_q$. Since utility is an approximation of DCMI, these conditions also approximate selection rule 4.4. Finally, to approximate the posterior based selection (rules 4.6 and 4.6), criterion three becomes:

$$I(X_t^i; Z_j) < \pi_{\tau_c}$$

which chooses a Z that should in general not reduce the entropy of X.

The parameter C_q for each q defines the set of classes that could be confusable with class q. These clusters can be determined either using high-level knowledge about the class definitions (e.g., phonemes, syllables, etc.) or could be derived using likelihood scores as a similarity measure in an automatic clustering algorithm (Povey & Woodland 1999; Leggetter 1995; Odell 1995).

With this heuristic, it is possible to end up with fewer than N (or even zero) dependencies if no satisfying $Z \in \mathcal{Z}$ exists for the current thresholds. Also, there may be cases where some feature elements have many dependencies and others have none. Note that the algorithm requires only the computation of pairwise (i.e., scalar) mutual information, conditional mutual information, or perhaps cross-context conditional mutual information for a given labeling scheme. In chapter 6, these heuristics are evaluated in a complete speech recognition system.

4.7 Buried Markov Models

The previous sections describe methods that can add statistical dependencies to an HMM. The results are called Buried Markov Models[7] (BMMs) because the hidden Markov chain in an HMM is further hidden (or buried) by specific cross-observation dependencies whose placement is determined via statistical measurements made on signals representing natural objects.

Like an HMM, a BMM can be viewed as a graphical model (Lauritzen 1996). Recall Figure 3.4 that showed the graphical model representation of an HMM. Figure 4.12 shows a similar diagram for a BMM. Relative to the HMM plot, the graph has been augmented with cross-observation dependency edges between individual observation elements. The resulting graph therefore depicts the conditional independence assumptions between individual observation elements made by a BMM.

The figure shows those dependencies only for a particular assignment to the hidden variables. A different hidden Markov chain assignment will result in different $\mathcal{Z}_q$'s at each time and therefore a different set of cross-observation dependencies. It is more precise therefore to categorize a BMM as a Bayesian multinet (Geiger & Heckerman 1996), or since there is a temporal element to the graph, perhaps a "dynamic Bayesian multinet". It is possible, of course, to use a graphical model to describe the dependencies under all hidden-variable assignments either using deterministic nodes as multiplexors or by drawing arcs for the dependency edges possible under all hidden state assignments. In this latter case, it is assumed that the parameters (e.g., zeros) of the model can cause the observation variables to ignore certain dependency variables for certain hidden state values. In any

[7]Other names considered include stealth Markov models, interred Markov models, natural Markov models, ecologically covered Markov models, vital statistics Markov models, dead and buried Markov models, hidden Markov super-models, etc.

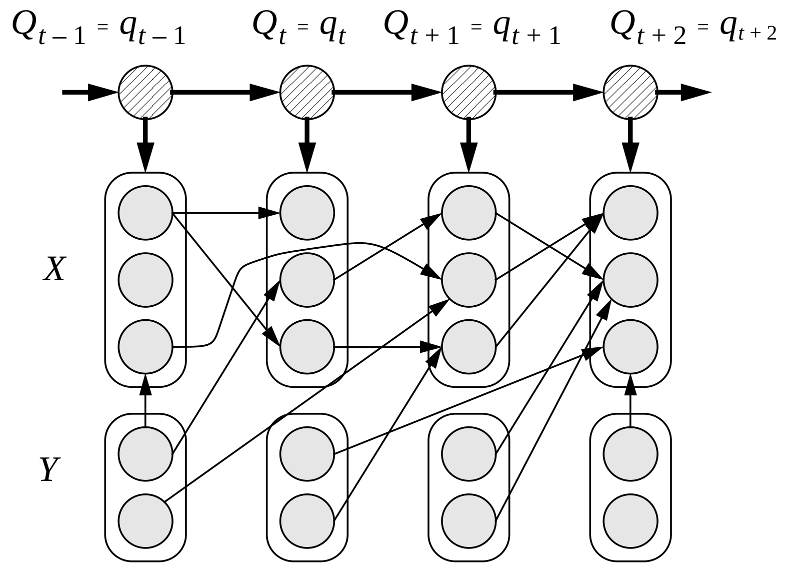

Figure 4.12: Graphical Model view of a BMM

case, such depictions are less desirable because the graphs very quickly become unwieldy and because they do not depict the various conditional independence properties for different hidden Markov chain values.

4.7.1 BMM Structure Learning

Viewing a BMM as a graphical model, it can be seen that the procedures described in previous sections essentially learn the structure of the model in terms of selecting new dependency edges. Bayesian network structure learning methods (Heckerman *et al.* 1994; Heckerman 1995; Friedman 1998; Chow & Liu 1968; Meila 1999; Sahami 1996) were described in Chapter 2. Most of the methods investigated in the literature so far, to the extent of this author's knowledge, concentrate on changing the structure of a network to better model the true class-dependent data distribution. As mentioned before, several of the BMM structure learning rules concentrate on increasing the model's structural discriminability — i.e., for each class, the goal is to adjust the corresponding model so that it better represents only the unique and natural statistical dependencies for each class. Many structure learning methods typically learn only the likelihood-maximizing structure.

While the BMM structure learning algorithms can be discriminative, there are still two training choices: likelihood based and discriminative. As argued in Section 3.3.10, a discriminatively structured model will represent only those statistical properties unique to its class. When such a model is evaluated in a different context with different statistical properties, the model would produce a lower score. While model's parameters might not have been trained to produce a lower score, its structure is simply incapable of representing the typical properties of rival classes. When trained using a maximum likelihood based scheme, the result could be called a discriminatively structured but non-discriminatively trained system. Chapter 5's presents an EM algorithm for maximizing the likelihood of

BMMs.

Discriminative training methods such as MMI (Bahl *et al.* 1986), MDI (Ephraim & Rabiner 1988; Ephraim *et al.* 1989), and MCE (Juang & Katagiri 1992; Juang *et al.* 1997) can also be used to train a discriminatively structured model such as a BMM. The result could be called a discriminatively structured and trained system.

Table 4.1 lists a few examples of different discriminative or maximum-likelihood based parameter or structure training procedures. Learning Bayesian networks is described in (Heckerman *et al.* 1994; Heckerman 1995). The structural EM algorithm is described in (Friedman 1998). Hybrid ANN/HMM systems are described in (Bourlard & Morgan 1994). The REMAP algorithm is described in Konig (1996). The various forms of HMM extensions (AR-HMM, Segmental HMM, Trended HMM, etc) were described in Section 3.5 and are summarized in Ostendorf *et al.* (1996).

	Discriminative Training	Distribution Estimation
Discriminitive Structure Determination	o BMM, rules 4.3-4.6 with MCE/MMI training.	o BMM, rules 4.3-4.6 with ML training.
Distribution Structure Determination	o BMM, rules 4.1,4.2 with MCE/MMI/MDI training	o BMM, rules 4.1,4.2 with ML training o Learning Baysian Networks o Structural EM
By-hand Structure Selection	o Hybrid ANN/HMM o REMAP o MCE/MMI/MDI trained: HMM, AR-HMM, Segmental HMM, Trended HMM, etc.	o ML Trained: HMM, AR-HMM Segmental HMM, Trended HMM, etc. o Other likelihood-based training.

Table 4.1: Examples of different parameter value and structure selection methods.

4.7.2 BMM Instantaneous Speaker Adaptation

The BMM graphical model in Figure 4.12 makes a distinction between two separate observation vector streams $X_{1:T}$ and $Y_{1:T}$. For an HMM,

$$p(X_{1:T}, Y_{1:T}) = \sum_{q_{1:T}} \prod_t p(X_t, Y_t | q_t) p(q_t | q_{t-1}).$$

and $p(X_t, Y_t|q_t) = p(X_t|Y_t, q_t)p(Y_t|q_t)$. In Figure 4.12, it is assumed that Y_t consists of a set of variables that are marginally independent of q_t but are not independent of q_t given X_t. This is the "explaining away" phenomena described in Section 2.

Modeling a relationship between Y_t and q_t when they are independent is at best superfluous and at worst detrimental. At best the addition of the irrelevant dependency $p(Y_t|q_t)$ to the model will have no affect on error performance, but it will require more memory and computation. At worst, including a dependency between two variables that are independent could lead to a larger demand on training data size, and the possibility of over-training would increase. The extraneous dependency and its parameters might interfere with the proper training of the remaining parameters. This is essentially a bias-variance issue (Bishop 1995). If Y_t and q_t are indeed independent, modeling a relationship between the two for a fixed training set size would cause the variance of the classifier to increase (i.e., the relationship potentially could allow the representation of idiosyncratic relationships in the data) while the bias would not decrease ($p(Y_t|q_t)$ adds no new information to help classification when Y_t and q_t are independent).

In automatic speech recognition, "speaker adaptation" refers to a method whereby the parameters of a previously trained system are updated with information obtained from a new speaker. For example, in (Leggetter & Woodland 1995; Leggetter 1995), the means of a Gaussian mixture HMM are re-trained with a form of EM using new speaker data. It has been found that adapting to a new speaker, essentially turning a speaker independent system into a speaker dependent one, reduces word error.

With the BMM described in the figure, $p(Y_t|q_t)$ is ignored but the dependency variables $\mathcal{Z}$ may consist of elements from both the X and Y streams. When using BMMs for speech recognition, X_t could be standard speech features that are known to convey information about the hidden state (e.g., MFCCs, LPC, etc.) and Y_t could be acoustic features that provide information about the current speaker. Such features could, for example, include higher-order cepstral coefficients (Rabiner & Juang 1993), or estimates of vocal tract length, gender, speaking rate, noise condition, etc. These types of features will influence the statistics of the acoustic features X_t but will have little if any dependence with the hidden state (e.g., phoneme, syllable, etc.). The BMM structure learning procedure can be used to decide which features are useful for each hidden state. The resulting system can be thought of as an instantaneous speaker adaptation procedure.

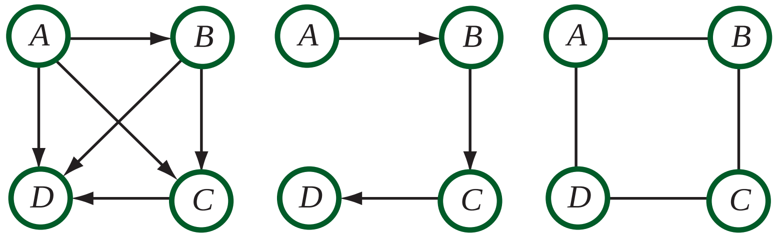

Figure 4.13: Decomposition of joint probability.

4.7.3 BMMs and MRFs

In section 4.4.4, it was suggested that dependencies could be added not just from the past but from the future as well. In other words, the set $\mathcal{Z}$ could take values from $X_{\neg t}$ rather than just $X_{<t}$. While this is possible, the danger is that the resulting dependencies

could create a directed cycle in the corresponding graph. The problem with directed cycles can easily be seen from the following simple example. Given four random variables, A, B, C, and D, the chain rule of probability allows the joint distribution to be written as

$$p(A,B,C,D) = p(A)p(B|A)p(C|A,B)p(D|A,B,C),$$

shown graphically on the left in Figure 4.13. If it is (validly) assumed that C is independent of A given B, and that D is independent of A and B given C, then we get

$$p(A,B,C,D) = p(A)p(B|A)p(C|B)p(D|C)$$

shown in Figure 4.13 in the center. This decomposition is still a valid probability distribution. If it was decided that A should also somehow depend on D, then

$$p(A,B,C,D) \neq p(A|D)p(B|A)p(C|B)p(D|C)$$

because the resulting product of conditional probabilities do not necessarily integrate to unity. In such a case, one must use a product of potential functions and globally normalize to re-obtain a valid distribution, as in:

$$p(A,B,C,D) = \frac{\phi_{A,D}(A,D)\phi_{B,A}(B,A)\phi_{C,B}(C,B)\phi_{D,C}(D,C)}{\Phi_{A,B,C,D}}$$

where $\Phi_{A,B,C,D}$ is a normalization constant. The resulting distribution becomes a Markov random field (e.g., the right in Figure 4.13) and the problem inherits all the associated complexities due to non-decomposable graphs (Lauritzen 1996).

There are several ways to overcome such difficulties with BMMs. First, dependencies could be allowed only from the past as was done for most of the selection rules and as is depicted in Figure 4.12. Second, since a BMM is a Bayesian multinet, as long as there are no directed cycles for any possible assignment to the hidden Markov chain, then the normalization problem does not arise. Extending the simple example above, suppose there is a 0/1-valued switching variable S. A valid decomposition of the joint probability of A,B,C, and D could be the mixture:

$$\begin{aligned}
&p(A,B,C,D)\\
&= \sum_s p(A,B,C,D,S=s)\\
&= p(A|D,S=0)p(B|A,S=0)p(C|B,S=0)p(D|S=0)p(S=0)\\
&\quad +p(A|S=1)p(B|A,S=1)p(C|B,S=1)p(D|C,S=1)p(S=1)
\end{aligned}$$

In this form, a direct dependence between A and D is represented when $S=0$ but not when $S=1$. For an BMM, the joint probability could be represented as:

$$p(X_{1:T}) = \sum_q \prod_t p(X_t|\text{parents}(q,t,X_t),q_t)p(q_t|q_{t-1})$$

where the parents() function is such that for a given value of q (i.e., hidden Markov chain assignment), no directed cycles result in the corresponding graph. An example is shown in

Figure 4.14. Ensuring that this condition holds, however, can be difficult because the existence of directed loops depend on the intra-vector dependencies implicit in the observation models $p(X_t|\cdot)$.

An alternative approach is to allow loops, and use a form of loopy belief propagation, an approximate inference procedure (Weiss Submitted) that tends to work well in practice. Finally, arbitrary dependencies could be allowed resulting in a MRF version of an BMM and necessitating a global normalization distribution of the form:

$$p(x_{1:T}) = \frac{\sum_q \prod_t p(x_t|z_t, q_t)p(q_t|q_{t-1})}{\int \sum_q \prod_t p(x_t|z_t, q_t)p(q_t|q_{t-1})dx_{1:T}}.$$

In this work, for simplicity, only the first case is considered. One case in Chapter 6, however, evaluates a form of a single-iteration loopy propagation where $\mathcal{Z}$ may contain variables from the future.

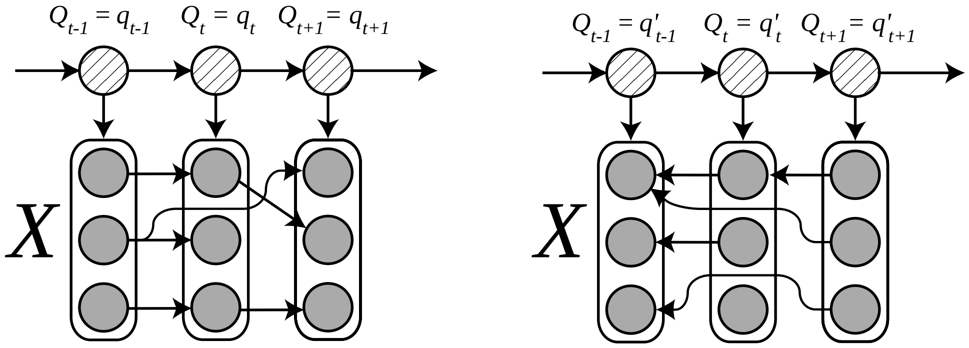

Figure 4.14: BMM dependencies depend on the value of the hidden variables.

Providing yet another alternative, the heuristic dependency selection algorithm described in Section 4.6 chooses dependencies individually for each element of X_t. This allows the possibility that a dependence variable could be chosen from the same time frame. For example, the dependencies for X_t^i could include X_t^j where $j \neq i$. If a full covariance Gaussian distribution is used for the observation models, such dependencies would be superfluous. With diagonal covariance or a mixture of diagonal covariance Gaussians, choosing such dependencies could enrich the model if appropriate normalization is performed. In this case, however, only a local normalization is required. These sparse covariance matrices are discussed further in Chapter 7.

4.7.4 BMM Complexity

In general, adding conditional dependencies in a DBN can significantly increase the complexity for probabilistic inference and learning. In fact, for the junction tree algorithm, the complexity (Smyth *et al.* 1996; Pearl 1988) is $O(\sum_{i=1}^{T} s(C_i))$ where T is the number of resultant cliques in the junction tree, and $s(C_i)$ is the size of the state space for clique C_i. For an HMM with T time-steps and N states, there are $O(T)$ cliques each with at most a state space size of N^2 resulting in the $O(TN^2)$ complexity we saw above. For a corresponding BMM, there are also $O(T)$ cliques, but because we are only adding conditional dependencies between observations, they do not increase the state-space size of the resulting cliques. There is, however, a constant cost associated with the number of additional dependency edges. The BMM complexity therefore is $O(TN^2M)$ where M is the maximum number of dependency edges per time frame. In a BMM, the extra dependency structure is sparse and can change for each hidden state value at each time. Therefore, M is a loose upper bound. In the average case, the computational and memory requirements of a BMM will be less than the $O(TN^2M)$ complexity figure implies. Even in the worst case, however, a BMM only adds a factor of M in complexity over an HMM, but potentially with significantly enhanced modeling power. And the complexity grows only linearly with M.

4.8 Discussion

This chapter has introduced a variety of information-theoretic discriminative criteria that can be used to extend an HMM's statistical dependencies. The method can seen as a way to produce structurally discriminative models. Up to now, a BMM implementation has not been described. In Chapter 5, Gaussian mixture HMMs are extended to allow for the dependencies specified by a BMM. The result will be called Gaussian mixture BMMs. This implementation is tested on speech corpora as reported in Chapter 6.

Chapter 5

Switching Gaussian Mixtures for BMMs

Contents

In Chapter 4, a method was presented that can augment the underlying statistical dependencies of an HMM but a specific implementation was not given. In this chapter, an new method is proposed for this purpose. The method is a generalization both of Gaussian mixture HMMs and auto-regressive (or conditionally Gaussian) HMMs. It allows the dependency variables to affect the underlying distribution in significant ways yet has relatively efficient EM update equations for maximum-likelihood parameter estimation.

In this chapter, maximum likelihood parameter estimation and the EM algorithm are reviewed first. Then, EM update equations are derived for Gaussian mixtures and for Gaussian mixture HMMs (i.e., the Baum-Welch algorithm).[1] Then *switching Gaussian*

[1]These three sections are a shortened version of Bilmes (1997).

mixtures, the method that implements BMM dependencies, are defined and new EM update equations are provided.

5.1 Maximum-likelihood parameter estimation

The maximum likelihood procedure for parameter estimation attempts to estimate the parameters values of a model that most accurately explain a given sample of data. There is some model density function $p(x|\Theta)$ that is governed by the set of parameters Θ (e.g., p could be a Gaussian and Θ could be the mean and covariance of that Gaussian). There is also a sample of data $\mathbf{x}$ of size N, supposedly drawn i.i.d. from the underlying true distribution where $\mathbf{x} = \{x_1, \ldots, x_N\}$. The probability of the data under the model is given by:

$$p(\mathbf{x}|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta) = \mathcal{L}(\Theta|\mathbf{X})$$

This function $\mathcal{L}(\Theta|\mathbf{x})$ is referred to as the likelihood of the parameters Θ given the data sample $\mathbf{x}$ (or just the likelihood function). The likelihood is viewed as a function of the parameters Θ for a fixed data set. The maximum likelihood problem is to find the Θ that maximizes $\mathcal{L}$, i.e.:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}}\ \mathcal{L}(\Theta|\mathbf{x}) \underset{\Theta}{\operatorname{argmax}}\ \log \mathcal{L}(\Theta|\mathbf{x})$$

The difficulty of this problem depends on the form of $p(x|\Theta)$. If, for example, $p(x|\Theta)$ is a single component Gaussian distribution and $\Theta = (\mu, \sigma^2)$, then setting the derivative of $\log(\mathcal{L}(\Theta|\mathbf{x}))$ to zero will result in analytical equations for μ and σ^2. For many problems it is not possible to find such analytical expressions and an iterative method such as EM or gradient descent (Bishop 1995) must be used.

5.2 The EM Algorithm

The EM algorithm (Dempster *et al.* 1977; Redner & Walker 1984; Ghahramami & Jordan 1995; Jordan & Jacobs 1994; Bishop 1995; Wu 1983; McLachlan & Krishnan 1997) can be used to find a maximum-likelihood parameter estimate for a model given a data set that is incomplete or has missing values.

There are two main applications of the EM algorithm. The first occurs when the data indeed lacks certain values due to problems with or limitations of the observation process. The second occurs when optimizing the original likelihood function is analytically intractable but the function can be simplified by assuming the existence of and values for additional but missing (or hidden) parameters. The latter application is more common in the computational pattern recognition community.

The random data $\mathbf{X}$ is assumed to be incomplete, and the complete data $\mathbf{V} = (\mathbf{X}, \mathbf{W})$, where $\mathbf{W}$ are the hidden or missing values, is described by the complete joint density function:

$$p(\mathbf{V}|\Theta) = p(\mathbf{X}, \mathbf{W}|\Theta)$$

With this new density function, a new likelihood function can be defined, $\mathcal{L}(\Theta|\mathbf{V}) = \mathcal{L}(\Theta|\mathbf{X},\mathbf{W}) = p(\mathbf{X},\mathbf{W}|\Theta)$, called the complete-data likelihood. Note that the complete-data likelihood is a true random variable because the missing information $\mathbf{W}$ is unknown, random, and presumably governed by an underlying distribution. The original likelihood $\mathcal{L}(\Theta|\mathbf{X})$ is referred to as the incomplete-data likelihood function.

The EM algorithm first finds the expected value of the complete-data log-likelihood $\log p(\mathbf{x},\mathbf{W}|\Theta)$ with respect to the unknown data $\mathbf{W}$ given the observed data $\mathbf{x}$ and some current parameter estimates governing the distribution of $\mathbf{W}$. An auxiliary function is defined as:

$$Q(\Theta,\Theta^{(i-1)}) = E\left[\log p(\mathbf{x},\mathbf{W}|\Theta)|\mathbf{x},\Theta^{(i-1)}\right] \tag{5.1}$$

where $\Theta^{(i-1)}$ is the current or guessed parameter estimate and is used to evaluate the expectation. The quantity Θ comprises the new parameters that are optimized to increase Q.

The right-hand side of Equation 5.1 can be written as:

$$E\left[\log p(\mathbf{x},\mathbf{W}|\Theta)|\mathbf{x},\Theta^{(i-1)}\right] = \int_{\mathbf{w}\in\Upsilon} \log p(\mathbf{x},\mathbf{w}|\Theta) f(\mathbf{w}|\mathbf{x},\Theta^{(i-1)})d\mathbf{w}. \tag{5.2}$$

The function $f(\mathbf{w}|\mathbf{x},\Theta^{(i-1)})$ is the marginal distribution of the unobserved data and is dependent on both the observed data $\mathbf{x}$ and on the current parameters, and Υ is the space of values $\mathbf{W}$ can take on. In the best of cases, this marginal distribution is a simple analytical expression of the assumed parameters $\Theta^{(i-1)}$ and perhaps the data. In the worst of cases, this density might be very hard to obtain. The evaluation of this expectation is called the E-step of the algorithm.

The second step (the M-step) of the EM algorithm is to maximize the expectation computed in the first step:

$$\Theta^{(i)} = \underset{\Theta}{\operatorname{argmax}}\ Q(\Theta,\Theta^{(i-1)}).$$

These two steps are repeated until convergence is achieved. Each iteration is guaranteed to increase the log-likelihood and the algorithm is guaranteed to converge to a local maximum of the likelihood function. The rate at which the likelihood converges to the true likelihood is discussed in Dempster *et al.* (1977); Redner & Walker (1984); Wu (1983); Jordan & Xu (1996); Xu & Jordan (1996); McLachlan & Krishnan (1997).

Instead of maximizing $Q(\Theta,\Theta^{(i-1)})$, an alternative form of the M-step is to find some $\Theta^{(i)}$ such that $Q(\Theta^{(i)},\Theta^{(i-1)}) > Q(\Theta,\Theta^{(i-1)})$. This form of the algorithm is called Generalized EM (GEM) and is also guaranteed to converge.

5.3 The EM Algorithm for Gaussian Mixture Parameter Estimation

The parameter estimation problem for a finite mixture of densities (Titterington *et al.* 1985) is probably one of the most widely used applications of the EM algorithm in the computational pattern recognition community. In this case, the following probabilistic model is assumed:

$$p(\mathbf{x}|\Theta) = \sum_{m=1}^{M} \alpha_m p_m(\mathbf{x}|\theta_m)$$

where the parameters are $\Theta = (\alpha_1, \ldots, \alpha_M, \theta_1, \ldots, \theta_M)$ such that $\sum_{m=1}^{M} \alpha_m = 1$ and each p_m is a density function governed by the parameters θ_m. There are M component densities mixed together with M mixing coefficients α_m. The graphical model for a finite mixture model is depicted in Figure 2.6.

The incomplete-data log-likelihood expression for this density is given by:

$$\log(\mathcal{L}(\Theta|\mathbf{x})) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} \log \left(\sum_{m=1}^{M} \alpha_m p_m(x_i|\theta_m) \right)$$

which is difficult to optimize directly because it contains the log of the sum. If the unobserved data $w_{1:N}$ identifies which component density "generated" each data item, the complete-date likelihood expression is much simpler. In this case, $w_i \in \{1, \ldots, M\}$ for each i, and $w_i = m$ only if the i^{th} sample was generated by the m^{th} mixture component. Given values for the unobserved data, the complete-data log likelihood becomes:

$$\log(\mathcal{L}(\Theta|\mathbf{x}, \mathbf{W})) = \sum_{i=1}^{N} \log \left(P(x_i|w_i)P(w_i)\right) = \sum_{i=1}^{N} \log \left(\alpha_{w_i} p_{w_i}(x_i|\theta_{x_i})\right).$$

The distribution of the unobserved data given the observed data and some guessed or current parameters can be obtained as follows. Using $\Theta^g = (\alpha_1^g, \ldots, \alpha_M^g, \theta_1^g, \ldots, \theta_M^g)$ as a guess of the parameter values, the distribution for the unobserved portion of the i^{th} sample can be specified using Bayes rule as:

$$p(w_i|x_i, \Theta^g) = \frac{\alpha_{w_i}^g p_{w_i}(x_i|\theta_{w_i}^g)}{p(x_i|\Theta^g)} = \frac{\alpha_{w_i}^g p_{w_i}(x_i|\theta_{w_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(x_i|\theta_k^g)}$$

and the distribution of the complete set of unobserved variables $\mathbf{w} = (w_1, \ldots, w_N)$ can be specified using the fact that the samples are i.i.d.:

$$p(\mathbf{w}|\mathbf{x}, \Theta^g) = \prod_{i=1}^{N} p(w_i|x_i, \Theta^g)$$

Equation 5.1 now takes the form:

$$
\begin{aligned}
Q(\Theta, \Theta^g) &= \sum_{\mathbf{w}\in\Upsilon} \log\left(\mathcal{L}(\Theta|\mathbf{x},\mathbf{w})\right) p(\mathbf{w}|\mathbf{x},\Theta^g) \\
&= \sum_{\mathbf{y}\in\Upsilon} \sum_{i=1}^{N} \log\left(\alpha_{w_i} p_{w_i}(x_i|\theta_{w_i})\right) \prod_{j=1}^{N} p(w_j|x_j,\Theta^g) \\
&= \sum_{w_1=1}^{M} \sum_{w_2=1}^{M} \cdots \sum_{w_N=1}^{M} \sum_{i=1}^{N} \log\left(\alpha_{w_i} p_{w_i}(x_i|\theta_{w_i})\right) \prod_{j=1}^{N} p(w_j|x_j,\Theta^g) \\
&= \sum_{w_1=1}^{M} \sum_{w_2=1}^{M} \cdots \sum_{w_N=1}^{M} \sum_{i=1}^{N} \sum_{m=1}^{M} \delta_{m,w_i} \log\left(\alpha_m p_m(x_i|\theta_m)\right) \prod_{j=1}^{N} p(w_j|x_j,\Theta^g) \\
&= \sum_{m=1}^{M} \sum_{i=1}^{N} \log\left(\alpha_m p_m(x_i|\theta_m)\right) \sum_{w_1=1}^{M} \sum_{w_2=1}^{M} \cdots \sum_{w_N=1}^{M} \delta_{m,w_i} \prod_{j=1}^{N} p(w_j|x_j,\Theta^g) \qquad (5.3)
\end{aligned}
$$

This equation can be greatly simplified by noting that for $m \in 1, \ldots, M$,

$$
\begin{aligned}
&\sum_{w_1=1}^{M} \sum_{w_2=1}^{M} \cdots \sum_{w_N=1}^{M} \delta_{m,w_i} \prod_{j=1}^{N} p(w_j|x_j,\Theta^g) \\
&= \left(\sum_{w_1=1}^{M} \cdots \sum_{w_{i-1}=1}^{M} \sum_{w_{i+1}=1}^{M} \cdots \sum_{w_N=1}^{M} \prod_{j=1, j\neq i}^{N} p(w_j|x_j,\Theta^g) \right) p(m|x_i,\Theta^g) \\
&= \prod_{j=1,j\neq i}^{N} \left(\sum_{w_j=1}^{M} p(w_j|x_j,\Theta^g) \right) p(m|x_i,\Theta^g) = p(m|x_i,\Theta^g) \qquad (5.4)
\end{aligned}
$$

since $\sum_{m=1}^{M} p(m|x_j,\Theta^g) = 1$. Using Equation 5.4, Equation 5.3 can be written as:

$$
\begin{aligned}
Q(\Theta, \Theta^g) &= \sum_{m=1}^{M} \sum_{i=1}^{N} \log\left(\alpha_m p_m(x_i|\theta_m)\right) p(m|x_i,\Theta^g) \\
&= \sum_{m=1}^{M} \sum_{i=1}^{N} \log(\alpha_m) p(m|x_i,\Theta^g) + \sum_{m=1}^{M} \sum_{i=1}^{N} \log(p_m(x_i|\theta_m)) p(m|x_i,\Theta^g) \qquad (5.5)
\end{aligned}
$$

To maximize this expression, the term containing α_m and the term containing θ_m can be maximized independently since they are not related.

To find the expression for α_m, the Lagrange multiplier λ is introduced, with the constraint that $\sum_m \alpha_m = 1$, in the following equation:

$$
\frac{\partial}{\partial \alpha_m} \left[\sum_{m=1}^{M} \sum_{i=1}^{N} \log(\alpha_m) p(m|x_i,\Theta^g) + \lambda \left(\sum_m \alpha_m - 1 \right) \right] = 0
$$

or

$$
\sum_{i=1}^{N} \frac{1}{\alpha_m} p(m|x_i,\Theta^g) + \lambda = 0
$$

Summing both sizes over m, we get that $\lambda = -N$ resulting in:

$$\alpha_m = \frac{1}{N}\sum_{i=1}^{N} p(m|x_i, \Theta^g)$$

For certain component densities, it is possible to get an analytical EM update expressions for θ_m as functions of everything else. For example, assuming a d-dimensional Gaussian component distributions with mean μ and covariance matrix Σ, i.e., $\theta = (\mu, \Sigma)$, yields

$$p_m(x|\mu_m, \Sigma_m) = \frac{1}{(2\pi)^{d/2}|\Sigma_m|^{1/2}} e^{-\frac{1}{2}(x-\mu_m)^T\Sigma_m^{-1}(x-\mu_m)}. \tag{5.6}$$

To derive the update equations for this distribution, some results from matrix algebra (Mardia *et al.* 1979; Harville 1997; Strang 1988) must be used.

The trace of a square matrix $\mathrm{tr}(A)$ is equal to the sum of A's diagonal elements. The trace of a scalar equals that scalar. Also, $\mathrm{tr}(A + B) = \mathrm{tr}(A) + \mathrm{tr}(B)$, and $\mathrm{tr}(AB) = \mathrm{tr}(BA)$. These properties imply that $\sum_i x_i^T A x_i = \mathrm{tr}(AB)$ where $B = \sum_i x_i x_i^T$. The quantity $|A|$ indicates the determinant of the matrix A, and the determinant of the inverse matrix satisfies $|A^{-1}| = 1/|A|$.

For notational simplicity, derivatives of a function of a matrix $f(A)$ with respect to multiple elements of that matrix are defined. In particular, define $\frac{\partial f(A)}{\partial A}$ to be the matrix with i, j^{th} entry $[\frac{\partial f(A)}{\partial a_{i,j}}]$ where $a_{i,j}$ is the i, j^{th} entry of A. The definition also applies to taking derivatives with respect to a vector. Using this notation, several derivative results for common matrix operations may be obtained. First, $\frac{\partial x^T A x}{\partial x} = (A + A^T)x$. Second, when A is a symmetric matrix:

$$\frac{\partial |A|}{\partial a_{i,j}} = \begin{cases} \mathcal{A}_{i,j} & \text{if } i = j \\ 2\mathcal{A}_{i,j} & \text{if } i \neq j \end{cases}$$

where $\mathcal{A}_{i,j}$ is the i, j^{th} cofactor of A. Given the above:

$$\frac{\partial \log |A|}{\partial A} = \left\{ \begin{array}{ll} \mathcal{A}_{i,j}/|A| & \text{if } i = j \\ 2\mathcal{A}_{i,j}/|A| & \text{if } i \neq j \end{array} \right\} = 2A^{-1} - \mathrm{diag}(A^{-1})$$

by the definition of the inverse of a matrix. Finally,

$$\frac{\partial \mathrm{tr}(AB)}{\partial A} = B + B^T - \mathrm{Diag}(B).$$

Taking the log of Equation 5.6, ignoring any constant terms (since they disappear after taking derivatives), and substituting into the right side of Equation 5.5, yields:

$$\begin{aligned} &\sum_{m=1}^{M}\sum_{i=1}^{N} \log\left(p_m(x_i|\mu_m, \Sigma_m)\right) p(m|x_i, \Theta^g) \\ &= \sum_{m=1}^{M}\sum_{i=1}^{N} \left(-\frac{1}{2}\log(|\Sigma_m|) - \frac{1}{2}(x_i - \mu_m)^T \Sigma_m^{-1}(x_i - \mu_m)\right) p(m|x_i, \Theta^g) \end{aligned} \tag{5.7}$$

Taking the derivative of Equation 5.7 with respect to μ_m and setting it equal to zero, yields:

$$\sum_{i=1}^{N} \Sigma_m^{-1}(x_i - \mu_m) p(m|x_i, \Theta^g) = 0$$

with can be solved for μ_m to obtain:

$$\mu_m = \frac{\sum_{i=1}^{N} x_i p(m|x_i, \Theta^g)}{\sum_{i=1}^{N} p(m|x_i, \Theta^g)}.$$

To find Σ_m, note that Equation 5.7 can be written as:

$$\begin{aligned}
&\sum_{m=1}^{M} \left[\frac{1}{2} \log(|\Sigma_m^{-1}|) \sum_{i=1}^{N} p(m|x_i, \Theta^g) - \frac{1}{2} \sum_{i=1}^{N} p(m|x_i, \Theta^g) \mathrm{tr}\left(\Sigma_m^{-1} (x_i - \mu_m)(x_i - \mu_m)^T \right) \right] \\
&= \sum_{m=1}^{M} \left[\frac{1}{2} \log(|\Sigma_m^{-1}|) \sum_{i=1}^{N} p(m|x_i, \Theta^g) - \frac{1}{2} \sum_{i=1}^{N} p(m|x_i, \Theta^g) \mathrm{tr}\left(\Sigma_m^{-1} A_{m,i} \right) \right]
\end{aligned}$$

where $A_{m,i} = (x_i - \mu_m)(x_i - \mu_m)^T$.

Taking the derivative with respect to Σ_m^{-1} yields:

$$\begin{aligned}
&\frac{1}{2} \sum_{i=1}^{N} p(m|x_i, \Theta^g) \left(2\Sigma_m - \mathrm{diag}(\Sigma_m) \right) - \frac{1}{2} \sum_{i=1}^{N} p(m|x_i, \Theta^g) \left(2A_{m,i} - \mathrm{diag}(A_{m,i}) \right) \\
&= \frac{1}{2} \sum_{i=1}^{N} p(m|x_i, \Theta^g) \left(2B_{m,i} - \mathrm{diag}(B_{m,i}) \right) \\
&= 2C - \mathrm{diag}(C)
\end{aligned}$$

where $B_{m,i} = \Sigma_m - A_{m,i}$ and where $C = \frac{1}{2} \sum_{i=1}^{N} p(m|x_i, \Theta^g) B_{m,i}$. Setting this derivative to zero, i.e., $2C - \mathrm{diag}(C) = 0$, implies that $C = 0$. This gives

$$\sum_{i=1}^{N} p(m|x_i, \Theta^g) \left(\Sigma_m - A_{m,i} \right) = 0$$

or

$$\Sigma_m = \frac{\sum_{i=1}^{N} p(m|x_i, \Theta^g) A_{m,i}}{\sum_{i=1}^{N} p(m|x_i, \Theta^g)} = \frac{\sum_{i=1}^{N} p(m|x_i, \Theta^g)(x_i - \mu_m)(x_i - \mu_m)^T}{\sum_{i=1}^{N} p(m|x_i, \Theta^g)}$$

Summarizing, the estimates of the new parameters in terms of the old parameters are as follows:

$$\alpha_m^{new} = \frac{1}{N} \sum_{i=1}^{N} p(m|x_i, \Theta^g)$$

$$\mu_m^{new} = \frac{\sum_{i=1}^N x_i p(m|x_i, \Theta^g)}{\sum_{i=1}^N p(m|x_i, \Theta^g)}$$

$$\Sigma_m^{new} = \frac{\sum_{i=1}^N p(m|x_i, \Theta^g)(x_i - \mu_m^{new})(x_i - \mu_m^{new})^T}{\sum_{i=1}^N p(m|x_i, \Theta^g)}$$

These three equations are the EM update equations for maximum-likelihood parameter estimation of Gaussian mixtures. The equations perform both the expectation step and the maximization step simultaneously. The algorithm iterates, using the newly derived parameters as the guess for the next iteration, until convergence has been achieved. In practice, it is important to start with good initial parameters, for otherwise the variances might collapse to zero. The k-means algorithm (Duda & Hart 1973; Jain & Dubes 1988; Anderberg 1973) often provides very good initial parameter estimates.

5.4 The EM Algorithm for Gaussian Mixture HMMs

Hidden Markov Models were described extensively in Chapter 3. In that chapter, it was shown that HMMs are a model of the joint probability of a collection of random variables $\{X_1, \ldots, X_T, Q_1, \ldots, Q_T\}$. In this section, the EM algorithm is used to produce update equations for maximum-likelihood parameter estimation of HMM parameters. This algorithm is also known as the Baum-Welch algorithm. The HMMs will use Gaussian mixture observation distributions.

It is assumed that the hidden Q_t variables are discrete with $\mathcal{Q}$ possible values $\{1 \ldots \mathcal{Q}\}$. The transition matrix is given by $A = \{a_{i,j}\} = p(Q_t = j|Q_{t-1} = i)$ for all t. The special case of time $t = 1$ is described by the (not necessarily stationary) initial state distribution, $\pi_i = p(Q_1 = i)$. A particular sequence of states is described by $\mathbf{q} = (q_{1:T})$ where $q_t \in \{1 \ldots \mathcal{Q}\}$ is the state at time t.

A particular observation sequence is described as $\mathbf{x} = x_{1:T}$ where x_t is a vector of features at time t. The observation probability distribution for state j is described by: $b_j(x) = p(X = x|Q = j)$. The complete collection of parameters for all observation distributions is represented by $B = \{b_j(\cdot), \forall j\}$. It is assumed that $b_j(x)$ is a mixture of M multivariate Gaussians so $b_j(x) = \sum_{\ell=1}^M c_{j\ell}\mathcal{N}(x|\mu_{j\ell}, \Sigma_{j\ell}) = \sum_{\ell=1}^M c_{j\ell} b_{j\ell}(x)$.

We describe the complete set of HMM parameters for a given model by: $\lambda = (A, B, \pi)$. There are three basic problems one typically wishes to solve using HMMs (Rabiner & Juang 1993):

1. Find $p(\mathbf{X} = \mathbf{x}|\lambda)$ for some $\mathbf{x} = x_{1:T}$. The forward or the backward procedure are typically used for this problem because they are much more efficient than direct evaluation. These procedures are special cases of the junction tree algorithm (Smyth *et al.* 1996; Jensen 1996) for probabilistic inference in Bayesian networks.

2. Given some $\mathbf{x}$ and some λ, find the state sequence $q_{1:T}^*$ that best explains $\mathbf{x}$. I.e., find

$$q_{1:T}^* = \underset{q_{1:T}}{\operatorname{argmax}}\, p(X_{1:T} = x_{1:T}, q_{1:T}|\lambda).$$

The Viterbi procedure, very similar to the forward procedure (and therefore not discussed in this thesis), solves this problem.

3. Find the parameters that best match the data, i.e., find $\lambda^* = \underset{\lambda}{\operatorname{argmax}}\, p(\mathbf{x}|\lambda)$. The Baum-Welch algorithm (also called the forward-backward procedure or EM for HMMs) solves this problem. This is the algorithm we develop presently.

5.4.1 Fast HMM Inference

As mentioned in Chapter 3, one of the advantages of HMMs is the relative efficiency of the probabilistic inference (Smyth *et al.* 1996) needed for the EM parameter estimation. Before deriving the update equations via the EM auxiliary equation, the procedure is intuitively reviewed as it is typically presented in the speech recognition literature (Rabiner & Juang 1993).

The forward procedure is defined as follows:

$$\alpha_i(t) = p(X_1 = x_1, \ldots, X_t = x_t, Q_t = i|\lambda)$$

which is the probability of seeing the partial sequence $x_1, \ldots, x_t$ and ending up in state i at time t. Using the HMM conditional independence properties (see Definition 3.3), the quantity $\alpha_i(t)$ can be efficiently and recursively defined as follows:

1. $\alpha_i(1) = \pi_i b_i(x_1)$
2. $\alpha_j(t+1) = \left[\sum_{i=1}^{Q} \alpha_i(t) a_{ij}\right] b_j(x_{t+1})$
3. $p(X|\lambda) = \sum_{i=1}^{Q} \alpha_i(T)$

The backward procedure is defined similarly:

$$\beta_i(t) = p(X_{t+1} = x_{t+1}, \ldots, X_T = x_T|Q_t = i, \lambda)$$

which is the probability of the ending partial sequence $x_{t+1}, \ldots, x_T$ given that we started at state i at time t. The quantity $\beta_i(t)$ can efficiently be defined as follows:

1. $\beta_i(T) = 1$
2. $\beta_i(t) = \sum_{j=1}^{Q} a_{ij} b_j(x_{t+1}) \beta_j(t+1)$
3. $p(X|\lambda) = \sum_{i=1}^{Q} \beta_i(1) \pi_i b_i(x_1)$

The quantity

$$\gamma_i(t) = p(Q_t = i|X_{1:T} = x_{1:T}, \lambda)$$

is the probability of being in state i at time t for the observation sequence $x_{1:T}$. Note that:

$$p(Q_t = i|x_{1:T}, \lambda) = \frac{p(x_{1:T}, Q_t = i|\lambda)}{P(x_{1:T}|\lambda)} = \frac{p(x_{1:T}, Q_t = i|\lambda)}{\sum_{j=1}^{Q} p(x_{1:T}, Q_t = j|\lambda)}$$

From the HMM conditional independence properties,

$$\alpha_i(t)\beta_i(t) = p(x_1, \ldots, x_t, Q_t = i|\lambda)p(x_{t+1}, \ldots, x_T|Q_t = i, \lambda) = p(X_{1:T} = x_{1:T}, Q_t = i|\lambda)$$

so $\gamma_i(t)$ can be defined in terms of $\alpha_i(t)$ and $\beta_i(t)$ as follows

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{Q} \alpha_j(t)\beta_j(t)}$$

Another useful quantity is

$$\xi_{ij}(t) = p(Q_t = i, Q_{t+1} = j|X_{1:T} = x_{1:T}, \lambda)$$

which is the probability of being in state i at time t and being in state j at time $t+1$. This can also be expanded as:

$$\xi_{ij}(t) = \frac{p(Q_t = i, Q_{t+1} = j, X_{1:T}|\lambda)}{p(X_{1:T}|\lambda)} = \frac{\alpha_i(t)a_{ij}b_j(x_{t+1})\beta_j(t+1)}{\sum_{i=1}^{Q}\sum_{j=1}^{Q} \alpha_i(t)a_{ij}b_j(x_{t+1})\beta_j(t+1)}$$

or equivalently as (Young *et al.* 1990's; Odell 1995):

$$\xi_{ij}(t) = \frac{p(Q_t = i|X_{1:T})p(x_{t+1} \ldots x_T, Q_{t+1} = j|Q_t = i, \lambda)}{p(x_{t+1} \ldots x_T|Q_t = i, \lambda)} = \frac{\gamma_i(t)a_{ij}b_j(x_{t+1})\beta_j(t+1)}{\beta_i(t)}$$

The quantity $\gamma_i(t)$ summed across time

$$\sum_{t=1}^{T} \gamma_i(t)$$

is the expected number of visits to state i and therefore is also the expected number of transitions away from state i for $X_{1:T}$. Similarly,

$$\sum_{t=1}^{T-1} \xi_{ij}(t)$$

is the expected number of transitions from state i to state j for $X_{1:T}$. These follow from the fact that

$$\sum_t \gamma_i(t) = \sum_t E[I_t(i)] = E[\sum_t I_t(i)]$$

and

$$\sum_t \xi_{ij}(t) == \sum_t E[I_t(i,j)] = E[\sum_t I_t(i,j)]$$

where $I_t(i)$ is an indicator random variable that is 1 when the Markov chain in state i at time t, and $I_t(i,j)$ is a random variable that is 1 when the Markov chain moves from state i to state j after time t.

To estimate new parameters for the HMM using the old parameters and the data, it seems intuitively reasonable to use the current relative frequencies to define the new parameters.

The quantity

$$\tilde{\pi}_i = \gamma_i(1) \tag{5.8}$$

is the relative frequency spent in state i at time 1.

The quantity

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{5.9}$$

is the expected number of transitions from state i to state j relative to the expected total number of transitions away from state i.

The probability that the ℓ^{th} component of the i^{th} mixture generated observation x_t is defined as:

$$\gamma_{i\ell}(t) = \gamma_i(t)\frac{c_{i\ell}b_{i\ell}(x_t)}{b_i(x_t)} = p(Q_t = i, C_{it} = \ell | X, \lambda)$$

where C_{it} is a random variable indicating the mixture component at time t for state i.

From the previous section on Gaussian Mixtures, a reasonable guess for the EM update equations might be:

$$c_{i\ell} = \frac{\sum_{t=1}^{T} \gamma_{i\ell}(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

$$\mu_{i\ell} = \frac{\sum_{t=1}^{T} \gamma_{i\ell}(t) x_t}{\sum_{t=1}^{T} \gamma_{i\ell}(t)}$$

$$\Sigma_{i\ell} = \frac{\sum_{t=1}^{T} \gamma_{i\ell}(t)(x_t - \mu_{i\ell})(x_t - \mu_{i\ell})^T}{\sum_{t=1}^{T} \gamma_{i\ell}(t)}$$

When there are E observation sequences the e^{th} being of length T_e, the resulting update equations are:

$$\pi_i = \frac{\sum_{e=1}^{E} \gamma_i^e(1)}{E}$$

$$c_{i\ell} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t)}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_i^e(t)}$$

$$\mu_{i\ell} = \frac{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t) x_t^e}{\sum_{e=1}^{E} \sum_{t=1}^{T_e} \gamma_{i\ell}^e(t)}$$

$$\Sigma_{i\ell} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^{e}(t)(x_t^e-\mu_{i\ell})(x_t^e-\mu_{i\ell})^T}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i\ell}^{e}(t)}$$

and

$$a_{ij} = \frac{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\xi_{ij}^{e}(t)}{\sum_{e=1}^{E}\sum_{t=1}^{T_e}\gamma_{i}^{e}(t)}$$

These relatively intuitive equations are in fact the EM (or Balm-Welch) algorithm for HMM parameter estimation. In the next section, they are derived using the typical EM auxiliary function.

5.4.2 Estimation Formula Using the Q Function.

The quantity $\mathbf{x} = (x_1, \ldots, x_T)$ is again the observed data. The underlying state sequence $\mathbf{q} = (q_1, \ldots, q_T)$ and the collection of mixture components $\mathbf{m} = (m_1, \ldots, m_T)$ constitute the missing information. The incomplete-data likelihood function is given by $P(\mathbf{x}|\lambda)$ and the complete-data likelihood function is $P(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda)$. The auxiliary function becomes:

$$Q(\lambda, \lambda^g) = \sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}} \log P(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda)P(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda^g)$$

where λ^g are the initial guessed parameter estimates, where $\mathcal{Q}$ is the space of all state sequences of length T, and $\mathcal{M}$ is the space of all possible mixture component assignments for the corresponding hidden variables.[2]

Given a particular state sequence $\mathbf{q}$ and set of mixtures $\mathbf{m}$, representing $P(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda^g)$ is fairly easy easy.[3] I.e.,

$$P(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda) = \pi_{q_0}\prod_{t=1}^{T} a_{q_{t-1}q_t}c_{q_t m_t}b_{q_t m_t}(x_t)$$

The auxiliary function then becomes:

$$\begin{aligned}
Q(\lambda, \lambda^g) \quad &= \quad \sum_{\mathbf{q}\in\mathcal{Q}} \log \pi_{q_0} P(\mathbf{x}, \mathbf{q}|\lambda^g) && (5.10a)\\
&+ \quad \sum_{\mathbf{q}\in\mathcal{Q}} \left(\sum_{t=1}^{T} \log a_{q_{t-1}q_t}\right) p(\mathbf{x}, \mathbf{q}|\lambda^g) && (5.10b)\\
&+ \quad \sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}} \left(\sum_{t=1}^{T} \log c_{q_t m_t}\right) p(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda^g) && (5.10c)\\
&+ \quad \sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}} \left(\sum_{t=1}^{T} \log b_{q_t m_t}(x_t)\right) P(\mathbf{x}, \mathbf{q}, \mathbf{m}|\lambda^g) && (5.10d)
\end{aligned}$$

[2]In general, the component assignment for a particular hidden state q_t at time t will depend on q_t since different hidden states might have different numbers of components. For notational simplicity, it will be assumed, in this and the following sections, that all states have the same number of components.

[3]It is assumed here that the initial distribution starts at $t = 0$ instead of $t = 1$, again for notational convenience.

The sum over $\mathbf{m}$ in terms 5.10a and 5.10b do not appear because they are marginalized away. Since the parameters to optimize are now split into a sum of four independent components, each can be optimized individually.

The first term (5.10a) becomes

$$\sum_{\mathbf{q}\in\mathcal{Q}} \log \pi_{q_0} P(\mathbf{x},\mathbf{q}|\lambda^g) = \sum_{i=1}^{\mathcal{Q}} \log \pi_i p(\mathbf{x}, q_0 = i|\lambda^g)$$

since by selecting all $\mathbf{q} \in \mathcal{Q}$, all hidden states are marginalized away except for time $t = 0$. Adding the Lagrange multiplier γ, using the constraint that $\sum_i \pi_i = 1$, and setting the derivative equal to zero yields:

$$\frac{\partial}{\partial \pi_i}\left(\sum_{i=1}^{\mathcal{Q}} \log \pi_i p(\mathbf{x}, q_0 = i|\lambda^g) + \gamma(\sum_{i=1}^{\mathcal{Q}} \pi_i - 1)\right) = 0$$

Taking the derivative, summing over i to get γ, and solving for π_i yields:

$$\pi_i = \frac{P(\mathbf{x}, q_0 = i|\lambda^g)}{P(\mathbf{x}|\lambda^g)} \tag{5.11}$$

The second term (5.10b) becomes:

$$\sum_{\mathbf{q}\in\mathcal{Q}}\left(\sum_{t=1}^{T} \log a_{q_{t-1}q_t}\right) p(\mathbf{x},\mathbf{q}|\lambda^g) = \sum_{i=1}^{\mathcal{Q}}\sum_{j=1}^{\mathcal{Q}}\sum_{t=1}^{T} \log a_{ij} P(\mathbf{x}, q_{t-1} = i, q_t = j|\lambda^g)$$

because for each term for time t, all other factors are again marginalized away. Similar to the previous term, the use of a Lagrange multiplier with the constraint $\sum_{j=1}^{\mathcal{Q}} a_{ij} = 1$ yields:

$$a_{ij} = \frac{\sum_{t=1}^{T} P(\mathbf{x}, q_{t-1} = i, q_t = j|\lambda^g)}{\sum_{t=1}^{T} P(\mathbf{x}, q_{t-1} = i|\lambda^g)}$$

The third term (5.10a) becomes:

$$\sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}}\left(\sum_{t=1}^{T} \log c_{q_t m_t}\right) P(\mathbf{x},\mathbf{q},\mathbf{m}|\lambda^g) = \sum_{i=1}^{\mathcal{Q}}\sum_{\ell=1}^{M}\sum_{t=1}^{T} \log(c_{i\ell}) p(\mathbf{x}, q_t = i, m_t = \ell|\lambda^g)$$

which again can be optimized using a Lagrange multiplier.

Finally, the forth term (5.10a) becomes:

$$\sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}}\left(\sum_{t=1}^{T} \log b_{q_t m_t}(x_t)\right) P(\mathbf{x},\mathbf{q},\mathbf{m}|\lambda^g) = \sum_{i=1}^{\mathcal{Q}}\sum_{\ell=1}^{M}\sum_{t=1}^{T} \log(b_{i\ell}(x_t)) p(\mathbf{x}, q_t = i, m_t = \ell|\lambda^g) \tag{5.12}$$

This term is almost identical to Equation 5.5, except for an addition sum component over the hidden state variables. Therefore, it can be optimized in the same way as described in Section 5.3.

The final resulting update equations are as follows:

$$c_{il} = \frac{\sum_{t=1}^{T} P(q_t = i, m_t = \ell | \mathbf{x}, \lambda^g)}{\sum_{t=1}^{T} \sum_{\ell=1}^{M} P(q_t = i, m_t = \ell | \mathbf{x}, \lambda^g)},$$

$$\mu_{il} = \frac{\sum_{t=1}^{T} x_t P(q_t = i, m_t = \ell | \mathbf{x}, \lambda^g)}{\sum_{t=1}^{T} P(q_t = i, m_t = \ell | \mathbf{x}, \lambda^g)},$$

and

$$\Sigma_{il} = \frac{\sum_{t=1}^{T} (x_t - \mu_{i\ell})(x_t - \mu_{i\ell})^T P(q_t = i, m_t = \ell | \mathbf{x}, \lambda^g)}{\sum_{t=1}^{T} P(q_t = i, m_t = \ell | \mathbf{x}, \lambda^g)}.$$

These are essentially the same set of update equations that were provided in the previous section.

5.5 BMM Implementation

The primary difference between an HMM and a BMM is the observation probability model. In an HMM, the probability model for x under state q is $p(x|q)$. In a BMM, there is a sized K set of additional dependency variables z for each q that are chosen either from the same or a different feature stream (see Figure 4.14) according to one of the selection rules listed in Chapter 4. In this case, each element of x will have a direct dependence on only a subset of the elements of z. One way of explicitly specifying this model is to expand the d-dimensional vector x into the scalars $x^1, x^2, \ldots, x^d$, where x^i is the i^{th} element of the vector x, and write the probability model as:

$$\begin{aligned} p(x|q,z) &= p(x^1, x^2, \ldots, x^d | q, z) \\ &= \prod_{i=1}^{d} p(x^i | x^1, x^2, \ldots, x^{i-1}, q, z) && \text{by the chain rule of probability} \\ &= \prod_{i=1}^{d} p(x^i | x^1, x^2, \ldots, x^{i-1}, q, z^{(i)}) && \text{by conditional independence} \end{aligned}$$

where $z^{(i)} \subseteq z$ is the set of additional dependency variables that x^i is directly dependent on. In other words, $x^i \perp\!\!\!\perp \{z \setminus z^{(i)}\} | \{q, z^{(i)}, x^{1:i-1}\}$. For notational simplicity, the observation model will often be specified simply as $p(x|q,z)$.

There are a variety of choices for the model $p(x|q,z)$. These include a Gaussian autoregressive process, a mixture-Gaussian autoregressive process, non-linear conditional mean regression using a neural network (Levin 1990; Levin 1992) or a mixture thereof, a conditional mean and conditional variance models such as a GARCH model (Engle & Bollerslev 1986; Bollerslev *et al.* 1988).

There are two criteria that such an implementation should satisfy. First, it should be possible that the entropy of the distribution on x can be significantly affected by the value of the z variable. In Chapter 4, it was shown that to produce a better model, the entropy of the distribution $p(x|q,z)$ should decrease in the context of q but not decrease as much in competing contexts. Therefore, any implementation must have the ability to achieve such an effect. Second, in the ideal case, an efficient and easy parameter estimation procedure should exist for the implementation.

In a Gaussian autoregressive process, only the mean of the conditional Gaussian can be affected by the additional dependency data. The entropy of a Gaussian, however, is a function only of the covariance matrix of that Gaussian (Kullback 1968). Therefore, such a simple model does not satisfy the first criterion.

A mixture Gaussian autoregressive process does have the potential to satisfy the first criterion if the mean of each Gaussian component is determined by different regression coefficients over the conditional data. In other words, an identical translation of all the components of a Gaussian mixture distribution does not affect the entropy of the distribution. The update equations for a mixture auto-regressive process are derived in Section 5.6.

A non-linear conditional mean can be implemented using a neural network (Levin 1992; Levin 1990). For example, a three-layer multi-layered perceptron with linear output units could control the mean of a Gaussian distribution. One such network could be used for each value of q to determine the mean of $p(x|q,z)$. Although the conditional mean would in this case contain potentially more interesting properties, the variance of the Gaussian is still unaffected by z. A mixture of Gaussians, where each mixture component has its own neural-network-derived conditional mean would, once again, solve this problem.

Finally, there are models where the variance is directly conditioned on the past. These are called conditional variance models, one example being the Generalized auto-regressive conditional heteroscedasticity (GARCH) model (Engle & Bollerslev 1986) and multivariate versions thereof (Bollerslev *et al.* 1988).

While these models satisfy the first criterion, parameter estimation is particularly easy using the model proposed in the next section which also satisfies the first criterion.

In the following sections, inference and learning procedures for BMMs that use new observation distributions are presented. In Section 5.6, EM update equations are derived for Gaussian mixture models with means linearly conditioned on z. In Section 5.7, these models are extended with an additional outer mixture, and EM update equations are provided in Section 5.8. And in Section 5.9, update equations are derived for BMMs that use these new observation models.

5.6 Mixture Gaussian Autoregressive Processes

In this section, the update equations are derived for a mixture Gaussian autoregressive process. The following probabilistic model is assumed:

$$p(x|z,\Theta) = \sum_{\ell=1}^{M} p(x|\Sigma_\ell, B_\ell, z)p(\ell) = \sum_{\ell=1}^{M} c_\ell p(x|\Sigma_\ell, B_\ell, z)$$

where

$$p(x|\Sigma_\ell, B_\ell, z) = \frac{1}{(2\pi)^{d/2}|\Sigma_\ell|^{1/2}} e^{-\frac{1}{2}(x-B_\ell z)^T \Sigma_\ell^{-1}(x-B_\ell z)}$$

is a Gaussian component with conditional mean determined by the matrix-vector product $B_\ell z$. The vector $z = [z_1 z_2 \ldots z_{K-1} 1]^T$ is a sized K observed vector, and $p(\ell) = c_\ell$ is the probability of component ℓ.

B_ℓ is a matrix that determines which elements of the vector z have influence over the individual elements of x. When used for a BMM, the matrix B_ℓ is sparse. The sparseness controls the fact that each individual element of x depends on a different set of elements of z. For example, if the i, j^{th} element of B_ℓ is not zero, then the j^{th} element of z will have an influence on the i^{th} element of x. For the sake of simplicity, the algorithm is presented without any assumptions made on the topology of B_ℓ.

As mentioned above, it is assumed that the last element of z is the constant 1, so if $K = 1$, then z is a scalar constant and the model degenerates to the standard mixture of Gaussians with a constant mean and covariance matrix.

Assume a sample of data $\mathbf{x} = x_{1:N}$ exists drawn according to the real distribution. Also, assume that $\mathbf{w} = w_{1:N}$ is the collection of hidden variables indicating the component of each sample, and that $\mathbf{z} = z_{1:N}$ is a sample of the additional dependency variables for each i.

The incomplete log-likelihood function is

$$\log p(\mathbf{x}|\mathbf{z}, \Theta) = \sum_{i=1}^{N} \log p(x_i|\Theta, z_i),$$

and the complete log-likelihood function is

$$\log p(\mathbf{x}, \mathbf{w}|\mathbf{z}, \Theta) = \sum_{i=1}^{N} \log(c_{w_i} p(x_i|\Sigma_{w_i}, B_{w_i}, z_i).$$

Similar to the analysis provided in Section 5.3, the auxiliary function may be specified as:

$$Q(\Theta, \Theta^g) = \sum_{\mathbf{w}\in\Upsilon} \log(p(\mathbf{x}, \mathbf{y}|\Theta, \mathbf{z}))p(\mathbf{w}|\mathbf{x}, \mathbf{z}, \Theta^g) = \sum_{\ell=1}^{M}\sum_{i=1}^{N} \log(c_\ell p(x_i|\Sigma_\ell, B_\ell, z_i))p(\ell|x_i, z_i, \Theta^g)$$

The quantity $p(\ell|x_i, z_i, \Theta^g)$ can be determined using Bayes rule as follows:

$$p(\ell|x_i, z_i, \Theta^g) = \frac{p(x_i|\ell, z_i, \Theta^g)p(\ell|z_i, \Theta^g)}{p(x_i|z_i, \Theta^g)} = \frac{p(x_i|\Sigma_\ell^g, B_\ell^g, z_i)c_\ell^g}{\sum_{k=1}^{M} p(x_i|\Sigma_k^g, B_k^g, z_i)c_k^g}$$

For notational simplicity, let $p_{\ell i} = p(\ell|x_i, z_i, \Theta^g)$. The auxiliary function therefore becomes:

$$Q(\Theta, \Theta^t) = \sum_{\ell=1}^{M}\sum_{i=1}^{N} \log(c_\ell)p_{\ell i} + \sum_{\ell=1}^{M}\sum_{i=1}^{N} \log(p(x_i|\Sigma_\ell, B_\ell, z_i))p_{\ell i} \qquad (5.13)$$

in which the individual terms can be optimized independently.

The first term is exactly analogous to the first term of Equation 5.5, yielding the update equation:

$$c_\ell = \frac{1}{N}\sum_{i=1}^{N} p_{\ell i}.$$

Ignoring any constants that vanish after taking derivatives, the second term of Equation 5.13 can be represented as:

$$\sum_{\ell=1}^{M}\sum_{i=1}^{N}\left[-\frac{1}{2}\log(|\Sigma_\ell|) - \frac{1}{2}(x_i - B_\ell z_i)^T \Sigma_\ell^{-1}(x_i - B_\ell z_i)\right] p_{\ell i} \tag{5.14}$$

Using the result from Appendix B, the derivative of Equation 5.14 can be taken with respect to B_ℓ and set it to zero yielding:

$$\frac{\partial}{\partial B_\ell}\sum_{i=1}^{N}\left[-\frac{1}{2}(x_i - B_\ell z_i)\Sigma_\ell^{-1}(x_i - B_\ell z_i)\right] p_{\ell i} = \sum_{i=1}^{N}\left[\Sigma_\ell^{-1}(x_i - B_\ell z_i) z_i^T p_{\ell i}\right] = 0 \tag{5.15}$$

Multiplying this by Σ_ℓ yields:

$$\sum_{i=1}^{N}(x_i - B_\ell z_i) z_i^T p_{\ell i} = 0$$

or equivalently

$$\sum_{i=1}^{N} x_i z_i^T p_{\ell i} = \sum_{i=1}^{N} B_\ell z_i z_i^T p_{\ell i}.$$

This equation can be easily solved for B_ℓ

$$B_\ell = \left(\sum_{i=1}^{N} x_i z_i^T p_{\ell i}\right)\left(\sum_{i=1}^{N} z_i z_i^T p_{\ell i}\right)^{-1}$$

which can be solved by a matrix subroutine such as LU-decomposition (Harville 1997; Golub & Loan 1996). If $K = 1$ so that z_i is just the scalar variable 1, then

$$B_\ell = \frac{\sum_{i=1}^{N} p_{\ell i} x_i}{\sum_{i=1}^{N} p_{\ell i}}$$

which, as expected, is the update equation for the mean for the $\ell^t h$ component of the usual Gaussian mixture model.

To find the update rule for the covariance matrices, let $v_{i\ell} = x_i - B_\ell z_i$ and $\mu_\ell = 0$. Equation 5.14 becomes

$$\sum_{\ell=1}^{M}\sum_{i=1}^{N}\left[-\frac{1}{2}\log(|\Sigma_\ell|) - \frac{1}{2}(v_{i\ell} - \mu_\ell)^T \Sigma_\ell^{-1}(v_{i\ell} - \mu_\ell)\right] p_{\ell i}$$

which has the same form as the unconditional Gaussian mixture case shown in Equation 5.7. Therefore, the update equation for Σ_ℓ is as follows:

$$\Sigma_\ell = \frac{\sum_{i=1}^{N} p_{\ell i}(x_i - B_\ell z_i)(x_i - B_\ell z_i)^T}{\sum_{i=1}^{N} p_{\ell i}}$$

5.7 Switching Gaussian Mixtures

A generalization of the mixture Gaussian autoregressive processes adds one additional outer mixture that is conditioned on the variable z. A switching Gaussian mixture (SGM) model is defined as follows:

$$p(x|z,q) = \sum_{s=1}^{S} \left(\sum_{m=1}^{M} p(x|m,s,z,q) p(m|s,q) \right) p(s|z) \tag{5.16}$$

The outer mixture uses the variable s which mixes over a set of inner mixtures. The variable s can be seen as a switching variable, the inner mixtures are probabilistically switched on or off according to the distribution $p(s|z)$. The inner mixture is a Gaussian mixture but where the mixing coefficients and the Gaussian components are dependent on the switching variable s. The inner mixing coefficients are given by $p(m|s,q)$ where m is the inner mixing variable. Each underlying component is a Gaussian autoregressive process on z with a sparse dependency matrix B_{qms}.

$$p(x|m,s,z,q) = \frac{1}{(2\pi)^{d/2}|\Sigma_{qms}|^{1/2}} e^{-\frac{1}{2}(x-B_{qms}z)'\Sigma_{qms}^{-1}(x-B_{qms}z)} \tag{5.17}$$

Therefore, $p(x|m,s,z,q)$ is a Gaussian distribution with conditional mean $B_{qms}z$ and covariance matrix Σ_{qms} with a discrete dependence on the variable s.

An SGM therefore can simulate the effect of the variable z controlling the covariances of a Gaussian mixture, but by using an additional mixture, it avoids complexities associated with a conditional variance model. The Bayesian network for a SGM is shown in Figure 5.1 and the sampling from an SGM is depicted in Figure 5.2.

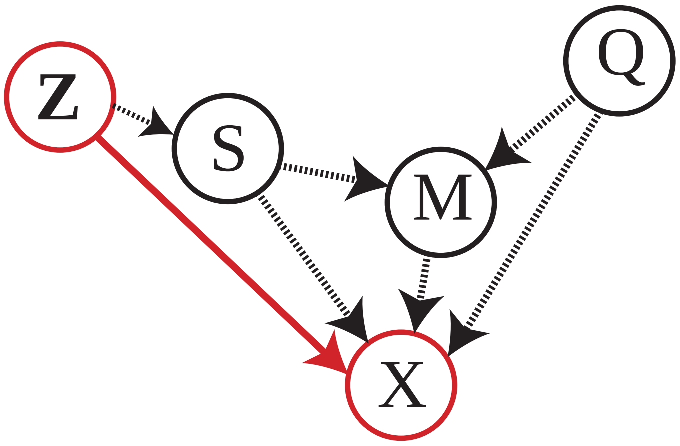

Figure 5.1: Graphical model showing a Switching Gaussian Mixture to represent the conditional density $p(X|Q,Z)$. Z is the set of continuous dependency variables, S is a discrete switching variable, M is a discrete mixture variable, Q is the hidden variable, and X is an observation variable. Continuous dependencies are shown using solid arrows, and discrete dependencies are shown using dashed arrows.

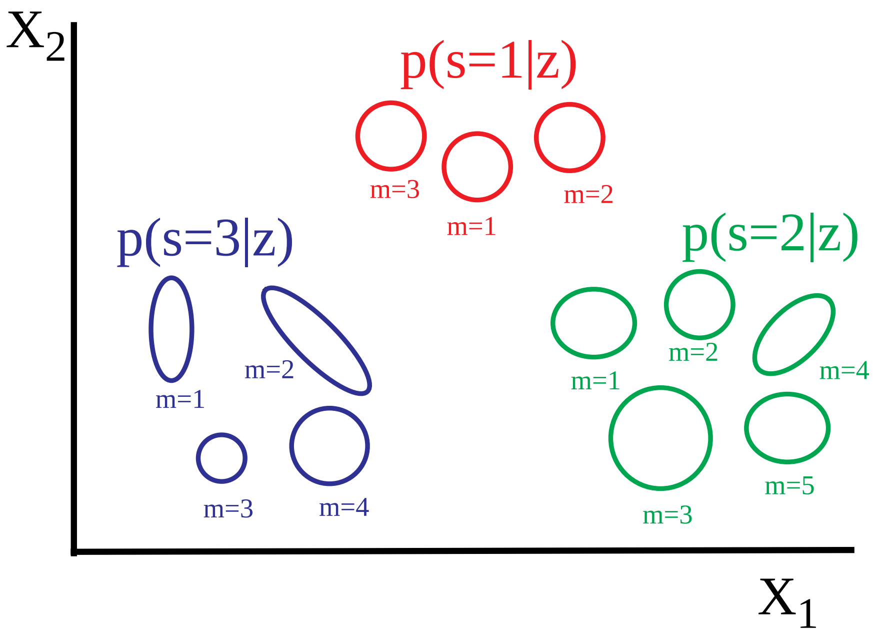

Figure 5.2: Two-dimensional density for a Switching Gaussian Mixture for a particular z. To sample from such a distribution given a q and z, first a Gaussian mixture is randomly chosen according to the distribution $p(s|z)$. Then an individual mixture component in that mixture is chosen using $p(m|s,q)$. Then, the mean of that component is set to the vector $B_{qms}z$ which corresponds to a translation of the chosen Gaussian component's mean. Finally, the resulting Gaussian density is sampled as usual.

5.8 Switching Gaussian Mixture EM Update Equations

In this section, EM update equations are derived for an SGM. For simplicity, update equations for an SGM alone are derived without considering the hidden Markov variables of the BMM (i.e., dependence on Q will be ignored for now). The hidden variables will be re-introduced in the next section.

Similar to the previous section, the following model is assumed:

$$p(x|z,\Theta) = \sum_{s=1}^{S} \left(\sum_{m=1}^{M} c_{ms} p(x|\Sigma_{ms}, B_{ms}, z) \right) p(s|z)$$

where

$$p(x|\Sigma_{ms}, B_{ms}, z) = \frac{1}{(2\pi)^{d/2}|\Sigma_{ms}|^{1/2}} e^{-\frac{1}{2}(x-B_{ms}z)^T \Sigma_{ms}^{-1}(x-B_{ms}z)},$$

c_{ms} are the mixing coefficients for mixture s, and $p(s|z)$ are the z specific mixing probabilities.

As in Section 5.6, individual elements of z may influence individual elements of x depending on the structure of the B_{ms} matrices. The conditional switching probabilities

$p(s|z)$ may be seen as an initial classifier of the z variables. Therefore, the entire x model is dependent on the class of the entire vector z. The more refined element-wise dependencies are represented by having a separate B_{ms} matrix both for each main component m and for each cluster s. In general, not only the parameters but also the structure of the B_{ms} matrices could be different for each value of s and m. This is discussed further in Chapter 6.

Let $\mathbf{x}$ be the incomplete data sample, and $\mathbf{w} = w_{1:N}$ and $\mathbf{s} = s_{1:N}$ designate the missing data indicating respectively the mixture component and the z-variable class. The complete log-likelihood function becomes:

$$\log p(\mathbf{x}, \mathbf{w}, \mathbf{s}|\mathbf{z}, \Theta) = \sum_{i=1}^{N} \mathit{log}\Big(c_{w_i s_i} p(x_i|\Sigma_{w_i s_i}, B_{w_i s_i}, z_i) p(s_i|z_i)\Big).$$

Using the same reasoning as was used in Section 5.3, the auxiliary function becomes:

$$\begin{aligned} Q(\Theta, \Theta^g) &= \sum_{\mathbf{w}} \sum_{\mathbf{s}} \log(p(\mathbf{x}, \mathbf{w}, \mathbf{s}|\Theta, \mathbf{z})) p(\mathbf{w}, \mathbf{s}|\mathbf{x}, \mathbf{z}\Theta^g) \\ &= \sum_{\mathbf{w}} \sum_{\mathbf{s}} \sum_{i=1}^{N} \log p(x_i, w_i, s_i|z_i, \Theta) \prod_j p(w_j, s_j|x_j, z_j, \Theta^g) \\ &= \sum_{m=1}^{M} \sum_{s=1}^{S} \sum_{i=1}^{N} \log p(x_i, m, s|z_i, \Theta) p(m, s|x_i, z_i, \Theta^g) \end{aligned}$$

Using the conditional independence properties implied by the Bayesian network shown in Figure 5.1, the probability of the hidden variables $p(m, s|x_i, z_i, \Theta^g)$ can be represented as

$$\begin{aligned} p(m, s|x_i, z_i, \Theta^g) &= p(m|x_i, z_i, s, \Theta^g) p(s|x_i, z_i, \Theta^g) \\ &= p(m|x_i, z_i, s, \Theta^g) p(s|z_i, \Theta^g) \end{aligned}$$

where

$$p(m|x_i, z_i, s, \Theta^g) = \frac{p(x_i|m, x_i, s, \Theta^g) p(m|s, \Theta^g)}{\sum_m p(x_i|m, x_i, s, \Theta^g) p(m|s, \Theta^g)}$$

and where $p(s|z_i, \Theta^g)$ is the posterior probability of class s given z_i.

Again using the conditional independence properties implied by Figure 5.1, the quantity to optimize as a function of Θ can be represented as:

$$\begin{aligned} p(x, m, s|z, \Theta) &= p(x|m, s, z, \Theta) p(m|s, z, \Theta) p(s|z, \Theta) \\ &= p(x|m, s, z, \Theta) p(m|s, \Theta) p(s|z, \Theta) \end{aligned}$$

This results in the following three separate equations to be optimized

$$\sum_{m=1}^{M} \sum_{s=1}^{S} \sum_{i=1}^{N} \log p(x_i|m, s, z_i, \Theta) p(m, s|x_i, z_i, \Theta^g), \tag{5.18}$$

$$\sum_{m=1}^{M}\sum_{s=1}^{S}\sum_{i=1}^{N}\log p(m|s,\Theta)p(m,s|x_i,z_i,\Theta^g) = \sum_{m=1}^{M}\sum_{s=1}^{S}\sum_{i=1}^{N}\log c_{ms}p(m,s|x_i,z_i,\Theta^g) \qquad (5.19)$$

where $c_{ms} = p(m|s,\Theta)$, and

$$\sum_{m=1}^{M}\sum_{s=1}^{S}\sum_{i=1}^{N}\log p(s|z_i,\Theta)p(m,s|x_i,z_i,\Theta^g) = \sum_{s=1}^{S}\sum_{i=1}^{N}\log p(s|z_i,\Theta)p(s|z_i,x_i,\Theta^g) \qquad (5.20)$$

Equation 5.18 is similar to Equation 5.13, so a procedure similar to that described in Section 5.6 may be used. In this case, each parameter has an additional subscript s that indicates the dependence on the switching variable (See Figure 5.1). The resulting update equations for the output probabilities are:

$$B_{ms} = \left(\sum_{i=1}^{N} x_i z_i^T p_{mis}\right)\left(\sum_{i=1}^{N} z_i z_i^T p_{mis}\right)^{-1}$$

and

$$\Sigma_{ms} = \frac{\sum_{i=1}^{N} p_{mis}(x_i - B_{ms}z_i)(x_i - B_{ms}z_i)^T}{\sum_{i=1}^{N} p_{mis}},$$

where $p_{mis} = p(m,s|x_i,z_i,\Theta^g)$.

Similar to Equation 5.5, the mixing coefficients in Equation 5.19 can be optimized by introducing a Lagrange multiplier, taking the partial derivatives, and setting the result to zero:

$$\frac{\partial}{\partial c_{ms}}\left(\sum_{m=1}^{M}\sum_{s=1}^{S}\sum_{i=1}^{N}\log(c_{ms})p_{mis} + \lambda(\sum_{m} c_{ms} - 1)\right) = 0$$

which yields

$$c_{ms} = \frac{1}{N}\sum_{i=1}^{N} p_{mis}.$$

Equation 5.20, performs the optimization of the quantity $p(s|z,\Theta)$, which performs an initial discrete classification of the variable z, and involves the quantity $p(s|z,x,\Theta^g)$ which can be represented as:

$$p(s|z,x,\Theta^g) = \frac{p(x|s,z,\Theta^g)p(s|z,\Theta^g)}{p(x|z,\Theta^g)} = \frac{p(x|s,z,\Theta^g)p(s|z,\Theta^g)}{\sum_r p(x|r,z,\Theta^g)p(r|z,\Theta^g)}$$

where

$$p(x|s,z,\Theta^g) = \sum_{m=1}^{M} c_{ms}p(x|\Sigma_{ms}^g, B_{ms}^g, z)$$

and where

$$p(x|\Sigma^g_{ms}, B^g_{ms}, z) = \mathcal{N}(x|\Sigma^g_{ms}, B^g_{ms}z)$$

Optimization of Equation 5.20 therefore depends on the form of the distribution $p(s|z,\Theta)$.

One option for $p(s|z,\Theta)$ is to use a form of multi-class logistic regression such as the softmax function, in which case

$$p(s|z,\Theta) = \frac{e^{u_s^T z}}{\sum_r e^{u_r^T z}}$$

where u_s is a length K vector of parameters for each s. It can be seen that Equation 5.20 corresponds to a maximization of the negative cross-entropy (Bishop 1995) between $p(s|z_i,\Theta)$ and $p(s|z_i,x_i,\Theta^g)$ so the optimization procedures such as gradient descent or iteratively reweighted least squares (Jordan & Xu 1993; Jordan & Jacobs 1994) can be used to learn u_s. Another (quite similar) approach uses a multi-layered perceptron with a softmax output non-linearity (Bishop 1995) trained at each iteration using the distribution $p(s|z,x,\Theta^g)$ as "soft" training targets. Both of these approaches requires an extra training iteration within each SGM EM training iteration.

Alternatively, to avoid the inner training iteration, it may be sufficient to approximate this learning problem by transforming the quantity to be optimized from a "diagnostic" to a "generative" (Jordan 1995) model and then ignoring terms that are difficult to optimize. By Bayes rule,

$$p(s|z,\Theta) = \frac{p(z|s,\Theta)p(s|\Theta)}{\sum_r p(z|r,\Theta)p(r|\Theta)} = \frac{p(z|s,\Theta)p(s|\Theta)}{p(z|\Theta)}$$

The right-hand side of Equation 5.20 then becomes:

$$\sum_{s=1}^{S}\sum_{i=1}^{N} \log p(z_i|s,\Theta)p(s|z_i,x_i,\Theta^g) + \sum_{s=1}^{S}\sum_{i=1}^{N} \log p(s|\Theta)p(s|z_i,x_i,\Theta^g)$$
$$- \sum_{s=1}^{S}\sum_{i=1}^{N} \log p(z_i|\Theta)p(s|z_i,x_i,\Theta^g)$$

If it is assumed that $p(z_i|s,\Theta)$ is a class conditional multivariate Gaussian (i.e., $p(z|s,\Theta) = \mathcal{N}(z|\mu_s,\Sigma_s)$ then the first two terms of this equation may be separately optimized as was done in Equation 5.5 if the third term (which is not independent of the first two) is ignored. This approximation eliminates the additional embedded training iteration within each SGM EM iteration by learning the models $p(z|s)$ and $p(s)$ rather than the model $p(s|z)$.

An even more severe approximation can be made by ignoring the dependence of s on x_i in $p(s|z_i,x_i,\Theta^g)$, in which case the right-hand side of Equation 5.20 becomes:

$$\sum_{s=1}^{S}\sum_{i=1}^{N} \log p(s|z_i,\Theta)p(s|z_i\Theta^g)$$

This quantity is maximized when $D(p(s|z_i,\Theta)||p(s|z_i,\Theta^g))$ is minimized, and this occurs when $\Theta = \Theta^g$ for those portions of Θ that affect these distributions. Therefore, under this approximation $p(s|z)$ does not change between EM iterations so any, perhaps unsupervised, probabilistic clustering method can be used (such as k-means followed by EM for Gaussian mixtures) prior to SGM EM training.

5.9 BMM Update Equations for SGM Observation Models

In this section, update equations for a BMM with SGM observation models are derived.

A BMM with SGM observations corresponds to the following probabilistic model:

$$p(\mathbf{x}|\mathbf{z}) = \sum_{\mathbf{q}}\sum_{\mathbf{m}}\sum_{\mathbf{s}} p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z})$$

where $\mathbf{x} = x_{1:T}$ is a collection of T feature vectors, $\mathbf{q} = q_{1:T}$ is an assignment to the hidden variables, $\mathbf{m} = m_{1:T}$ is a mixture variable assignment, and $\mathbf{s} = s_{1:T}$ is a switching variable assignment. Because of the HMM and SGM conditional independence properties, the following factorization is valid:

$$\begin{aligned} p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z}) &= \prod_t p(x_t|q_t, m_t, s_t, z_t)p(m_t|s_t, q_t)p(s_t|z_t)p(q_t|q_{t-1}) \\ &= \pi_{q_0}\prod_{t=1}^{T} b_{q_t}(x_t|m_t, s_t, z_t)c_{m_t s_t q_t}a_{q_{t-1}q_t}d_{z_t}(s_t) \end{aligned}$$

where π is the initial distribution over Markov states, $b_i(x|\ell, s, z)$ is the mean-conditional Gaussian for state i and switch value s, $c_{\ell s q}$ is the mixture coefficient for mixture ℓ of switch s and hidden state q, a_{ij} is the Markov transition probability from state i to state j, and $d_z(s) = p(s|z)$ is the probability of mixture s given dependency variables z.

The auxiliary function is similar to the one provided in Section 5.4.2 except in this case there is an additional sum over the switching variables:

$$\begin{aligned} Q(\lambda,\lambda^g) &= \sum_{\mathbf{q}}\sum_{\mathbf{m}}\sum_{\mathbf{s}} \log p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda)p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda^g) \\ &= \sum_{\mathbf{q}}\sum_{\mathbf{m}}\sum_{\mathbf{s}} \log \pi_{q_0} \sum_{t=1}^{T} \log\Big(b_{q_t}(x_t|m_t, s_t, z_t)c_{m_t s_t q_t}a_{q_{t-1}q_t}d_{z_t}(s_t)\Big)p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda^g) \end{aligned}$$

In this form, it can be seen that there are five independent components to be optimized: the initial distribution π, the mean-conditional Gaussians $b_i(x|\ell, s, z)$, the mixture coefficients $c_{\ell s q}$, the Markov transition probabilities a_{ij}, and the dependency variable classifier $d_z(s)$.

The auxiliary function for first term becomes:

$$\sum_{\mathbf{q}}\sum_{\mathbf{m}}\sum_{\mathbf{s}} \log \pi_{q_0} p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda^g) = \sum_{i=1}^{Q} \log \pi_i p(\mathbf{x}, q_0 = i|\mathbf{z},\lambda^g)$$

giving an update equation that is very similar to Equation 5.11:

$$\pi_i = \frac{p(\mathbf{x}, q_0 = i|\mathbf{z},\lambda^g)}{p(\mathbf{x}|\mathbf{z},\lambda^g)}$$

The auxiliary function for the second term becomes:

$$\sum_{\mathbf{q}}\sum_{\mathbf{m}}\sum_{\mathbf{s}}\sum_{t=1}^{T}\log\left(b_{q_t}(x_t|z_t,m_t,s_t)\right)P(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda^g)$$
$$=\sum_{i=1}^{\mathcal{Q}}\sum_{\ell=1}^{M}\sum_{j=1}^{S}\sum_{t=1}^{T}\log\left(b_i(x_t|z_t,m_t=\ell,s_t=j)\right)p(\mathbf{x},q_t=i,m_t=\ell,s_t=j|\mathbf{z},\lambda^g)$$

This equation is almost identical to Equation 5.18 except for the additional sum over the hidden Markov variable at each time q_t. For notational simplicity, let:

$$\gamma_{i\ell j}(t)=p(q_t=i,m_t=\ell,s_t=j|\mathbf{x},\mathbf{z},\lambda^g)$$

By the same logic that optimized Equation 5.13 in Section 5.6, the update equation for $B_{i\ell j}$, the dependency matrix for hidden Markov state i, mixture component ℓ, and switch j is

$$B_{i\ell j}=\left(\sum_{t=1}^{T}\gamma_{i\ell j}(t)x_tz_t'\right)\left(\sum_{t=1}^{T}\gamma_{i\ell j}(t)z_tz_t'\right)^{-1} \tag{5.21}$$

and the update equation for the corresponding covariance is:

$$\Sigma_{i\ell j}=\frac{\sum_{t=1}^{T}\gamma_{i\ell j}(t)(x_t-B_{i\ell j}z_t)(x_t-B_{i\ell j}z_t)'}{\sum_{t=1}^{T}\gamma_{i\ell j}(t)} \tag{5.22}$$

The auxiliary function for the third term becomes:

$$\sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}}\sum_{\mathbf{s}\in\mathcal{S}}\sum_{t=1}^{T}\log\left(c_{m_ts_tq_t}\right)p(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda^g)$$
$$=\sum_{i=1}^{\mathcal{Q}}\sum_{\ell=1}^{M}\sum_{j=1}^{S}\sum_{t=1}^{T}\log\left(c_{\ell ji}\right)p(\mathbf{x},q_t=i,m_t=\ell,s_t=j|\mathbf{z},\lambda^g)$$

which is similar to Equation 5.19. The update equation becomes:

$$c_{\ell ji}=\frac{\sum_{t=1}^{T}p(q_t=i,m_t=\ell,s_t=j|\mathbf{x},\mathbf{z},\lambda^g)}{\sum_{t=1}^{T}\sum_{\ell=1}^{M}p(q_t=i,m_t=\ell,s_t=j|\mathbf{x},\mathbf{z},\lambda^g)}$$

For the forth term which optimize the transition probabilities, the form of the auxiliary equation is similar to Equation 5.10b. The update equation therefore is:

$$a_{ij}=\frac{\sum_{t=1}^{T}P(\mathbf{x},q_{t-1}=i,q_t=j|\mathbf{z},\lambda^g)}{\sum_{t=1}^{T}P(\mathbf{x},q_{t-1}=i|\mathbf{z},\lambda^g)}$$

Finally, the auxiliary function for the fifth term becomes:

$$\sum_{\mathbf{q}\in\mathcal{Q}}\sum_{\mathbf{m}\in\mathcal{M}}\sum_{\mathbf{s}\in\mathcal{S}}\sum_{t=1}^{T}\log d_{z_t}(s_t)P(\mathbf{x},\mathbf{q},\mathbf{m},\mathbf{s}|\mathbf{z},\lambda^g)=\sum_{i=1}^{S}\sum_{t=1}^{T}\log d_{z_t}(i)P(\mathbf{x},s_t=i|\mathbf{z},\lambda^g)$$

The right-hand side of this equation has the same form as Equation 5.20 and can be optimized as described in that section.

Notice that if $K = 1$ (i.e., $z_t = 1$ for all t) and $S = 1$, these update equations reduce to the standard update equations for Gaussian mixture HMMs.

5.9.1 Fast Recursive Equations for BMMs with SGMs

As in Section 5.4.1, we can define α and β recursions that can be computed efficiently and then used to produce the needed probabilistic quantities in the previous section.

The alpha and beta recursions become:

$$\alpha_i(t) = p(x_{1:t}, q_t = i|\mathbf{z}) = \sum_j \alpha_j(t-1) a_{ji} b_i(x_t|z_t)$$

and

$$\beta_i(t) = p(x_{t+1:T}|q_t = i, \mathbf{z}) = \sum_{i=1}^{Q} a_{ij} b_j(x_{t+1}|z_{t+1}) \beta_j(t+1)$$

In this case, the recursions use the more complex observation probability model $b_j(x|z)$ for the j^{th} hidden Markov state. Using the conditional independence properties, it follows that

$$p(\mathbf{x}, q_t = i|\mathbf{z}) = \alpha_i(t)\beta_i(t)$$

The following quantities may also be defined:

$$\gamma_i(t) = p(q_t = i|\mathbf{x}, \mathbf{z}) = \frac{p(\mathbf{x}, q_t = i|\mathbf{z})}{\sum_{j=1}^{Q} p(\mathbf{x}, q_t = i|\mathbf{z})} = \sum_\ell \gamma_{i\ell}(t)$$

where

$$\gamma_{i\ell}(t) = \gamma_i(t) \frac{c_{\ell i} b_{i\ell}(x_t|z_t)}{b_i(x_t|z_t)} = p(q_t = i, m_t = \ell|\mathbf{x}, \mathbf{z}) = \sum_j \gamma_{i\ell j}(t),$$

and where

$$\gamma_{\ell j i}(t) = \gamma_i(t) \frac{c_{\ell i j} b_{i\ell j}(x_t|z_t) d_{z_t}(j)}{b_i(x_t|z_t)} = p(q_t = i, m_t = \ell, s_t = j|\mathbf{x}, \mathbf{z}),$$

and where $b_{i\ell j}(x_t|z_t) = p(x_t|z_t, m_t = \ell, s_t = j)$ and $c_{\ell ji} = \sum_j c_{\ell ji}$.

The expected state transitions may also be defined:

$$\xi_{ij}(t) = p(q_t = i, q_{t+1} = j|\mathbf{x}, \mathbf{z})$$

as

$$\xi_{ij}(t) = \frac{p(q_t = i, q_{t+1} = j, \mathbf{x}|\mathbf{z})}{p(\mathbf{x}|\mathbf{z})} = \frac{\alpha_i(t) a_{ij} b_j(x_{t+1}|z_{t+1}) \beta_j(t+1)}{\sum_{i=1}^{Q} \sum_{j=1}^{Q} \alpha_i(t) a_{ij} b_j(x_{t+1}|z_{t+1}) \beta_j(t+1)}$$

Therefore, all the quantities needed for the EM update equations can obtained efficiently.

The most likely assignment to the hidden state variables (i.e., the Viterbi path) can be found as usual (Rabiner & Juang 1993):

$$\phi_j(t) = \left[\max_i \phi_i(t-1)a_{ij}\right] b_j(x_t|z_t)$$

for $1 < t \leq T$ and where the initial conditions are given by:

$$\phi_j(1) = \pi_j b_j(x_1|z_1).$$

5.10 Simplifications with Diagonal Covariance Matrices

As stated, equations 5.21 and 5.22 require the outer products zz' and xz' and the matrix-vector products $B_{i\ell j}z_t$ for each time frame, each hidden Markov state, each mixture component, and each switch variable value. Significant simplifications can be achieved if diagonal rather than full covariance matrices are used. And as will be described in Section 7.3, diagonal covariance matrices are not any less general than full covariance matrices for these observation distributions.

Let $\text{diag}(\Sigma_{i\ell j}, r)$ be the r^{th} diagonal element of the (now diagonal) covariance matrix $\Sigma_{i\ell j}$. Also, let $B_{i\ell j}(r)$ be a row-vector consisting of the non-zero (i.e., existing) elements from the r^{th} row of the matrix $B_{i\ell j}$, and let $z_t(r)$ be a column vector consisting of the elements of the vector z_t that correspond to the non-zero entries in the r^{th} row of $B_{i\ell j}$. In other words, the quantity $B_{i\ell j}(r)z_t(r)$ equals the r^{th} element of the vector $B_{i\ell j}z_t$ but $B_{i\ell j}(r)z_t(r)$ does not sum terms containing just zero. This is a slight abuse of notation as the elements contained in $z_t(r)$ depend on the matrix it is being multiplied with. The meaning should be clear nevertheless. Finally, let $x_t(r)$ be the r^{th} scalar element of the vector x_t. It follows immediately that the update equations for the diagonal entries of the covariance matrices are:

$$\text{diag}(\Sigma_{i\ell j}, r) = \frac{\sum_{t=1}^{T} \gamma_{i\ell j}(t)\Big(x_t(r) - B_{i\ell j}(r)z_t(r)\Big)^2}{\sum_{t=1}^{T}\gamma_{i\ell j}(t)} \tag{5.23}$$

Note that this is a scalar update equation for each of the diagonal elements of the covariance matrix. Depending on the sparse patterns of the matrices, the cost of the dot products $B_{i\ell j}(r)z_t(r)$ can be significantly smaller than before.

The update equations for the B matrices can also be significantly simplified. When using diagonal covariance matrices, Equation 5.15 can be represented as:

$$\frac{\partial}{\partial B_\ell(r)} \sum_{i=1}^{N}\sum_{r=1}^{d}\left[-\frac{1}{2}\big(x_i(r) - B_\ell(r)z_i(r)\big)\Sigma_\ell^{-1}(r)\big(x_i(r) - B_\ell(r)z_i(r)\big)\right] p_{\ell i}$$

which consists of a set of independent update equations, one each for the non-zero portions of each row of B_ℓ. Taking the derivative by applying the result of Appendix B to each update equation yields:

$$\sum_{i=1}^{N} \left[\Sigma_\ell^{-1}(r)(x_i(r) - B_\ell(r)z_i(r))z_i(r)^T p_{\ell i}\right] = 0$$

for each r. This leads to independent update equations for the non-zero portion of each row of the B matrix.

$$B_{i\ell j}(r) = \left(\sum_{t=1}^{T} \gamma_{i\ell j}(t)x_t(r)z_t(r)'\right)\left(\sum_{t=1}^{T} \gamma_{i\ell j}(t)z_t(r)z_t(r)'\right)^{-1}$$

In this case, only the outer products $z_t(r)z_t(r)'$ and the SAXPY (Golub & Loan 1996) operations $x_t(r)z_t(r)'$ are required, a significant computational complexity and memory requirement reduction.

5.11 Discussion

In this chapter, the switching mixture of Gaussians was introduced and suggested as an implementation for the dependencies used by a BMM. EM update equations for maximum likelihood parameter estimation were derived in detail. This implementation with $S = 1$ and $M = 1$ is similar to the conditionally Gaussian HMMs as was described in Chapter 3 — the key difference between an HMM and a BMM when $S = M = 1$ is that the dependence structure is sparse, derived discriminatively from the data, and hidden-variable dependent (see Chapter 4).

As described in section 4.7, the BMM parameters could also be found using discriminative training schemes such as MMI (Bahl *et al.* 1986), MDI (Ephraim & Rabiner 1988; Ephraim *et al.* 1989), and MCE (Juang & Katagiri 1992; Juang *et al.* 1997). As argued in Chapter 3, however, because the BMM learning schemes (Chapter 4) attempt to produce more structurally discriminative models, these more complex training methods might not be necessary.

Chapter 6

BMM Word Error Results

Contents

In this chapter, word error results are reported for a BMM-based speech recognition system and are compared with an HMM-based system. It is shown that it is possible for a BMM-based system to outperform a comparable HMM based system. The implementation of the additional BMM dependencies and their EM parameter update equations was described in Chapter 5. In Section 6.1, the procedure used to boot both HMMs and BMMs are described. In Section 6.2 BMMs are compared with HMMs on a small-vocabulary isolated-word speech corpus. Then, in Section 6.3, results are reported for a large-vocabulary isolated-word speech corpus. In Section 6.4, by comparing word error results for a variety of models, it is empirically demonstrated that good performance relies crucially on the structural discriminability of the models, at least when trained using a maximum-likelihood procedure.

6.1 Training Procedure

A BMM-based speech recognition system requires the computation of conditional mutual information $(X;Z|Q = q)$ where the individual conditions q correspond to the hidden state values of a particular baseline HMM. Computing these quantities therefore requires a working HMM baseline system. The following general training procedure is used for all of the results reported in this chapter — any differences between this procedure and the actual procedure used for a particular experiment are described in the section where that experiment is presented.

1. From the training data, initialize the parameters of a set of Gaussian mixtures using the uniform segmental k-means procedure (Rabiner & Juang 1993; Jelinek 1997) where k is the number of mixture components per Gaussian.

2. Construct a boot HMM with the initial Gaussian means set from the result of the k-means procedure. Compute the initial covariance matrices using the samples closest to each mean.

3. Using the same training data, train the resulting HMM system using EM until convergence is achieved.

4. Compute the Viterbi (most likely) path through the HMM's hidden variables for all examples in the training set.

5. Compute the conditional mutual information $I(X;Z|Q=q)$ for class q using the samples in the training set that are labeled q according to the Viterbi path. The computation of mutual information is described in Appendix C.

6. Compute the appropriate loss function(s).

7. Choose a set of parameter values for the heuristic listed in Figure 4.11. Run the heuristic and produce a BMM with a new set of cross-observation dependencies.

8. Produce a Gaussian mixture BMM with sparse B matrices. The B matrices will have trainable (i.e., existing) entries only where there is a corresponding direct dependency in the BMM. If the entry of B in the i^{th} row and j^{th} column is greater than zero, it means that the i^{th} element of x has a direct dependency on the j^{th} element of z in the model for $p(x|z,q)$. Other locations in the B matrix are permanently fixed to the value zero and therefore are not trained (and in fact do not exist, see Section 5.10).

9. Train the resulting BMM either using a forced Viterbi procedure or using the full EM update equations. Recall that the collection of dependency variables z have a unity constant as the last element. That is $z = [z_1 z_2 \ldots z_{K-1} 1]^T$ is a sized K observed vector where the K^{th} element is the constant 1. Therefore, the right-most column of B can be seen as the residual means of each component distribution irrespective of the dependency variables. The initial values for the B matrices are set as follows: the right-most column of B is set to the mean of the boot HMM's corresponding mixture component. The remaining trainable entries in B are set to zero. After one EM iteration, these entries will take on values other than zero. The initial covariance matrices for the BMMs are set to the corresponding HMM covariance matrices.

10. Test the resulting trained BMM on a development set and compute the word error rate. The utterance probability scores are evaluated using a Viterbi procedure (i.e., the score for the most likely path is used rather than for all the paths). Since utterances in both corpora consist only of isolated words, a full-beam Viterbi score is used. The word-error results therefore do not include any decoding errors (Chase 1997).

11. For random restarts, go to step 7 , but use different parameter values for the heuristic.

12. Report the best word error rate found.

All the experiments described in this chapter use feature vectors consisting of 12 mel-frequency cepstral coefficients (MFCCs) (Young 1996; Rabiner & Juang 1993; Deller *et al.* 1993) plus c_0 (log energy) and their deltas, all of which are derived from FFT-based log spectra. While it is known that MFCCs work very well with Gaussian mixture HMMs alone, it was desirable to show that an improvement could be obtained even using this baseline configuration, one that typically results in state-of-the-art speech recognition performance. Potential BMM benefits when using different features are discussed Chapter 7.

As described in Section 3.5, the inclusion of delta features often negates any improvements resulting from HMMs extended with direct dependencies between feature vectors. Deltas are included in all the experiments to verify that BMM improvements can be obtained even in this, probably more difficult, case. In total, the features with their deltas resulted in $d = 26$ element feature vectors. The feature extraction process also included cepstral mean subtraction (Deller *et al.* 1993), a pre-emphasis filter with a coefficient of 0.97, and Hamming windowed FFTs computed for each time frame. In all cases, the frame sampling rate was set to 10 ms where each frame had a window width of 25 ms.

Strictly left-to-right Markov chains are used in each experiment. This means that each HMM state has only a self-loop and an exit transition. Only a single pronunciation is used for each word. All experiments use mixture components containing only diagonal covariance matrices. The use of diagonal-covariance matrices with BMMs are not inherently less flexible than full-covariance matrices as BMM dependencies can also exist within a single frame. Test results using such "sparse" covariance matrices are not reported in this work, but they are discussed further in Section 7.3.

The reported results all use a switching variable value of unity, so $S = 1$. The observation model therefore becomes a mixture Gaussian autoregressive process with sparse dependency matrices, as described in Section 5.6. In general, the B_{qms} dependency matrix in Equation 5.17 is a function of the hidden state q, the mixture component m, and the switching variable s. For the results presented in this chapter, the structure of this matrix is shared by all mixture components. In other words, the sparse structure of each matrix changes only when the hidden state q changes. Different parameters are used for different values of m. In general, a different structure could be used for each q, m, and s if conditional mutual information quantities such as $I(X; Z|Q = q, M = m, S = s)$ were computed. Since the goal of the procedure, however, is not to discriminate between different values of m or s, the structure is a function only of q. Structures could, however, also be shared by groups of different values of q — for example, when multiple states are used for a phoneme model, each state used for a portion of a phone might share the same structure. The mutual information quantities required in this case would be $I(X; Z|Q \in \rho_i)$ where ρ_i is the set of Markov states corresponding to the i^{th} phoneme. Such sharing could increase the robustness of the conditional mutual information estimates (see Section 7.7). This type of shared structure, however, is not investigated in this work.

In the following sections, results typically include the word error rate, the number of parameters in the final system, the number of states (either per word or per phone model), and the number of Gaussian mixtures per state.

The reported number of parameters indicates the final parameters of each system. This, in fact, might be less than the number of parameters before training. This is because during EM training certain mixture components with very small covariances (e.g., small eigenvalues) are pruned away. Such covariance matrices occur for one of two reasons. First, the additional dependency variables can at times decrease the resulting variances because of the BMM's time-varying means. In other words, in a Gaussian mixture BMM, the distance between a particular frame and its "customized" mean are sometimes very small, much smaller than for an HMM. This can be seen by examining Equation 5.22. Second, a particular component might at times become extremely improbable — as a result, that component "receives" no new training data during the next iteration of EM via the component's posterior probability.

Two mechanisms were used to avoid such covariance matrices. First, the variances (diagonal elements of the covariance matrices) were "floored" if they became to small. This is done by using the corresponding variance at the previous EM iteration. Second, any mixture component whose probability falls below a threshold is eliminated. A dynamically changing threshold is used and is determined via a *mixture coefficient vanishing ratio* (MCVR). If an EM iteration resulted in a mixture component with a coefficient less than $(1/K)/\text{MCVR}$, then the component is eliminated and K is decremented accordingly. This scheme can therefore reduce the number of parameters in the system — it is somewhat similar to a complexity penalty like MDL or BIC (Burnham & Anderson 1998). The results presented in this chapter use a mixture coefficient vanishing ratio of 40.

6.2 BMM Bellcore Digits+ Results

The first test results were obtained using a small isolated word speech corpus. The Bellcore *digits+* corpus (Bellcore Digits+ Database 1990's) consists of isolated digits ("zero," "one," ..., "nine") and control words ("oh," "yes," and "no"). The corpus is telephone quality, was sampled at 8 kHz, and consists of utterances spoken in a variety of channel settings.

The word error rates (WER) reported for this case were obtained using data from 200 speakers totaling 2600 examples from 4 jack-knifed cuts – scores shown are the average of 4 tests in which 150 speakers were used for training and 50 different speakers used for testing.

The following procedure is performed independently for each cut and number of states per word. Whole-word, strictly left-to-right HMM models bootstrapped using the uniform segmental k-means procedure are created. Since the test words occur in the training set and since the corpus consists only of a small number of isolated words, whole-word models are feasible. Each HMM state in each model therefore corresponds to some not necessarily linguistic portion of the corresponding word. And since each state occurs only in one word in one position, the states can be considered context dependent although the meaning of "context" is not defined in any high-level phonetic sense like context dependent phone models typically are.

Full EM training is performed on these HMMs until convergence has been achieved. In this case, "convergence" means that the relative difference between the log likelihood of the data at two successive EM iterations has fallen below 0.2%. Using the trained models, the test set words are evaluated and word error is calculated.

Using the resulting HMMs, the Viterbi path is computed for each word determining the HMM state of each frame. Conditional mutual information is then computed using the resulting labels. The BMM dependency selection heuristic described in Section 4.6 is performed. The heuristic is slightly different in this case in that, rather than percentile thresholds specified as input parameters, the actual values are specified. This means, for example, that rather than π_{τ_u} being the value at the percentile τ_u, the quantity τ_u is the actual value in units of bits. The values of the threshold parameters are listed in the caption in Table 6.1.

For this corpus, only selection rule 4.2 (summarized in Section 4.5) was used. The set C_q for all q was set to all states except for q so the confusability refined utility was not used. Dependency links were allowed to span a maximum of 70 ms (7 frames) on either side of t. Therefore, directed cycles were possible with this structure. Other experiments were performed with this corpus where dependency variables were taken only from the past (thereby avoiding directed cycles) — the results, however, were not strongly affected by the existence of future dependency variables.

As described in the previous section, the BMMs are trained starting with the means and covariances given by the corresponding HMM and with initial dependency link values set to zero. In this experiment, forced-Viterbi training is then performed on the BMMs using the labels derived from the HMM.

HMM	BMM	HMM
1.73% (3, 10k)	0.96% (3, 19k)	1.19% (6, 20k)
1.34% (4, 14k)	0.85% (4, 26k)	0.89% (8, 27k)
1.15% (5, 17k)	0.96% (5, 32k)	1.08% (10, 34k)
1.19% (6, 20k)	0.73% (6, 39k)	1.00% (12, 41k)
	0.54% (6, 62k)	

Table 6.1: The results for the Bellcore digits+ corpus. The dependency selection parameters are $\tau_u = 5 \times 10^{-4}, \tau_q = 10^{-3}$, $\tau_g = 75\%$, $\tau_c = 5 \times 10^{-2}$, $N_q = 2$ for all q, and C_q is the set of all states except q.

Table 6.1 shows the word error results. Each entry in the table contains three numbers, the first shows the word error percentage, the second is the number of HMM states used for each word, and the third is the total number of system parameters. The table consists of three columns, the left- and right-most columns correspond to HMM word-error rates and the middle column to BMM results. In all cases, five mixture components per HMM state are used.

The left-most and center columns compare HMM and BMM performance using the same number of hidden states per word. The numbers range from three to six. As can be seen from the table, in each case a BMM yields a performance improvement over the corresponding HMM.

The center and right-most columns compares the performance of an BMM and HMM system with approximately the same number of parameters. This was done by increasing the number of hidden states per word for an HMM and then choosing the case with the most closely matching number of parameters. Again, BMMs seem to outperform HMMs, even when comparing systems with comparable numbers of parameters.

The best known word error rate for this corpus was achieved with a BMM using 61877 parameters (built by increasing the upper limit on dependency variables to $N_q = 7$) and is 0.54% using 6 states per word.

The results of this section indicate that it is possible for a BMM to produce smaller error rates than an HMM even with a comparable or fewer number of parameters and even when delta features are included in the feature stream. As argued in the beginning of Chapter 4, one possible reason for this is that a BMM augments an HMM only with those additional dependencies that fix the deficiencies found in a particular HMM.

6.3 BMM NYNEX Phonebook Results

Phonebook is a large-vocabulary "phonetically-rich isolated-word telephone-speech database" (Pitrelli *et al.* 1995) that was created to investigate real-world isolated-word recognition systems because it has repeatedly been found that continuous speech system training is not as good as isolated word training when the underlying goal is isolated word recognition.

Phonebook contains a rich collection of vocabulary words including poly-syllabic words such as "exhaustion," "immobilizing," "sluggishness," and "overambitious" as well monosyllabic words such as "awe," "biff," and "his."

In all experiments, the training and test sets were determined as defined in Dupont *et al.* (1997). The training corpus consisted of 19,421 isolated word utterances, and consisted of the following phonebook lists (Pitrelli *et al.* 1995): aa, ah, am, aq, at, ba, bh, bm, bq, bt, ca, ch, cm, cq, ct, da, dh, dm, dq, dt, and ea. The development set consisted of the eight lists: ad, ar, bd, br, cd, cr, dd, and dr. The independent "test-only-once" test set consisted of the eight lists: ao, ay, bo, by, co, cy, do, and dy. There is neither any speaker overlap nor vocabulary word overlap between these training and test sets. Therefore, it is necessary to produce embedded HMMs (or BMMs) and represent a test word as a concatenation of a series of phoneme models structured according to a pronunciation of the word. All presented results use pronunciations taken from the phonebook distribution dictionary (Pitrelli *et al.* 1995). Strictly left-to-right transition matrices were used according to the pronunciation. The pronunciations also included a silence model used both at the beginning and the end of each word.

All pronunciations in these experiments are defined using a set of 42 phonemes — 41 of the 42 phonemes defined in the phonebook distribution plus a silence model. The phonebook pronunciation dictionary includes lexical stress markings but they are not used for the current results. Also, all pronunciations using the syllabic consonant 'N' are translated into a schwa ('x') followed by an 'n'. The phoneme set used in this work is listed in Table 6.2.

1 - i - bEAt	12 - Y - bIte	23 - m - Meet	34 - p - Pea
2 - I - bIt	13 - O - bOY	24 - G - siNG	35 - t - Tea
3 - e - bAIt	14 - W - bOUt	25 - h - Heat	36 - k - Key
4 - E - bEt	15 - R - bIRd	26 - s - See	37 - b - Bee
5 - @ - bAt	16 - x - sofA	27 - S - SHe	38 - d - Day
6 - a - bOb	17 - X - buttER	28 - f - Fee	39 - g - Geese
7 - c - bOUGHt	18 - l - Let	29 - T - THigh	40 - C - CHurCH
8 - o - bOAt	19 - w - Wet	30 - z - Zoo	41 - J - JuDGe
9 - ˆ bUt	20 - r - Red	31 - Z - meaSure	42 - # - Silence
10 - u - bOOt	21 - y - Yet	32 - v - Van	
11 - U - bOOk	22 - n - Neat	33 - D - THy	

Table 6.2: Phone list used for phonebook experiments.

Each test case includes word error results for four different vocabulary sizes: 75, 150, 300, and 600. The vocabulary size determines the number of possible competing words that a given unknown test word might take on. Therefore the perplexity (Jelinek 1997) in each case is equal to the vocabulary size. It is assumed that each test word occurs in the test vocabulary so there are never any "out-of-vocabulary" words.

The development set consists of the eight phonebook lists mentioned above; each list has a number of utterances as given in Table 6.3. The independent test set sizes are given in Table 6.9. Each test list consists of a set of about 75 possible words, and no two test lists share a single word or speaker in common. The results for the 75 word vocabulary size are obtained by performing a separate recognition on all eight lists, and averaging the eight results. The results for the 150 word vocabulary size are obtained by performing a recognition on groups of two lists, and then averaging the four results. The 300 word (average of two grouped lists) and 600 word cases (all lists grouped together) are defined similarly.

Phonebook List	ad	ar	bd	br	cd	cr	dd	dr
Num. Utterances	764	672	780	1007	964	785	725	901

Table 6.3: Phonebook development set lists and list sizes.

6.3.1 No Transition Matrix Training

The first experiment compares an HMM and a BMM on the development set with training performed only on the observation models, so the transition matrices are not trained. In other words, the transition matrices are fixed and are not adjusted via the EM algorithm — Markov state exit transitions are fixed at 0.9 (and self loops are fixed at 0.1). One state per phoneme is used so there are a total of 42 Markov states. Each observation distribution is shared among any word model whose pronunciation uses the corresponding phoneme.

This first experiment evaluated dependency selection rule 4.3 (listed in Section 4.5) with confusability refined utility, and a phonetically motivated clustering. Eight phonetically derived clusters were used to define the confusable classes C_q in the utility function: silence (phone model 42), frontal vowels which possess high second (F2) and third (F3) formants and large F2-F1 distances (phoneme models 1, 2, 21, 4, 5, 9, 16, and 18), diphthongs which are characterized by spectral changes over time (phoneme models 3, 8, and 12-14), tongue retraction which is characterized by a small F2-F1 distance and high F1, and rounding characterized by a downward shift of F1, F2, and F3 (phoneme models 6,7,10,11, and 19), retroflex which have small F4-F3 distance (models 15,17, and 20), nasals which have a weak F1 (models 22-24), fricatives which have a noise like spectrum with much high frequency energy (models 25-33, 40, and 41), and plosives which have dynamic energy characteristics such as a closure followed by a sudden burst in energy (models 34-39). Of course, it is not guaranteed that such phonetically derived clusters will actually lead to confusability among the corresponding members of the statistical models. As an alternative, one would produce clusters in a data-derived way (Woodland 1991; Woodland 1992; Leggetter 1995). The utility was computed using the upper bound approximation described in Section 4.6.

As described above, an HMM baseline system bootstrapped using uniform segmental k-means was developed using the 42 phoneme models. In this experiment, each phoneme model used a mixture of 48 diagonal covariance Gaussians.

Lex. Size	75	150	300	600	Params
HMM	5.7%	7.6%	9.8%	14.1%	105k
BMM	5.1%	7.1%	9.3%	13.4%	115k

Table 6.4: Phonebook results, one Markov state per phoneme, no transition training

The results are summarized in Table 6.4. As can be seen, for each vocabulary size, a BMM outperforms the corresponding HMM. The heuristic algorithm thresholds were again specified using values rather than percentiles. The thresholds for this set are as follows: $\tau_u = 0.0$, $\eta_u = -\infty$, $\tau_q = 2.25 \times 10^{-2}$, $\tau_g = 75\%$, $\tau_c = 6.1 \times 10^{-3}$, $M = 20$, and C_q is the clustering defined above. These values were obtained by performing a search over parameter values and evaluating each result. In the above, dependency links were allowed to span a maximum of 100 ms (10 frames) only into the past. That means that directed cycles in the dependency graph do not occur. The baseline HMM system used 105k observation model parameters and the BMM used an additional 10k parameters resulting in 115k parameters.

6.3.2 One Markov State per Phoneme

The results in this section, and in Sections 6.3.3 through 6.3.5, are produced with HMMs and BMMs that include Markov transition matrix training. Word-error results are again provided for the development set. The main purpose is to demonstrate that it is possible for a BMM to outperform an HMM even when delta features are included in the feature stream.

Again, strictly left-to-right Markov chains are used. Also, the approximation to dependency selection rule 4.5 was used for the heuristic in Figure 4.11. As described in Chapter 4, this selection rule is directly related to Bayes error. In all of the following results, dependency links were allowed to span a maximum of 100ms (10 frames) only into the past. Therefore, directed cycles in the Bayesian network graph do not occur.

Unlike the previous section, the results presented here are produced with a system trained using the embedded BMM update equations listed in Appendix A. These equations are used because in order to train the Markov transitions, it is required to embed BMMs inside a larger Markov structure. Each embedded BMM corresponds to a phoneme model consisting of at least one observation distribution and a local Markov chain controlling the probabilistic sequence through the observation distributions. The number of observations per phone model is in all cases equal to the number of Markov states per phone model, i.e., the case was not evaluated where multiple Markov states are used to produce a different phone duration distribution, with all such states sharing an observation distribution.

Vocab. Size	75	150	300	600	Params	Mixtures/State
Comp. HMM	7.1%				162k	48
HMM A	5.3%	7.2%	9.5%	13.5%	105k	48
HMM B	5.0%	7.0%	9.2%	13.3%	157k	72
BMM	4.6%	6.5%	8.7%	12.3%	157k	48

Table 6.5: Phonebook results, one Markov state per phone, transition training

This section reports the case where one Markov state per phoneme is used, so there are a total of 42 Markov states. The results are shown in Table 6.5. The first row shows the result of a comparable HMM as reported in the literature (Dupont *et al.* 1997). This HMM result was produced using cepstral features, and used the same word pronunciation dictionary. It also uses double-delta (acceleration) features (Wilpon *et al.* 1991; Lee *et al.* 1991), minimum duration modeling (i.e., each phoneme corresponded to three HMM states strung together in series where each state shared the same output distribution), and 48 mixture components per state.

The results for HMM A show the new baseline HMM results. The HMM baseline reported here was significantly better than Dupont *et al.* (1997) probably because of better parameter initialization via the k-means procedure, although the exact reason for HMM A's improvement is not known. The HMM A system uses one state per phoneme, no duration modeling, and 48 components per mixture. The next row, HMM B, shows the word error performance when the number of parameters in the HMM system is increased by increasing the number of mixture components per state to 72. This was close to the maximum number of components possible with this system as initial parameter values for more components could not be found via the k-means procedure (i.e., k-means would not converge to a configuration with more than zero samples in all of the $k > 72$ clusters). Again, the EM algorithm pruned away irrelevant components using the MCVR so 72 is the maximum number of components used by this system.

The forth column shows the result of a BMM using 48 component densities per mixture. As can be seen, the BMM achieves better performance than any of the previous HMMs. It is apparent, in this case, that adding dependencies between features yields a greater performance improvement per parameter than increasing mixture components. The

BMM was derived using heuristic algorithm percentiles: $\eta_u = 0.0$, $\tau_u = 60\%$, $\tau_q = 80\%$, $\tau_c = 40\%$, $\tau_g = 75\%$, and C_q was undefined since selection rule 4.5 is used.

6.3.3 Two Markov States per Phoneme

Vocab. Size	75	150	300	600	Params	Mixtures/State
Comp. HMM	6.2%				162k	34
HMM A	2.8%	4.0%	6.2%	8.6%	105k	24
HMM B	2.4%	3.5%	5.6%	7.7%	182k	43
BMM	2.3%	3.4%	5.4%	7.5%	182k	24

Table 6.6: Phonebook results, two Markov states per phoneme, transition training

This section presents performance results when two Markov states per phoneme model are used. When the number of states per phoneme is increased, performance gets uniformly better as shown in Table 6.6.

The first row again shows a comparable HMM result again from (Dupont *et al.* 1997). This result is not exactly the same as using two states per phoneme — in that case, three observations and three states were used per phoneme. The first and third state in each phone used observations that were globally tied, the first observation being the global "entry to phoneme model" observation, and the third observation being the "exit phoneme model" observation. The middle observation is unique to each phoneme model. Although the HMM associated with the results in the first row of the table are different than the other HMMs, they are included because they most closely match the current conditions.

The second row, HMM A, shows the new baseline HMM results which are again better than the first row. These results were obtained with an HMM using 24 mixture components per state. Moving up to 48 mixture components per state, the HMM results do improve, as shown in the third row. The last row shows the BMM result with 24 mixture components per state which is again the best, and achieves its performance with fewer parameters. The BMM was derived using heuristic algorithm percentiles: $\eta_u = 0.0$, $\tau_u = 80\%$, $\tau_q = 60\%$, $\tau_c = 40\%$, and $\tau_g = 60\%$.

6.3.4 Three Markov States per Phoneme

Vocab. Size	75	150	300	600		Params	Mixtures/State
Comp. HMM	5.0%					162k	16
HMM A	1.8%	3.2%	4.9%	7.0%	+	105k	16
HMM B	1.7%	2.8%	4.6%	6.2%		163k	26
BMM	1.5%	2.6%	4.4%	6.0%		166k	16

Table 6.7: Phonebook results, three Markov states per phoneme, transition training

This section presents performance results for the case when three Markov states per phoneme model are used. Performance again uniformly increases as shown in Table 6.7.

The first row again shows a comparable HMM result from (Dupont *et al.* 1997). This HMM uses 16 components per mixture, 38 features (as described above), and three separate Markov states with different observation distributions for each phoneme model.

The second row, HMM A, shows the new baseline HMM results which are significantly better than the first row. These results were obtained with an HMM using 16 mixture components per state. Moving up to 32 mixture components per state, the HMM results do improve, as shown in the third row. But the last row shows the BMM result which is again the best. The BMM was derived using heuristic algorithm percentiles: $\eta_u = 0.0$, $\tau_u = 60\%$, $\tau_q = 60\%$, $\tau_c = 30\%$, and $\tau_g = 50\%$.

6.3.5 Four Markov States per Phoneme

Vocab. Size	75	150	300	600	Params	Mixtures/State
HMM A	1.7%	3.1%	4.9%	6.5%	105k	12
HMM B	1.5%	2.7%	4.3%	5.8%	200k	24
BMM	1.4%	2.6%	4.2%	5.6%	207k	12

Table 6.8: Phonebook results, four Markov states per phone, transition training

This section provides performance when four Markov states per phoneme model are used. Once again, performance uniformly increases over the cases where fewer states are used. The results are presented in Table 6.6.

The fist row, HMM A, shows the baseline HMM performance. These results were obtained with an HMM using 12 mixture components per state. Moving up to 24 mixture components per state, the HMM results again improve, as shown in the third row. The last row shows the BMM result using 12 mixtures per state which is again the best and uses a comparable number of parameters as HMM B. The BMM was derived using heuristic algorithm percentiles: $\eta_u = 0.0$, $\tau_u = 60\%$, $\tau_q = 60\%$, $\tau_c = 40\%$, and $\tau_g = 70\%$.

6.3.6 Independent Test Set

In the previous sections as described in Section 6.1, several sets of heuristic parameter values were evaluated to produce the reported word error performance. In this section, the HMM and the final resulting BMMs from each of the four cases above (one through four states per phoneme) are evaluated on an independent test set. The test lists and test list sizes are given in Table 6.9. In each of the following, the reported word error was produced by a recognition system that was run once and only once using the test set — no parameter tuning was done on these test sets in an attempt to produce better performance.

Again four different test cases are shown corresponding to different numbers of Markov states per phoneme. The results are presented in Tables 6.10 through 6.13. As the results show, a BMM consistently outperforms an HMM with the same number of mixture components per state. As the parameters of the HMM are increased by increasing the number of HMM mixture components per state, the performance of the HMM system improves as predicted by the theory presented in Chapter 3. In all but the last case, the BMM still has a slight advantage over the HMM. This is discussed further in Section 6.5.

Phonebook List	ao	ay	bo	by	co	cy	do	dy
Num. Utterances	922	971	734	895	918	1104	830	917

Table 6.9: Phonebook independent test set lists and list sizes.

Vocab. Size	75	150	300	600	Params	Mixtures/State
HMM A	6.3%	7.6%	11.4%	14.8%	105k	48
HMM B	6.4%	7.7%	11.4%	14.7%	157k	72
BMM	5.8%	7.2%	11.1%	14.3%	157k	48

Table 6.10: Phonebook results, one Markov state per phoneme, independent test set

Vocab. Size	75	150	300	600	Params	Mixtures/State
HMM A	3.4%	4.5%	7.4%	9.7%	105k	24
HMM B	3.0%	4.1%	7.0%	9.2%	182k	43
BMM	3.0%	4.1%	7.1%	9.1%	182k	24

Table 6.11: Phonebook results, two Markov states per phoneme, independent test set

6.4 Structural Discriminability

In this section, several of the 75 word vocabulary development set results from the previous sections are presented along with results obtained using other methods to extending the HMM's structure.

The results are presented in Table 6.14. The first five results (cases 1-5) all use one state per phoneme and 48 mixtures per state.

Case 1 shows the performance of an HMM that has been extended with additional dependencies according to selection rule 4.2 (dependency selection using just conditional mutual information). As described in Chapter 4, this rule adds dependencies such that the resulting structure has a higher potential likelihood. Once these models are trained, the likelihood scores of each model both on training data and test data dramatically increase relative to the baseline HMM (case 5). But as can be seen from the table, the performance dramatically decreases. The reason is that, although the likelihood of each word for each model is much higher, the models no longer discriminate well. Essentially, the models have been extended with structure that describes the statistics of speech but not the characteristics distinguishing one speech utterance relative to other utterances. As a result, each model reports a high likelihood score for all utterances and discriminability decreases.

Vocab. Size	75	150	300	600	Params	Mixtures/State
HMM A	2.4%	3.4%	5.7%	7.5%	105k	16
HMM B	2.3%	3.2%	5.6%	7.0%	163k	26
BMM	2.2%	3.1%	5.3%	6.7%	166k	16

Table 6.12: Phonebook results, three Markov states per phoneme, independent test set

Vocab. Size	75	150	300	600	Params	Mixtures/State
HMM A	2.3%	3.4%	5.8%	7.4%	105k	12
HMM B	2.0%	2.8%	5.0%	6.2%	200k	24
BMM	2.0%	2.9%	5.0%	6.2%	207k	12

Table 6.13: Phonebook results, four Markov states per phoneme, independent test set

Case 2 and 3 shows the performance of vector auto-regressive (or conditional Gaussian, see Chapter 3) HMMs on this corpus. AR2 (case 2) corresponds to the case where dependencies are added from the current observation variable to the two preceding observation variables. More precisely, each element of the current observation vector now also includes a dependency on the corresponding element of the two preceding observation vectors. AR1 (case 3) is the same except that a dependency is added only to the preceding observation vector. As can be seen from the results, the auto-regressive HMM performance is also very poor. This is consistent with the literature which shows that auto-regressive HMMs are beneficial only when delta features are not included in the feature stream (see Chapter 3).

The results from cases 1-3 indicate that some dependencies can be damaging — these are the ones which produce a structure with increased potential likelihood, but decreased discriminability. As mentioned above, the likelihoods reported by the models for cases 1-3 were extremely large when evaluated both on the training data and on test data which consisted of different words spoken by different speakers. This suggests that overfitting to the training data did not cause the performance decrease.

Case 4 shows the performance when a random set of dependencies between observation elements are added. Different random dependencies are added for each hidden state. Case 4 is much better than cases 1 through 3 which suggests that truly harmful and anti-discriminative dependencies have not been added. The performance, however, is still not as good as case 5 (to be described shortly) which implies that just adding dependencies arbitrarily, even if they are not the "wrong" ones, can hurt performance.

Case 5 shows a baseline HMM result given in Table 6.5. As can be seen, this result is better than all of the previous HMM extensions. Taken together, the above results imply that just adding dependencies to a model because they are missing does not necessarily improve performance.

Case 6 shows the comparable best Gaussian-mixture HMM performance using the same pronunciation dictionary as given in (Dupont *et al.* 1997). This number is the same

Case	Type	WER	Num. Params	States/Phone	Mixtures/State
1	CMI	32.0%	207k	1	48
2	AR2	27.6%	207k	1	48
3	AR1	20.9%	156k	1	48
4	RAND	8.3%	207k	1	48
5	HMM	5.3%	105k	1	48
6	Comp HMM	5.0%	162k	3	16
7	HMM	5.0%	157k	1	72
8	BMM	4.6%	157k	1	48
9	HMM	2.8%	105k	2	24
10	HMM	2.4%	200k	2	48
11	BMM	2.3%	182k	2	24
12	HMM	1.8%	105k	3	16
13	HMM	1.7&	201k	3	32
14	HMM	1.7%	105k	4	12
15	HMM	1.5%	200k	4	24
16	BMM	1.5%	166k	3	16
17	BMM	1.4%	207k	4	12

Table 6.14: 75 Word Vocabulary Size Comparisons

as what was provided in Table 6.7.

With one Markov state per phoneme, cases 5 and 7 compare an HMM and a BMM with equal numbers of mixtures per component (48 in this case). Cases 5 and 8 compare an HMM and a BMM with a comparable number of parameters (157k). In each case, the BMM outperforms the HMM.

With two Markov states per phoneme, cases 9 and 11 compare an HMM and a BMM with equal numbers of mixtures per component (24 in this case) and cases 10 and 11 compare an HMM and a BMM with a comparable number of parameters (200k for the HMM and 182k for the BMM). Again, the BMM yields a lower word error rate in either case.

With three Markov states per phoneme, the situation is similar. Cases 12 and 16 compare an HMM and a BMM with equal numbers of mixtures per component (16 in this case) and cases 13 and 16 compare an HMM and a BMM with a comparable number of parameters (201k for the HMM and 166k for the BMM). Again, the BMM yields a lower word error rate in either case, even with significantly fewer parameters.

Finally, with four Markov states per phoneme, the gap narrows between an HMM and a BMM, with the BMM still having a slight advantage. Cases 14 and 17 compare an HMM and a BMM with equal numbers of mixtures per component (12 in this case) and cases 15 and 17 compare an HMM and a BMM with a comparable number of parameters (200k for the HMM and 207k for the BMM). The BMM yields a slightly lower word error rate in both cases.

There are several general points that can be made from the information provided by Table 6.14. First, case 1 suggests that augmenting the model structure to increase the potential likelihood might actually cause the error rate to dramatically increase. Many of the methods in the statistical model selection literature (Burnham & Anderson 1998;

Linhart & Zucchini 1986) attempt to select a model that provides the best description of the data. But even if this is done in a class conditional way, and the resulting models are used for classification, the above results show that discrimination can get worse. Admittedly, model selection procedures also typically include a term which penalizes overly complex structures (e.g., MDL, BIC, etc.). But these penalties typically do not select for discriminative structures, something that seems required to achieve good classification performance. This could explain the mixed success of AR-HMMs in the past (Kenny *et al.* 1990; Ostendorf *et al.* 1996) where "dependencies" are fixed a priori without regard to their affect on information gain and discriminability.

Second, dependencies should not be added arbitrarily just because they are missing. Cases 2 and 3 adds dependencies between adjacent observation vectors, an approach sometimes justified by noting that they are not directly represented by an HMM. But as the performance for these augmented models shows, adding missing dependencies can decrease classification performance.

Third, adding random dependencies does not produce as poor performance as in the previous cases, but there is no benefit either. It is unlikely that just choosing random dependencies, even if they are different for different q, will select the discriminative dependencies because the space of possible sets of dependencies is so large.

Forth, as mentioned above, when delta features are also used in the observation stream, HMMs with extra dependencies between observations often do not provide any improvement. The results if this section show that it is possible to gain an improvement even using delta features if the right structurally discriminative set of additional dependencies are used.

Finally, the theory given in Chapter 3 argues that an HMM can approximate a distribution arbitrarily well if it has enough hidden states, mixtures per component, and training data. The results in Table 6.14 (and throughout this chapter) support this claim as each increase in the number of hidden states or the number of mixtures per component resulted in a performance improvement. The performance improvement obtained by adding more hidden states is dramatic, but on top of that the additional BMM dependencies are able to produce an additional improvement in each case. As argued in Chapter 4, a BMM adds parameters only where an HMM is found to be deficient and therefore could lead to better or equal word error performance with fewer parameters. The data also supports this claim as an HMM is often better than an HMM with an equivalent number of parameters and the same number of hidden states.

The key point of this section is that it is important to produce structurally discriminative models, at least when the parameters are trained using a maximum likelihood training procedure. Adding more hidden states increases structural discriminability as can adding BMM dependencies.

6.5 Discussion

For most of the results presented in this chapter, dependency variables are chosen only from the past and not from the present (see Section 7.3) or from the future. This avoids a MRF and the need for a global renormalization as described in Section 4.7.3. Therefore, at each time frame, each observation model is a function only of the current hidden state and previous feature frames. At the beginnings of the utterances, however, the observation models contain dependencies to time regions before the beginning of the speech waveform. It therefore might be necessary to, prior to feature extraction, prepend samples to the beginnings of each speech utterance providing enough data for these extra dependencies. For the results above, the speech signals were prepended with extra random Gaussian noise samples since specific hidden state conditions to determine beginning-of-utterance dependencies were not used. Therefore, the BMM results might have been penalized by this constraint. In the ideal case, context dependent phoneme models would be used and dependencies appropriately directed according to the hidden state.

	Min	1st Quartile	2nd Quartile	3rd Quartile	Max	Avg
1 state/phone	2k	20k	30k	45k	115k	46k
2 state/phone	700	9k	14k	25k	82k	23k
3 state/phone	450	6k	10k	16k	49k	16k
4 state/phone	300	4k	7k	12k	56k	12k

Table 6.15: Number of speech frames per CMI estimate

Regarding the results in this chapter, it is important to realize that as the number of hidden states per phoneme increases, the robustness of the conditional mutual information estimates get worse for a fixed amount of training data. This is because conditional mutual information is computed using Viterbi paths as described in Section 7.4 and Chapter C. As the number of Markov states increases, there are fewer frames of speech that are labeled as any given state. The minimum, quartiles, maximum, and average number of frames per state used for to produce conditional mutual information estimates is given in Table 6.15. As can be seen, on average about eight minutes of speech are used to produce each conditional mutual information estimate when using one Markov state per phone model. But only about 2 minutes of speech on average are used when using four states per phoneme model, a drop that could have a significant affect on the robustness of the estimate.

In general, the HMM theory presented in Chapter 3 argued that HMMs can approximate a distribution arbitrarily well given enough hidden states and parameters. The results of this chapter seem to support this claim — increasing the number of hidden states and the number of mixture components improves performance. By adding discriminative structure only where it is found to be missing, BMMs are an endeavor to produce an equal or better performing model with the same or fewer parameters. The results of this chapter appear to support this claim as well. Furthermore, as the results of Section 6.4 show, it is crucial to produce structurally discriminative models when training uses a maximum likelihood procedure, as otherwise performance can get dramatically worse.

Chapter 7

Conclusions and Future Work

Contents

In this thesis, it has been argued that statistical models should represent the natural statistical dependencies extant in real-world objects. For a classification task, it is particularly important to produce models that are structurally discriminative — each model should have the capability to represent the defining and unique attributes of each object class. It was hypothesized that such an approach can satisfy the principle of parsimony in that the resulting models represent only what is important for the classification task, not attempting to explain more than necessary.

This general approach was applied to the automatic speech recognition task. It was claimed that some criticisms of HMMs do not address inherent problems that cause errors in a speech recognition system and that HMMs are a more powerful statistical model than they are sometimes portrayed. To extend HMMs in a potentially parsimonious way, it was shown how one can exploit statistical regularity in speech to automatically determine statistical dependencies between observation elements that can be added to an HMM. Thereby, the HMM conditional independence assumptions can be relaxed in a data-driven way.

In particular, an HMM makes the assumption that $X_t {\perp\!\!\!\perp} X_{\neg t} | Q_t$ for all t. Given enough hidden states and observation distributes with large enough capacity, an HMM can at least in theory approximate any real-world distribution arbitrarily well. This might also be true in practice assuming sufficient training data. For a particular HMM, however, the validity of this conditional independence property can be measured using conditional mutual information, a quantity that can be seen as a measure of an HMM's loss. If $I(X_t, X_{\neg t} | Q_t) =$

0 then it is possible for the HMM to be accurate. If $I(X_t, X_{\neg t}|Q_t) > 0$ then an HMM can be extended with additional cross-observation dependencies to minimize the remaining loss. But because structural discriminability is the goal, discriminative versions of an HMMs loss were developed in this thesis and used to determine additional dependencies. The result is called a BMM. It was found that a BMM can outperform an HMM with fewer parameters if discriminative dependencies are used, but if "likelihood maximizing" dependencies are chosen, word-error performance can get dramatically worse.

This thesis leaves open many possible avenues of research to further explore. Some of the more promising ones are postulated and discussed in the following sections.

7.1 Discriminative Training

The structure learning schemes presented in this work attempt to produce models that are structurally discriminative, but only maximum likelihood parameter training was implemented and tested. While it was mentioned in Chapter 3 that structurally discriminative models might not need a discriminative training procedure, on the other hand, there might be an additional advantage to using both discriminatively structured and discriminatively trained models.

7.2 Simultaneous Multi-level Speech System Improvements

It has repeatedly been observed that Gaussian mixture HMMs and MFCC-based features seem uniquely suited to each other. Promising new feature extraction methods are sometimes proposed and then compared with MFCCs using a Gaussian mixture HMM-based system only to find that the MFCCs work better. One possible explanation is that the nature of the statistical properties that are represented by a particular Gaussian mixture HMM better match the properties of the MFCCs than the new features. In other words, the HMMs that yield success with MFCCs do not yield success with the new features. It would perhaps be advantageous to move to a fundamentally different HMM, or to tune the HMM to better match the statistical properties of the speech signals as represented by the new features.

More generally, as was hypothesized in Chapter 1, speech recognition systems will perhaps yield significant performance improvements when multiple components of the system are simultaneously optimized. For example, if new features are used in an otherwise fixed system, that system might be incapable of successfully using those features while a different system might find them beneficial. Alternatively, if new pronunciations are added to a dictionary using an otherwise fixed system, the increased pronunciation accuracy can be offset by greater confusability at the acoustic level.

The BMM structure learning procedure can be seen as adapting a model to represent the statistics of the speech signal as represented by the current set of features and using the current set of word pronunciations. BMMs might therefore yield greater benefits with different non-MFCC feature sets. And since BMMs possess observation distributions that have lower within-class average entropy, richer pronunciation models could be used but with a smaller confusability increase. In general, the BMM approach of automatically augmenting the statistical model can be one part of a set of improvements applied simultaneously to multiple levels of a large speech recognition system.

7.3 Sparse Inverse Covariance Matrices

In many speech recognition systems, diagonal covariance Gaussian components are used even though full covariance matrices can provide additional improvements. Another option, for example, factors the covariance matrices using an eigen-analysis and then ties different factors together (Gales 1999).

In a Gaussian distribution, zeros in the inverse covariance matrix determine conditional independence properties of the Gaussian model (Lauritzen 1996; Mardia *et al.* 1979). Another possibility therefore is to use sparse inverse covariance matrices. The sparse patterns impose a structure over the intra-frame statistical dependencies. With the right discriminative structure, the same advantages might be obtained as with any discriminatively structured model.

The BMM structure learning procedure described in Chapter 4 can also be used to produce sparse inverse covariance matrices by restricting the set $\mathcal{Z}$ in the algorithm in Figure 4.11 to contain only elements from the vector at the current time frame. Consider Equation 5.17 with $S = 1$ slightly modified so that the z variables do not contain 1 as the last element. Also, the fixed means normally contained in the right-most column of the B matrices are now explicitly represented using the quantities μ_{qm}:

$$p(x|m,z,q) = \frac{1}{(2\pi)^{d/2}|\Sigma_{qm}|^{1/2}} e^{-\frac{1}{2}(x-B_{qm}z-\mu_{qm})'\Sigma_{qm}^{-1}(x-B_{qm}z-\mu_{qm})}$$

If the set z contains elements only from x itself, the above procedure would use square B_{qm} matrices that have zeros along the diagonal. These matrices represent dependencies between the individual elements of x. These dependencies are, of course, superfluous if Σ is a full covariance matrix. Suppose, however, that the inverse covariance matrices, referred to as Λ, contains only diagonal elements. If the element of B_{qm} in the i^{th} row and j^{th} column (for $j \neq i$) is non-zero, then it says that i^{th} element of x is directly dependent on the j^{th} element of x. This can be viewed as a Bayesian network similar to the simplification of the chain rule of probability, as shown in Figure 2.5. Therefore, it is sufficient to consider only the case where B_{qm} is an upper triangular matrix with zeros along the diagonal.

The argument of the exponential above can be transformed as follows (subscripts are dropped for notational simplicity):

$$\begin{aligned}
(x + Bx - \mu)^T\Lambda(x + Bx - \mu) &= ((I + B)x - \mu)^T\Lambda((I + B)x - \mu) \\
&= (Cx - \mu)^T\Lambda(Cx - \mu) \\
&= (CC^{-1}(Cx - \mu))^T\Lambda(CC^{-1}(Cx - \mu)) \\
&= (C(x - C^{-1}\mu))^T\Lambda(C(x - C^{-1}\mu)) \\
&= (C(x - \hat{\mu}))^T\Lambda(C(x - \hat{\mu})) \\
&= (x - \hat{\mu}))^T C^T\Lambda C(x - \hat{\mu})
\end{aligned}$$

where $C = I + B$ and $\hat{\mu} = C^{-1}\mu$ is the new mean. This procedure defines new discriminatively structured sparse inverse covariance matrix $C^T\Lambda C$ where the sparse structure is determined by the B matrices used by the BMM. The Gaussian normalization constant becomes $(2\pi)^{d/2}|C^T \Lambda C|^{1/2} = (2\pi)^{d/2}|\Lambda|^{1/2}$ since $det(C) = 1$.

A matrix A is positive semi-definite if $x^T A x > 0$ for all non-zero x. Since Λ is positive semi-definite, so is the matrix $C^T \Lambda C$ since:

$$\begin{aligned} x^T(C^T \Lambda C)x &= x^T(C^T \Lambda^{1/2} \Lambda^{1/2} C)x \\ &= x^T(\Lambda^{1/2} C)^T(\Lambda^{1/2} C)x \\ &= (\Lambda^{1/2} C x)^T(\Lambda^{1/2} C x) \geq 0 \end{aligned}$$

The last form is a sum of squares which is always non-negative, so $C^T \Lambda C$ is also positive semi-definite and therefore is a valid covariance matrix for a Gaussian.

Note also that the matrix C^{-1} is guaranteed to exist since $C = I + B$ is an upper triangular matrix with ones along the diagonal, a matrix that is guaranteed to have full rank (Harville 1997).

7.4 Iterative Re-estimation of Mutual Information

The BMM boot scheme presented in Chapter 5 requires the computation of the conditional mutual information. The mutual information is obtained from frames labeled using the Viterbi path of the HMM. Once the BMM has been designed, however, a new Viterbi path can be computed and used to re-estimate the conditional mutual information. This leads to the following possible algorithm for iteratively refining a BMM.

1) Train a normal HMM using EM
2) Compute Viterbi paths using the HMM
3) Compute conditional mutual information and discriminative loss functions
4) Build a BMM as described in Section 4.7
5) Train the new BMM
6) Compute Viterbi paths using the BMM
7) Go to step 3 if the new alignment is sufficiently different.

This procedure iteratively refines the Viterbi path, the conditional mutual information, and the BMM model itself.

A possible modification of this procedure computes mutual information not using the Viterbi path (which makes hard decisions about the hidden state assignment at each frame) but instead uses the state occupation probabilities at each frame (this is called $\gamma_i(t)$ in Chapter 5).

7.5 Instantaneous Speaker Adaptation

In Figure 4.12, the Bayesian network for a BMM was presented making a distinction between the two feature streams X and Y. It was suggested that X could consist of standard speech features and that Y could consist of features that are more informative about characteristics of a particular speaker but which might be independent of the hidden variables. Such an approach is similar to the speaker adaptation procedures commonly used in large speech recognition systems (Leggetter & Woodland 1995; Leggetter 1995). In this case, however, the adaptation data changes the statistics of the observations at each time frame. Such a procedure therefore could be referred to as instantaneous speaker adaptation.

7.6 Switching Variable $S > 1$

In this work, switching Gaussian mixture BMMs were tested only with $S = 1$ in Equation 5.17. A potentially greater affect on the entropy can be achieved by setting $S > 1$.

7.7 Miscellaneous

A list of additional research topics to explore follows.

- Better dependency selection heuristic. Several gross simplifications were made to obtain the heuristic presented in Figure 4.11. As was demonstrated in Chapter 6, the selected dependencies can have a dramatic affect on recognition error. A more advanced heuristic might produce better results. And since the heuristic is only run once per training, it need not have severe computation complexity restrictions.

- The dependency selection procedure could be applied to any Bayesian network to produce an additional discriminatively structured set of dependency edges.

- As described in Chapter 1, BMMs might yield further advantages in noisy acoustic environments.

- Dependency selection rule 4.1 could be useful for producing good speech synthesis models.

- As discussed in Section 6.5, BMMs word-error results should be produced using context-dependent hidden Markov state phoneme models. Such a procedure would solve the problem where, at the beginning of utterances, dependency variables in the past correspond to random background noise. Such a hidden-state definition could perhaps also result in more robust estimates of conditional mutual information. This is because the data samples used to produce conditional mutual information (see Appendix C) for each hidden state would contain less noise from the frames prior to the beginning of an utterance.

- The BMM results presented in this thesis used dependency variables limited to no more than 100 ms into the past. There could be an additional advantage to using dependency variables at more distant temporal locations, or at a lower level of granularity.

- The heuristic listed in Figure 4.11 could be useful as a general feature selection procedure for a pattern classification task. In this case, a modification of dependency selection rule 4.5 that includes the $I(Q; \mathcal{Z})$ term would be useful.

- The test results given in Chapter 6 were obtained with BMMs that used a different structure for each q. In other words, the sparse structure of the matrices B_{qms} in Equation 5.17 would change as a function of q. It might be beneficial to first cluster the q and then share the structure among cluster members. This could reduce the amount of computation required for and increase the robustness of the conditional mutual information estimates (see Section C). One example would be to share the structure within each state of a phoneme when using multi-state phoneme models.

In summary, there are many more ideas, heuristics, and experiments that follow directly from the work presented in this thesis.

Appendix A

Embedded BMM Update Equations

This appendix describes the equations for computing the alpha and beta probabilities, and for EM parameter update, in the context of embedded BMMs. The BMMs use mixtures of Gaussian auto-regressive processes as observations (so they correspond to switching Gaussian mixtures with $S = 1$). An embedded BMM is a collection of BMMs that are concatenated together forming an implicit much larger BMM. The BMMs are concatenated together according to the Markov transition probabilities in the matrix A. The equations given here are similar to those presented in (Young *et al.* 1990's) except that here A can be an arbitrary left-to-right Markov "pronunciation" model. Therefore, the models can be trained with multiple possible pronunciations for each word.

There are K BMMs concatenated together typically indexed by k. BMM 1 and BMM K are non-emitting (i.e., they only consist of a single non-emitting Markov state) so there are $K-2$ emitting BMMs in the collection. The k^{th} BMM consists of N_k states, the first and the N_k^{th} states being non-emitting. The transition matrix of BMM k is the matrix $a^{(k)}$.

The transition probability between the non-emitting exit state of BMM i and the non-emitting entrance state of BMM j is element A_{ij} from the matrix A. In all the following equations, a left-to-right transition structure is assumed both between BMMs and within BMMs. This implies that the matrix A is upper triangular with zero diagonal entries (no BMM self loops), and that the matrices $a^{(k)}$ with i, j^{th} element $a_{ij}^{(k)}$ are also upper triangular with not necessarily zero diagonal entries (self loops allowed) except that $a_{1,1}^{(k)} = a_{N_k,N_k}^{(k)} = 0$.

The time variable is t and ranges from 1 to T. One can think of the non-emitting first state of a BMM that starts at time t as occurring at time t^- and the non-emitting last state of a BMM that ends at time t as occurring at time t^+.

For convenience, recall here the standard definition of the alpha, beta, and gamma probabilities after (Rabiner & Juang 1993; Young *et al.* 1990's).

$$\alpha_j^{(k)}(t) = p(o_1, o_2, \ldots, o_t, Q_t = k_j)$$

$$\beta_j^{(k)}(t) = p(o_{t+1}, o_{t+2}, \ldots, o_T | Q_t = k_j)$$

$$\gamma_j^{(k)}(t) = \frac{\alpha_j^{(k)}(t)\beta_j^{(k)}(t)}{\sum_i \alpha_i^{(k)}(t)\beta_i^{(k)}(t)} = p(Q_t = k_j | o_{1:T})$$

where Q_t is the hidden random variable at time t and k_j is the j^{th} state of the k^{th} BMM.

The observation vector for time t is x_t and the set of dependency variables at time t is given by z_t. The functions $b_j^{(k)}(x_t|z_t)$ is the acoustic model for state j of BMM k. The variables z_t have an implicit dependence on j (the hidden state) but this subscript is dropped for simplicity. The construct $x_t|z_t$ is abbreviated using o_t.

The most difficult aspect of these equations is caused by the existence of non-emitting states and their adjacent transition probabilities. These occur between BMMs (to define the embedded BMM Markov structure), within a BMM from the non-emitting start state to the emitting states, within a BMM from the emitting states to the non-emitting end state, and also within a BMM from the non-emitting start state to the non-emitting end state (i.e., the "tee" transition). These states and their adjacent transitions essentially distribute probability to all connecting emitting states at a given time before actually accounting for any observations at that time.

In the following, all transitions within a BMM are learned via the EM algorithm. While update equations could also be presented for A, it is assumed here that these are fixed and exist simply to define the (say pronunciation) structure.

A.1 Alpha computation

The alpha computation for the start state of each BMM is as follows. Like all start-state alpha computations, it corresponds to a time slightly earlier than its argument.

$$\alpha_1^{(k)}(1^-) = \begin{cases} 1 & \text{if } k = 1 \\ A_{1k} + \sum_{r=2}^{k-1} \alpha_1^{(r)}(1) a_{1N_r}^{(r)} A_{rk} & \text{otherwise} \end{cases}$$

For $1 < j < N_k$, the initial alpha probabilities are:

$$\alpha_j^{(k)}(1) = a_{ij}^{(k)} b_j^{(k)}(o_1) \alpha_1^{(k)}(1^-)$$

Since this next alpha computation corresponds to the non-emitting last state, it corresponds to a time slightly later than its argument.

$$\alpha_{N_k}^{(k)}(1^+) = \sum_{i=2}^{N_k-1} \alpha_i^{(k)}(1) a_{iN_k}^{(k)}$$

The following are the alpha equations for $t > 1$ and $k < K$.

$$\alpha_1^{(k)}(t^-) = \begin{cases} 0 & \text{if } k = 1,2 \\ \sum_{r=2}^{k-1} \left[\alpha_{N_r}^{(r)}((t-1)^+) + \alpha_1^{(r)}(t^-) a_{1Nr}^{(r)}\right] A_{rk} & \text{otherwise} \end{cases}$$

This next equation is like the standard alpha update equation except consideration has to be given to the entrance states at each time point.

$$\alpha_j^{(k)}(t) = \left[\alpha_1^{(k)}(t^-) a_{1j}^{(k)} + \sum_{i=j}^{N_k-1} \alpha_i^{(k)}(t-1) a_{ij}^{(k)}\right] b_j^{(k)}(o_t)$$

The final alpha update equation is for the end state.

$$\alpha_{N_k}^{(k)}(t^+) = \sum_{i=2}^{N_k-1} \alpha_i^{(k)}(t) a_{iN_k}^{(k)}$$

To manage tee transitions at time T from each BMM's non-emitting final state, several additional quantities are defined. The first is an extra alpha probability that can be thought of as occurring at time $(T+1)^-$, i.e., at time at the end of an utterance where all observations have been accounted for by previous BMMs, and it is only possible to reach the final Markov state by traveling through tee transitions.

$$\alpha_1^{(k)}((T+1)^-) = \begin{cases} 0 & \text{if } k = 1, 2 \\ \sum_{r=2}^{k-1} \left[\alpha_{N_r}^{(r)}(T^+) + \alpha_1^{(r)}((T+1)^-) a_{1Nr}^{(r)} \right] A_{rk} & \text{otherwise} \end{cases}$$

The final alpha computation is used to obtain the total observation probability.

$$\alpha^{(K)} = \sum_{k=2}^{K-1} \left[\alpha_1^{(k)}((T+1)^-) a_{1N_k}^{(k)} + \alpha_{N_k}^{(k)}(T^+) \right] A_{kK}$$

Note that $p(o_{1:T}) = \alpha^{(K)}$.

A.2 Beta computation

The following equations define the beta computation. They are analogous to the alpha computation, so there is less accompanying descriptive text given.

$$\beta_{N_k}^{(k)}(T^+) = \begin{cases} 1 & \text{if } k = K \\ A_{kK} + \sum_{r=k+1}^{K-1} \beta_{N_r}^{(r)}(T^+) a_{1N_r}^{(r)} A_{kr} & \text{otherwise} \end{cases}$$

$$\beta_i^{(k)}(T) = a_{iN_k}^{(k)} \beta_{N_k}^{(k)}(T^+)$$

$$\beta_1^{(k)}(T^-) = \sum_{j=2}^{N_k-1} a_{ij}^{(k)} b_j^{(k)}(o_T) \beta_j^{(k)}(T)$$

The following are for $t < T$.

$$\beta_{N_k}^{(k)}(t^+) = \begin{cases} 0 & \text{if } k = K, K-1 \\ \sum_{r=k+1}^{K-1} \left[\beta_1^{(r)}((t+1)^-) + \beta_{N_r}^{(r)}(t^+) a_{1Nr}^{(r)} \right] A_{kr} & \text{otherwise} \end{cases}$$

$$\beta_i^{(k)}(t) = a_{iN_k}^{(k)} \beta_{N_k}^{(k)}(t) + \sum_{j=i}^{N_k-1} a_{ij}^{(k)} b_j^{(k)}(o_{t+1}) \beta_j^{(k)}(t+1)$$

$$\beta_1^{(k)}(t) = \sum_{j=2}^{N_k-1} a_{1j}^{(k)} b_j^{(k)}(o_t)\beta_j^{(k)}(t)$$

Again, special beta quantities are needed to support tee transitions.

$$\beta_1^{(k)}((T+1)^-) = \begin{cases} 1 & \text{if } k = K \\ a_{1N_k}^{(k)}\left[\sum_{r=k+1}^{K} A_{kr}\beta_1^{(r)}((T+1)^-)\right] & \text{otherwise} \end{cases}$$

$$\beta_{N_k}^{(k)}(-1^+) = \begin{cases} 0 & \text{if } k = K, K-1 \\ \sum_{r=k+1}^{K-1}\left[\beta_1^{(r)}(1^-) + \beta_{N_r}^{(r)}(-1^+)a_{1N_r}^{(r)}\right] A_{kr} & \text{otherwise} \end{cases}$$

The earliest beta probability is computed as follows:

$$\beta^{(1)} = \sum_{k=2}^{K-1}\left[\beta_{N_k}^{(k)}(-1^+)a_{1N_k}^{(k)} + \beta_1^{(k)}(1)\right] A_{1k}$$

The final utterance probability is given by:

$$p(o_{1:T}) = \beta^{(1)} = \alpha^{(K)}$$

A.3 Update equations

This section provides the EM update equations for embedded BMMs. The update equations are for multiple utterances. There are a total of U utterances, indexed by the variable u. Each utterance contains T_u time frames. The quantity P_u is the total probability for utterance u and is defined as $P_u = p(o_{1:T_u}^u) = \alpha^{(K)u}$. The $^+$ and $^-$ superscripts are dropped in this section to save space.

The first update equation gives the new values for the transitions within a BMM.

$$\hat{a}_{ij}^{(k)} = \frac{\sum_{u=1}^{U}\frac{1}{P_u}\sum_{t=1}^{T_u-1}\alpha_i^{(k)u}(t)a_{ij}^{(k)}b_j^{(k)}(o_{t+1}^u)\beta_j^{(k)u}(t+1)}{\sum_{u=1}^{U}\frac{1}{P_u}\sum_{t=1}^{T_u}\alpha_i^{(k)u}(t)\beta_i^{(k)u}(t)}$$

The next equation is for the transition from the emitting states to the non-emitting exit state of a BMM.

$$\hat{a}_{iN_k}^{(k)} = \frac{\sum_{u=1}^{U}\frac{1}{P_u}\sum_{t=1}^{T_u}\alpha_i^{(k)u}(t)a_{iN_k}^{(k)}\beta_{N_k}^{(k)u}(t)}{\sum_{u=1}^{U}\frac{1}{P_u}\sum_{t=1}^{T_u}\alpha_i^{(k)u}(t)\beta_i^{(k)u}(t)}$$

The quantity $\mathcal{D}$ is used as a denominator by several of the update equations.

$$\mathcal{D} = \sum_{u=1}^{U}\frac{1}{P_u}\left(\sum_{t=1}^{T_u}\left[\alpha_1^{(k)u}(t)\beta_1^{(k)u}(t) + \alpha_1^{(k)u}(t)a_{1N_k}^{(k)}\left\{\sum_{r=k+1}^{K-1}A_{kr}\beta_1^{(r)u}(t)\right\}\right]\right.$$

$$\left. + \quad \alpha_1^{(k)u}(T+1)a_{1N_k}^{(r)}\left\{\sum_{r=k+1}^{K}A_{kr}\beta_1^{(r)u}(T+1)\right\}\right)$$

The next equation is for the transition from the non-emitting entry state to the emitting states of a BMM.

$$\hat{a}_{1j}^{(k)} = \frac{\sum_{u=1}^{U} \frac{1}{P_u} \sum_{t=1}^{T_u} \alpha_1^{(k)u}(t) a_{1j}^{(k)} b_j^{(k)}(o_t^u) \beta_j^{(k)u}(t)}{\mathcal{D}}$$

The next equation is for the tee transition.

$$\hat{a}_{1N_k}^{(k)} =$$

$$\frac{\sum_{u=1}^{U} \frac{1}{P_u} \left(\sum_{t=1}^{T_u} \alpha_1^{(k)u}(t) a_{1N_k}^{(k)} \left\{ \sum_{r=k+1}^{K-1} A_{kr} \beta_1^{(r)u}(t) \right\} + \alpha_1^{(k)u}(T+1) a_{1N_k}^{(r)} \left\{ \sum_{r=k+1}^{K} A_{kr} \beta_1^{(r)u}(T+1) \right\} \right)}{\mathcal{D}}$$

Given the definitions of alpha and beta from the previous section, we can define a gamma probability for each emitting state and use that to update the acoustic models, and a gamma probability for each component of each mixture model associated with each state. That is:

$$\gamma_j^{(k)u}(t) = \frac{\alpha_j^{(k)u}(t) \beta_j^{(k)u}(t)}{P_u}$$

and

$$\gamma_{jm}^{(k)u}(t) = \gamma_j^{(k)u}(t) \frac{c_{jm}^{(k)} b_{jm}^{(k)}(o_t^u)}{b_j^{(k)}(o_t^u)}$$

Where $c_{jm}^{(k)}$ is the mixture coefficient, $b_j^{(k)}(o_t^u)$ is the acoustic model, and $b_{jm}^{(k)}(o_t^u)$ is just the m^{th} component of the acoustic model for BMM k state j.

With these definitions, the BMM update equations become:

$$B_{jm}^{(k)u} = \left(\sum_{u=1}^{U} \sum_{t=1}^{T_u} \gamma_{jm}^{(k)u}(t) x_t^u z_t^{u\prime} \right) \left(\sum_{u=1}^{U} \sum_{t=1}^{T_u} \gamma_{jm}^{(k)u}(t) z_t^u z_t^{u\prime} \right)^{-1},$$

$$\Sigma_{jm}^{(k)u} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T_u} \gamma_{jm}(t)(x_t - B_{jm}^{(k)u} z_t)(x_t - B_{jm}^{(k)u} z_t)'}{\sum_{u=1}^{U} \sum_{t=1}^{T_u} \gamma_{jm}^{(k)u}(t)},$$

and

$$c_{jm}^{(k)} = \frac{\sum_{u=1}^{U} \sum_{t=1}^{T_u} \gamma_{jm}^{(k)u}(t)}{\sum_{u=1}^{U} \sum_{t=1}^{T_u} \gamma_j^{(k)u}(t)}$$

All these equations were used in the training program that produced the results listed in Chapter 6.

Appendix B

Matrix Derivative

In this section, it is shown that

$$\frac{\partial}{\partial B}(x - Bz)^T A(x - Bz) = -(A + A^T)(x - Bz)z^T$$

where A (resp. B) is an $N \times N$ (resp. $N \times K$) matrix, and x (resp. z) is an $N \times 1$ (resp. $K \times 1$) vector.

For notational simplicity, assume that:

$$\begin{aligned} \mu &= Bz \\ y &= x - \mu \\ \mu_i &= \sum_p B_{ip} z_p \end{aligned}$$

and

$$y_i = x_i - \mu_i = x_i - \sum_p B_{ip} z_p.$$

where B_{ip} is the i, p^{th} element of the matrix B and z_p is the p^{th} element of the vector z. It immediately follows that:

$$\begin{aligned} (x - Bz)^T A(x - Bz) &= \sum_{i,j} (x_i - \mu_i) A_{ij} (x_j - \mu_j) \\ &= \sum_{i,j} \left(x_i - \sum_p B_{ip} z_p \right) A_{ij} \left(x_j - \sum_q B_{jq} z_q \right) \\ &= \sum_{ij} x_i A_{ij} x_j - \sum_{ij} x_i A_{ij} \sum_q B_{jq} z_q \\ &\quad - \sum_{ij} \left(\sum_p B_{ip} z_p \right) A_{ij} x_j + \sum_{ij} \left(\sum_p B_{ip} z_p \right) A_{ij} \left(\sum_q B_{jq} z_q \right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{ij} x_i A_{ij} x_j - \sum_{ijq} x_i A_{ij} B_{jq} z_q \\
&\quad - \sum_{ijp} B_{ip} z_p A_{ij} x_j + \sum_{ijpq} B_{ip} z_p A_{ij} \sum_q B_{jq} z_q
\end{aligned}
$$

The derivative with respect to $B_{k\ell}$ for each of the above terms must therefore be found. The first term is zero since it does not involve $B_{k\ell}$. Taking the derivative of the second term yields:

$$\frac{\partial}{\partial B_{k\ell}} \sum_{ijq} x_i A_{ij} B_{jq} z_q = \frac{\partial}{\partial B_{k\ell}} \sum_i x_i A_{ik} B_{k\ell} z_\ell = \sum_i x_i A_{ik} z_\ell$$

Doing the same for the third term yields:

$$\frac{\partial}{\partial B_{k\ell}} \sum_{ijp} B_{ip} z_p A_{ij} x_j = \frac{\partial}{\partial B_{k\ell}} \sum_j B_{k\ell} z_\ell A_{kj} x_j = \sum_j z_\ell A_{kj} x_j$$

The forth term yields:

$$
\begin{aligned}
&\frac{\partial}{\partial B_{k\ell}} \sum_{ijpq} B_{ip} z_p A_{ij} \sum_q B_{jq} z_q \\
&= \frac{\partial}{\partial B_{k\ell}} \left(\sum_{(j,q)\neq(k,l)} B_{k\ell} z_\ell A_{kj} B_{jq} z_q + \sum_{(i,p)\neq(k,l)} B_{ip} z_p A_{ik} B_{k\ell} z_\ell + B_{k\ell}^2 z_\ell^2 A_{kk} \right) \\
&= \sum_{(j,q)\neq(k,l)} z_\ell A_{kj} B_{jq} z_q + \sum_{(i,p)\neq(k,l)} B_{ip} z_p A_{ik} z_\ell + 2 B_{k\ell} z_\ell^2 A_{kk}
\end{aligned}
$$

Combining all four terms leads to:

$$
\begin{aligned}
\frac{\partial}{\partial B_{k\ell}} y^T A y &= -\sum_i x_i A_{ik} z_\ell - \sum_j z_\ell A_{kj} x_j + \sum_{jq} z_\ell A_{kj} B_{jq} z_q + \sum_{ip} B_{ip} z_p A_{ik} z_\ell \\
&= z_\ell \left[-\sum_i x_i A_{ik} - \sum_j A_{kj} x_j + \sum_{jq} A_{kj} B_{jq} z_q + \sum_{ip} B_{ip} z_p A_{ik} \right] \\
&= z_\ell \left[\sum_i \left(-x_i (A_{ik} + A_{ki}) + A_{ki} \sum_q B_{iq} z_q + A_{ik} \sum_p B_{ip} z_p \right) \right] \\
&= z_\ell \left[\sum_i \left(-x_i + \sum_q B_{iq} z_q \right) (A_{ik} + A_{ki}) \right] \\
&= -z_\ell \left[\sum_i \left(x_i - \sum_q B_{iq} z_q \right) (A_{ik} + A_{ki}) \right] \\
&= \left[-(A + A^T)(x - Bz) z^T \right]_{k\ell}
\end{aligned}
$$

This last quantity is the k, ℓ^{th} element of the matrix $-(A + A^T)(x - Bz)z^T$, as desired.

Appendix C

Computation of MI and CMI

This appendix describes the computation of (conditional) mutual information given a sample of data. After making the approximations described in Section 4.6, only pairwise conditional mutual information is needed, the computation of which is described herein.

C.1 Mutual Information Computation Using EM

The mutual information between two continuous random variables X and Y is defined as (Cover & Thomas 1991):

$$I(X;Y) = \int p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy.$$

In practice, access is only given to samples $D = \{(x_i, y_i) : 1 \leq i \leq N\}$ drawn from the distributions governing X and Y. Therefore, an estimation method must be used.

One method first discretizes the samples in D, computes a 2-dimensional histogram, and then performs a discrete version of the above computation:

$$I(X;Y) = \sum_{x,y} p_d(x,y) \log \frac{p_d(x,y)}{p_d(x)p_d(y)}$$

where the p_d are the discrete histograms. Like all histogram methods, this method requires a very large sample size to accurately estimate the histogram.

Another method assumes that X and Y are jointly Gaussian distributed with correlation coefficient ρ_{xy}. If so, the quantity can be computed analytically (Kullback 1968) as $I_l(X;Y) = -\frac{1}{2}\log_2(1-\rho_{xy}^2)$ where ρ_{xy} is the correlation coefficient of X and Y. Because ρ_{xy} captures the linear dependence between X and Y regardless of their joint distribution, this estimate could be called the linear mutual information.

Another parametric option is to fit a Gaussian mixture distribution to the sampled data using, say, EM, producing an estimate $\hat{p}$ of the true distribution p of X and Y. Unfortunately, there is no analytical formula that, like in the single component case, maps from mixture parameters to mutual information. Given such a mixture distributions, however, there are a variety of ways to produce a mutual information estimate.

One way would perform some form of numerical integration method (Press *et al.* 1992). Alternatively, a law-of-large-numbers type of computation can be performed as follows:

$$I = \frac{1}{N}\sum_{i=1}^{N} \log \frac{\hat{p}(x_i, y_i)}{\hat{p}(x_i)\hat{p}(y_i)}$$

The sample used in this sum can either be the original samples used to estimate the distribution (resulting in a cross-entropy-like estimation of mutual information) or could also be drawn anew from the estimated distributions.

As yet another alternative, once $\hat{p}$ has been produced, the distribution itself can be sampled and used for the discrete form of the mutual information computation.

While the first non-parametric method is fairly simple, it suffers from several problems including the need for bias correction (Morris 1992), a large number of histogram bins, and large amounts of "training" data. Because we need to compute thousands of mutual information values, this approach is not viable since thousands of simultaneously maintained 2-dimensional histograms would be required. Also, while the linear mutual information approximation is computationally simple, it does not capture potentially important non-linear statistical dependencies contained in distributions not well approximated by a single component Gaussian.

Therefore, all of the (conditional) mutual information results in this work use the following parametric method. The EM algorithm is used to fit a 5-component full-covariance Gaussian mixture to each data set. The resulting density is sampled at points on a 250×250 evenly spaced grid. In each dimension $d = 1, 2$, the grid spans the range $[\min_i(\mu_{i,d} - 2.4\sigma_{i,d}), \max_j(\mu_{j,d} + 2.4\sigma_{j,d})]$ where $\mu_{i,d}$ and $\sigma_{i,d}$ are the mean and standard deviation of component i for dimension d. This grid is normalized and used in the discrete computation of mutual information. With one mixture component, the method produces results almost identical to linear mutual information. For a larger number of components, the resulting values almost always get larger, indicating the addition of non-linear ingredients of the true mutual information. This method also produces results that are comparable to the law-of-large numbers method but requires significantly less computation.

Bibliography

ANDERBERG, M.R. 1973. *Cluster Analysis for Applications.* 111 Fifth Avenue, New York, N.Y. 10003: Academic Press, INC.

ATAL, B. S. 1974. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America* 55.1304–1312.

ATICK, J.J. 1992. Could information theory provide an ecological theory of sensory processing? *Network* 3.

ATTIAS, H., & C.E. SCHREINER. 1997. Temporal low-order statistics of natural sounds. *NIPS* 9.

BADDELEY, R., L.F. ABBOTT, M.C.A. BOOTH, F. SENGPIEL, T. FREEMAN, E.A. WAKEMAN, & E.T. ROLLS. 1997. Responses of neurons in primary and inferior temporal visual cortices to natural scenes. *Proceedings of the Royal Society B* 264.1775–1783.

BADDELEY, R.J. 1997. The correlational structure of natural images and the calibration of spatial representations. *Cognitive Science* 21.351–372.

BAHL, L.R., P.F. BROWN, P.V. DE SOUZA, & R.L. MERCER. 1986. Maximum mutual information estimation of HMM parameters for speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 49–52, Tokyo, Japan.

BARINAGA, M. 1998. Researchers go natural in visual studies. *Science* 282.614–616.

Bellcore Digits+ Database, 1990's. Bellcore digits+ database.

BILMES, J.A. 1997. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, ICSI.

—— 1998. Data-driven extensions to HMM statistical dependencies. In *Proc. ICSLP*, Sidney, Australia.

BISHOP, C. 1995. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford.

BOLLERSLEV, T., R.F. ENGLE, & J.M. WOOLDRIDGE. 1988. A capital asset pricing model with time varying covariances. *Journal of Political Economy* 96.116–131.

Bourlard, H., 1999. Personal communication.

——, & N. Morgan. 1994. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers.

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

Breiman, L., J.H. Friedman, R.A. Olshen, & C.J. Stone. 1984. *Classification and Regression Trees*. Wadsworth and Brooks.

Brown, P.F., 1987. *The Acoustic Modeling Problem in Automatic Speech Recognition*. CMU dissertation.

Buntine, W. 1994. A guide to the literature on learning probabilistic networks from data. *IEEE Trans. on Knowledge and Data Engineering* 8.195–210.

Burnham, K.P., & D.R. Anderson. 1998. *Model Selection and Inference : A Practical Information-Theoretic Approach*. Springer-Verlag.

Chase, L. L., 1997. *Error-Responsive Feedback Mechanisms for Speech Recognizers*. CMU dissertation.

Chellappa, R. 1985. *Progress in Pattern Recognition 2*, chapter Two-Dimensional Discrete Gaussian Markov Random Field Models for Image Processing, 79–112. Elsevier.

Chickering, D.M. 1996. *Learning from Data: Artificial Intelligence and Statistics*, chapter Learning Bayesian networks is NP-complete. Springer Verlag.

Chow, C.K., & C.N. Liu. 1968. Approximating discrete probability distributions with dependence trees. *IEEE Trans. on Info. Theory* 14.

Clark, J., & C. Yallop. 1995. *An Introduction to Phonetics and Phonology*. Blackwell.

Cooper, G., & E. Herskovits. 1990. Computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 393–405.

Cover, T.M., & J.A. Thomas. 1991. *Elements of Information Theory*. Wiley.

Dagum, P., & M. Luby. 1993. Approximating probabilistic inference in Bayesian belief networks is np-hard. *Artificial Intelligence* 60.

DARPA Broadcast News Workshop, 1999. Darpa 1999 broadcast news workshop. DARPA Notebooks and Proceedings. Hilton at Washington Dulles Airport.

Davis, S.B., & P. Mermelstein. 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing* ASSP-28.357–366.

Dean, T., & K. Kanazawa. 1998. Probabilistic temporal reasoning. *AAAI* .

Deller, J.R., J.G. Proakis, & J.H.L. Hansen. 1993. *Discrete-time Processing of Speech Signals*. MacMillan.

Dempster, A.P., N.M. Laird, & D.B. Rubin. 1977. Maximum-likelihood from incomplete data via the EM algorithm. *J. Royal Statist. Soc. Ser. B.* 39.

Deng, L., M. Aksmanovic, D. Sun, & J. Wu. 1994. Speech recognition using hidden Markov models with polynomial regression functions as non-stationary states. *IEEE Trans. on Speech and Audio Proc.* 2.101–119.

——, & C. Rathinavelu. 1995. A Markov model containing state-conditioned second-order non-stationarity: application to speech recognition. *Computer Speech and Language* 9.63–86.

Derin, H., & P. A. Kelley. 1989. Discrete-index Markov-type random processes. *Proc. of the IEEE* 77.1485–1510.

Digalakis, V., M. Ostendorf, & J.R. Rohlicek. 1989. Improvements in the stochastic segment model for phoneme recognition. *Proc. DARPA Workshop on Speech and Natural Language* .

Drullman, R. 1995. Temporal envelope and fine structure cues for speech intelligibility. *Journal of the Acoustical Society of America* 97.585–592.

——, J. M. Festen, & R. Plomp. 1994a. Effect of reducing slow temporal modulations on speech reception. *JASA* 95.2670–2680.

——, J.M. Festen, & R. Plomp. 1994b. Effect of temporal envelope smearing on speech reception. *Journal of the Acoustical Society of America* 95.1053–1064.

Dubes, R.C., & A.K. Jain. 1989. Random field models in image analysis. *Journal of Applied Statistics* 16.131–164.

Duda, R.O., & P.E. Hart. 1973. *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc.

Dudley, H. 1939. Remaking speech. *Journal of the Acoustical Society of America* 11.169–177.

Dupont, S., H. Bourlard, O. Deroo, J.-M. Fontaine, & J.-M. Boite. 1997. Hybrid HMM/ANN systems for training independent tasks: Experiments on phonebook and related improvements. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

Dusenbery, D.B. 1992. *Sensory Ecology: How Organisms Acquire and Respond to Information*. W.H. Freeman and Company.

Elenius, K., & M. Blomberg. 1982. Effects of emphasizing transitional or stationary parts of the speech signal in a discrete utterance recognition system. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 535–538. IEEE.

Engle, R.F., & T. Bollerslev. 1986. Modeling the persistence of conditional variances. *Econometric Reviews* 5.1–50.

Ephraim, Y., A. Dembo, & L. Rabiner. 1989. A minimum discrimination information approach for HMM. *IEEE Trans. Info. Theory* 35.1001–1013.

——, & L. Rabiner. 1988. On the relations between modeling approaches for information sources. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 24–27.

——, & ——. 1990. On the relations between modeling approaches for speech recognition. *IEEE Trans. Info. Theory* 36.372–380.

Field, D.J. 1987. Relations between the statistics of natural images and the response properties of cortical cells. *J. Optical Soc. of America A* 4.2397–2394.

Frey, B. 1998. *Graphical Models for Machine Learning and Digital Communication*. MIT Press.

Friedman, N. 1998. The Bayesian structural EM algorithm. *14th Conf. on Uncertainty in Artificial Intelligence* .

——, D. Geiger, & M. Goldszmidt. 1997. Bayesian network classifiers. *Machine Learning* 29.131–163.

Fukunaga, K. 1990. *Introduction to Statistical Pattern Recognition, 2nd Ed.*. Academic Press.

Furui, S. 1981. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.254–272.

—— 1986a. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.52–59.

Furui, Sadaoki. 1986b. On the role of spectral transition for speech perception. *Journal of the Acoustical Society of America* 80.1016–1025.

Gales, M.J.F. 1999. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing* 7.272–281.

——, & S.J. Young. 1993. Segmental hidden Markov models. *Proc. Euro. Conf. Speech Commun. Technol. (EUROSPEECH93)* 1579–1582.

——, & ——. 1995. Robust speech recognition in additive and convolutional noise using parallel model combination. *Computer Speech and Language* 9.289–307.

Gallager, R.G. 1968. *Information Theory and Reliable Communication*. Wiley.

Geiger, D., & D. Heckerman. 1996. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence* 82.45–74.

Ghahramami, Z., & M. Jordan. 1995. Learning from incomplete data. Technical Report AI Lab Memo No. 1509, CBCL Paper No. 108, MIT AI Lab.

Ghahramani, Z. 1998. *Lecture Notes in Artificial Intelligence*, chapter Learning Dynamic Bayesian Networks. Springer-Verlag.

——, & M. Jordan. 1997. Factorial hidden Markov models. *Machine Learning* 29.

Godfrey, J. J., E. C. Holliman, & J McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 517–520.

Golub, G.H., & C.F. Van Loan. 1996. *Matrix Computations*. Johns Hopkins.

Gray, R.M., & A. Gersho. 1991. *Vector Quantization and Signal Compression*. Klewer.

Greenberg, S. 1996. Understanding speech understanding: Towards a unified theory of speech perception. In *Workshop on the Auditory Basis of Speech Perception*, ed. by William Ainsworth & Steven Greenberg, 1–8, Keele University, UK.

Grimmett, G.R., & D.R. Stirzaker. 1991. *Probability and Random Processes*. Oxford Science Publications.

Harville, D.A. 1997. *Matrix Algebra from a Statistician's Perspective*. Springer.

Haykin, S. 1989. *Modern Filters*. MacMillan.

—— 1999. *Neural Networks: A Comprehensive Foundation, 2nd Ed.*. Prentice Hall.

Heckerman, D. 1995. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft.

——, D. Geiger, & D.M. Chickering. 1994. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft.

——, A. Mamdani, & M. Wellman. 1995. Real-world applications of Bayesian networks. *Communications of the ACM* 38.

Hermansky, H. 1990. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America* 87.1738–1752.

——, & N. Morgan. 1994. RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing* 2.578–589.

Hertz, J., A. Krogh, & R.G. Palmer. 1991. *Introduction to the Theory of Neural Computation*. Allan M. Wylde.

Huang, X.D., Y. Ariki, & M. Jack. 1990. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press.

Jaakkola, T.S., & M.I. Jordan. 1998. *Learning in Graphical Models*, chapter Improving the Mean Field Approximations via the use of Mixture Distributions. Kluwer Academic Publishers.

Jain, A.K., & R.C. Dubes. 1988. *Algorithms for Clustering Data*. Englewood Cliffs, New Jersey 07632: Prentice Hall, Advanced Reference Series.

Jelinek, F. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

Jensen, F.V. 1996. *An Introduction to Bayesian Networks*. Springer.

Jordan, M.I. 1995. Why the logistic function? A tutorial discussion on probabilities and neural networks. Technical Report 9503, MIT Computational Cognitive Science.

—— (ed.) 1998. *Learning in Graphical Models*. Kluwer Academic Publishers.

——, Z. Ghahramani, T.S. Jaakkola, & L.K. Saul. 1998. *Learning in Graphical Models*, chapter An Introduction to Variational Methods for Graphical Models. Kluwer Academic Publishers.

——, & R. Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation* 6.181–214.

——, & L. Xu. 1993. Convergence results for the EM approach to mixtures of experts architectures. Technical Report 1458, MIT AI LAB.

——, & L. Xu. 1996. Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks* 8.1409–1431.

Juang, B.-H., W. Chou, & C.-H. Lee. 1997. Minimum classification error rate methods for speech recognition. *IEEE Trans. on Speech and Audio Signal Processing* 5.257–265.

Juang, B-H, & S. Katagiri. 1992. Discriminative learning for minimum error classification. *IEEE Trans. on Signal Processing* 40.3043–3054.

Juang, B.-H., & L.R. Rabiner. 1985. Mixture autoregressive hidden Markov models for speech signals. *IEEE Trans. Acoustics, Speech, and Signal Processing* 33.1404–1413.

Kadirkamanathan, M., & A.P. Varga. 1991. Simultaneous model re-estimation from contaminated data by composed hidden Markov modeling. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 897–900.

Kashyap, R.L. 1981. *Progress in Pattern Recognition*, chapter Analysis and Synthesis of Image Patterns by Spatial Interaction Models, 149–186. North-Holland.

Kenny, P., M. Lennig, & P. Mermelstein. 1990. A linear predictive HMM for vector-valued observations with applications to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38.220–225.

Konig, Y., 1996. *REMAP: Recursive Estimation and Maximization of A Posterior Probabilities in Transition-based Speech Recognition*. U.C. Berkeley dissertation.

Krause, P. 1998. Learning probabilistic networks. *Philips Research Labs Tech. Report.* .

Kullback, S. 1968. *Information Theory And Statistics*. Dover.

Langley, P., W. Iba, & K. Thompson. 1992. An analysis of Bayesian classifiers. *Proc. Tenth National Conference on Artificial Intelligence* 223–228.

Lauritzen, S.L. 1996. *Graphical Models*. Oxford Science Publications.

Lee, C.-H., E. Giachin, L.R. Rabiner, R. Pieraccini, & A.E. Rosenberg. 1991. Improved acoustic modeling for speaker independent large vocabulary continuous speech recognition. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* .

Leggetter, C.J., 1995. *Improved Acoustic Modeling for HMMs using Linear Transformations*. University of Cambridge, England dissertation.

——, & P.C. Woodland. 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language* 9.171–185.

Levin, E. 1990. Word recognition using hidden control neural architecture. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 433–436. IEEE.

—— 1992. Hidden control neural architecture modeling of nonlinear time varying systems and its applications. *IEEE Trans. on Neural Networks* 4.109–116.

Linhart, H., & W. Zucchini. 1986. *Model Selection*. Wiley.

Linsker, R. 1990. Perceptual neural organization: Some approaches based on network models and information theory. *Annual Review of Neuroscience* 13.257–281.

Logan, B.T., & P.J. Moreno. 1998. Factorial HMMs for acoustic modeling. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* .

LVCSR Workshop, 1998. 9th Hub-5 conversational speech recognition (LVCSR) workshop. DARPA Notebooks and Proceedings. Maritime Institute of Technology, Maryland.

MacDonald, I.L., & W. Zucchini. 1997. *Hidden Markov and Other Models for Discrete-valued Time Series*. Chapman and Hall.

Machine Learning, 1997. Machine learning. Kluwer Academic Publishers. 23(2/3).

MacKay, D.J.C. 1998. *Learning in Graphical Models*, chapter Introduction to Monte Carlo Methods. Kluwer Academic Publishers.

Makhoul, J. 1975. Linear prediction: A tutorial review. *Proc. IEEE* 63.561–580.

Mardia, K.V., J.T. Kent, & J.M. Bibby. 1979. *Multivariate Analysis*. Academic Press.

McAllaster, D., L. Gillick, F. Scattone, & M. Newman. 1998. Fabricating conversational speech data with acoustic models: A program to examine model-data mismatch. In *ICSLP*.

McLachlan, G.J., & T. Krishnan. 1997. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics.

Meila, M., 1999. *Learning with Mixtures of Trees*. MIT dissertation.

Morgan, N., & H. Bourlard. 1995. Continuous speech recognition. *IEEE Signal Processing Magazine* 12.

Morris, A.C., 1992. *An information-theoretical study of speech processing in the peripheral auditory system and cochlear nucleus*. Institut National Polytechnique De Grenoble dissertation.

——, J.-L. Schwartz, & P. Escudier. 1993. An information theoretical investigation into the distribution of phonetic information across the auditory spectrogram. *Computer Speech and Language* 2.121–136.

Noda, H., & M.N. Shirazi. 1994. A MRF-based parallel processing algorithm for speech recognition using linear predictive HMM. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* .

Odell, J.J., 1995. *The Use of Context in Large Vocabulary Speech Recognition*. University of Cambridge, England dissertation.

Ostendorf, M., V. Digalakis, & O. Kimball. 1996. From HMM's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Trans. Speech and Audio Proc.* 4.

——, A. Kannan, O. Kimball, & J.. Rohlicek. 1992. Continuous word recognition based on the stochastic segment model. *Proc. DARPA Workshop CSR* .

Paliwal, K.K. 1993. Use of temporal correlations between successive frames in a hidden Markov model based speech recognizer. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* II–215/18.

Papoulis, A. 1991. *Probability, Random Variables, and Stochastic Processes, 3rd Edition*. McGraw Hill.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 2nd printing edition.

Pitrelli, J., C. Fong, S.H. Wong, J.R. Spitz, & H.C. Lueng. 1995. PhoneBook: A phonetically-rich isolated-word telephone-speech database. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

Poritz, A.B. 1982. Linear predictive hidden Markov models and the speech signal. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* 1291–1294.

—— 1988. Hidden Markov models: A guided tour. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* 7–13.

Povey, D., & P.C. Woodland. 1999. Frame discrimination training of HMMs for large vocabulary speech recognition. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

PRESS, W. H., S. A. TEUKOLSKY, W. T. VETTERLING, & B. P. FLANNERY. 1992. *Numerical Recipes in C, 2nd Edition*. Cambridge University Press.

RABINER, L.R., & B.-H. JUANG. 1993. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series.

REDNER, R., & H. WALKER. 1984. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review* 26.

RIEKE, F., D. WARLAND, R.R. STEVENINCK, & W. BIALEK. 1997. *Spikes, Exploring the Neural Code*. MIT Press.

RIPLEY, B.D. 1981. *Spatial Statistics*. Wiley.

RUDERMAN, D. L. 1994. The statistics of natural images. *Network: Computation in Neural Systems* 5.517–548.

RUSSEL, S.J., & P. NORVIG. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

SAFFRAN, J. R., R.N. ASLIN, & E.L. NEWPORT. 1996. Statistical learning by 8-month-old infants. *Science* 274.1926–1928.

SAHAMI, M. 1996. Learning limited dependence Bayesian classifiers. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*.

SAUL, L., & M. RAHIM. 1998. Markov processes on curves for automatic speech recognition. *NIPS* 11.

SAUL, L.K., T. JAAKKOLA, & M.I. JORDAN. 1996. Mean field theory for sigmoid belief networks. *JAIR* 4.61–76.

SHACHTER, R.D. 1998. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams. In *Uncertainty in Artificial Intelligence*.

SMYTH, P., D. HECKERMAN, & M.I. JORDAN. 1996. Probabilistic independence networks for hidden Markov probability models. Technical Report A.I. Memo No. 1565, C.B.C.L. Memo No. 132, MIT AI Lab and CBCL.

STIRZAKER, D. 1994. *Elementary Probability*. Cambridge.

STRANG, G. 1988. *Linear Algebra and its applications, 3rd Edition*. Saunders College Publishing.

SUGA, N. 1988. Auditory neuroethology of speech processing: Complex-sound processing by combination-sensitive neurons. In *Auditory Function*, ed. by G.M. Edelman, W.E. Gall, & W.M. Cowan, 679–720. John Wiley and Sons.

—— 1996. Basic acoustic patterns and neural mechanisms shared by humans and animals for auditory perception: A neuroethologist's view. In *Workshop on the Auditory Basis of Speech Perception*, ed. by William Ainsworth & Steven Greenberg, 31–38, Keele University, UK.

TAKAHASHI, S., T. MATSUOKA, Y. MINAMI, & K. SHIKANO. 1993. Phoneme HMMs constrained by frame correlations. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* .

TITTERINGTON, D.M., A.F.M. SMITH, & U.E. MAKOV. 1985. *Statistical Analysis of Finite Mixture Distributions*. John Wiley and Sons.

TVERSKY, A. 1977. Features of similarity. *Psychological Review* 84.327–352.

VAPNIK, V. 1998. *Statistical Learning Theory*. Wiley.

VARGA, A.P., & R.K. MOORE. 1990. Hidden Markov model decomposition of speech and noise. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, 845–848, Alburquerque.

——, & ——. 1991. Simultaneous recognition of concurrent speech signals using hidden makov model decomposition. In *Proceedings Eurospeech*.

WEISS, Y. Submitted. Correctness of local probability propagation in graphical models with loops. *Neural Computation* .

WELLEKENS, C.J. 1987. Explicit time correlation in hidden Markov models for speech recognition. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* 384–386.

WILPON, J.G., C.-H. LEE, & L.R. RABINER. 1991. Improvements in connected digit recognition using higher order spectral and energy features. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* .

WOODLAND, P.C. 1991. Optimizing hidden Markov models using discriminative output distributions. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*.

—— 1992. Hidden Markov models using vector linear prediction and discriminative output distributions. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, I–509–512.

WU, C.F.J. 1983. On the convergence properties of the EM algorithm. *The Annals of Statistics* 11.95–103.

XU, L., & M.I. JORDAN. 1996. On convergence properties of the EM algorithm for gaussian mixtures. *Neural Computation* 8.129–151.

YANG, H.H., S.J.V. VUUREN, & H. HERMANSKY. 1999. Relevancy of time-frequency features for phonetic classification measured by mutual information. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing* .

YOUNG, S. 1996. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine* 13.45–56.

——, J. JANSEN, J. ODELL, D. OLLASON, & P. WOODLAND, 1990's. *The HTK Book*. Entropic Labs and Cambridge University, 2.1 edition.

ZWEIG, G., 1998. *Speech Recognition with Dynamic Bayesian Networks*. U.C. Berkeley dissertation.