IDIAP RESEARCH REPORT

# Speech Recognition with Auxiliary Information

Todd Andrew Stephenson[a]

IDIAP RR 03-28

May 2003

TO APPEAR AS

Stephenson, T. A. (2003). *Speech Recognition with Auxiliary Information*. Docteur ès Sciences thesis, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland.

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a]with the Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP), CH-1920 Martigny, Switzerand, and with the Swiss Federal Institute of Technology Lausanne (EPFL), CH-1015 Lausanne, Switzerland

# Summary

Automatic speech recognition (ASR) is a very challenging problem due to the wide variety of the data that it must be able to deal with. Being the standard tool for ASR, hidden Markov models (HMMs) have proven to work well for ASR when there are controls over the variety of the data. Being relatively new to ASR, dynamic Bayesian networks (DBNs) are more generic models with algorithms that are more flexible than those of HMMs. Various assumptions can be changed without modifying the underlying algorithm and code, unlike in HMMs; these assumptions relate to the variables to be modeled, the statistical dependencies between these variables, and the observations which are available for certain of the variables.

The main objective of this thesis, therefore, is to examine some areas where DBNs can be used to change HMMs' assumptions so as to have models that are more robust to the variety of data that ASR must deal with. HMMs model the standard observed features by jointly modeling them with a hidden discrete state variable and by having certain restraints placed upon the states and features. Some of the areas where DBNs can generalize this modeling framework of HMMs involve the incorporation of even more "auxiliary" variables to help the modeling which HMMs typically can only do with the two variables under certain restraints. The DBN framework is more flexible in how this auxiliary variable is introduced in different ways. First, this auxiliary information aids the modeling due to its correlation with the standard features. As such, in the DBN framework, we can make it directly condition the distribution of the standard features. Second, some types of auxiliary information are not strongly correlated with the hidden state. So, in the DBN framework we may want to consider the auxiliary variable to be conditionally independent of the hidden state variable. Third, as auxiliary information tends to be strongly correlated with its previous values in time, I show DBNs using discretized auxiliary variables that model the evolution of the auxiliary information over time. Finally, as auxiliary information can be missing or noisy in using a trained system, the DBNs can do recognition using just its prior distribution, learned on auxiliary information observations during training.

I investigate these different advantages of DBN-based ASR using auxiliary information involving articulator positions, estimated pitch, estimated rate-of-speech, and energy. I also show DBNs to be better at incorporating auxiliary information than hybrid HMM/ANN ASR, using artificial neural networks (ANNs). I show how auxiliary information is best introduced in a time-dependent manner. Finally, DBNs with auxiliary information are better able than standard HMM approaches to handling noisy speech; specifically, DBNs with hidden energy as auxiliary information — that conditions the distribution of the standard features and which is conditionally independent of the state — are more robust to noisy speech than HMMs are.

# Version abrégée


La reconnaissance automatique de la parole (ASR) est un grand défi, vu la grande variété des données concernées. Lorsque les types de données sont suffisamment homogènes, les modèles de Markov cachées (HMMs) sont efficaces et representent l'état de l'art. Plus récents en ASR, les réseaux dynamiques bayésiens (DBNs) utilisent des algorithmes plus génériques et plus flexibles : deux groupes d'hypothèses intrinsèques aux HMMs ne sont plus nécessaires, à savoir celles concernant les dépendance statistiques entre les variables et celles concernant les données observables. Dans le contexte des DBNs, ces deux types d'hypothèses peuvent être ajoutées ou relaxées sans changement de code.

L'objectif principal de cette thèse est donc d'examiner des situations où les DBNs peuvent être utilisés pour changer les hypothèses des HMMs, afin de construire des modèles plus robustes face à la grande variété de données utilisées en ASR. Les HMMs modèlent conjointement les variables observées et les variables d'état cachées discrètes, en imposant des contraintes aux états et aux variables observées. Les DBNs permettent de généraliser cette modélisation en incorporant d'autres variables "auxiliaires". La flexibilité apportée par les DBNs a plusieurs aspects. Premièrement, l'information auxiliaire facilite la modélisation car on peut exploiter sa corrélation avec la variable observée. Ainsi, en utilisant les DBNs l'information auxiliaire peut conditionner directement les distributions de probabilités des variables observées. Deuxièmement, certains types d'information auxiliaire ne sont que faiblement corrélés avec les variables d'état. Dans ce cas, les DBNs permettent de considérer les variables d'état comme conditionellement indépendantes des variables auxiliares. Troisièmement, comme l'information auxiliaire est souvent caractérisée par une forte corrélation entre valeur présente et valeurs passées, les DBNs peuvent utiliser des variables auxiliaires discrètes pour en modéliser les variations dans le temps. Enfin, si lors de l'utilisation d'un système DBN entraîné, l'information auxiliaire est manquante ou bruitée, il est possible de la remplacer par sa distribution *a priori* apprise pendant l'entraînement.

Dans cette thèse, j'étudie ces différents avantages des DBNs. Les informations auxiliaires j'ai analysées comprennent la position du système articulatoire, le ton de la voix, la cadence de parole et l'énergie. Je montre aussi que les DBNs incorporent mieux l'information auxiliaire que les systèmes hybrides HMM/ANN (HMM et réseaux de neurones artificiels). De plus, je montre qu'utiliser l'information auxiliaire est nettement plus efficace si l'on introduit des dépendances temporelles. Enfin, les DBNs utilisant une information auxilaire sont plus efficaces que les HMMs normaux pour la reconnaissance de parole bruitée. En particulier, utiliser l'énergie comme variable auxiliaire - indépendante de l'état et non observée lors du décodage - apporte une amélioration significative par rapport aux HMMs.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

This thesis would not have been possible without God's help to me over these many years in Switzerland. My family's love and support for me from afar has strengthened and encouraged me and has reminded me of how valuable they are to me.

My friends at IDIAP have shown much kindness and patience towards me. Their friendship, outings, meals, etc., have been a source of refreshment for me. Those of the Assemblée Evangelique de Martigny have warmly welcomed me into their fold with much hospitality during most of my time in Martigny. Their love for Jesus and for the Bible has been an example to me.

IDIAP has been kind enough to supervise the work for this thesis. My supervisor Hervé Bourlard has had a substantial impact on my research. My work also benefited directly from collaborations with Samy Bengio, Jaume Carmona Escofet, Mathew Magimai-Doss, and Andrew Morris. More generally, I have learned a lot from many discussions with colleagues at IDIAP. This work has also benefited from discussions with the research community at conferences, from reviews of papers written as part of this work, and from my extended visit to AT&T Labs-Research. My thesis jury provided a lot of valuable input for improving this thesis. Maël Guillemot, Guillaume Lathoud, and Samy Bengio provided their French translation services.

Finally, This work was carried out in the framework of the Swiss National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM)2. The NCCR are managed by the Swiss National Science Foundation on behalf of the Federal Authorities. This work was also possible through financial support both from the Swiss National Science Foundation under the grant BN_ASR (20-64172.00) and from IDIAP.

# Chapter 1

# Introduction

Variety in the speaker types, conversation styles, environment, and topics of speech is one of the primary reasons why automatic speech recognition (ASR) is such a challenging task. Regarding speaker types, we may be dealing with a male verses female or, in a more general manner, people with very different vocal tracts. Regarding conversation styles, we may be attempting to recognize speech that at times is very fast and at other times is very slow; or we may be faced to cope with someone that mumbles his speech. Regarding environment, we may be dealing with a single speaker with minimal background noise or we may be dealing with multiple (overlapping) speakers with traffic noise in the background. Regarding topics of speech, we may be dealing with a travel reservation request or we may be dealing with something as different as medical records transcription.

In light of this variety, one of the main problems facing ASR is how to deal with it, either by determining how to remove it or by having the ASR system learn how to adapt to the variety of situations that it can face. A given type of variety can often be dealt with in different stages of the recognition process, these stages being known as the feature extraction, the acoustic modeling, and the language modeling.

## 1.1 Feature Extraction

Feature extraction, at the lowest levels of automatic speech recognition (ASR), is the task of extracting the limited amount of useful information from high-dimensional data. In reducing the dimensionality of the speech signal on the typical order of 80:1, feature extractors maintain much of the characteristics of the original speech and eliminate much of the extraneous information. Nevertheless, the speech is degraded through the information loss.

One of the goals in such a process is to extract purely the information that distinguishes a given sub-unit of a word from another sub-unit. These sub-units are often represented in ASR by phonemes, which are the elemental units in human speech. For example, the pronunciation of the word "cat" is represented by the sequence of the three phonemes: /k/-/æ/-/t/. So, there will be some information which is fundamental for distinguishing between these different phonemes; the rest of the information is the "variety" that changes and should be removed. Two of the areas where researchers have attempted to remove this variety in feature extraction are in speaker-dependent information and in noise.

First, variations due to speaker-dependent information is one source of information that we would hope the feature extraction to remove. This is the case with mel-frequency cepstral coefficients (MFCCs) (Owens, 1993), where "mel-frequency" is a warping done upon the spectrum so as to give finer resolution to lower-frequencies and, thus, to simulate the sensitivities of the human ear. In cepstral analysis, an inverse Fourier transform is done on the logarithm of the spectrum; the result is that the contributions to the signal from the vocal tract and from the speaker-dependent pitch (the fundamental frequency of the signal) are, in theory, separated into the lower-ordered MFCCs and the higher-ordered MFCCs, respectively. The higher-ordered MFCCs are, therefore, discarded in an attempt to remove the speaker-dependent information. However, as I will show later on, it appears that not all of the effects of the pitch are removed during cepstral analysis. A further method for attempting to remove speaker-dependent information is in using perceptual linear prediction coefficients cepstral coefficients (PLPs), which takes psychoacoustic findings into account (Hermansky *et al.*, 1985) in processing the speech.

Second, variations in background noise from one utterance to the next are ideally removed during feature extraction. Boll (1979) presented an early form of removing noise variation during speech. He makes the assumption that the noise is locally stationary. If the signal can be separated into speech and non-speech regions (see Rabiner and Juang (1993, Section 4.2) for a discussion about speech detection), then the noise spectrum for a given speech region can be estimated using the spectrum of its preceding non-speech region. By assuming that the noise is stationary during a speech region, the estimated noise spectrum can be subtracted from the spectrum of the signal; the effect, in theory, is to have spectrum resulting purely from the clean speech. As changing (non-stationary) noise is a more realistic scenario, he reestimates the noise spectrum during each period of non-speech. In addition to removing noise variation in the spectral domain, the noise can be dealt with in the cepstral domain with cepstral mean subtraction (Atal, 1974). This also assumes that there is stationary noise in the speech and that this noise, therefore, shows up as a constant addition to each MFCC. So, for a given utterance, the mean of all the MFCCs is subtracted from each MFCC. The result is, in theory, the coefficients resulting purely from the speech signal (Owens, 1993)–as if the speech had been recorded without any noise.

Another method for dealing with background noise is found with RASTA (RelAtive SpecTrAl) PLPs. RASTA processing in PLP analysis (Hermansky *et al.*, 1992) attempts to extract purely the speech while leaving out the noise. It assumes that the characteristics of the speech spectrum alone change at a certain rate. Those characteristics that change either much slower or much faster than the rate that speech changes at are assumed to be non-speech and, hence, are filtered out (Hermansky and Morgan, 1994).

## 1.2 Acoustic Modeling

An early method of adaptation of the acoustic models to a variety of potential inputs is known as dynamic time warping (DTW) (Rabiner and Juang, 1993). In DTW, the acoustic model for a word consists of an acoustic template which is a sort of generic pronunciation of the word. The warping comes into play because the number of frames in the template is fixed and will usually not be equal to the number of frames in the test utterance. There may be more frames, in which certain of the frames in the templates will have to be replicated, or there may be fewer frames, in which case, some of the templates will have to be deleted. As a distance is computed between each given frame in the template and each given sample frame, the sequence of templates is stretched and shrunk ("warped"), as illustrated in Figure 1.1, so as to minimize the sum of all the distances across all frames.

This thesis addresses variation in ASR on the acoustic modeling level. Acoustic modeling involves taking the computed acoustic features and determining which phones and, ultimately, which words were spoken when they were produced. ASR is a complex pattern recognition task involving time-dependent sub-models which having overlapping feature distributions. Several statistical assumptions are typically made which, while sometimes counter-intuitive, render this complex task

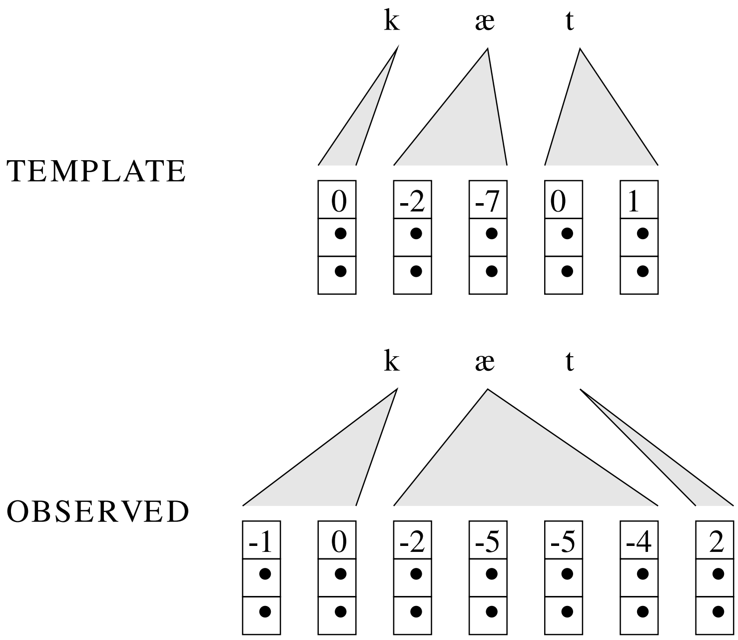


Figure 1.1: DTW: Acoustic models and dynamic time warping. On top is the template and on bottom is the observed features warped to fit the template. In this case, the /k/ is expanded by one frame; the /æ/ is expanded by two frames; and the /t/ is shrunk by one frame.

more manageable. One is that the features $X_n$ at time $1, \ldots, n, \ldots, N$ are independent through time, given a hidden state $Q_n$ (i.e., they are conditionally independent, identically distributed (c.i.i.d.), given $Q_n$); in practice, this assumption is partially-relaxed by also using approximated first and second derivatives of the features in $X_n$. Another is that, in the case of modeling $X_n$ using Gaussian mixture models (GMMs) with $J$ mixture components, the dimensions of $X_n$ are conditionally independent of each other, given $Q_n$ and the mixture component $J_n$ of the GMM. The standard model used for such a process is the hidden Markov model (HMM) (Rabiner, 1989), which, for each time frame $n$, models each mixture component of its observed features $x_n$ as being conditioned upon the hidden value of its state $Q_n$: $\sum_{j=1}^{J} p(X_n = x_n, J_n = j|Q_n)$. Furthermore, the distribution of $Q_n$ itself is conditioned upon its value at the previous time frame: $P(Q_n|Q_{n-1})$. Such a model is referred to as a first-order Markov process: the states evolve over time, depending only upon their previous value, and emit features depending only upon their current value, which will be discussed in detail in Section 2.1. Some methods discussed below for better modeling variation in acoustics include sparse time-dependencies of $X_n$, gender information, auxiliary information, and hybrid HMM/ANNs.

In making the features $X_n$ c.i.i.d. in HMM modeling, it can be difficult to model the various ways that $X_n$ can evolve over time. This is because the correlation of $X_n$ with, for example, $X_{n-1}$ can only be modeled indirectly via the hidden value of $Q_n$; hence, some important information relating $X_n$ with $X_{n-1}$ may have been lost in making this assumption if the values of $Q_n$ were not defined appropriately. Recent work in buried Markov models (BMMs) (Bilmes, 1999) has been addressing relaxing this assumption in a controlled, data-driven manner so as to keep a compact parameter set. Each individual element of $X_n$ is modeled dependent upon a small subset of elements (not necessarily from the same position in the feature vector) from the feature vectors of the recent past, $X_{n-T}, \ldots, X_{n-1}, T > 0$. The selection of these dependencies is data-driven so as to maximize the discrimination between the different possible values $1, \ldots, K$ of the hidden state $Q_n$ so that only the variations relevant to this discrimination are modeled; variations irrelevant to this discrimination are not to be modeled. Wellekens (1987) presents an earlier form of simple time-dependencies that can be used to model the evolution of $X_n$ across time; he takes the typical emission distribution $p(X_n = x_n | Q_n = q_n)$ and augments it with the features for $X_{n-1}$: $p(X_n = x_n | X_{n-1} = x_{n-1}, Q_n = q_n)$.

A simple and effective way to model inter-speaker variations in acoustic modeling is to use gender information. As male and female voices have different acoustics, such as different pitch ranges, gender-based modeling addresses these differences with a different set of acoustic models for both males and females. These models, therefore, are each more tuned to the characteristics of male and female voices. In recognition, where the gender of the speaker is unknown, the models can then be used jointly (Konig and Morgan, 1992). In this thesis, I am providing a more general framework for the incorporation of high-level information, such as gender information, into the modeling of the standard acoustic features; included in this framework is the capacity to handle this high-level information as missing during recognition. I call this high-level information "auxiliary" information, as it is not intended to be the primary information for acoustic modeling. Rather, it is higher-level information, with respect to the standard information; being correlated with standard information, it can be used to provide acoustic models that are more robust to different speaker types.

Within this framework for incorporating auxiliary information, I am focusing on the conditional independence assumption within the feature elements of a given mixture component and state at given time frame; this differs to the approach of BMMs, which addresses the c.i.i.d. assumption between the feature vectors of different time frames. The goal here is to model those dependencies which are most useful to the modeling. Likewise, if we were to model all of the dependencies, we would need a lot of data to properly learn the resulting large number of parameters; furthermore, we may be adding noise to the modeling so that the dependencies irrelevant to modeling the variation actually corrupt the modeling. Consequently, I am focusing here on the direct dependency between two groups of features: the standard features $X_n$ and the auxiliary features $A_n$. The auxiliary features are so named because of their intended secondary role in the modeling: first, they are not emitted by the state $Q_n$ (assuming they contain "high enough" information) and, hence, are not used directly in discriminating between the states; second, they are correlated with $X_n$ and, by modeling directly their dependency with $X_n$, can help give better models for $X_n$ that better handle the variations in $X_n$. In having a secondary role, these auxiliary features carry higher-level information which arises from the variations between speaker types and between conversation styles. Thus, auxiliary information can help in system adaptation, an important area within robust speech recognition.

The auxiliary variable, the focus of this thesis, was initially investigated in Zweig (1998), where it was a latent variable that was used to model "contextual" information; that is, the auxiliary variable was used to recapture some of the correlation across feature elements and across time that was lost in incorporating the standard assumptions. In studying this work, I have been using the framework of dynamic Bayesian networks (DBNs) (Cowell *et al.*, 1999; Dean and Kanazawa, 1988) for part of the study. This is also the framework that BMMs were presented in. DBNs, which are a type of graphical model (Lauritzen, 1996), model the evolution in time of a set of variables. Thus, we are assuming them to model the evolution of the states over time: $Q_{1:N} = \{Q_1, \ldots, Q_n, \ldots, Q_N\}$, of the standard features over time: $X_{1:N} = \{X_1, \ldots, X_n, \ldots, X_N\}$, and of the (optional) auxiliary features over time: $A_{1:N} = \{A_1, \ldots, A_n, \ldots, A_N\}$. As hidden Markov models (HMMs) can also model such a process, this puts HMMs and DBNs in the same family of models. The advantage of using DBNs is that they provide a more flexible framework for investigating the addition of variables to the modeling, the addition and deletion of statistical dependencies between the component variables, and the "hiding" of variables (i.e., marginalizing them out, through integration/summation, of each time-frame's distribution where it is hidden).

Artificial neural networks (ANNs) are a non-linear model that crudely simulates the operation of the human brain; one application of ANNs is to do classification of given features. Hybrid HMM/ANN ASR, which uses ANNs to model the emission distributions in the context of HMMs, can be more robust in handling the variety in speech than (GMM) HMMs. This is partly due to the fact that the ANN uses inputs of whole feature vectors covering a window of time (e.g., a window of nine frames). Given the strengths of HMM/ANN in modeling the correlation both within a feature vector and across feature vectors from nearby time frames, I also investigate auxiliary information in the context of HMM/ANN ASR.

## 1.3 Language Modeling

Language modeling (Rabiner and Juang, 1993, Sections 8.5, 8.6, & 8.7) is the task of determining the valid and likely sequences of words that have been spoken. It builds upon, and works with, the results of the acoustic modeling. It takes the various potential sequences of words from the acoustic models and assigns a probability to each sequence of words. Each probability is combined with the acoustic models' likelihood assigned to the respective sequence; this results in the overall probability of pronouncing the sequence of words with the given acoustics.

A standard approach to language modeling is to use an $n$-gram model. Given the previous $n-1$ words, an $n$-gram language model gives the probability of the current word. Common, feasible $n$-gram models are tri-grams ($n = 3$ and where $P(w^3|w^1, w^2)$ is modeled for words $w^1$, $w^2$, and $w^3$ ) and bi-grams ($n = 2$ and where $P(w^2|w^1)$ is modeled). As the value of $n$ increases, the size of the probability table holding its parameters increases exponentially. If we have a large vocabulary size, then the amount of data needed to learn an $n$-gram model with a large value for $n$ is prohibitive. Hence, some of the elements in the table will be poorly estimated and may even have a value of zero where they should have a non-zero probability. One solution to this problem is to have different $n$-gram models, for example, a tri-gram, bi-gram, and uni-gram ($n = 1$), and to use of weighted sum of all of them (Jelinek and Mercer, 1980; Rabiner and Juang, 1993, Section 8.6):

$$
\begin{aligned}
P^*(w^3|w^2, w^1) \quad = \quad & P(n = 3|N(w^2, w^1), N(w^1)) \cdot \frac{N(w^3, w^2, w^1)}{N(w^2, w^1)} \\
& + P(n = 2|N(w^2, w^1), N(w^1)) \cdot \frac{N(w^2, w^1)}{N(w^1)} \\
& + P(n = 1|N(w^2, w^1), N(w^1)) \cdot \frac{N(w^1)}{N(\cdot)},
\end{aligned} \tag{1.1}
$$

where $P(n|N(w^2, w^1), N(w^1))$ is the weight of each $n$-gram, which depends upon the counts of the joint occurrence in the training data of $w^2$ and $w^1$ held in $N(w^2, w^1)$ and upon the counts in the training data of the occurrence of $w^1$ held in $N(w^1)$, as explained in Jelinek and Mercer (1980); $N(\cdot)$ represents the total number of words in the training data.

The language model itself is dependent upon the varieties of the type of language. Using adaptation techniques with language modeling could involve, for example, first identifying the type of language (e.g., customer service requests, technical presentation, etc.) and then choosing the language model that is specified to that domain. Lagus and Kurimo (2002) used a self-organizing map and the words preliminarily recognized from the acoustics to determine which language model to use.

## 1.4 Organization

In this thesis I propose a framework to increase the recognition robustness of automatic speech recognition by using an acoustic model with both the typical acoustic feature $X_n$ and with the auxiliary information $A_n$ containing high-level information; this is done in a way such that the correlation between $A_n$ and the rest of the system is utilized advantageously and such that the system can deal with missing/unreliable $A_n$. As such, this thesis is broken down as follows.

- In Chapter 2 I discuss how time-series modeling works in the context of ASR. I review hidden Markov model (HMM) theory before discussing the more general dynamic Bayesian networks (DBNs). This chapter gives a simplified tutorial of learning in HMMs. Also, this chapter contributes to DBN theory by explaining issues regarding tractability from the viewpoint of "$d$-separation". It also provides, in Section 2.5.4, further discussion, in addition to that in the literature, on the relation between HMMs and DBNs.

- In Chapter 3 I continue the discussion of DBNs from Chapter 2 by discussing their inference process. This chapter contributes to the DBN literature by giving an explanation of propagation in mixed Bayesian networks from an implementation perspective.

- I then motivate and discuss auxiliary information, the core concept of this thesis, in Chapter 4; the purpose of auxiliary information is to provide information outside of the standard acoustics that can be used to provide more robust distributions for the standard acoustics. One of the novel approaches presented here is the use of auxiliary information only in training; however, in recognition, the auxiliary variable is hidden, and its inferred distribution is used instead in modeling. A further contribution made in this chapter is one of the few uses of factorial HMMs (implemented as a DBN) in ASR, where a second Markov chain carries auxiliary information. In general, this chapter serves as a unifying work summarizing many of the different approaches taken by myself and others for incorporating auxiliary information into ASR; as such, it includes an investigation into the different ways that auxiliary can be incorporated into the modeling.

- In Chapter 5 I review more practical information of how DBNs are actually constructed for ASR, based on Zweig (1998). I also discuss the various types of auxiliary information to be investigated in the experimental studies: articulator positions, pitch, rate-of-speech, energy, and graphemes.

- Experimental results using the DBNs and auxiliary information of Chapter 5 are given in Chapter 6. In addition to drawing together the various aspects of the thesis, these results contribute to ASR theory by showing how auxiliary features can be constructively used to aid ASR performance, particularly in noisy, spontaneous speech.

- Conclusions are drawn in Chapter 7. Additionally, ideas for further development of this work are discussed.

## 1.5 Notation

In this thesis, I use the notation of upper-case letters $A, B, C$ to refer to random variables (continuous or discrete) or to sets of variables. and lower-case letters $a, b, c$ to refer to actual observations of variables. A subscript refers to the point in time that the variable is instantiated (e.g., $A_n$) while a subscript with two colon separated items, refers to a range of instantiations in time of the variable (e.g., $A_{1:n}$). When referring to discrete variables, an upper-case letter (e.g., $K$) refers to the maximum number of values that it can take while the respective lower-case letter (e.g., $k$) refers to any given value of the variable.

# Chapter 2

# Time Series Modeling for ASR

Time series modeling in the context of ASR assumes that the system is in a certain state (discrete-valued or continuous-valued) at each given (discrete) point in time. As we consider this to be a "dynamic" problem, the evolution of the values of the state is time-dependent: the value of the state at time $n$ is directly influenced by the value of the state at time $n-1$. The common, basic model for doing time series modeling in the ASR framework is the hidden Markov model (HMM). This theory underlying HMMs can be generalized to that of dynamic Bayesian networks (DBNs). So, in this chapter I first review HMM theory in Section 2.1, where I present an introduction to learning the parameters of an HMM. This is followed in Section 2.2 by a discussion of the potential problems involved with HMM assumptions. Section 2.3 touches upon artificial neural network (ANN) theory, which can be incorporated into HMMs to make HMM/ANN hybrids. I then explain DBNs in Section 2.4, where I include my own explanation of complexity issues in mixed DBNs (having continuous and discrete variables) from the viewpoint of "$d$-separation." I then make the bridge between HMMs and DBNs in Section 2.5, where I review some of the issues in the literature as well as contributing my own points on their relation. The discussion on probabilistic inference in mixed DBNs will be presented later in Chapter 3.

## 2.1 HMM Framework

Given a signal $s$ for an utterance, the first step in the recognition process is to transform it into a series of features for each successive time frame $n$, where it is the acoustic feature $x_n$ that carries the relevant information from $s$ for doing speech recognition. So, given the features $x_{1:N} = \{x_1, \ldots, x_n, \ldots, x_N\}$ for all time frames $n = 1, \ldots, N$, they are then compared against a series of recognition models and classified according to the one that it most closely resembles. In current speech recognition systems, these models are typically hidden Markov models (HMMs). Being a generative model, an HMM $m$ will give the likelihood that it generated the features $x_{1:N}$:

$$p(X_{1:N} = x_{1:N} | \lambda_m), \tag{2.1}$$

given the parameters $\lambda_m$ for model $m$.

In the HMM framework we are dealing with a finite number of models $\{1, \ldots, m, \ldots, M\}$ according to which we classify a given set of features $x_{1:N}$, using the likelihoods $p(X_{1:N} = x_{1:N}|\lambda_m)$ associated with each model $m$, $1 \leq m \leq M$. Note that $x_{1:N}$ may have non-zero likelihood for many different models $m$. However, a given model $m$ tends to have certain values for its features $x_{1:N}$; assuming that each model typically produces different features from each other, we can recognize the different models with HMMs.

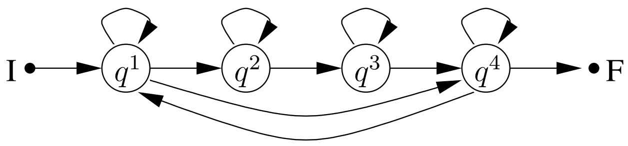


Figure 2.1: Four state Markov chain, with initial and final states, with transitions $< q^1, q^1 >, < q^1, q^2 >, < q^1, q^4 >, < q^2, q^2 >, < q^2, q^3 >, < q^3, q^3 >, < q^3, q^4 >, < q^4, q^1 >, < q^4, q^4 >$.

Each model $m$, moreover, is typically composed of several states, or sub-models, which are in common with the other models. These states are denoted $Q$ and take on the value $q^k$, where $1 \leq k \leq K$. Hence model $m$ is composed of $N(m)$ (non-unique) instantiations of $Q$: $\{q^{m_1}, \ldots, q^{m_{N(m)}}\}$. Furthermore, the temporal ordering of these states can be restricted; for example, we can impose a "left-to-right" model in which if the model is exiting state $q^{m_i}$ at time $n$, then the state of the model at time $n+1$ must have value $q^{m_j}$, where $j \geq i$. Each value $k$ of $Q_n$, the state at time $n$, has its own local likelihood, $p(X_n = x_n | Q_n = k)$, if it is an emitting state, which contributes to the likelihood $p(X_{1:N}|\lambda_m)$ (an emitting state produces a feature vector, as opposed to a non-emitting null state, which is used to indicate the initial and final states of the model at time $n = 0$ and $n = N + 1$, respectively). That is, $p(X_{1:N}|\lambda_m)$ is computed, in part, by the product of $p(X_n = x_n|Q_n = k)$ over all time frames. An HMM is 'hidden' due to the fact that, for the $N(m)$ states of model $m$, we are uncertain as to which of the valid states $q^{m_1}, \ldots, q^{m_{N(m)}}$ is associated with each feature $x_1, \ldots, x_N$. Given that the value of the state $Q_n$ is hidden, algorithms must be introduced to account for all of the possible state values of $Q_n$, which will be discussed below in the context of the Baum-Welch training in Section 2.1.2.

I now proceed to present the overall structure and parameters of an HMM in Section 2.1.1. One of the methods for learning the parameters, the expectation-maximization algorithm, will be presented for HMMs in Section 2.1.2. Given the trained HMM, Section 2.1.3 presents how to use it for recognition.

### 2.1.1 Hidden Markov Model Structure

An HMM consists, first of all, of the set of state variables in time: $Q_1, \ldots, Q_N$; for each point in time, the HMM state $Q_n$ is considered to have value $q^k$, $1 \leq k \leq K$ (I abbreviate $q^k$ simply as $k$). Given these states, we then specify transitions between them and, thus, create a Markov chain; this allows the HMM to change the value of its state as time progresses. For example, we can have the set of states $\{q^1, q^2, q^3, q^4\}$; we can then form this into a Markov chain by adding the transitions as illustrated in Figure 2.1. Given this Markov chain, we then specify "observation" vectors that can be "emitted" depending upon the value of the state at each time frame, thus creating a Markov model (as illustrated in Figure 2.2), with a probability associated with the state sequence and emitted observations. If it is not always possible to determine which value the state was in for each observation vector, then the Markov model is known as a *hidden* Markov model (see Figure 2.3).

While there are a variety of HMMs, the type that is typically used in ASR has the following items:

- set of states $1, \ldots, k, \ldots, K$. The state at time frame $n$ is represented by the variable $Q_n$. In basic ASR, each emitting state (that which produces an observation $x_n$) represents either (a part of ) a phoneme within a word or a non-speech part of the signal (e.g., silence); a non-emitting state does not have any associated feature vector (for example, states 'I' and 'F' in Figures 2.1, 2.2, & 2.3 are non-emitting states used to specify that the model must start in state 'I' and end in state 'F').

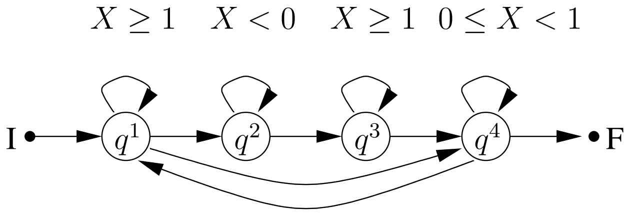


Figure 2.2: Four state Markov model, with initial and final states. It is based on Figure 2.1 but with emissions added. The sequence $\{1, -1, 1, 0, 1, 0\}$ can only come from the state sequence $\{I, q^1, q^2, q^3, q^4, q^1, q^4, F\}$.

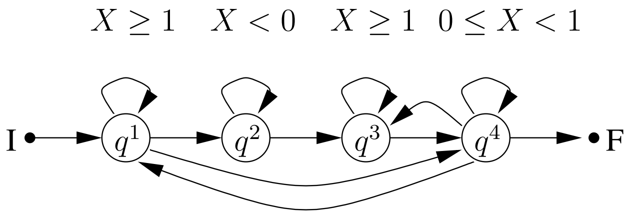


Figure 2.3: Four state hidden Markov model, with initial and final states. It is based on Figure 2.2 but with the transition $< q^4, q^3 >$ added. The sequence $\{1, -1, 1, 0, 1, 0\}$ can come from different state sequences: $\{I, q^1, q^2, q^3, q^4, q^1, q^4, F\}$ or $\{I, q^1, q^2, q^3, q^4, q^3, q^4, F\}$.

- set of emission distributions $P = \{p_1(\cdot), \ldots, p_k(\cdot), \ldots, p_K(\cdot)\}$, one for each state.
- a transition matrix R of size $K \times K$, indicating the probability of changing to a different state, i.e., $R(k, k') = P(Q_n = k' | Q_{n-1} = k)$, and $\sum_{k'=1}^{K} R(k, k') = 1 \, \forall k$. That is, the sum of any column is 1.

At a time $n$, the state $k$ emits an observation vector $x_n$, with the likelihood specified by $p_k(X_n = x_n)$. Being in state $k$ at time $n$, the probability of being in state $k'$ at time $n + 1$ is $R(k, k')$. For time $n = 1, \ldots, N$, we, hence, are modeling the evolution of the hidden discrete states $Q_{1:N} = \{Q_1, \ldots, Q_n, \ldots, Q_N\}$ and the corresponding observations $x_{1:N} = \{x_1, \ldots, x_n, \ldots, x_N\}$:

$$p(Q_{1:N}, x_{1:N}). \tag{2.2}$$

By summing over all of the possible values of $Q_{1:N}$ in (2.2), we can obtain the likelihood of $x_{1:N}$; alternately, by instantiating $Q_{1:N}$ with $q_{1:N}$, we can obtain the likelihood of the given "path" $q_{1:N}$.

### 2.1.2 Training Hidden Markov Models

#### Model Formation

When training an HMM, we are typically given a list of utterances and a list of corresponding labels with the text of each utterance. These utterances could be individual words or could be one or more sentences. Sometimes the labels are given with time stamps, thus indicating which words and/or phones were being spoken at each point in time; this is referred to as the segmentation. To use such information for training, we need to build word and/or phoneme models using the individual low-level HMM states (Rabiner and Juang, 1993, Chapter 8).

If we are given a segmentation according to, for example, which words are spoken at given points in time, we can first extract the speech corresponding to each word and run the training on a model for each of the words. To form each word model, we either need the labeling of the word according to its phonemes or we need a lexicon (a dictionary) which gives which phonemes, in order, that are used to pronounce the word (assume here that there is only one valid pronunciation for each word). So, to form the word model, the state(s) that correspond to each phoneme are concatenated together (typically as a left-to-right model) to form the word model. By using these word models for training, the component phoneme models will be trained.

After training according to a segmentation, or even if we are not given a segmentation, we can run training on the whole utterance level. This is formed, first, by forming the word models by concatenation of the phoneme models, as explained above. The utterance model is then formed by concatenation of these word models along with any possible silence or non-speech models. The resulting utterance models are thus used in training. Being composed, ultimately, of a series of state models representing (parts of) phonemes, we are, likewise, learning the distributions and transitions of these states.

### Expectation-Maximization Algorithm in HMMs

If we knew the values of all of the variables, $X_{1:N}$ and $Q_{1:N}$, then the maximum likelihood estimates of the parameters for the transition distribution $P(Q_n|Q_{n-1})$ and the emission distribution $p(X_n|Q_n)$ could be easily learned. That is, the discrete probabilities $P(Q_n = k'|Q_{n-1} = k)$ can be learned by counting the occurrences $N(Q_n = k, Q_{n-1} = k)$ and $N(Q_n = k)$, where $N(\cdot)$ signifies the number of times the indicated variable assignments occurred in the data. The maximum likelihood estimate for the transitions $R(k, k')$ would then be calculated as:

$$P(Q_n = k'|Q_{n-1} = k) = \frac{N(Q_n = k', Q_{n-1} = k)}{N(Q_{n-1} = k)} \tag{2.3}$$

Likewise, the parameters for the probability distribution for the continuous variables $X_n$ for each state $k$ can be learned by using only those data points $(X_n, Q_n)$ where $Q_n = k$. For example, if $p(X_n|Q_n = k)$ were modeled with a single Gaussian, its parameters $\mu_k$ and $\Sigma_k$ would could then simply be estimated, using standard maximum likelihood estimators of the mean and variance of a distribution (Papoulis, 1991, Section 5.3), as

$$\mu_k = \frac{1}{N(Q_n = k)} \sum_{n, \forall q_n = k} x_n \tag{2.4}$$

$$\Sigma_k = \frac{1}{N(Q_n = k)} \sum_{n, \forall q_n = k} (\mu_k - x_n)(\mu_k - x_n)^T. \tag{2.5}$$

However, since the values $q_{1:N}$ are hidden in an HMM, there is no closed-form equation for estimating these parameters. This leaves us the problem of estimating the distribution of $P(Q_{1:N})$; as each $Q_n$ has $K$ possible values, this results in dealing with $K^N$ different state sequences. If we had not made the assumptions inherent to an HMM framework, this would have been a difficult task.

The expectation-maximization (EM) algorithm (Dempster *et al.*, 1977), as expressed in the the Baum-Welch algorithm for HMMs, is one common method that takes advantage of an HMM's statistical assumptions for efficiently learning an HMM's parameters with hidden $Q_{1:N}$. It uses parameters $\lambda^i = \{R^i, \{\mu_k^i\}_{k=1}^K, \{\Sigma_k^i\}_{k=1}^K\}$. For simplicity here, I explain the training using a single Gaussian per state; in practice, however, a Gaussian mixture model (GMM) is used, being re-estimated according to Rabiner and Juang (1993, Section 6.6). EM is an iterative process where we try to compute parameters $\lambda^{i+1}$ whose log data likelihood is at least as great as the data likelihood of a model with parameters $\lambda^i$:

$$\log p(X_{1:N} = x_{1:N}|\lambda^{i+1}) \geq \log p(X_{1:N} = x_{1:N}|\lambda^i), \tag{2.6}$$

as explained in Rabiner and Juang (1993, Section 6.4.3). This is, in practice, updated using the joint likelihood of $x_{1:N}$ with the hidden states $Q_{1:N}$:

$$p(X_{1:N} = x_{1:N}, Q_{1:N}|\lambda^i). \tag{2.7}$$

As explained in Dempster *et al.* (1977, Section 2), a simple way to (locally) maximize (2.6) is to maximize the expectation of the logarithm of (2.7):

$$\begin{aligned} &E\{\log p(X_{1:N} = x_{1:N}, Q_{1:N}|\lambda^{i+1})|X_{1:N} = x_{1:N}, \lambda\} \\ &\qquad \geq E\{\log p(X_{1:N} = x_{1:N}, Q_{1:N}|\lambda^{i})|X_{1:N} = x_{1:N}, \lambda\}. \end{aligned} \tag{2.8}$$

As discussed in Rabiner and Juang (1993, Section 6.3.4), increasing the expectation (2.8) implies an increase in the likelihood (2.6); furthermore, a (local) maximization of (2.8) implies a (local) maximization of (2.6). If we had not been in the HMM framework, such a maximization problem would have involved operations of complexity $O(K^N)$, taking into account all unknown values of $Q_{1:N}$; however, by being in the HMM framework, (2.8) can easily be factored such that the complexity of the operations is reduced to $O(K + K \cdot N + K^2 \cdot N)$ (Rabiner and Juang, 1993, Section 6.4.3), where there are $K$ parameters for a prior $\pi = Q_0$, $K \cdot N$ sets of parameters for $X_1, \ldots, X_N$ and $K^2 \cdot N$ parameters for the conditionals $Q_1|Q_0, \ldots, Q_N|Q_{N-1}$. Specifically, in comparison with computing $K^N$ elements in (2.7), we are left to computing the joint of $X_{1:N} = x_{1:N}$ with individual elements in time $Q_n$ and with neighboring elements in time $(Q_{n-1}, Q_n)$:

$$p(Q_n, X_{1:N} = x_{1:N}|\lambda^i) \tag{2.9}$$

$$p(Q_{n-1:n}, X_{1:N} = x_{1:N}|\lambda^i). \tag{2.10}$$

The main task, therefore, in EM training is to estimate (2.9) and (2.10). First, to estimate (2.9), two recursions are used together to compute this joint distribution: the forward algorithm and the backward algorithm (Rabiner and Juang, 1993, Section 6.4.1). The forward algorithm considers the past and current observations $x_{1:n}$ while the backward algorithm considers the future observations $x_{n+1:N}$, as explained in more detail below.

This forward recursion at time $n$ is done so as to obtain the current likelihood of $x_n$ being emitted by state $k$, considering only the past and current frames as well as the parameters $\lambda^i$. This is obtained recursively by using the previous likelihoods (those at time $n-1$) of all of the states; by summing over the the previous likelihoods, weighted by the transition probability of their respective state into state $k$, and then by multiplying it with the emission likelihood of the current frame by state $k$, we obtain the desired likelihood. It is initialized at time 0 using $\pi$ (the prior distribution over all the states) and then continues "forward" in time, as follows:

$$\alpha_k^i(0) = \pi(k) \tag{2.11}$$

$$\alpha_k^i(n) = p(Q_n = k, X_{1:n} = x_{1:n}|\lambda^i) \tag{2.12}$$

$$= p_k^i(X_n = x_n) \sum_{k'=1}^{K} \alpha_{k'}^i(n-1) R^i(k', k), \tag{2.13}$$

where $p_k^i(\cdot)$ is the emission distribution for state $k$ with parameters $\lambda^i$.

This backward recursion at time $n$ is done so as to obtain the likelihood of the future frames, given that we are currently in state $k$ with parameters $\lambda^i$. This is obtained recursively by using its future values (those at time $n+1$) of all of the states; by summing over these future values, weighted both by the transition probability of state $k$ into their respective state and by the emission likelihood of the respective state for frame $x_{n+1}$, we obtain the current "future" likelihood. It is initialized at time $N$ with probabilities of 1 and then continues "backward' in time, as follows:

$$\begin{aligned}\beta_k^i(N) &= 1, \forall k &(2.14)\\ \beta_k^i(n) &= p(X_{n+1:N} = x_{n+1:N}|Q_n = k, \lambda^i) &(2.15)\\ &= \sum_{k'=1}^{K} \beta_{k'}^i(n+1) p_{k'}^i(X_{n+1} = x_{n+1}) R^i(k, k'). &(2.16)\end{aligned}$$

As each $\alpha_k^i(N)$, for all states $k$, $1 \leq k \leq K$, contains the likelihood of each state at time $N$, given both $x_{1:N}$ and $\lambda^i$, we can obtain the likelihood of all $x_{1:N}$, given $\lambda^i$, by summing over all of the values for $\alpha_k^i(N)$ at the final frame $N$. With such, we can compute the data likelihood with only one recursion (the forward recursion):

$$\mathcal{L}(X_{1:N} = x_{1:N}|\lambda^i) = \sum_{k=1}^{K} \alpha_k^i(N). \tag{2.17}$$

After computing the likelihood with the forward recursion, the posteriors can be computed with a second pass through the model with the backward recursion. Then, the product of $\alpha_k^i(n) \cdot \beta_k^i(n)$ gives the joint probability of (2.9).

$$\begin{aligned}\alpha_k^i(n) \cdot \beta_k^i(n) &= p(Q_n = k, X_{1:n} = x_{1:n}|\lambda^i) \cdot p(X_{n+1:N} = x_{n+1:N}|Q_n = k, \lambda^i) &(2.18)\\ &= p(Q_n, X_{1:N} = x_{1:N}|\lambda^i). &(2.19)\end{aligned}$$

The $\alpha$ and $\beta$ values can be further used to compute (2.10), as shown in Rabiner and Juang (1993, Section 6.4.3.1). The posterior of being in state $k$ at time $n$, given $x_{1:N}$, can then be computed using (2.9), which is then used for counting the number of times that we estimate state $k$ to have occurred.

$$\begin{aligned}P(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i) &= \frac{p(Q_n = k, X_{1:N} = x_{1:N}|\lambda^i).}{\sum_{k'=1}^{K} p(Q_n = k', X_{1:N} = x_{1:N}|\lambda^i)} &(2.20)\\ N(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i) &= \sum_{n=1}^{N} P(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i) &(2.21)\end{aligned}$$

Similarly, we have

$$\begin{aligned}&P(Q_n = k', Q_{n-1} = k|X_{1:N} = x_{1:N}, \lambda^i)\\ &\qquad = \frac{p(Q_n = k', Q_{n-1} = k, X_{1:N} = x_{1:N}|\lambda^i).}{\sum_{k''=1}^{K} \sum_{k'''=1}^{K} p(Q_n = k''', Q_{n-1} = k'', X_{1:N} = x_{1:N}|\lambda^i)} &(2.22)\end{aligned}$$

$$N(Q_n = k', Q_{n-1} = k|X_{1:N} = x_{1:N}, \lambda^i) = \sum_{n=1}^{N} P(Q_n = k', Q_{n-1} = k|X_{1:N} = x_{1:N}, \lambda^i) \tag{2.23}$$

(2.20), (2.21), (2.22), and (2.23) can then be used to update the parameters for $\lambda^i$:

$$\begin{aligned}R(k, k') &= \frac{N(Q_n = k', Q_{n-1} = k|X_{1:N} = x_{1:N}, \lambda^i)}{N(Q_{n-1} = k|X_{1:N} = x_{1:N}, \lambda^i)} &(2.24)\\ \mu_k^i &= \frac{\sum_{n=1}^{N} P(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i)\, x_n}{N(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i)} &(2.25)\\ \Sigma_k^i &= \frac{\sum_{n=1}^{N} P(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i)\, (\mu_k^i - x_n)\, (\mu_k^i - x_n)^T}{N(Q_n = k|X_{1:N} = x_{1:N}, \lambda^i)}, &(2.26)\end{aligned}$$

which are explained in more detail in Rabiner and Juang (1993, Section 6.4.3.1).

### 2.1.3 Recognition using Hidden Markov Models

Recognition with HMMs involves a set of models $1, \ldots, m, \ldots, M$ and features $x_{1:N}$ for a given utterance. As done in training in Section 2.1.2, these models could be formed by concatenating phoneme models into word models and word models into sentence models. The task is to determine which model $m$ was the most likely candidate to have generated $x_{1:N}$. The utterance that $m$ represents is the recognition result. That is, we are finding the utterance associated with

$$\arg\max_{1 \leq m \leq M} p(X_{1:N} = x_{1:N} | \lambda_m) \tag{2.27}$$

$p(X_{1:N} = x_{1:N} | \lambda_m)$ can be computed using the forward recursion of the Baum-Welch algorithm, producing the likelihood, as given in (2.17), of the model $m$ (Rabiner, 1989). This likelihood takes into account every possible instantiation of the states $Q_{1:N}$. If $M$ is small, we can compute the actual likelihood of $x_{1:N}$ for each model so as to use it in making our recognition decision in (2.27). One method for approximating $p(X_{1:N} = x_{1:N} | \lambda_m)$ is to calculate the Viterbi score, which is the likelihood of the single instantiation of the states $Q_{1:N}$ that is the most likely of all of its possible instantiations (Rabiner and Juang, 1993, Section 6.4.2.1):

$$p(X_{1:N} = x_{1:N} | \lambda_m) \approx \arg\max_{q_{1:N}} p(X_{1:N} = x_{1:N}, Q_{1:N} = q_{1:N} | m) \tag{2.28}$$

In more complicated tasks, such as connected word recognition, $M$ becomes too large for us to consider calculating or estimating the likelihood for all possible models as this would involve constructing and doing inference in $M$ different models. Suppose that each model $m \in M$ is composed of individual words from a lexicon. In this case, we can create a single, large HMM which has the respective word-level HMMs for all of the words in the lexicon; the final null state from each word-level HMM is connected to each initial null state for each word-level HMM (assuming no grammar is used to govern transitions between words (Rabiner and Juang, 1993, Section 7.4.5). We can then calculate the Viterbi score along with its single associated utterance. However, this can be cumbersome if the size of the word lexicon used to form the models $M$ is large, as the Viterbi algorithm is tracking the scores in each possible word at each time frame. One adaptation would be to do pruning during the Viterbi algorithm such that, at any given frame during processing, scores that are much smaller than the most likely score are removed (Lowerre and Reddy, 1980). Alternately, methods such as stack decoding can be used in these cases to limit the number of words to examine at a given time frame (Jelinek, 1969; Paul, 1992; Renals and Hochberb, 1999). With methods such as stack decoding, a stack of hypotheses is maintained as a given sentence is processed. For each iteration in stack decoding, the hypothesis currently having the best score is popped off the stack. As this given hypothesis covers only part of the utterance, the most likely successive words to it are determined. This given hypothesis is then augmented multiple times, each time with a different one of these most likely words; these new hypotheses (each of which contains the given hypothesis with an additional word) are pushed onto the stack before re-iterating.

## 2.2 HMM Problems

### 2.2.1 Review of HMM Assumptions

With discrete states $Q_{1:N}$ and continuous features $X_{1:N}$, we want to model their joint distribution $p(X_{1:N}, Q_{1:N})$. If we were to make no statistical assumptions on this distribution, then we could factor it as follows:

$$p(Q_{1:N}, X_{1:N}) = P(Q_{1:N})\, p(X_{1:N}|Q_{1:N}) \tag{2.29}$$

$$= \prod_{n=1}^{N} P(Q_n|Q_{1:n-1})\, p(X_n|X_{1:n-1}, Q_{1:N}). \tag{2.30}$$

Thus, in (2.30), we are modeling $Q_n$ as being dependent upon every variable $Q_{1:n-1}$ that preceded it in time and $X_n$ as being dependent both upon every variable $X_{1:n-1}$ that preceded it in time as well as upon all $Q_{1:N}$. We are making no assumption in the simple factorizations of (2.29) and (2.30). It is, therefore, the $2 \cdot N$ individual ("local") distributions on (2.30) that we are attempting to model. However, modeling these distributions can be complex from both the learning viewpoint and from the inference viewpoint. In learning, we would need a large amount of data (or, alternately, Bayesian priors) to learn the distributions. For example, the distribution $P(Q_N|Q_{1:N-1})$ has a discrete-space of the order of $K^N$ (assuming $K$ discrete values for each of the $N$ discrete variables $Q_{1:N}$). Thus, for large $N$ and large $K$, there would be a prohibitive number of parameters to learn; and to learn each parameter, we would need a minimum number of training examples assigned to each parameter in order to properly learn it. Furthermore, as discussed in the literature on probabilistic modeling (e.g., Heckerman (1999); Smith and Whittaker (1999); Bilmes (1999, Section 2.5)), we want to limit the number of dependencies upon a variable as much as possible. This is because we only want to model the important dependencies upon a variable and to eliminate the unimportant variables that, at most, can be of no harm and, at worst, can be of harm if used to condition a variable's distribution. For example, a certain state at some time $n$, $Q_n$, may have no effect on the distribution $P(Q_N|Q_{1:N-1})$, that is, $Q_N \perp\!\!\!\perp Q_n \,|\, \{Q_{1:n-1}, Q_{n+1:N-1}\}$ (the expression $A \perp\!\!\!\perp B \,|\, C$ reads "$A$ is conditionally independent of $B$, given $C$"). This means that we can remove the dependency of $Q_n$ upon $Q_N$:

$$P(Q_N|Q_{1:N-1}) = P(Q_N|Q_{1:n-1}, Q_n, Q_{n+1:N-1}) \tag{2.31}$$

$$= P(Q_N|Q_{1:n-1}, Q_{n+1:N-1}). \tag{2.32}$$

So, suppose that we then learned such a distribution with the unnecessary $Q_n$ included in (2.31). Assuming we had a finite amount of training data, the maximum likelihood estimate for (2.31) may cause a strong dependence upon $Q_n$, even through the true, underlying distribution is not dependent upon $Q_n$. Using (2.31) instead of (2.32) could, then, give undesired likelihoods. An additional reason for wanting to limit the number of dependencies is so that we can take advantage of the sparse dependencies so as to have more efficient probabilistic inference, as utilized extensively, for example, in BNs (Pearl, 1988; Cowell *et al.*, 1999).

Therefore, in light of the desire to decrease the parameter space and to have more robust distributions, the distribution (2.30) is simplified in different ways until we arrive at what is actually modeled by HMMs:

1. First, an independence assumption is assumed between all $X_{1:N}$ such that they are conditionally, independent, identically distributed (c.i.i.d.), thus simplifying the distribution of $X_n$ in (2.30). This makes both $X_n$ and $X_{n'}$ ($n' < n$) dependent only via the hidden state variables $Q_{1:N}$:

$$p(Q_{1:N}, X_{1:N}) \approx \prod_{n=1}^{N} P(Q_n|Q_{1:n-1})\, p(X_n|Q_{1:N}). \tag{2.33}$$

2. A second assumption is also made, which is similar to the c.i.i.d. assumption made in Assumption 1. We assume that the distribution of $X_n$ is time-independent, i.e., $p(X_n|Q_{1:N}) =$

$p(X_n|Q_n)$, which simplifies (2.33) to

$$p(Q_{1:N}, X_{1:N}) \approx \prod_{n=1}^{N} P(Q_n|Q_{1:n-1})\, p(X_n|Q_n). \tag{2.34}$$

The difference between the assumption made here in (2.34) and that made above in (2.33) is that before we were referring to the interaction between the different $X_n$ and $X_{n'}, n' < n$, whereas here we are addressing the (conditional) independency between $X_n$ and $Q_{n'}, n' \neq n$.

3. Third, a time window is imposed, such that we say that all that is needed to accurately model the distribution for $(Q_n, X_n)$ is contained within the previous $T$ time frames. This is referred to as a $T^{th}$ order Markov process:

$$p(Q_{1:N}, X_{1:N}) \approx \prod_{n=1}^{N} P(Q_n|Q_{n-T:n-1})\, p(X_n|Q_n). \tag{2.35}$$

So, in the case of a $1^{st}$ order Markov process, which is what I use in this thesis, we have:

$$p(Q_{1:N}, X_{1:N}) \approx \prod_{n=1}^{N} P(Q_n|Q_{n-1})\, p(X_n|Q_n). \tag{2.36}$$

4. A fourth assumption is typically made (though not necessary in order to be considered a "regular" HMM) in modeling the distribution of $X_n$: instead of being modeled by a single Gaussian with a full covariance matrix, it is modeled by a GMM whose components have diagonal covariance matrices. Defining $X_n[1], \ldots, X_n[p], \ldots, X_n[P]$ as the $P$ elements of $X_n$, we can then model the distribution of $p(X_n|Q_n)$ as indicated by the assumption in (2.38) with mixture component $J_n$ with values $j$, $1 \leq j \leq J$:

$$\begin{aligned} p(X_n|Q_n) &= \sum_{j=1}^{J} P(J_n = j|Q_n) \prod_{p=1}^{P} p(X_n[p]|X_n[p+1], \ldots, X_n[P], J_n = j, Q_n) && (2.37) \\ &\approx \sum_{j=1}^{J} P(J_n = j|Q_n) \prod_{p=1}^{P} p(X_n[p]|J_n = j, Q_n). && (2.38) \end{aligned}$$

While the mixture components do have diagonal covariance matrices, indicating independence of the dimensions within a given mixture component of a given state, the elements $X_n$ are typically not totally independent of each other. Rather, they are conditionally independent, given $Q_n$ and $J_n$: $X_n[p] \perp\!\!\!\perp X_n[p'] \,|\, \{Q_n, J_n\}, \forall p' \neq p, 1 \leq p' \leq P$. That is, their joint dependence upon $(Q_n, J_n)$ allows the correlation between the dimensions of $X_n$ to be modeled indirectly; this is due to their shared impact in calculating the posterior distribution for $(Q_n, J_n)$.

Hence, to summarize, when I refer to standard HMMs, as used in ASR, I am referring to a first-order Markov process with time-independent features (continuous or discrete) whose elements are considered (conditionally) independent of each other:

$$p(Q_{1:N}, X_{1:N}) \approx \prod_{n=1}^{N} p(Q_n|Q_{n-1})\, p(X_n|Q_n), \tag{2.39}$$

where $p(X_n|Q_n)$ is modeled according to (2.38). As illustrated in Section 2.1.3, with $Q_{1:N}$ being unknown, we must then either marginalize them out (so as to obtain the data likelihood) or find their single joint instantiation giving the highest likelihood (so as to obtain the Viterbi score), respectively:

$$
\begin{aligned}
p(X_{1:N}) &= \sum_{q_{1:N}} p(Q_{1:N} = q_{1:N}, X_{1:N} = x_{1:N}) && (2.40) \\
q^*_{1:N} &= \arg\max_{q_{1:N}} p(Q_{1:N} = q_{1:N}, X_{1:N} = x_{1:N}) && (2.41)
\end{aligned}
$$

### 2.2.2 Problems with HMM Assumptions

In practice we assume that there is strictly a first-order Markov process (see Assumption 3 on page 17) and do not attempt to have a higher-order Markov process (where the Markov order $T$ is greater than 1). One thing, however, that is commonly done is to use minimum-state duration, where we stipulate that once a state is entered, the model must stay in that state for a minimum number of frames; this is doing a pseudo-collapsing of states from several time frames into one. Nevertheless, it may be necessary to carry over information beyond that minimum number of frames so that $Q_n$ knows several of the previous values of the state. While this long distance modeling of the state is typically ignored on the actual HMM level, this is sometimes captured at a higher level in ASR modeling, such as in pronunciation models or in language models.

The main concern relating to the HMM assumptions, therefore, is with those related to the modeling of features $X_n$. Assumption 1 (as well as Assumption 2) removes a lot of information, as show in Bilmes (1998). That is, $X_n$ and $X_{n'}, n' < n$ are assumed to be *conditionally* independent given the state $Q_n$: $X_n \perp\!\!\!\perp X_{n'} \mid Q_n$. So, the state $Q_n$ does model some of the dependency between $X_n$ and $X_{n'}$ but not all of it. That is, the value of $Q_n$ does indicate to $X_n$ what the value of both $Q_{n-1}$ and $X_{n'}$ might have been; this means that $X_n$ assumes $X_{n'}$ has a certain distribution if $Q_n$ has a given value. However, as $X_n$ lacks the precise value of $X_{n'}$ that is lost with no direct dependency between the two, it can not model the dependency exactly. One solution, as proposed in Bilmes (1998, 1999) is to use *buried* Markov models (BMMs), which selectively add dependencies directly between $X_n$ and $X_{n'}$. These dependencies vary according to the value of the state $Q_n$ and are chosen according to information theoretic criteria in such a way so as to increase the discriminability between the different states. Therefore, (2.38), as used in standard HMMs, is replaced by:

$$p(X_n|Q_n) = \sum_{j=1}^{J} P(J_n = j|Q_n) \prod_{p=1}^{P} p(X_n[p]|Z(p, Q_n), J_n = j, Q_n), \tag{2.42}$$

where $Z(p, Q_n)$ is the set of variables (and dimensions) from $X_1, \ldots, X_{n-1}$ that $X_n[p]$ is directly dependent upon, with the members of this set depending on the value of $Q_n$

In addition to looking at the modeling of the features $X_n$ across time, there is also the assumption of modeling of the elements of $X_n$ using a diagonal covariance within a single time frame of a given mixture component and state, as addressed in Assumption 4. This assumption can be relaxed by having selected features be dependent upon each other. That is, (2.38) can be replaced by

$$p(X_n|Q_n) = \sum_{j=1}^{J} P(J_n = j|Q_n) \prod_{p=1}^{P} p(X_n[p]|Z(p), J_n = j, Q_n), \tag{2.43}$$

where $Z(p)$, in this context, means the set of variable dimensions $X_n[p+1], X_n[p+2], \ldots$ that $X_n[p]$ is to be dependent upon. It is this assumption, of those listed above, that I look at in more detail in this thesis. While I do not want to eliminate the conditional independence between the features entirely, I am looking at how a key "auxiliary" feature should be conditioning the distributions of the regular features. In terms of (2.43), $Z(X_n[p]) = X_n[P+1] = A_n$, where $A_n$ is the auxiliary feature (or, the $(P+1)^{th}$ feature). This is inspired by the *latent* auxiliary variable used to model "contextual" information in Zweig (1998)

In addition to looking at relaxing Assumption 4, I am looking at *introducing* an additional assumption (which adds more (conditional) independencies than does the related Assumption 2 on page 16 by removing the dependencies upon all states $Q_{1:N}$), one of state independence:

1. Some high level features are not emitted based on the current state. If we consider the state level to be on the short-term, some features that are considered auxiliary do not evolve so much according to this short-term information. Rather, they evolve according to higher, longer-term information, perhaps relating to the prosody. Therefore, if we concatenate $S$ auxiliary features to $X_n$ so that we have the "standard" $X_n[1],\ldots,X_n[P]$ and the auxiliary features $X_n[P+1],\ldots,X_n[P+S]$, we first obtain (2.44), based on (2.39) and (2.43); then, introducing this assumption, we obtain (2.45) for the modeling of $X_n$:

$$p(X_n|Q_n) = \sum_{j=1}^{J} P(J_n=j|Q_n)\left[\prod_{n=1}^{N} p(Q_n|Q_{n-1})\prod_{p=1}^{P} p(X_n[p]|Z(p),J_n=j,Q_n)\right] \cdot \prod_{s=P+1}^{P+S} p(X_n[s]|Z(s),Q_n) \tag{2.44}$$

$$p(X_n|Q_n) = \sum_{j=1}^{J} P(J_n=j|Q_n)\left[\prod_{n=1}^{N} p(Q_n|Q_{n-1})\prod_{p=1}^{P} p(X_n[p]|Z(p),J_n=j,Q_n)\right] \cdot \prod_{s=P+1}^{P+S} p(X_n[s]|Z(s)). \tag{2.45}$$

This is how the auxiliary information of Fujinaga *et al.* (2001) was modeled. The features that are "concatenated" here, that is, $X_n[P+1],\ldots,X_n[P+S]$, will later be referred to as $A_n$.

Issues related to assumptions with the auxiliary variable are discussed in more detail in Section 4.4.1 on page 47 ff and Section 4.4.2 on page 51 ff.

## 2.3 Artificial Neural networks

While DBNs, as presented here, use Gaussian mixture models (GMMs) or conditional GMMs to compute the likelihood of the data, artificial neural networks (ANNs) can also be used in the context of HMMs to compute (scaled) likelihoods. Such a system is referred to as an HMM/ANN hybrid (Bourlard and Morgan, 1993). The standard ANN used in such systems has three layers. The input layer contains the observations for the current time frame and a window of $c$ frames: $x_{n-c:n+c}$, where $c \geq 0$. The hidden layer enables the correlations between the elements of $X_n$ to be modeled. The output layer consists of a separate output unit for each phonetic state, giving a posterior probability between 0 and 1 for each of the states. These posteriors $P(Q_n|X_{n-c:n+c}=x_{n-c:n+c})$ are then scaled by the prior distributions $P(Q_n)$ to get scaled likelihoods:

$$p(X_{n-c:n+c}=x_{n-c:n+c}|Q_n) = \frac{p(X_{n-c:n+c}=x_{n-c:n+c})\cdot P(Q_n|X_{n-c:n+c}=x_{n-c:n+c})}{P(Q_n)} \tag{2.46}$$

$$\propto \frac{P(Q_n|X_{n-c:n+c}=x_{n-c:n+c})}{P(Q_n)} \tag{2.47}$$

Like DBNs, ANNs also have some flexibility in changing the statistical assumptions involved. So I also present ANNs having some standard and non-standard assumptions in them (work with ANNs was done in collaboration with Mathew Magimai-Doss). We can model some features independent of $Q_n$ by having a different ANN for each value of a discretized feature (in a similar way to (2.45)). We can also model some features conditionally independent of the others (in a similar way to (2.38)). These different approaches are explained in more detail in Section 4.4.2.

## 2.4 Dynamic Bayesian Networks

Section 2.1 presented the HMM, the workhorse of much current speech recognition. HMMs can be viewed graphically as dynamic Bayesian networks (DBNs), which are extensions of Bayesian networks (BN). Whereas HMMs have predefined probabilistic dependencies, all using the same underlying algorithms, BNs can work with a wide range of various probabilistic dependencies, all using the same underlying algorithms. HMMs can work in the same variety of probabilistic dependencies but need to be reimplemented in order to incorporate the change in dependencies. Therefore, given this flexibility of DBNs in allowing the underlying distributions to be modified, I have done my work in the framework of DBNs.

DBNs are actually part of a larger group of probabilistic models called graphical models (GMs). Within the family of graphical models, three specializations are possible: Markov fields, whose component graph is undirected; Bayesian networks (BNs), whose component graph is a directed acyclic graph (DAG, as defined in Section A on page 101); and chain graph networks, which use a combination of a DAG and an undirected graph. Markov fields are common in the field of computer vision (Li, 1995). DBNs are themselves a specialization of BNs, where a given set of variables is modeled dynamically (e.g., over a series of windows in time) (Dean and Kanazawa, 1988). This chapter assumes a certain knowledge of graph theory; for an introduction to graph theory, please refer to Appendix A;

A Bayesian network (BN) is merely a tool to aid in the modeling of the joint distribution of a set of variables $V = \{V^1, \ldots, V^M\}$. It enables the joint distribution to be factored into a set of local distributions, one for each $V^i \in V$, where $V^i$ represents a generic variable, in this case of ASR with auxiliary information, $A$, $Q$, or $X$. With such a factorization, individual distributions can be utilized by themselves or the joint distribution of only a subset of $V$ can be easily constructed.

After defining DBNs in Section 2.4.1 above, I elaborate more on their probability distribution in Section 2.4.2. I then discuss complexity issues in Section 2.4.3. I then give an overview of learning in BNs in 2.4.4. For an in depth discussion on inference, please see Chapter 3.

### 2.4.1 DBN Definition

Using the same notation as in Lauritzen (1996, Chapter 6), DBNs have at their base a set of variables $V = \Delta \cup \Gamma$, where $\Delta$ is the set of discrete variables and $\Gamma$ is the set of continuous variables. See Cowell *et al.* (1999) for a good introduction to BNs. In the case of ASR with auxiliary information, $\Delta = \{Q\}$, and $\Gamma = \{A, X\}$, where $Q$ is the discrete hidden state, $A$ the continuous auxiliary information, and $X$ the continuous acoustic information ($A$ and $X$ can, alternately, be discrete, as illustrated in Stephenson *et al.* (2001)). In working with a dynamical system, these variables occur at each discrete step in time: $V_{1:N} = \Delta_{1:N} \cup \Gamma_{1:N} = Q_{1:N} \cup A_{1:N} \cup X_{1:N}$ for time $n = 1, \ldots, N$. These variables then serve as the vertices in a directed acyclic graph (DAG) $G =< V_{1:N}, E >$, with $E$ being the directed edges between vertices in the DAG, as illustrated in Figure 2.4. For each ordered edge pair $< V_n^{i^p}, V_n^{i^c} >$, the first vertex $V_n^{i^p}$ is considered the "parent" and the second vertex $V_n^{i^c}$ the "child." I also require the following conditions:

1. Edges do not go back in time (required because of how I have implemented the DBN algorithms, which reflects the notion that the future does not affect the past).

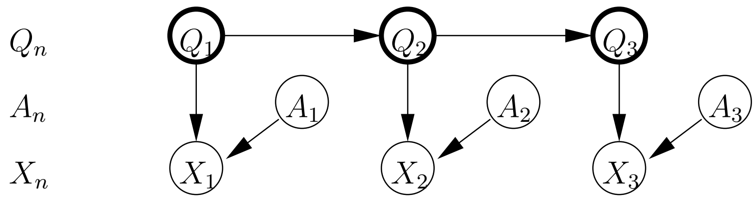


Figure 2.4: Example DBN, where $\Delta = \{Q\}$; $\Gamma = \{A, X\}$; $N = 3$; and $G =< V_{1:3}, E >$, where $V_{1:3} = \{Q_1, A_1, X_1, \ldots, Q_3, A_3, X_3\}$ and $E = \{< Q_1, X_1 >, < A_1, X_1 >, < Q_1, Q_2 >, \ldots, < Q_2, Q_3 >, < Q_3, X_3 >, < A_3, X_3 >\}$. Note that there is no connection such as $< X_1, Q_2 >$ as this is not allowed as $X_1 \in \Gamma$ but $Q_2 \notin \Gamma$; also there is no connection such as $< A_2, X_1 >$ as this is a connection going back in time ($2 \not\leq 1$). Discrete variables have bold vertices.

2. Edges span, at most, one time frame (required so as to aid in efficient probabilistic inference (Zweig, 1998, Section 3.6.3)).

3. Edges from continuous variables go only to continuous variables (required by the algorithm (Lauritzen and Jensen (2001)) that I use).

Figure 2.4 gives an example of a DBN meeting these conditions. Defining $\text{pa}(V_n^i)$ as all of the parent nodes of an arbitrary vertex $V_n^i$ in $G$, as specified in $E$, each variable has an associated "local" probability distribution:

$$P(V_n^i|\text{pa}(V_n^i)). \tag{2.48}$$

For example, the vertices $Q_3$,$A_3$, and $X_3$ from Figure 2.4 have distributions of the form $P(Q_3|Q_2)$, $p(A_3)$, and $p(X_3|Q_3, A_3)$, respectively. The joint probability distribution of $V_{1:N}$ is then the product of all of the local probability distributions:

$$P(V_{1:N}) = \prod_{V_n^i \in V_{1:N}} P(V_n^i|\text{pa}(V_n^i)), \tag{2.49}$$

Thus, for Figure (2.4), we have

$$\begin{aligned} p(V_{1:3}) &= P(Q_1) \cdot P(Q_2|Q_1) \cdot P(Q_3|Q_2) \\ &\quad \cdot p(A_1) \cdot p(A_2) \cdot p(A_3) \\ &\quad \cdot p(X_1|Q_1, A_1) \cdot p(X_2|Q_2, A_2) \cdot p(X_3|Q_3, A_3), \end{aligned} \tag{2.50}$$

which shows one of the main purposes of DBNs: a sparse parameterization of the joint distribution by introducing certain independencies (assumptions) between variables. Note that, for these ASR variables shown in Figure 2.4, local distributions can be used for (2.48) that do not make any statistical assumptions by using the product rule of probability. That is, the following factorization represents the joint distribution of $V_{1:3}$ without any statistical assumptions:

$$\begin{aligned} p(V_{1:3}) &= p(Q_{1:3}, A_{1:3}, X_{1:3}) \\ &= P(Q_1) \cdot P(Q_2|Q_1) \cdot P(Q_3|Q_{1:2}) \\ &\quad \cdot p(A_1|Q_{1:3}) \cdot p(A_2|A_1, Q_{1:3}) \cdot p(A_3|A_{1:2}, Q_{1:3}) \\ &\quad \cdot p(X_1|A_{1:3}, Q_{1:3}) \cdot p(X_2|X_1, A_{1:3}, Q_{1:3}) \cdot p(X_3|X_{1:2}, A_{1:3}, Q_{1:3}) \end{aligned} \tag{2.51}$$

A DBN representing this factorization would have a lot more edges (i.e., it would be a fully-connected DAG) and, hence, a lot more parameters in its local probability distributions than that

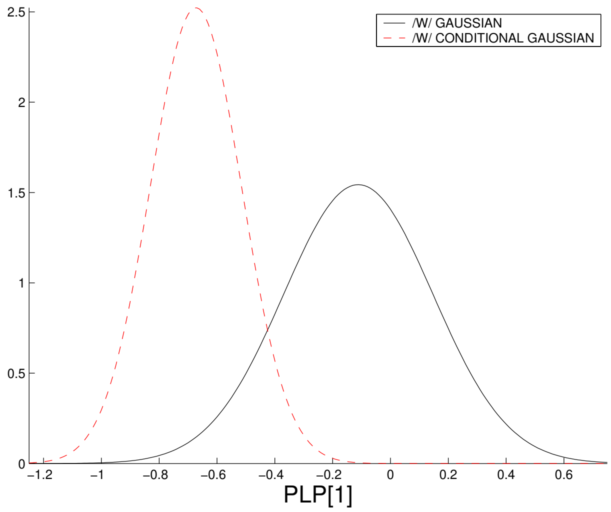


Figure 2.5: Lower variance with conditional Gaussian mixture models (GMMs), illustrated by the conditioning variable of energy. Both plots use the same data used for estimating the parameters for the first PLP coefficients with energy as of first state of the phoneme /w/ with energy as conditioning variable (taken from the "$A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$" of Table 6.11 on page 88). When using a conditional Gaussian, part of the variance is accounted for by the conditioning auxiliary variable. This results in a smaller variance for the conditional Gaussian, giving a more compact distribution that is hopefully more separable from the other states' distributions.

represented in Figure 2.4. Furthermore, it does not meet the condition within my DBN framework that edges are not to go back in time; we could have factored it so as to meet this condition, but then it would not have met the other condition within my DBN framework that edges do not go from continuous variables to discrete variables. Hence, in order to get a more sparsely connected and more robust DBN, if a variable can be shown (or assumed) to be independent of a variable that it is conditioned upon, then that conditioning variable can be removed from its distribution along with the corresponding edge in the graph. For example, if $X_3$ were conditionally independent of all the other variables given $Q_3$ and $A_3$, then the factor $p(X_3|X_{1:2}, A_{1:3}, Q_{1:3})$ could be simplified to $p(X_3|A_3, Q_3)$, as represented in (2.50) and the corresponding Figure 2.4.

### 2.4.2 Probability Distributions Definition

The distribution $P(V_n^i|\text{pa}(V_n^i))$ of (2.48) is defined dependent on what types of variables are in $V_n^i$ and $\text{pa}(V_n^i)$. It the framework that I am using, based on Lauritzen and Jensen (2001), the allowed distributions are defined using Gaussians, conditional Gaussians, discrete probability tables, and tables of (conditional) Gaussians:

- $V_n^i \in \Delta$
  - $\text{pa}(V_n^i) \subset \Delta$: a table of discrete probabilities (e.g., $P(Q_n|Q_{n-1})$)
  - $\text{pa}(V_n^i) \cap \Gamma \neq \{\emptyset\}$: undefined since a continuous parent can not have a discrete child $V_n^i$
- $V_n^i \in \Gamma$
  - $\text{pa}(V_n^i) \subset \Delta$: a table of Gaussian distributions: for each possible instantiation $h$ of $\text{pa}(V_n^i)$, a Gaussian defined by $\mathcal{N}(\mu_{V_n^i,h}, \Sigma_{V_n^i,h})$, for mean $\mu_{V_n^i,h}$ and covariance $\Sigma_{V_n^i,h}$, (e.g.,

| |
|---|
| $P(Q_n = \text{`a'} \mid Q_{n-1} = \text{`a'}) = 0.8$ <br> $P(Q_n = \text{`b'} \mid Q_{n-1} = \text{`a'}) = 0.2$ <br> $P(Q_n = \text{`a'} \mid Q_{n-1} = \text{`b'}) = 0.3$ <br> $P(Q_n = \text{`b'} \mid Q_{n-1} = \text{`b'}) = 0.7$ |
| $p(X_n \mid Q_n = \text{`a'}, A_n) \sim \mathcal{N}\left( \begin{bmatrix} 0.1 \\ -2.1 \\ 1.5 \end{bmatrix} + \begin{bmatrix} -1.5 & 2.2 & 0 \\ 0.1 & 0 & 2.2 \end{bmatrix}^T A_n, \begin{bmatrix} 3.0 & 0.3 & -1.1 \\ 0.3 & 2.0 & 0.2 \\ -1.1 & 0.2 & 1.0 \end{bmatrix} \right)$ <br> $p(X_n \mid Q_n = \text{`b'}, A_n) \sim \mathcal{N}\left( \begin{bmatrix} -0.2 \\ -1.0 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 2.1 & 0.3 & 2.0 \end{bmatrix}^T A_n, \begin{bmatrix} 2.2 & 0.4 & 0.2 \\ 0.4 & 0.5 & 0.1 \\ 0.2 & 0.1 & 1.1 \end{bmatrix} \right)$ |
| $p(A_n) \sim \mathcal{N}\left( \begin{bmatrix} -3.0 \\ 8.5 \end{bmatrix}, \begin{bmatrix} 8.2 & -8.0 \\ -8.0 & 12.0 \end{bmatrix} \right)$ |

Table 2.1: Sample distributions for the DBN in Figure 2.4. For purposes of illustration, assume that $Q_n$ has values 'a', 'b', $A_n$ is a two-dimensional vector, and that $X_n$ is a three dimensional vector.

$p(X_n|Q_n, J_n)$, $Q_n$ being the discrete state variable and $J_n$ being the discrete mixture component variable)

- $\mathrm{pa}(V_n^i) \subset \Gamma$: a conditional Gaussian distribution, defined by $\mathcal{N}(\mu_{V_n^i} + B_{V_n^i}^T a, \Sigma_{V_n^i})$, where $B_{V_n^i}$ is a matrix of regression weights upon the value $a$ ($B_{V_n^i}$ is a vector if $a$ is a scalar) of the continuous parents. (e.g., $p(A_n|A_{n-1})$. This condition does not exist in the DBNs used in this work.
- $\mathrm{pa}(V_n^i) \cap \Gamma \neq \{\emptyset\}$ and $\mathrm{pa}(V_n^i) \cap \Delta \neq \{\emptyset\}$: a table of conditional Gaussian distributions (e.g., $p(X_n|Q_n, J_n, A_n)$).

A conditional Gaussian, which occurs when a continuous variable is itself conditioned by another continuous variable, can be viewed as a Gaussian whose mean changes dynamically. By allowing the mean to change, more of the variation in the data can be captured by this changing of the mean instead of just being measured by the variance. See Figure 2.5. Also, I note that there are other possibilities for structuring the probability distributions besides just the discrete tables and (conditional) Gaussians above. For example, a "noisy or" (Jensen, 1996, Section 3.3.2) can be used to produce more robust probabilities of discrete variables with many parents through the use of few parameters. Also, ANNs can be used for estimating the probabilities of discrete or continuous variables (Murphy, 2002, Section C.3.4). We can even suppose that the first moment of the conditional Gaussian, $\mu_{V_n^i} + B_{V_n^i}^T a$, can be estimated in some other, perhaps non-linear, manner. While not allowed in my framework, the probabilities of discrete variables with continuous parents can be estimated in the framework of Koller *et al.* (1999). See Table 2.1 for example parameters for some of these distributions.

### 2.4.3 Complexity

DBN theory places few restrictions on valid topologies, depending on the reference framework. However, I have found that there are some DBNs which are intractable due to both their topologies and to which variables are hidden. Since this is not discussed much in the BN literature, I discuss the problem here both from a graph viewpoint and from a probability viewpoint. For additional discussion in the BN framework, see Lerner and Parr (2001). For additional explanation of the terms paths, neighbors, clique tree, moralized, triangulated (decomposable), elimination order, and other graph theory concepts, as used in this section, I refer the reader to Appendix A on page 101.

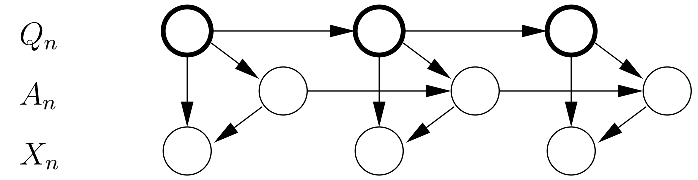


Figure 2.6: DBN for auxiliary information ASR. While illustrated here as a continuous-valued variable, $A_n$ can also be discrete-valued.

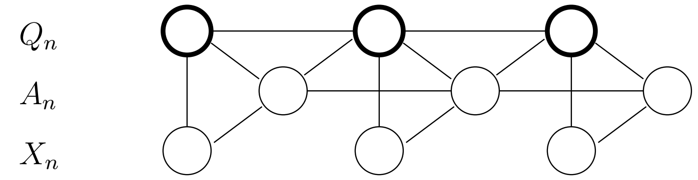


Figure 2.7: "Non-strongly" decomposable graph, using elimination nodes of $Q_1$, $A_1$, $X_1$, $Q_2$, $A_2$, $X_2$, $Q_3$, $A_3$, $X_3$.

The focus of this thesis lies in adding an auxiliary variable that is in similar spirit to the 'context' variable in Zweig (1998), except that here I give the variable a definite meaning by giving it training data. The base form for including the auxiliary variable '$A_n$' is shown in Figure 2.6.

### Graphical Analysis

In doing inference in a DBN, we will be needing a clique tree (as defined on page 104) with a strong root (Cowell *et al.*, 1999), which, informally defined, means that cliques containing only discrete variables are found at the top of the clique tree while cliques containing only continuous variables are found at the bottom of the clique tree. A necessary condition for this is that, after "triangulation" (as defined on page 103) there can not be a path between two non-neighboring discrete variables that has only continuous variables between the two non-neighbors (Cowell *et al.*, 1999, Proposition 7.9). For example, Figure 2.7 violates this condition with the path $< Q_1, A_2, A_3, Q_3 >$. The solution is that, during the triangulation phase of forming the clique tree of the graph, edges must be added between such discrete variables (in this case, $Q_1$ and $Q_3$ must be connected), as illustrated in Figure 2.8. This becomes intractable for large $N$, where there is a long chain of discrete variables, $Q_1, \ldots, Q_N$. Since, for any two of the non-neighboring discrete variables, $Q_i$ and $Q_j$, there is such a violating path $< Q_i, \ldots, Q_j >$; therefore, all of these discrete variables must be fully connected, forming part of one large clique, with too large of a state space to allow for tractable, exact inference. The solution is to use approximation methods (Boyen and Koller, 1998; Lerner and Parr, 2001) for inference, which is not the focus of this thesis. This was not an issue in Zweig (1998), as he only used discrete vertices.

The conflict arises in the condition that in the numbering of the "elimination order" (as defined in page 103) used for triangulation that (1) all variables at time $n+1$ are higher than those at time $n$ (Zweig, 1998, Section 3.4) and that (2) all continuous variables are numbered higher than the discrete variables (Cowell *et al.*, 1999, Section 7.3). If we meet the first condition, then the discrete variables from time $n+1$ will be numbered higher than the continuous variables from time $n$, thus producing a graph that is not strongly decomposable. However, if we meet the second condition, then we will have cliques whose variables may come from more than two neighboring frames, thus potentially increasing immensely the complexity of the graph. The first condition is for complexity

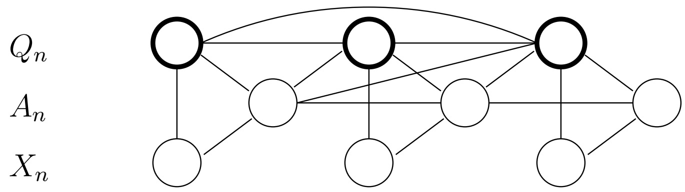


Figure 2.8: Strongly decomposable graph, using elimination order of $Q_1$, $Q_2$, $Q_3$, $A_1$, $A_2$, $A_3$, $X_1$, $X_2$, $X_3$

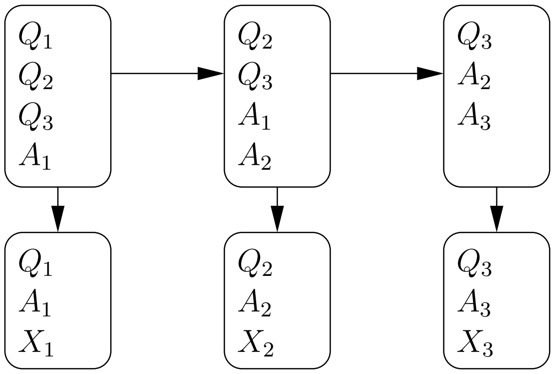


Figure 2.9: Clique tree of graph in Figure 2.8. Using such an elimination order as in Figure 2.8 results in each variable $A_n$ appearing in a clique with all variables $Q_{n,N}$. In this figure, $A_1$ appears in a clique with $Q_{1,3}$ and $A_2$ appears in a clique with $Q_{1,3}$. For large $N$, the cliques with low numbered $A_n$ (e.g., $A_1$, $A_2$, $A_3$) would have a very large number of values in their tables. So, if $Q_n$ has $K$ values, there would be cliques involving $A_1, A_2, A_3, \ldots$ with $K^N, K^{N-1}, K^{N-2}, \ldots$ values, respectively.

reasons, the second for theoretical reasons. Therefore, the theoretical requirement must be adhered to. As a result, doing exact inference in a DBN with a continuous, time-dependent variable is infeasible due to complexity reasons. The resulting, strongly decomposable graph is shown in Figure 2.8 (though there can be other valid triangulations). As shown in Figure 2.9, a clique tree formed using Figure 2.8, has cliques whose parameter space sizes are $K^N, K^{N-1}, K^{N-2}, \ldots$, giving complexity of the order of $K^N$; hence, doing operations on cliques of these sizes would be computationally complex.

**Probability Analysis**

The complexity problem in DBNs such as in Figure 2.6 involves marginalizing over discrete variables when there is a continuous conditioning variable in the distribution, which is not defined in Lauritzen and Jensen (2001, Section 4.3). That is, as we can not sum over the values of the marginalized discrete variable, we must maintain each individual value of the discrete variables from each time frame to the next. Assume that we were to have one single clique for all the variables contained in a single frame (the conclusions are the same if we were to assume more than one clique within a time frame but the analysis is simpler if we assume only one clique in a single time frame). In looking at the final time frame $N$, we would be modeling the following distribution (by

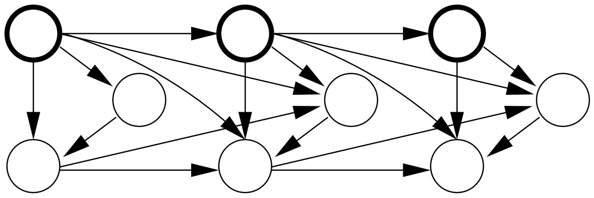


Figure 2.10: Tractable DBN if $X_n$ is observed.

taking into account the time-dependency of two of the variables, $A_N$ and $Q_N$):

$$p(Q_N, X_N, A_N | Q_{N-1}, A_{N-1}). \tag{2.52}$$

We can not instantiate the known value of $X_N$ at this point, as their is a continuous conditioning variable (i.e., $A_{N-1}$) (Lauritzen and Jensen, 2001, Section 7); however, we can marginalize out $A_N$:

$$\int_{-\infty}^{\infty} p(Q_N, Q_{N-1}, X_N, A_N | A_{N-1}) \, \mathrm{d}A_N = p(Q_N, Q_{N-1}, X_N | A_{N-1}). \tag{2.53}$$

As we can not marginalize out $Q_N$ (due to the continuous conditioning variable $A_{N-1}$, we must maintain a list of all $K$ values of $Q_n$:

$$\sum_{\forall q_N} p(Q_N = q_N, Q_{N-1}, X_N | A_{N-1}) = \{p(Q_N = q_N, Q_{N-1}, X_N | A_{N-1})\}_{\forall q_N}. \tag{2.54}$$

That is, to marginalize out the discrete variable (e.g., $Q_N$), we must maintain all $K$ the values for time frame $N$. Continuing on to time frame $N-1$, this will become $K^2$ that must be maintained, as there will be yet another continuous conditioning variable (e.g., $A_{N-2}$); this grows exponentially until at time frame 1, we have $K^N$ possible values.

**Always Tractable**

The complexity of inference in a clique tree is a function of the space size of the component cliques. Therefore, for this inference to be tractable, DBNs must have a few restrictions upon them in addition to the edges not going back in time and not going from continuous to discrete variables (though, these two restrictions are only limited to the framework that I do my work in; Koller *et al.* (1999) allows connections from continuous to discrete variables and Deviren and Daoudi (2001) allows certain connections back in time). Suppose we are given one discrete variable per time frame, $Q_n$, and the two continuous variables per time frame, $A_n$ and $X_n$, where $X_n$ is always observed. The most connections that such a DBN can have is illustrated in Figure 2.10, with the reason for the missing connections explained below. Generally speaking, a discrete variable can receive connections from any other discrete variables in the current or previous frame (e.g., $P(Q_n|Q_{n-1})$). Also, a continuous variable can receive connections from any other variables in the current frame (e.g., $P(X_n|Q_n, A_n)$). However, there are restrictions on a continuous variable's having connections from the previous time frame, though this does not occur in the studies in this thesis.

A continuous variable such as $X_n$ can receive connections from discrete variables in the previous time frame (e.g., $p(X_n|Q_{n-1})$. It can also receive connections from continuous variables such as $X_{n-1}$ in the previous time frame if they observed (e.g., $p(X_n|X_{n-1} = x_n)$; this is allowed because that observation can be incorporated into the distribution to produce an updated distribution $p^*(\cdot)$ with no dependency on a continuous variable in the previous time frame (e.g., $p(X_n|X_{n-1} = x_N) \rightarrow p^*(X_n)$). If we were to allow a conditioning variable such as $X_{n-1}$ to be hidden, then inferring the distribution of the variable in the current time frame (in this case, $X_n$) would be directly dependent on the inferred distribution of $X_{n-1}$ and indirectly dependent upon continuous variables, such as $X_{n-2}, \ldots, X_1$, further back in time. So to model these indirect dependencies, we would have the intractable situation where the discrete variables in between such variables are contained in the same clique (a result of the strong root condition mentioned earlier in Section 2.4.3), as indicated in Figure 2.9. Restricting the connections from a hidden continuous variable in the previous time frame is because the continuous variables at a given time frame must be '$d$-separated' (Pearl, 1988, Section 3.3) from continuous variables more than one time frame away, given only the discrete variables. '$d$-separation' means that two sets of variables $A$ and $B$ are conditionally independent of each other given a third set $C$; I notate this by $A \perp\!\!\!\perp B \,|\, C$. So, while $X_{n+1}$ is dependent upon $X_{n-1}$, given the continuous variable $X_n$, the dependency is broken by the observation $X_n = x_n$; thus, we have $X_{n+1} \perp\!\!\!\perp X_{n-1} \,|\, \{Q_n, X_n = x_n\}$. To illustrate the concept of $d$-separation better, $A_1$ is $d$-separated from $A_2$ in Figure 2.4 on page 21 given only the discrete variables $Q_{1,2}$ because any path between the two must go through $Q_{1,2}$; but $A_2$ is not $d$-separated from $A_2$ in Figure 2.6 on page 24 given only the discrete variables $Q_{1,2}$ because there is a path between the two that does not go through $Q_{1,2}$ (that is, in this case, $A_1$ and $A_2$ have a direct path between each other as they are connected).

**Tractable only with approximations**

Theoretically, it is not necessary that the continuous variables from a given time be $d$-separated from continuous variables at other time frames, given only the discrete variables However, in order to accommodate continuous variables that do not meet this criterion, it is necessary for the resulting clique tree to have a strong root (Cowell *et al.*, 1999), as discussed above, which, informally defined, means that cliques containing only discrete variables are found at the top of the clique tree while cliques containing only continuous variables are found at the bottom of the clique tree. This becomes intractable for large $N$, where there is a long chain of discrete variables, $Q_1, \ldots, Q_N$. Since, for any two of the non-neighboring discrete variables, $Q_i$ and $Q_j$, there is such a violating path $< Q_i, \ldots, Q_j >$, all of these discrete variables must be fully connected, forming one large clique, with too large a state space to allow for tractable, exact inference. The solution is to use approximation methods (Boyen and Koller, 1998; Lerner and Parr, 2001) for inference, which is not the focus of this thesis.

### 2.4.4 Learning

The inferred posterior distributions can be used in the expectation stage of expectation-maximization (EM) training. To get the expectation for a variable $V_n^i$, we get each joint posterior distribution $P(V_n^i, \text{pa}(V_n^i)|e)$ (where $e$ is the observations for the observed variables of $V$; in my case, this will be the observations $x_{1:N}$, as well as $a_{1:N}$, if available) for all time samples $n = 1, \ldots, N$ and use them in the maximization stage to compute the updated version $P^*(V_n^i, \text{pa}(V_n^i))$. This can then be factored as $P^*(V_n^i, \text{pa}(V_n^i)) = P^*(\text{pa}(V_n^i))P^*(V_n^i|\text{pa}(V_n^i))$, with the final factor being used as the new estimate.

Note that in the case of a continuous variable $V_n^i$ with at least one continuous parent, that the counts will be collected from the joint occurrences of $(V_n^i, \text{pa}(V_n^i))$ even though the final output after maximization will be a *conditional* Gaussian $P^*(V_n^i|\text{pa}(V_n^i))$. That is, we are actually maximizing the regular, joint GMM for $V_n^i$ and $\text{pa}(V_n^i)$ even though we only want a conditional GMM. After collecting the counts for $(V_n^i, \text{pa}(V_n^i))$ for $n = 1, \ldots, N$, we use the standard maximum likelihood formulas for estimating the weights, means, and variances of this multi-variate Gaussian mixture model $p(V_n^i, \text{pa}(V_n^i))$; we then factor the estimated distribution, using the formulas in Lauritzen and Jensen (2001, Section 4.5) to obtain the conditional $p^*(V_n^i|\text{pa}(V_n^i))$. So, there is (to my knowledge) no formula for directly computing the maximum likelihood estimate for a conditional GMM; rather, we maximize the joint, multi-variate GMM and then factor it to get the conditional GMM. This explanation is lacking in previous literature.

## 2.5 Benefits of DBNs

In Section 2.1 I presented the standard HMM, followed by a detailed discussion on dynamic Bayesian networks (DBNs) in Section 2.4. I give here a discussion on the relation between HMMs and DBNs. I first discuss the similarities in probability distributions and probabilistic inference in Section 2.5.1 and Section 2.5.2, respectively. I then close with a review on how to convert between the two in Section 2.5.3 and on the advantages of DBNs in Section 2.5.4.

### 2.5.1 Probability distributions

In time-series modeling, with the assumptions as discussed in Section 2.2 on pages 15 ff, both an HMM and a DBN model the evolution of discrete variables $Q = \{Q[1], \ldots, Q[R]\}$ and continuous variables $X = \{X[1], \ldots, X[P]\}$. Note that in typical HMM modeling in the context of ASR, we typically have $R = 1$ and $P >> 10$. However, it is possible to have $R > 1$; in such a case, the states would actually be combined into a single compound state whose values are the possible combined instantiations of the individual state dimensions. Certain statistical independencies are assumed to exist between the dimensions in $Q$, as well as between the dimensions in $X$; these are defined arbitrarily, according to the problem being worked with (as defined by a human "expert") or can be learned as in, for example, Monti and Cooper (1999). Furthermore, certain temporal statistical independencies are assumed to exist between $\{Q_n, X_n\}$ and $\{Q_{n'}, X_{n'}\}$, $n \neq n'$. For instance, both HMMs and DBNs assume a first-order process (variables at time $n$ being conditionally independent of all variables at previous time frames, given the variables at time $n-1$) so that we have $\{Q_n, X_n\} \perp\!\!\!\perp \{Q_{1:n-2}, X_{1:n-2}\} \mid \{Q_{n-1}, X_{n-1}\}$.

One restriction in defining the topology of the distributions is that no discrete variable's distribution can be dependent upon a continuous variable's value (though a current research area within Bayesian networks is ways to allow this, as done, for example, in Koller *et al.* (1999)). That is, for subsets of continuous variables $X_n^1$ and $X_n^2$ and subsets of discrete variables $Q_n^1$ and $Q_n^2$, the following types of distributions are allowed: $p(X_n^1)$, $p(X_n^1|X_n^2)$, $p(X_n^1|Q_n^1)$, $p(X_n^1|X_n^2, Q_n^1)$, $P(Q_n^1)$, and $P(Q_n^1|Q_n^2)$, as explained in Section 2.4.2 on page 22. However, $P(Q_n^1|X_n^1)$ and $P(Q_n^1|Q_n^2, X_n^1)$ are not defined within the framework that I am using in this thesis.

So, effectively, both HMMs and DBNs model the evolution of a set of discrete variables and/or a set of continuous variables. If there are both discrete and continuous variables, the continuous variables are dependent upon the discrete variables. For example, assume three discrete variables ($R = 3$) and three continuous variables ($P = 3$). Figure 2.11 presents the generic structure for a DBN/HMM that is a first-order process.

### 2.5.2 Probabilistic inference

Both HMMs and DBNs consist of two pass inference: the first to compute the likelihood of the observed data given the prior distribution and the second to compute the variables' posterior distributions, given the observed data. During expectation-maximization (EM) training, the posteriors from the second pass are used as the expected counts in both types of models. Likewise, in the trained models in recognition, only the data likelihood from the first pass is used in determining which candidate model most likely produced the utterance.

In standard HMMs, the probabilistic dependencies, and hence the actual inference operations, are determined at compile time. However, in DBNs, this is more flexible in being determined at

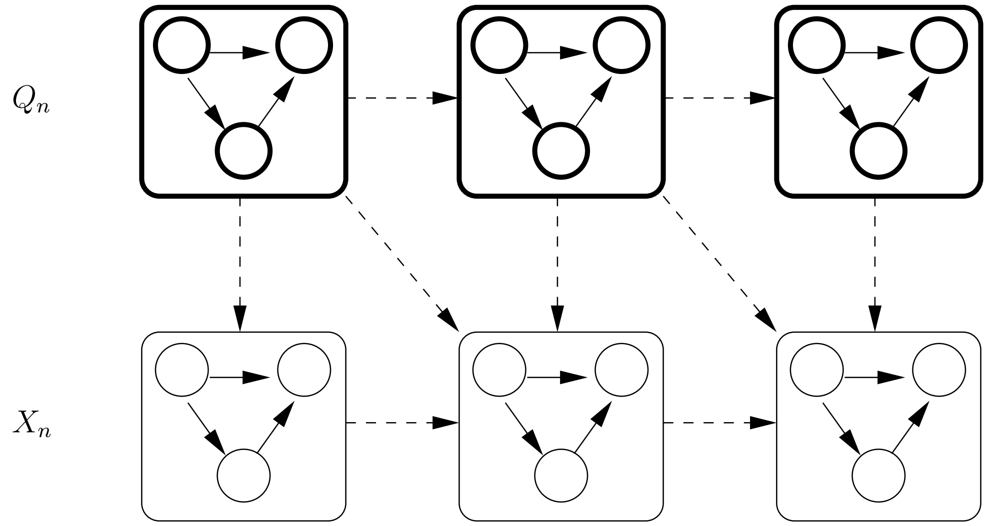


Figure 2.11: DAG for a Generic DBN. Discrete variables have bold vertices. Dashed line arrows between groups of variables indicate full connection from the vertices in the parent group to the vertices in the child group.

run time (Zweig, 1998). With DBNs, we can have both the component variables as well as the statistical dependencies changed as desired; with HMMs, we assumed two variables $Q_n$ and $X_n$ and two dependencies ($Q_{n-1}$ conditioning $Q_n$ and $Q_n$ conditioning $X_n$). For example, in the DBN in Figure 2.4 we were able to introduce the variable $A_n$ into the modeling without having to change the underlying algorithms of the DBNs; we could also have given different dependencies in that DBN (e.g., having $Q_n$ condition $A_n$ and not having $A_n$ condition $X_n$). The HMM framework is not suited for easily incorporating such changes.

### 2.5.3 DBN to HMM Equivalence

As explained in Smyth *et al.* (1997); Roweis and Ghahramani (1999), DBNS are a generalization of HMMs. In fact, they both can model the same processes. Zweig (1998) outlined how, given a DBN, to construct its HMM equivalent, which, in general terms and modified, works as follows:

1. The values of the state space of the HMM will be all the possible instantiations of all the discrete variables in a time frame $n$ of the DBN.

2. The feature vector of the observed space of the HMM will be the concatenation of the features of each observation variable.

3. The transition matrix for the state space is the combination of the transition matrices for all of the component discrete variables.

4. The emission distribution for the observed space is the combination of all the emission distributions for all of the component observed variables.

Doing the reverse process of generating a DBN from an HMM is not straightforward. That is, one of a DBN's strengths is that it can factor the elements in both its state space and its observation space. So, to construct the DBN, we would need to determine the conditional independence assumptions existing within the state space, within the observation space, and between the state space and the observed space. The determination of these assumptions is not absolutely necessary but needs to be done if we want to properly make use of the advantages that DBNs offer over HMMs (such as that of having conditional statistical independence between multiple discrete state variables); else, we have no need to work in the DBN framework and could just use the HMM framework.

### 2.5.4 Advantages of DBNs over HMMs

Both HMMs and DBNs, within the above framework, can, in theory, have arbitrary dependencies within the topology of the probability distributions. The difference lies in the implementation. For each new type of dependency desired in an HMM, the algorithms have to be enhanced and code has to be added to the program to allow it. However, the algorithms used with a DBN are generic enough to allow various dependencies, which are defined at run-time, not at the earlier stage of compile-time. Also, the arbitrary hiding of any variable in the discrete or continuous space for any time is built into the DBN; for HMMs, this will need to be added to the code.

HMMs are not designed for handling arbitrary statistical independencies within their state space $Q$ or within their feature space $X$. DBNs, rather, have a rich framework for allowing dependencies to be selectively removed among $Q_n[1], \ldots, Q_n[R]$ and among $X_n[1], \ldots, X_n[P]$ as well as between $X$ and $Q$ groups:

1. $Q_n[r] \to Q_n[1], \ldots, Q_n[r-1]$ - dependencies can be deleted within multiple state variables of Figure 2.11. In normal HMM modeling, all dependencies are used within the state.

2. $Q_n \to X_n$ - dependencies can be deleted between the state and the features of Figure 2.11. In normal HMM modeling, all dependencies are used. In this thesis, I look at removing this dependency between $Q_n$ and one of the features (the "auxiliary" feature).

3. $X_n[p] \to X_n[1], \ldots, X_n[p-1]$ - dependencies can be deleted within multiple feature variables of Figure 2.11. With the factor rule of probability, without assumptions, we have

$$p(X_n[1, \ldots, P]) = \prod_{p=1}^{P} p(X_n[p] | X_n[p+1], \ldots, X_n[P]), \tag{2.55}$$

   which explains why the standard dependencies go only from dimension $p$ to lower order dimensions (the ordering of the dimensions is arbitrary). In normal HMM modeling, no dependencies are used within models for features of a given state and mixture component (though it is allowable, reflecting full covariance matrices being used in each mixture component).

4. $Q_{n-1} \to Q_n$ - dependencies can be deleted between different state variables across time of Figure 2.11. In normal HMM modeling, all dependencies are used within the state.

5. $Q_{n-1} \to X_n$ - dependencies can be deleted between different state variables and different feature variables across time of Figure 2.11. In normal HMM modeling, no dependencies are used between the state and features across time.

6. $X_{n-1} \to X_n$ - dependencies can be deleted between different observations variables across time of Figure 2.11. In normal HMM modeling, no dependencies are used between observation variables across time.

In Zweig (1998, Chapter 4), Dependencies 1, 2, 3, and 4 were examined (he did not look at Dependencies 5 and 6 as these fall outside the realm of normal HMM modeling). In summary, he argues for the case of using DBNs instead of HMMs where not only the state space (Dependencies 1 and 4) can be sparsely factored but where the observation space (Dependency 3) as well as the dependencies between the state and observation (Dependency 2) can be sparsely factored as well. If there are no such possible sparse factorizations, then DBNs would offer no advantages over the HMMs, as least in regards to these modeling aspects.

In this thesis, in addition to Dependencies 2 and 3, I advocate an additional advantage of DBNs over HMMs:

1. Hiding $X_n[p]$ : DBNs, in theory, can handle any arbitrary variable (state or feature) being hidden or partially-observed through time. In practice, there must be certain conditions imposed on the hidden variable (see Section 2.4.3).

It is true that this can be handled in the framework of HMMs (as done in missing feature theory (Morris *et al.*, 1998)), but DBNs provide a more general framework to do it in, regardless of the statistical dependencies imposed.

## 2.6 Conclusion

In this chapter I have discussed how time series modeling is performed, in the context of ASR. One fundamental model for time series in ASR is the HMM. I have shown the basic structure of the HMM as well as the general framework for how it does inference and, more specifically, how its parameters are learned. HMMs have a set of fundamental assumptions that make their modeling more computationally feasible; however, these assumptions can remove some important information for robustly dealing with the variation in modeling.

A more generic model for time series in ASR is the DBN. It can be used to relax or tighten some of the assumptions imposed in standard HMMs so as to produce more robust models. While theoretically in the same family of probabilistic models as HMMs, they can easily handle the changing of many of the statistical dependencies assumed in HMMs, due to their generic inference algorithm, which will be presented in Chapter 3. I have shown, from the perspective of $d$-separation, how the relaxing of certain assumptions related to having time-dependent, hidden continuous information can pose complexity problems. The assumptions that I will examine in detail in Chapter 4 involve independence between the state and the “auxiliary” feature as well as dependence between this “auxiliary” feature and the the standard features.

# Chapter 3

# DBN Inference

In this chapter, I continue the discussion on DBNs, as introduced in Chapter 2, by explaining in detail how to do probabilistic inference in them. Probabilistic inference is the task of taking prior distributions for a set of variables and a set of "evidence" (also known as observations) for a subset of these variables and then using this to determine the posterior distribution of the variables without evidence (that is, the hidden variables). Lauritzen and Jensen (2001) explain the theory of how to do probabilistic inference in mixed Bayesian networks (BNs), where "mixed" means that there are both continuous and discrete variables, any of which can be hidden. The algorithms in Lauritzen and Jensen (2001) were developed to correct for numerical stability problems in an earlier algorithm (Lauritzen, 1992). Here I would like to expand on some of the implementation details of Lauritzen and Jensen (2001) in regards to the DBNs used in my work.

In a similar manner to that of HMMs in Section 2.1.2, calculating the probabilities of DBNs, as well as learning their parameters, would be a lot simpler if all of the variables were observed. However since ASR deals with models with hidden variables, we have a more complex problem. In HMMs, I showed how the calculation of the various probabilities associated with an HMM (the data likelihood, the posterior distribution of a given state $Q_n$, etc.) can be more easily factored due to the framework that the HMMs are set in.

In DBNs, we are faced with a generalization of this problem: we are given an arbitrary set of variables, dependencies, and observations of certain of those variables (all within the limits described in Section 2.4.1), and we need to be able to efficiently compute the data likelihood, the posterior distributions of the hidden variables, etc. Like with HMMs, if we try to compute this using only the joint probability of all of the variables, $P(V_{1:N})$, we will have a complex problem; for example, if the distribution $P(V_{1:N})$ has hidden variables $Q_{1:N}$, each of which has $K$ possible values, then there will be $K^N$ unknown parameters to jointly estimate. However, within the context of DBNs, we can also take advantage of local computations. These local computations done in probabilistic inference are possible because of conditional independence assumptions introduced when forming the DBN.

To give a better understanding of the general workings of inference, I first give an overview of the "clique tree" as well as how it is used for training. I then give a high-level overview of inference, according to Lauritzen and Jensen (2001). Then, with this background, I propose methods (in addition to those from Zweig (1998)) from my own coding experience as to how this algorithm should be implemented in practice in order to more efficiently do inference in mixed dynamic Bayesian networks (DBNs). This is meant as a complement to Lauritzen and Jensen (2001); so, for further explanations and details, please refer there. For those readers unfamiliar with probabilistic inference in Bayesian networks, please refer to Appendix B; for an explanation of some of the terminology (DAG, cycles, cliques, moralize, triangulate, etc.), please refer to Appendix A.

## 3.1 The Clique Tree

A clique tree is a transformed version of the base DBN, where vertices in the base DBN serve as the members of the cliques in the clique tree (for more information on clique trees, please refer to Section B.2 on page 109). Inference directly upon a DBN is not always possible if there are cycles among the variables; however, we can always, in theory, do inference upon a clique tree formed from the DBN. The clique tree is formed from the DBN so as to contain all of the probabilistic dependencies contained in the DBN. As all of the DBN's dependencies are captured in the clique tree, the inference results from the clique tree apply to the underlying DBN as well. In contrast to the case of HMMs, the topology of the distribution (i.e., which variables were dependent upon which other variables) is not known beforehand; hence, this determination of what cliques are to be used for the local computations is determined on a network-by-network basis using the generic clique tree building algorithm involving moralization and triangulation.

When doing inference in the context of training the DBN, one of the goals is to obtain the joint posterior distribution $P(V_n, \mathrm{pa}(V_n)|\mathbf{e})$ for each variable $V_n$ and its parents $\mathrm{pa}(V_n)$:

$$P(V_n, \mathrm{pa}(V_n)|\mathbf{e}), \tag{3.1}$$

given all of the observations ("evidence") $\mathbf{e}$. In the case of HMMs, $\mathbf{e} = \{x_{1:N}\}$; however, in DBNs this can involve any arbitrary subset of the variables $V_{1:N}$ within the DBN (subject to the complexity constraints discussed in Section 2.4.3). (3.1), in the case of EM learning, can then be used in collecting the counts during the expectation step. For example, in collecting the counts for a discrete $V_n$, we would be collecting posteriors for the counts, in a similar fashion as done with (2.21) on page 14. Also, in collecting the data points for a continuous $V_n$, we have the option of leaving certain continuous $V_n$ hidden, if necessary, and using its inferred value from (3.1) in collecting the data for the maximization of the continuous variables, using formulae such as used for HMMs in (2.25) and (2.26). I note that, in this thesis, I never have a hidden continuous variable during training; however I do, as noted later, take advantage of being able to use the posterior distribution of a hidden continuous variable during recognition.

Inference, as used in this thesis, which is described in detail in Lauritzen and Jensen (2001), then proceeds in two stages, an upward pass and a downward pass, corresponding to the forward and backward passes used in HMMs (Baum, 1972; Zweig, 1998), respectively. The results of the inference algorithm can then be used to simply compute the inferred distribution $P(V_n, \mathrm{pa}(V_n)|\mathbf{e})$. More details regarding this whole process can be found in Pearl (1988); Zweig (1998); Lauritzen and Jensen (2001); Cowell *et al.* (1999).

## 3.2 Training

EM training also exists in the context of DBNs (Lauritzen, 1995), thus allowing a more general form of training than the Baum-Welch training used with HMMs, as discussed in Section 2.1.2. As EM can be done in the context of the DBN probabilistic inference, we can take advantage of the local computations involved in such calculations so as to achieve efficient EM training. It involves maximizing the following expectation (Lauritzen, 1995, Section 3):

$$E\{\log P(\mathbf{e}, V_{1:N}|\lambda^i)|\mathbf{e}, \lambda\}. \tag{3.2}$$

As done in (2.8), (3.2) can be used to increase the data likelihood $P(\mathbf{e}|\lambda^i)$ in DBNs. Thus, we can use DBN probabilistic inference to learn the joint distributions of the cliques. These local joint distributions can then be marginalized to subsets of their component variables so as to obtain the desired distributions that we want to learn, such as:

$$P(V_n|\mathrm{pa}(V_n), \mathbf{e}, \lambda^i). \tag{3.3}$$

## 3.3 Probabilistic Inference Theory

As a note of introduction, each clique f will be performing a set of local computations related to its component variables. It stores some of its calculated values in a "potential" $\psi_{\mathrm{f}}$; a potential $\psi$ is an unnormalized distribution (i.e., its sum may be less than 1, which will occur as we compute the likelihood of its component variables). Thus, a potential is composed of a table, where each element $i$ has an associated likelihood $p(i)$, a mean vector $\mathsf{A}(i)$, a regression matrix $\mathsf{B}(i)$, and a covariance matrix $\mathsf{C}(i)$. As with probability distributions, we can have conditioning variables in the potential, such as $\psi(H|T)$; more generally, we can have, in the framework of Lauritzen and Jensen (2001), $\psi(D; H|T)$, where $T$ are continuous "tail" variables that condition the continuous "head" variables $H$ and where $D$ are discrete variables that, in the algorithm of Lauritzen and Jensen (2001) do not need to be distinguished as to being conditioning or not. Each of the element $i$ in the potentials' table is associated with an instantiation of the discrete variables in the potential, and the values of $\mathsf{A}(i)$, $\mathsf{B}(i)$, and $\mathsf{C}(i)$ for each $i$ refer to the continuous variables. Thus, the element resembles a conditional Gaussian, as discussed in Section 2.4.2. However, if $T = \{\emptyset\}$, then the element resembles a Gaussian, and if $H = \{\}$ (which also implies in this framework of Lauritzen and Jensen (2001) that $T = \{\emptyset\}$), then the element is a discrete probability. The continuous head variables are associated with the elements of the mean vector $\mathsf{A}(i)$, the rows of the regression matrix $\mathsf{B}(i)$, and the rows (and columns) of the covariance matrix $\mathsf{C}(i)$; the continuous tail variables are associated with the columns of $\mathsf{B}(i)$ (each column represents the regression weights of a tail variable on the head variables). Before discussing how to initialize the clique tree and do propagation in Section 3.3.4 & 3.3.5, respectively, I discuss some of the operations needed to do them: combination, marginals, and evidence incorporation in Sections (3.3.1), (3.3.2), & (3.3.3), respectively.

### 3.3.1 Combination

Combination involves taking two potentials and forming one distribution out of the two:

$$\psi^3 = \psi^1 \otimes \psi^2, \tag{3.4}$$

where $\otimes$ is the combination operator. It is used first in initialization of the clique tree; as the cliques contain several variables, we may need to initialize certain cliques by "combining" the prior distributions of more than one of their component variables. It is also used in propagation, where messages (Pearl, 1988, Chapter 5) are passed between cliques; when a clique receives a message (which is itself a potential $\psi_S$), it needs to combine it to its own potential $\psi_{\mathrm{f}}$ to form a new potential $\psi^*_{\mathrm{f}}$:

$$\psi^*_{\mathrm{f}} = \psi_{\mathrm{f}} \otimes \psi_S \tag{3.5}$$

This receiving of a message is equivalent to when, in the forward and backward recursions of HMMs, the current $\alpha$ (or $\beta$) uses information from a neighboring $\alpha$ (or $\beta$) value.

### 3.3.2 Marginals and Complements

Marginalizing involves taking a potential and factoring it into the marginal and the complement. For example, if we have $\psi(D_1, D_2; H_1, H_2|T)$, we can marginalize out $H_1$ so as to obtain the marginal $\psi(D_1, D_2; H_2|T)$ and the complement $\psi(D_1, D_2; H_1|H_2, T)$, which if re-"combined", equal the original potential:

$$\psi(D_1, D_2; H_1, H_2|T) = \psi(D_1, D_2; H_2|T) \otimes \psi(D_1, D_2; H_1|H_2, T). \tag{3.6}$$

Also, assuming there is no tail, (marginalization over discrete variables with a tail in the potential is not possible (Lauritzen and Jensen, 2001, Section 4.3)), we can marginalize out the discrete variables $D_1$ out of $\psi(D_1, D_2; H_1, H_2)$ to obtain the marginal $\psi(D_2; H_1, H_2)$ and the complement $\psi(D_1, D_2; H_1, H_2)$:

$$\psi(D_1, D_2; H_1, H_2) = \psi(D_2; H_1, H_2) \otimes \psi(D_1, D_2; H_1, H_2). \tag{3.7}$$

Marginalizing is necessary primarily for generating the messages that are passed during propagation (and which are then "combined" with a clique's potential, as referred to in Section 3.3.1). Its analogy in the HMM framework is when, in the forward and backward recursions, we do a sum (i.e., a marginalization) over all the values from a neighboring time frame, as shown in (2.13) on page 13, as well as in (2.16). However, in the DBN framework, the marginalization operation is more general and could involve a sum or an integral or a combination of the two.

It is in the marginalization operation that the Viterbi algorithm used in the HMM framework can be implemented in the DBN framework. That is, in DBNs, the marginal $\psi(D_2; H_1, H_2)$ in (3.7) is computed by

$$\psi(D_2; H_1, H_2) = \sum_{d_1} \psi(D_1 = d_1, D_2; H_1, H_2). \tag{3.8}$$

To do Viterbi in the DBN framework, this summation is replaced by a maximization:

$$\psi(D_2; H_1, H_2) = \max_{d_1} \psi(D_1 = d_1, D_2; H_1, H_2), \tag{3.9}$$

that is, for a given value $d_2$ in the new potential, the value for $\psi(D_2 = d_2; H_1, H_2)$ is determined by the whichever value $d_1$ has the highest likelihood (i.e., the $p(i)$ value in each element of the table) in $\psi(D_1 = d_1, D_2 = d_2; H_1, H_2)$

### 3.3.3 Incorporating Evidence

Incorporating evidence is the process of taking the observed value for a variable within a clique and of updating the parameters (the likelihood $p(i)$, the mean vector $\mathsf{A}(i)$, the regression matrix $\mathsf{B}(i)$, and the covariance matrix $\mathsf{C}(i)$ for all values $i$ of the discrete variables in the distribution) of its potential in light of this observation. When incorporating evidence for a discrete variable $V = v$ in $\psi_\mathrm{f}$, this involves setting $p(i) = 0$ all of the values in the $\psi_\mathrm{f}$'s table where $V \neq v$. When incorporating evidence for a continuous variable $V = v$ in $\psi_\mathrm{f}$, it is incorporated differently, depending on whether it is a "tail" variable or a "head" variable in the potential. If $V$ is in the tail $T$ of $\psi_\mathrm{f}$, then its corresponding regression weights are removed from its corresponding column in $\mathsf{B}(i)$ for each element $i$ in $\psi_\mathrm{f}$'s table; these regression weights are then used, weighted by $v$, to produce $\mathsf{A}^*(i)$, the updated version of $\mathsf{A}(i)$ for each element $i$, in light of this evidence

$$\mathsf{A}^*(i) = \mathsf{A}(i) + \mathsf{B}_V(i) \cdot v, \tag{3.10}$$

where $\mathsf{B}_V(i)$ is the column in $\mathsf{B}(i)$ corresponding to $V$. The effect is to shift the conditional Gaussian's mean $\mathsf{A}(i)$ so as to take into account the conditioning variable's observation. As the conditioning variable's value is known, its weights are no longer needed in the matrix $\mathsf{B}_V(i)$.

If $V$ is in the head $H$ of $\psi_\mathrm{f}$ and if $\psi_\mathrm{f}$ has no tail variables, then we remove its corresponding elements from $\mathsf{A}(i)$ and $\mathsf{C}(i)$: $\mathsf{A}_V(i)$ and $\mathsf{C}_{VV}(i)$. We then compute $p^*(i)$, the updated likelihood for $p(i)$ for each element $i$ in $\psi_\mathrm{f}$'s table, using the $\mathsf{A}_v(i)$ and $\mathsf{C}_{VV}(i)$ as the mean and covariance, respectively, the standard likelihood estimation for Gaussians (Papoulis, 1991, Section 4-3):

$$p^*(i) = p(i)\mathcal{L}(V = v | \mathsf{A}_V(i), \mathsf{C}_{VV}(i)). \tag{3.11}$$

The elements of $\mathsf{A}(i)$ and $\mathsf{C}(i)$ that were not removed are updated, so as to take into account the covariance between the remaining head variables and $V$, as explained in Lauritzen and Jensen (2001, Section 7): while not given in detail here, the remaining mean $\mathsf{A}_{\bar{V}}(i)$ is shifted (where $\bar{V} = H\backslash V$) and the remaining covariance $\mathsf{C}_{\bar{V}\bar{V}}(i)$ is updated (note that the covariances $C_{\bar{V}V}$ and $C_{V\bar{V}}$ do not remain after the incorporation). Note that there are exceptions to this for the case of having a variance of 0 in $\mathsf{C}(i)$, which is also dealt with in Lauritzen and Jensen (2001, Section 7). If $V$ is in the head of $\psi_{\mathrm{f}}$ and if $\psi_{\mathrm{f}}$ has a tail variable(s), then the evidence can not be incorporated into $\psi_{\mathrm{f}}$; rather, $\psi_{\mathrm{f}}$ is factored and a message containing $V$ is passed to the parent clique of f, which will then try to incorporate it, as explained in more detail in the "push" operation of Lauritzen and Jensen (2001, Section 6.2).

### 3.3.4 Initializing

With a strongly rooted clique tree, initial potentials are assigned to each of the cliques. This is done by first assigning each variable $V_n^i$ to a clique where it occurs with its parents $\mathrm{pa}(V_n^i)$. Then, for each clique, we do a "combine" of the (prior) distributions of all its assigned variables to obtain the clique's potential $\psi_{\mathrm{f}}$. For example, say that variables $V_n^1$ and $V_n^2$ as well as their respective parents, $\mathrm{pa}(V_n^1)$ and $\mathrm{pa}(V_n^2)$, all occur in clique f. Then, clique f's initial potential would be formed as:

$$\psi_{\mathrm{f}} = P(V_n^1|\mathrm{pa}(V_n^1)) \otimes P(V_n^2|\mathrm{pa}(V_n^2)).$$

The initialization is a simple process whose purpose is ensure that all of the local variable distributions are incorporated into the clique tree. Once done, this ensure that we can compute the joint potential (i.e., the unnormalized joint distribution) as such:

$$\psi = \bigotimes_{\mathrm{f}\in\text{ clique tree}} \psi_{\mathrm{f}}. \tag{3.12}$$

As at this point no evidence has been incorporated into the potentials, the joint potential is the same as the joint distribution of all the variables $V$:

$$\psi = P(V_{1:N}) = \prod_{V_n^i \in V_{1:N}} P(V_n^i|\mathrm{pa}(V_n^i)) \tag{3.13}$$

### 3.3.5 Propagation

While the initialization above does allow the clique tree to represent the joint distribution of all of the variables, we need a method for the cliques in the tree to "communicate" with each other. That is, if a clique's potential is changed (e.g., evidence is incorporated into it), it needs to let the other cliques know how its potential has been changed. To do so, messages are passed between neighboring cliques. A message is itself a potential whose component variables are those variables in common between the two cliques involved in the message passing (these variables are referred to as the "separators" between the two cliques). For clique $\mathrm{f}^1$ to form its message to $\mathrm{f}^2$, with separator variables $S_{\mathrm{f}^1\mathrm{f}^2}$, the potential $\psi_{\mathrm{f}^1}$ is factored into its marginal $\psi_{\mathrm{f}^1}^{\downarrow S_{\mathrm{f}^1\mathrm{f}^2}}$ and complement $\psi_{\mathrm{f}^1}^{|S_{\mathrm{f}^1\mathrm{f}^2}}$

$$\psi_{\mathrm{f}^1} = \psi_{\mathrm{f}^1}^{\downarrow S_{\mathrm{f}^1\mathrm{f}^2}} \otimes \psi_{\mathrm{f}^1}^{|S_{\mathrm{f}^1\mathrm{f}^2}}, \tag{3.14}$$

where $\psi_{\mathrm{f}^1}^{\downarrow S_{\mathrm{f}}^1\mathrm{f}^2}$ indicates $\psi_{\mathrm{f}^1}$ with the non-$S_{\mathrm{f}^1\mathrm{f}^2}$ variables marginalized out, with $\psi_{\mathrm{f}^1}^{|S_{\mathrm{f}}^1\mathrm{f}^2}$ being the respective complement.

More specifically, this message passing proceeds in two phases, "COLLECT" and "DISTRIBUTE" (Lauritzen and Jensen, 2001, Sections 5.1 & 5.2) (see also the discussion in Pearl (1988, Chapter 4)), which resemble the backward recursion and forward recursion, respectively, used in HMMs, as explained in Section 2.13. In COLLECT, we are starting at the bottom (the "leaves") of the clique tree, passing up messages recursively to each clique f related to the data likelihood of the cliques below f in the clique tree. DISTRIBUTE is performed after COLLECT. In DISTRIBUTE we are starting at the top (the "root") of the clique tree, passing down messages recursively to each clique f informing it of the posterior distribution of the f's separator variables (the variables in common between f and its parent).

- COLLECT: for each clique f in reverse topological order, letting $P$ be f's parent and $S_{\mathrm{f}P}$ the separator variables between the two:
  - compute the marginal $\psi_{\mathrm{f}}^{\downarrow S_{\mathrm{f}P}}$ and the complement $\psi_{\mathrm{f}}^{|S_{\mathrm{f}P}}$ (as discussed in Section 3.3.2). These are factors of $\psi_{\mathrm{f}}$: $\psi_{\mathrm{f}} = \psi_{\mathrm{f}}^{\downarrow S_{\mathrm{f}P}} \otimes \psi_{\mathrm{f}}^{|S_{\mathrm{f}P}}$.
  - "combine" the marginal with $\psi_P$ to form the updated form $\psi_P^*$ so that $P$ can take into account the potential $\psi_{\mathrm{f}}$:

    $$\psi_P^* = \psi_P \otimes \psi_{\mathrm{f}}^{\downarrow S_{\mathrm{f}P}}$$

  - assign the complement to $\psi_{\mathrm{f}}^*$, the updated version of $\psi_{\mathrm{f}}$:

    $$\psi_{\mathrm{f}}^* = \psi_{\mathrm{f}}^{|S_{\mathrm{f}P}}$$

  Note that as both factors of the original $\psi_{\mathrm{f}}$ are still accounted for (one combined with $\psi_P$ and the other left in the updated $\psi_{\mathrm{f}}^*$), that the joint distribution of all the variables in the BN is still represented by doing a "combine" over all of the potentials, as represented by (3.12).

- DISTRIBUTE: for each clique f in topological order, with parent $P$, its parent's separator variables $S_{PG}$ ($G$ is the parent of $P$) and its own separator variables $S_{\mathrm{f}P}$:
  - compute the marginal $\psi^{\downarrow S_{\mathrm{f}P}}$:

    $$\psi^{\downarrow S_{\mathrm{f}P}} = (\psi^{\downarrow S_{PG}} \otimes \psi_P)^{\downarrow S_{\mathrm{f}P}}$$

  - assign this marginal to $\psi_{S_{\mathrm{f}P}}$, producing the updated version $\psi_{S_{\mathrm{f}P}}^*$:

    $$\psi_{S_{\mathrm{f}P}}^* = \psi^{\downarrow S_{\mathrm{f}P}}$$

  Note that DISTRIBUTE does not change the potentials of the cliques. Rather, it merely gives the potentials to the separators for each clique. So, (3.12) and (3.13) still hold.

## 3.4 Probabilistic Inference Implementation

For reasons of optimizing both computational costs and memory size, the above algorithm can be altered while being theoretically equivalent. For the DBNs used in my work, where there is a high-dimensional, sparse discrete space, it is very important to implement the `Enumerate_Legal_Values` algorithm found in Figure 3.4 of Zweig (1998), the purpose of which is to take advantage of the known sparsity in the cliques; in doing so, the computational load is reduced substantially as we are not doing operations on the many zero values in the cliques. As there are discrete probability tables in my DBNs that are sparse (i.e., having mainly probabilities of zero), we only want the inference to deal with those values in the DBN corresponding to the non-zero values in the tables. To do otherwise would make the inference computationally infeasible in my DBNs.

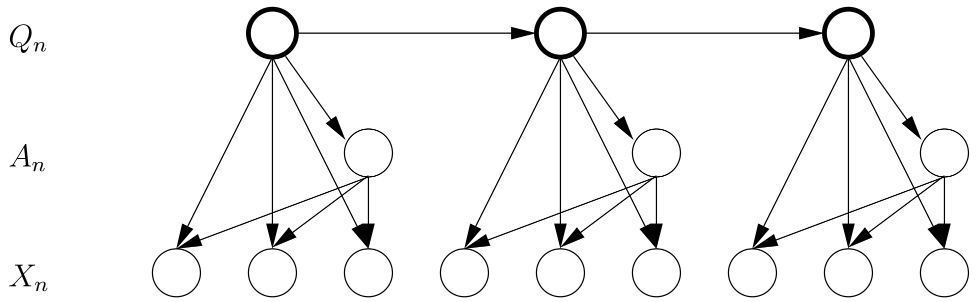


Figure 3.1: DBN for "auxiliary" ASR, here representing feature vectors $X_n$ with three elements each.

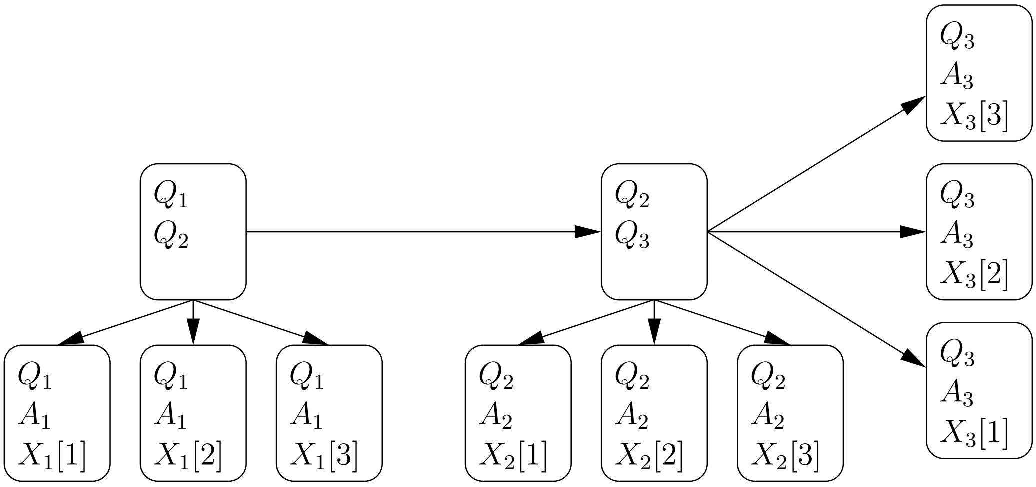


Figure 3.2: Clique tree corresponding to Figure 3.1.

### 3.4.1 Initializing and Incorporating Evidence

This initial assigning of the variables' distributions is incorporated into the algorithm entitled `Enumerate_Legal_Values`. I have also chosen to integrate as much as possible of the evidence incorporation in the initial assigning of the variables' distributions to the cliques. For example, take the distribution $p(X_n|A_n, Q_n)$, where $H = \{X_n\}$, $T = \{A_n\}$, and $\Delta_\text{f} = \{Q_n\}$. This distribution would have a weight $p_k$, mean $\mu_k$, regression weights $B_k$, and covariance $\Sigma_k$ for each discrete value $k$ of $Q_n$. Using this distribution for each time frame $n = 1, \ldots, N$ would mean copying all of the "legal" parameters for $p_k$, $\mu_k$, $B_k$, and $\Sigma_k$ for each value $k$ of $Q_n$. This would take up lots of memory, in addition to any time spent doing such copying. So, if we have evidence for, say, both $X_n$ and $A_n$ for all time frames, we can proceed as follows: when we are assigning the distribution for $X_n$ at a given time frame, we can access the $p_k$, $\mu_k$, $B_k$, and $\Sigma_k$ parameters and incorporate the corresponding evidence for $X_n$ and $A_n$ at that time frame. We then only need to assign the updated $p_k$ (the likelihood) for the potential for each value of $Q_n$.

### 3.4.2 Propagation

Furthermore, I have chosen to integrate the above initialization and evidence incorporation into the COLLECT algorithm. This has, first, to do with the fact that many of the cliques, after having their evidence incorporated, have the same remaining component variables in their potentials. For example, take the cliques, "$Q_1 - A_1 - X_1[1]$", "$Q_1 - A_1 - X_1[2]$", and "$Q_1 - A_1 - X_1[3]$" from Figure 3.2 for the DBN in Figure 3.1; after incorporating the evidence for $A_1$, $X_1[1]$, $X_1[2]$, and $X_1[3]$ into these cliques, they are each left with the same variables: "$Q_1$", though with different '$p$' values. Therefore, after "enumerating" these legal values for all such cliques for a given time frame, the memory used for them can be taken back right after the evidence incorporation by combining them together into their parent and before moving on to do the same in another time frame. Furthermore, when DISTRIBUTE is called, it will not have to deal many times with what have become cliques with identical topologies.

### 3.4.3 Combination

Lauritzen and Jensen (2001) presents a simple algorithm called "Extension" which enables the combination to be presented in simple terms of matrix multiplications and additions. However, the Extension only adds zero terms to the potentials; in other words, they are just place-holders in the potential for simplifying the matrix operations. Furthermore, the combination will be wasting lots of time with additions and multiplications upon these "place-holders". Therefore, I have chosen to perform combinations in arranging the calculations such that the equivalent outputs are produced but without having to do the "Extension." This involves doing matrix operations on only selected parts of vectors and matrices. While not used in my code itself, this could potentially be optimized, in C++, using the "slice" functionality provided with the `valarray` class (Josuttis, 1999).

## 3.5 Conclusion

Having presented DBNs previously in Chapter 2, I have elaborated in this chapter how inference is done in them. While the general theory presented in Section 3.1 and 3.3 is not original to this thesis, I have contributed to this field by illustrating in Section 3.3 different methods that can be used when actually implementing DBNs for ASR; some of the methods in Section 3.3 were taken from Zweig (1998) (namely, the `Enumerate_Legal_Values`). Incorporating such implementations makes the examination of the auxiliary information presented in the following chapters more feasible.

# Chapter 4

# Auxiliary Information

Auxiliary information is any information outside of the standard acoustic features that aids in modeling the distribution of the standard features. They aid the modeling in that, because of their correlation with the standard features, they help to better handle the wide variation that the standard features can have. The auxiliary information needed for these more robust distributions could come from different sources, such as a second source of acoustic information; visual information; articulator values; or, as one if its best examples, gender.

Standard ASR uses only the standard acoustic feature vectors $x_{1:N} = \{x_1, \ldots, x_N\}$ for time $n = 1, \ldots, N$, for its modeling task. These standard feature vectors are usually derived using some form of cepstral analysis. These feature vectors are typically assumed to be conditionally independent, identically distributed (c.i.i.d.) and are an attempt to extract the information from the signal that is the most useful for the modeling. However, the information contained within $x_n$ appears to be insufficient for modeling the acoustics as ASR performance typically degrades when given a variety of input acoustics (e.g., background noises not seen during system training). I show that improved performance can come from better modeling of the distribution of $X_n$ by conditioning its distribution upon auxiliary information $A_n$.

Auxiliary information only has use in the context of standard information. So I here first describe what standard information is. I then describe the relation between the auxiliary information and both the standard information and the emitting states. The auxiliary information that I refer to can come in different ways: it can be static, semi-static (i.e., slowly changing), or it can be dynamic (varying each time frame); it can be precisely known by a source outside $x_n$ or it can be estimated from the same sources as $x_n$. In this chapter, I will present the various types of auxiliary information that I am investigating. First I will give some motivation as to why auxiliary information should be used in Section 4.1 as well as how this relates to previous work in Section 4.2.1. I will then discuss discrete auxiliary information in Section 4.3 and continuous auxiliary information in Section 4.4. I close with a discussion on a special type of discrete auxiliary information, in the form of a second chain, in Section 4.5. The main novelties of this chapter are the use of auxiliary information in training but not in recognition; the illustration of a factorial HMM with two different types of chains, each representing different types of states; and a unified overview of different attempts at auxiliary information, including an investigation of different ways of incorporating "real" auxiliary information into the modeling;

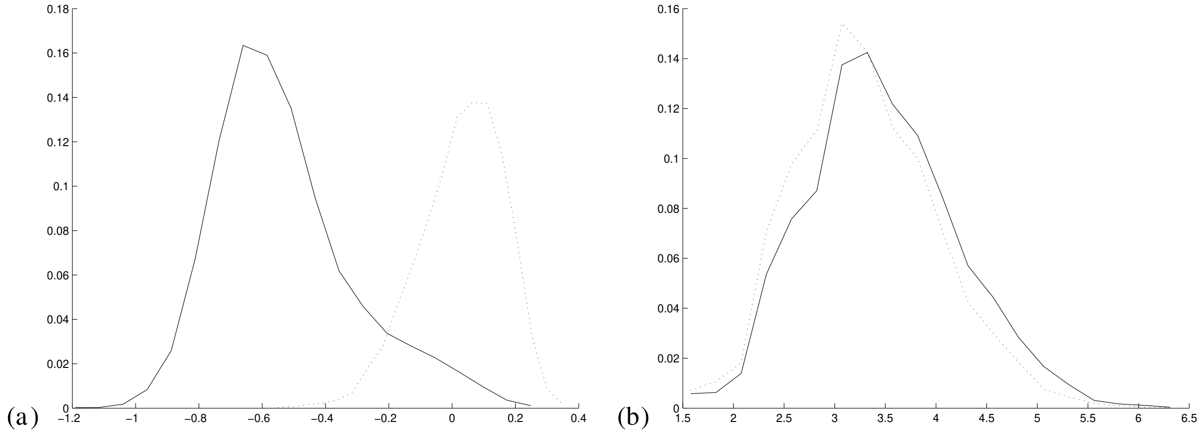

Figure 4.1: Illustration of the *ideal* type of distributions according to my definition of auxiliary information: (a) $X_n$ having different, discriminant distributions (shown is the distribution of the first PLP coefficient for the phonemes /ao/ and /f/); (b) $A_n$ of rate-of-speech (ROS) having similar, non-discriminant distributions (shown is the distribution of ROS for the phonemes /ao/ and /f/). These are normalized, empirical distributions using all of the training data for the respective phonemes, using the segmentation used for EM initialization (the segmentation obtained using a forced-alignment as discussed in Section 6.1 on page 77).

## 4.1 Standard vs. Auxiliary Information

The standard information is the features $x_{1:N} = \{x_1, \ldots, x_n, \ldots, x_N\}$ that are useful for discriminating between the different possible values $1, \ldots, K$ of the hidden states $Q_n$ for each time frame. That is, the nature of the features is that we would expect the distribution of $X_n$ to lie in a distinctly different area for each value of $Q_n$. If any of the elements in the feature $X_n$ are found to not discriminate between the different values of $Q_n$, then it may be better to remove such elements from the feature vector.

In modeling the standard features, we are hoping that the distributions will be separated enough for each state so as to allow good discrimination between the states. Furthermore, we would want that the distributions be robust to noise and speaker/environmental changes that may arise during utilization of the trained system. The current performance of state-of-the-art systems suggests that these standard features are not robust enough. Therefore, there may be some other features, referred to as *auxiliary* features, that, by conditioning the distributions of the standard features, can make them more robust; so, even in noise, the distributions for $X_n$ could hopefully discriminate well between the states due, in part, to the information carried in $A_n$.

The information in the auxiliary features is different from the standard features in that it is not directly dependent upon the emitting states, as illustrated in Figure 4.1. That is, while the low-level standard features are changing as the hidden state changes, the higher-level auxiliary features change due to other factors and may be more slowly changing than the standard features. See Figure 4.2. Since the auxiliary information is not dependent directly upon the states, it makes sense to model it independently of the states. Now, this auxiliary information is a fundamental trait of the speech signal, meaning that there will quite possibly be a correlation between the standard features and the auxiliary features. Hence, I propose that, while the auxiliary information is independent of the state, it can aid the modeling of the standard features.

There are different possible sources for this auxiliary information. It can be known precisely; for example, with discrete auxiliary information of gender (Konig and Morgan, 1992) or the contin-

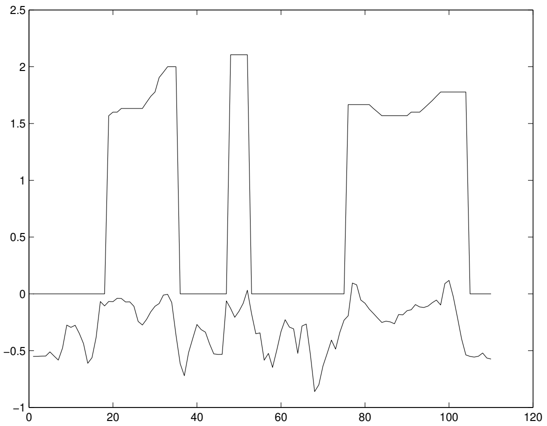


Figure 4.2: Correlation between $X_n$ and $A_n$. Training utterance from the Numbers database for a female uttering "thirty-six twelve." The first PLP coefficient, PLP[1], (one of the "standard" features) is plotted below its corresponding pitch estimate (an "auxiliary" feature). The $x$-axis represents the frame number; the $y$-axis represents the value of PLP[1] for the lower plot and the pitch estimate divided by 100 for the upper plot. Note the positive correlation between PLP[1] and pitch: when there is non-zero estimated pitch, PLP[1] tends to have higher values. Furthermore, while PLP[1] changes its value rapidly, the pitch estimate changes slowly over time, hence carrying higher-level information (that is, information related to the speaker or utterance, as opposed to the phoneme).

uous auxiliary information of a child's age (to take into account the changing attributes of a child's voice as he matures). It can be obtained using precise measurements, for example, using auxiliary information of the articulator positions (Westbury *et al.*, 1994; Wrench, 2000). Finally, it can be estimated information, for example, information regarding pitch, rate-of-speech, and energy, all estimated from the signal.

## 4.2 Relation to previous work

### 4.2.1 Discrete conditioning variables

The concept of an auxiliary variable was in the first work that seriously addressed issues related to using DBNs in ASR (Zweig, 1998), where it was typically referred to as a "context" variable. The term *context* refers to how it attempts to model the features in relation to the features at the previous time frame as well as how it attempts to models the correlation between features at the same time frame. This is captured in the four different topologies that he used, as illustrated in Figure 4.3.

Zweig (1998) showed the benefit of having an auxiliary variable to model this contextual information. The auxiliary variable was always hidden, both in training and in testing. Thus, in training a latent variable, we can not be certain of what information it might be modeling. We can, however, attempt to bias what information it is carrying; for example, Zweig gave the training initial distributions for the variable that reflected what sort of information (e.g., voicing) that he wanted them to model. In treating the auxiliary variable as a latent variable, he showed that the "Chain" BN in Figure 4.3 can be used to do unsupervised clustering according to both speaker types and word types. His thesis gave a lot of the theoretical background for pursuing auxiliary variables

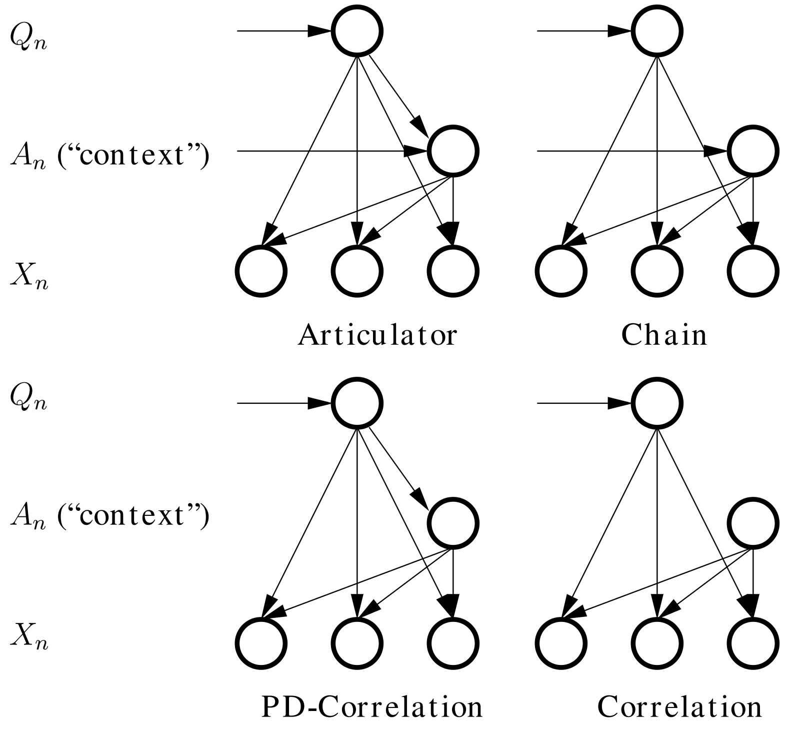


Figure 4.3: Four BNs for modeling "context" information with their names ("PD" means phoneme-dependent), as used in Zweig (1998). The context variable, while possibly given initial distributions with certain characteristics before training, remains hidden in both training and testing.

that are trained on actual data, instead of being latent in training. He, however, did not do any experimental investigation on actual data. Furthermore, his thesis dealt with discrete DBNs, with minimal discussion related to mixed DBNs. So, in this thesis, I am continuing his line of research by using auxiliary variables with real data and in using mixed DBNs.

Discrete auxiliary information has also been investigated using the gender (male/female) of the speaker. Konig and Morgan (1992) had two general approaches, each of which uses a "Gender Classification Network" ANN to try to determine the gender of the speaker. In one approach, the two outputs of the gender classifier are treated as additional features. In the other approach, the outputs of the gender classifier are used to determine which of two gender-dependent acoustic models to use during decoding. In a similar manner to using the gender as a discrete auxiliary information, the discretized speaking rate has also been used. Martínez *et al.* (1998) proposed having a set of HMM parameters adapted to slow speech and another adapted to fast speech. In conjunction with the two sets of parameters, they proposed a "speech rate classifier" (SRC) that gave an estimation of the speaking rate. Using the two sets of HMM parameters in conjunction with the estimates given by the SRC, reduced the word error rate for slow speech as well as for fast speech (but increased the word error rate of normal speech a small amount). In this thesis, my discrete auxiliary information changes from frame-to-frame; it is not static (like gender) or "semi-static" (like speaking rate). Furthermore, instead of taking the most likely auxiliary information at each frame, I take into account the likelihood of each possible auxiliary value.

Logan and Moreno (1998) introduced factorial HMMs, in practice, to speech recognition. With factorial HMMs (Ghahramani, 1997), the idea is to factor the discrete state space into two separate discrete chains, each sharing the same observation space, as illustrated in Figure 4.8 on page 62.

Logan and Moreno (1998) actually had a modified factorial HMM where the same discrete space was used for each chain and where each chain had different observations; with such modifications, it resembles a multi-stream approach to ASR (Dupont, 2000, Chapter 5). In this thesis, I further this work by simultaneously training two chains (one of them being "auxiliary") which have different discrete spaces and where the two chains have the same observations; however, in recognition, I remove the time-dependency for one of the chains.

### 4.2.2 Continuous conditioning variables

A continuous auxiliary variable with data in both training and testing was used in Fujinaga *et al.* (2001). This was not done in the context of DBNs–though it could easily have been done. They illustrated how continuous auxiliary information can be used in a regression to better model the Gaussian distribution of the regular features. They showed how using auxiliary information in a standard manner (that is, appending it to the standard feature vector used in the HMMs) in speaker-dependent phoneme or isolated word recognition typically increases the error rate (they hypothesized that the cause of this increase in error was the "curse of dimensionality" of having too large a feature vector after the auxiliary feature(s) were appended). However, if this auxiliary information conditions the standard features by shifting their expected mean value, then the error rate decreases. A further benefit of conditioning the standard features on the auxiliary information is that the variances within the resulting trained Gaussians are smaller (assuming the auxiliary information is correlated with the standard features). They used auxiliary information related to pitch and energy characteristics.

The results with Fujinaga *et al.* (2001) give an indication as to why there has not until recently been much use of "auxiliary" information in ASR. While some of the auxiliary features that they used provided some marginal improvement in standard HMM phoneme recognition, none of them provided any improvement in standard HMM isolated word recognition. This fits in with my argument that certain auxiliary information needs to condition the standard features as well to be independent of the state, both of which were done successfully in Fujinaga *et al.* (2001). I also investigate continuous pitch and energy related features (as well as speaking rate) that condition Gaussian distributions. I further this work both by looking at the effect of hiding the continuous auxiliary feature in the testing phase and by looking at the effect of having this auxiliary feature itself modeled dependent upon the state. Additionally, I am looking at the broader task of speaker-independent, spontaneous speech recognition in noisy conditions.

While a discretized speaking rate was shown in Martínez *et al.* (1998), as mentioned in Section 4.2.1, a continuous speaking rate was used in Morgan *et al.* (1997). Instead of conditioning all of the HMM parameters, it conditioned only the transition probabilities out of the discrete states. Using such an approach provided a 14% reduction in word error rate. Furthermore, note that these results held for the case where they used a speaking rate estimated from the signal (Morgan *et al.*, 1997) or from a transcription. In this thesis I show how the acoustics can be conditioned by the speaking rate (like Martínez *et al.* (1998)), and how the speaking rate can be used in a continuous way to "shift" the distributions of the acoustics.

While not using "auxiliary" variables, as used in my thesis, Bilmes (1999), which used buried Markov models (BMMs), and Wellekens (1987) conditioned the distribution of the observations upon observations from the previous time frame(s). Bilmes (1999) showed the benefit of selectively including dependencies from previous time frames, with this set of dependencies changing according to the value of the discrete state. My work deals in conditioning the standard features not upon features from other time frames but on, rather, conditioning them on the "auxiliary" variable from the same time frame. I do note that, while not doing any experimental investigation into auxiliary variables, Bilmes (1999, Section 4.7.2) does give the background for incorporating them, referring to the auxiliary variable by $Y_t$, into BMMs[1].

## 4.3 Discrete Auxiliary Information

My work with discrete auxiliary information was done with discrete DBNs (where all variables are discrete). The exception where discrete auxiliary information is used in mixed DBNs (i.e., DBNs with both discrete and continuous variables) is the case of "auxiliary" chain information, which is dealt with separately in Section 4.5. Discrete DBNs have, in a way, a much simpler inference algorithm. As such, they serve as a good tool for the initial investigations into auxiliary information in DBNs. They do not have the computational problems that (hidden) continuous variables can pose to probabilistic inference; they do, however, have their own computational problems when the discrete space gets too large.

In discrete DBNs for ASR, we are modeling the discrete acoustic observation $x_n$, which, in the framework of Zweig (1998), involves having three dimensions $x_n[1]$, $x_n[2]$, and $x_n[3]$ (representing the MFCCs, the approximate first-derivative of the MFCCs, and the zeroth MFCC with its approximate first-derivative, respectively; note that the number of dimensions of $x_n$ can be changed if other feature representations are desired), and the hidden discrete state for each time $n$:

$$P(X_n = x_n|Q_n) = P(X_n[1] = x_n[1]|Q_n) \cdot P(X_n[2] = x_n[2]|Q_n) \cdot P(X_n[3] = x_n[3]|Q_n), \quad (4.1)$$

thus making the standard assumption that the dimensions of $X_n$ are (conditionally) independent of each other and that $X_n$ is (conditionally) time-independent. That is, the uncorrelated dimensions of $X_n$ are conditionally independent, identically distributed (c.i.i.d.) (see Section 2.2.1 on page 15 ff). My first investigations into auxiliary information $A_n$, then, is to incorporate it as time-dependent information that conditions $X_n$ and can potentially be hidden:

$$P(X_n, A_n|Q_n, A_{n-1}) = P(X_n|Q_n, A_n) \cdot P(A_n|A_{n-1}, Q_n). \quad (4.2)$$

Note, therefore, that the auxiliary feature here is not treated as if it were just another dimension of the standard feature $X_n$; if it had been, then $A_n$ would not be conditioning $X_n$ and $A_{n-1}$ would not be conditioning $A_n$. Figure 4.4 (a) illustrates the BN that depicts this distribution. As some auxiliary information may have its origin above the phone-level, it may not be related to the phonetic-state $Q_n$. However, it remains a time-dependent information that does condition $X_n$ itself, as represented in Figure 4.4 (b):

$$P(X_n, A_n|Q_n, A_{n-1}) = P(X_n|Q_n, A_n) \cdot P(A_n|A_{n-1}), \quad (4.3)$$

which, because there are two conditionally independent Markov chains, can be viewed as a factorial HMM, as discussed in Section 4.2.1. In either case, the auxiliary information may be either missing or noisy at any or all time frames. So, in discrete DBNs (specifically, with discrete $A_n$), we can marginalize out the $A_n$ by a summation over all $L$ values of $A_n$. In the case of (4.1) and (4.2),

[1] I thank an anonymous reviewer of a paper (submitted to a journal) written in preparation for this thesis for pointing this out.

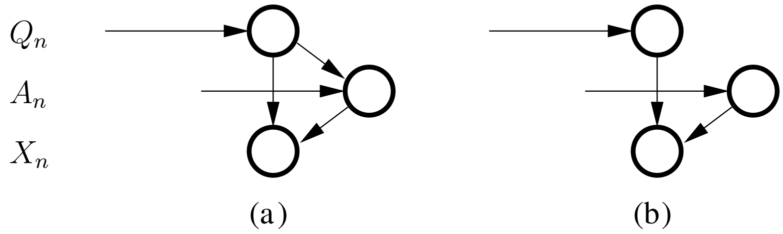


Figure 4.4: Discrete BN carrying time-dependent auxiliary information $A_n$ that conditions $X_n$. (a) has state-dependent $A_n$, as in (4.2); (b) has state-independent $A_n$, as in (4.3).

respectively:

$$\begin{aligned}
P(X_n|Q_n, A_{n-1}) &= \sum_{a_n=1}^{L} P(X_n, A_n = a_n|Q_n, A_{n-1}) \\
&= \sum_{a_n=1}^{L} P(X_n|Q_n, A_n = a_n) \cdot P(A_n = a_n|A_{n-1}, Q_n) \qquad (4.4)
\end{aligned}$$

$$\begin{aligned}
P(X_n|Q_n, A_{n-1}) &= \sum_{a_n=1}^{L} P(X_n, A_n = a_n|Q_n, A_{n-1}) \\
&= \sum_{a_n=1}^{L} P(X_n|Q_n, A_n = a_n) \cdot P(A_n = a_n|A_{n-1}) \qquad (4.5)
\end{aligned}$$

## 4.4 Continuous Auxiliary Information

### 4.4.1 Statistical Assumptions-DBNs (with GMMs)

These various methods outlined below for including $A_n$ in GMM based modeling can also be done in the framework of HMMs. However, different software would need to be developed for handling each change in the assumptions for handling $A_n$. DBNs provide the general framework for being able to handle all of these assumptions in the same software, using the same algorithms.

In general, all of the DBNs in my experiments with continuous auxiliary information $A_n$ assume that $A_n$ is time-independent. That is, $A_n$ is conditionally independent of $A_{n-1}$. This has been done because of the complexity issues involved when having $A_n$ dependent upon a hidden $A_{n-1}$, as discussed in Section 2.4.3 on pages 23 ff. As a result, the following time-dependent ways to model $A_n$ have not, to the best of my knowledge, been examined experimentally by anyone in the context of ASR with continuous, hidden $A_n$ (nor are they examined experimentally in this thesis):

$$p(A_n|Q_n, A_{n-1}) \qquad (4.6)$$

$$p(A_n|A_{n-1}). \qquad (4.7)$$

(4.6) could have been incorporated into both (4.10) and (4.14) below and where (4.7) could have been incorporated into (4.12) below. (Note that Bilmes (1999) does provide a framework for investigating (4.7) with observed $A_n$ in the context of BMMs but, as far as I am aware, has not experimentally tested it). (4.6) and (4.7) could have been investigated, without the complexity issues with only observed $A_n$. This is because the $d$-separation property (cf. Section 2.4.3) between $A_n$ and $Q_{1:N}$ remains the same whether $A_n$ is dependent upon the observed $A_{n-1}$ or not: $A_n \perp\!\!\!\perp A_{n'} \,|\, Q_{1:N}$, $(n' <$

$n-1$ or $n' > n+1$). The only case with observed, time-dependent $A_n$ that I have not investigated yet which is of interested to this thesis is that of incorporating (4.6). (4.7) would not have been of interest with only observed $A_n$ as, since there are no hidden variables in its distribution, it will add nothing to the modeling.

### $X_n$ only (Baseline, Figure 4.5 (a))

The baseline systems with no auxiliary variable are theoretically equivalent to standard HMMs, as represented by the emission distribution $p(X_n|Q_n)$, which when using Gaussian mixture models (GMMs) in the DBNs is expanded as:

$$p(X_n|Q_n) = \sum_{j=1}^{J} P(J_n = j|Q_n) \cdot p(X_n|Q_n, J_n = j), \tag{4.8}$$

where $j$ is the mixture component and $J$ the number of mixture components. The elements of these standard features, furthermore, are assumed to be statistically independent of each other, within a given mixture component of a given state. This means that, in modeling them as multi-Gaussian distributions, they have diagonal covariance matrices. Doing so reduces the complexity in the models — each covariance matrix has only $P$ parameters instead of $\frac{(P+1)P}{2}$. With this reduced complexity, more robust models can be learned without having to get the large amounts of data that very complex models would demand for effective learning. Furthermore, as discussed in Assumption 4 on page 17, there is still some indirect modeling of the correlation between the elements of $X_n$ via $J_n$ (as well as via $Q_n$). So, given the $P$ parameters used for the variances in each of the $J$ components of the GMM used in (4.8), (not including the parameters for the mixture component weights) the covariance of the full GMM is effectively modeled with $JP$ parameters.

### $X_n, A_n$ (no assumptions, Figure 4.5 (b))

Without any statistical assumptions between $A_n$, $X_n$, and $Q_n$, the emission distribution $p(X_n, A_n|Q_n)$ is modeled as:

$$\begin{aligned} p(X_n, A_n|Q_n) &= p(X_n|A_n, Q_n) \cdot p(A_n|Q_n) & (4.9)\\ &= \sum_{j=1}^{J} P(J_n = j|Q_n) \cdot p(X_n|A_n, Q_n, J_n = j) \cdot p(A_n|Q_n), & (4.10) \end{aligned}$$

Thus, $A_n$ is serving as an auxiliary variable to $X_n$ by conditioning its distribution. By having $A_n$ condition the distribution for $X_n$, some of the covariance between the elements is further modeled implicitly. That is, by themselves, the elements of $X_n$ are assumed to be uncorrelated (given $Q_n$ and $J_n$). However, each element of $X_n$ has a different regression weight upon the value of $A_n$, found in the $B$ matrix described in Section 2.4.1. Therefore, each element of $X_n$ will then be correlated with each other, via $A_n$. So, in effect there is then a "full" covariance matrix (i.e., every element can be non-zero) for mixture component of the GMM for $X_n$ but with only a limited number of free parameters. That is, when the distributions of $X_n$ and of $A_n$ are combined (with $A_n$ being an $S$-dimensional vector), then the resulting $(P+S)\times(P+S)$ covariance matrix for each discrete mixture component will have $P + 2S + PS$ free parameters; in other words, the free parameters are the $P$ variances of $X_n$, the $S$ means of $A_n$, the $S$ variances of $A_n$, and the $P \cdot S$ regression weights of $A_n$ upon $X_n$, all of which are used as specified in Lauritzen and Jensen (2001, Section 4.4). This results in $J(P + 2S + PS)$ total parameters for such a "conditional" GMM (not including the parameters for the mixture component weights). So, in effect, we have two options available for improving the modeling of the elements of $X_n$: we can use the standard discrete mixture component variable

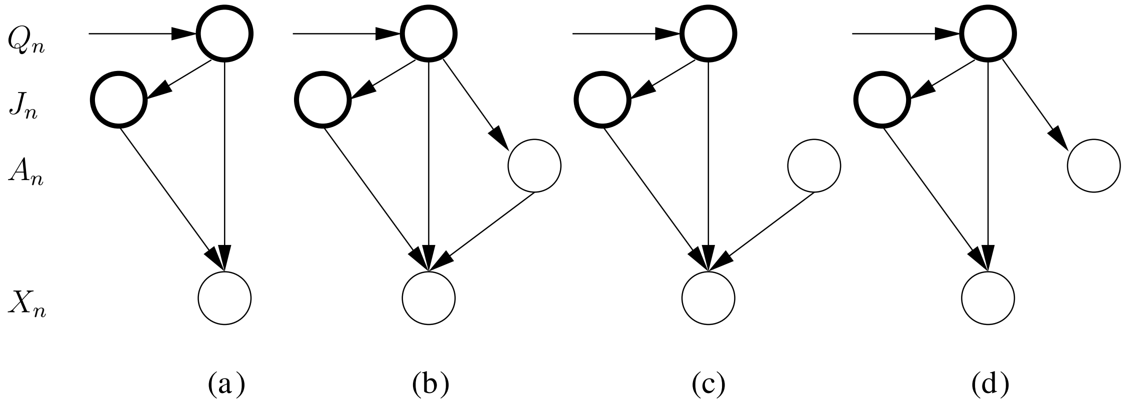


Figure 4.5: BNs for ASR: (a) has only $x_n$; (b)-(d) have different ways of incorporating $a_n$. Figure 5.4 shows the full DBN, with the control layer and multi-dimensional $x_n$, for the case of (b).

$J_n$ and we can use a continuous "mixture" component variable $A_n$. We would want to use the $A_n$ directly conditioning $X_n$ (instead of making them both part of the same GMMs), if there was a substantial correlation between $A_n$ and $X_n$ which could not be modeled effectively using only a reasonable number of Gaussian mixtures components.

Note that the use of (4.10) instead of (4.8) represents a small increase in computation for each mixture component. That is, as (4.10) uses a conditional Gaussian, there is an additional multiplication and addition to shift each mean according to the value of $A_n$ (assuming $A_n$ is observed).

### $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$ (Figure 4.5 (c))

Equation (4.10) does not make the assumption that $A_n$ is independent of the state. However, my standard way of incorporating $A_n$ involves treating $A_n$ independent of $Q_n$:

$$\begin{aligned} p(X_n, A_n|Q_n) &= p(X_n|A_n, Q_n) \cdot p(A_n) && (4.11)\\ &= \sum_{j=1}^{J} P(J_n = j|Q_n) \cdot p(X_n|A_n, Q_n, J_n = j) \cdot p(A_n). && (4.12) \end{aligned}$$

The correlation between the elements of $X_n$ is still modeled implicitly through their mutual dependence upon $A_n$ (as upon $Q_n$ and $J_n$)–see Figure 4.6. However, $A_n$ is merely given a simpler distribution, that is, one independent of $Q_n$. Additionally, in using conditional GMMs for each state, DBNs both with $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$ and with $X_n, A_n$ above, there will be smaller variances for $X_n$, as part of the variance is accounted for by $A_n$, as illustrated in Figure 2.5 on page 22.

At this point I would like to draw attention to the above two ways of incorporating $A_n$: $X_n, A_n$ and $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$ regarding two different points. First, by conditioning the distribution for $X_n$, the auxiliary variable $A_n$ can be viewed as a continuous mixture component variable. In being a (discrete) mixture component variable, $J_n$ itself indicates which Gaussians component(s) to use for a given time frame as well as their respective weight(s) (i.e., probability(ies) within $J_n$ itself). In a similar manner, the continuous "mixture" component variable, $A_n$, gives a continuous range of Gaussians to use for a given time frame as well as the respective continuous range of weights (i.e., the probability distribution of $A_n$ itself). The advantage of having a continuous mixture component variable is that there is, in a sense, an infinite number of mixture components, with those closest to the mean of $A_n$ being the most likely ones. Conversely, the advantage of having a traditional discrete mixture component variable is that the parameters of each discrete mixture component

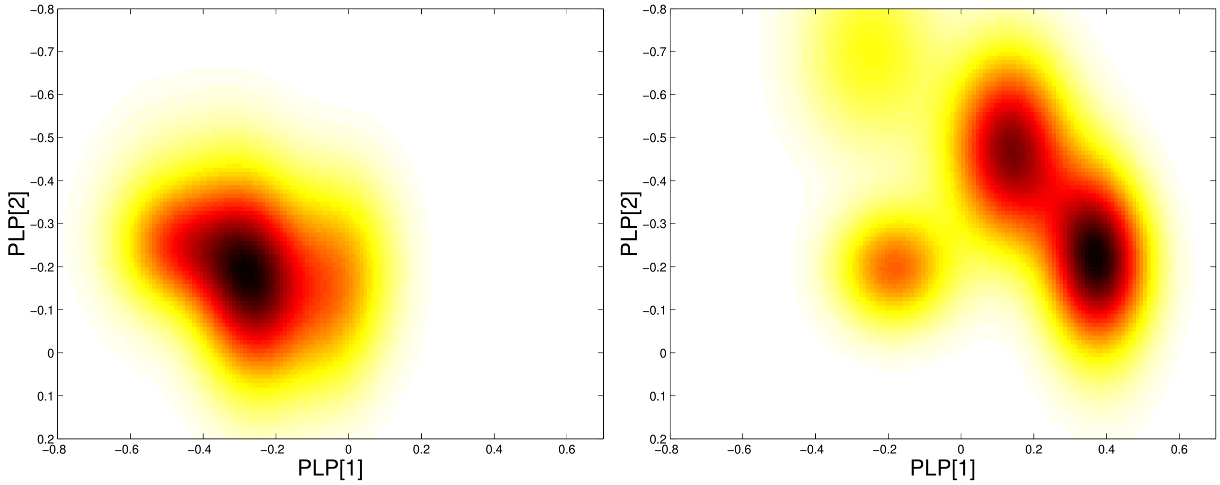


Figure 4.6: Conditional Gaussian mixture models, illustrated by the first state of the phoneme /w/ and the first and second PLP coefficients with energy as the auxiliary variable. These two graphs are taken from a single conditional GMM that was learned from the OGI Numbers data for the "$A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$" system in Table 6.11. On the left is the result of conditioning the (conditional) GMM on a low energy value; on the right is the result of conditioning the same (conditional) GMM on a high energy value. The resulting GMMs after conditioning change with different conditioning values. Furthermore, with different energy values, the covariance (as indicated by the shape of the GMMs) changes as a function of the energy value.

can be tailored for each of the discretized mixture components; that is, the weight, mean, and variance can be different for each component, unlike with using $A_n$, whose weight upon $X_n$ is a function of $A_n$ (in this work, this is a linear function) and whose variance is uniform (though an area of research would be to investigate changing this variance also using a function of $A_n$). I do not argue for or against the exclusive use of a continuous mixture component variable verse a discrete mixture component variable (I actually use both in the same models). Rather, I point out that they each have their own aspects to contribute to better modeling. Second, I have chosen in this work on continuous $A_n$ to not investigate how to best model $A_n$ itself, except for allowing it in certain cases to be dependent upon $Q_n$. As a result, I have not made $A_n$ dependent upon the mixture component variable $J_n$. When using continuous $A_n$, I am concentrating on how to best model $X_n$ itself; any other ways to model continuous $A_n$ (in addition to its dependency or independency upon $Q_n$) is left for future work (as explained in Chapter 7).

### $X_n \perp\!\!\!\perp A_n \,|\, Q_n$ (Figure 4.5 (d))

Conversely, should $A_n$ be dependent upon $Q_n$ but independent of $X_n$ (and, thus, not truly auxiliary), we would model:

$$\begin{aligned} p(X_n, A_n|Q_n) &= p(X_n|Q_n) \cdot p(A_n|Q_n) && (4.13)\\ &= \sum_{j=1}^{J} P(J_n = j|Q_n) \cdot p(X_n|Q_n, J_n = j) \cdot p(A_n|Q_n). && (4.14) \end{aligned}$$

Note that this is equivalent to having included $A_n$ among $X_n$ except that $A_n$ is not modeled here by mixture distributions. That is, $X_n$ and $A_n$ are assumed to be uncorrelated with each other (given $Q_n$).

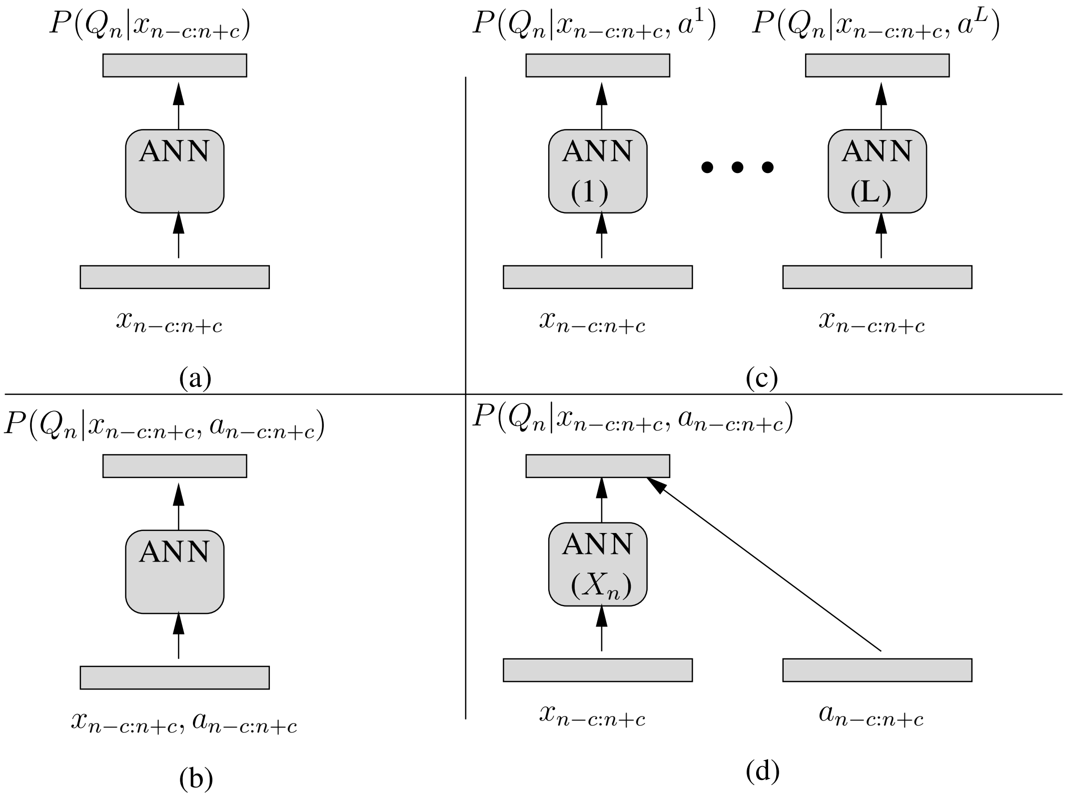


Figure 4.7: ANNs for hybrid HMM/ANN ASR. Presented are (a), the BASELINE $X_n$ only ANN; (b), the ANN with $a_n$ appended to $x_n$ and used as a standard feature; (c), the multiple ANNs using discretized auxiliary information (with $L$ values); and (d), the two ANNs treating $a_n$ outside of the hidden layer for $X_n$.

### 4.4.2 Statistical Assumptions-HMM/ANNs

As hybrid HMM/ANN ASR is a competitive method compared to GMM based ASR (as presented in Section 4.4.1), I have investigated[2] how to incorporate $A_n$ with the same assumptions as with the GMMs. In investigating using auxiliary information in HMM/ANN systems, I only look at how to change the ANNs so as to accommodate auxiliary information; the HMM portion of the system is always used in the same standard way. That is, I look at how to take advantage of using auxiliary information so as to get better scaled likelihoods from the ANNs. As typically done in HMM/ANN systems for ASR, the ANNs are trained independently of the HMM (the HMMs themselves are not trained).

#### $X_n$ only (Baseline)

The baseline HMM/ANN system uses the scaled likelihood from (2.47), using the observations $x_{n-c:n+c}$. It has a window size of nine frames (four frames to the past and four frames to the future, plus the current frame). It is illustrated in Figure 4.7 (a).

[2]In collaboration with Mathew Magimai-Doss

### $X_n, A_n$ (no assumptions)

In incorporating $A_n$ in the HMM/ANN context with no additional assumptions, we treat it as a normal feature by appending it to the standard feature $X_n$. That is, like with $X_n$, we use the contextual auxiliary features (a window of nine frames) and by including them in the same input layer as the standard features, the hidden layer models correlation between $A_n$ and $X_n$ as well as the correlation between $A_n$ and $Q_n$ (similar to what is done in the BN presented in Figure 4.5 (b)). Equation (2.47) is expanded with the observations $x_{n-c:n+c}$ and $a_{n-c:n+c}$ as shown in (4.15).

$$\begin{aligned} &p(X_{n-c:n+c} = x_{n-c:n+c}, A_{n-c:n+c} = a_{n-c:n+c}|Q_n) \\ &\quad = \frac{p(X_{n-c:n+c} = x_{n-c:n+c}, A_{n-c:n+c} = a_{n-c:n+c}) \cdot P(Q_n|X_{n-c:n+c} = x_{n-c:n+c}, A_{n-c:n+c} = a_{n-c:n+c})}{P(Q_n)} \\ &\quad \propto \frac{P(Q_n|X_{n-c:n+c} = x_{n-c:n+c}, A_{n-c:n+c} = a_{n-c:n+c})}{P(Q_n)}. \end{aligned} \tag{4.15}$$

The use of conditional Gaussians in DBNs for modeling $X_n, A_n$ with no assumptions lets $A_n$ have a linear impact on the distribution. However, here incorporating $X_n, A_n$ with no assumptions brings a non-linear dependency between $X_n$ and $A_n$. Therefore, the relation between the two can potentially be modeled better in the ANNs.. It is illustrated in Figure 4.7 (b).

### $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$

In treating $A_n$ as a normal feature in the $X_n, A_n$ above, the hidden layer carried information about $A_n$ to the output layer representing $Q_n$. Hence, that ANN had a dependency between $A_n$ and $Q_n$. Breaking this dependency is a subject of research. The approach taken here is that to break this dependency, there should be no input from $A_n$ into the hidden layer. This is achieved by having a separate ANN for each value of a discretized $A_n$. Therefore, the modeling of $X_n$ is done differently depending on which discrete value $1, \ldots, l, \ldots, L$ that $A_n$ has; however, the value of $A_n$ does not directly affect $Q_n$ (only indirectly via $X_n$). Likewise, in a DBN, if $A_n$ had been discretized (which is not the case in my DBN—though the systems in Section 4.3 used a discretized $A_n$ with discretized $X_n$), the effect would have been to have a different set of Gaussians for each discrete value of $A_n$. As in the other HMM/ANN setups, each ANN has a window size of nine frames. It is illustrated in Figure 4.7 (c). For a given value $A_n = l$, the scaled likelihood is computed using only the one ANN associated with $A_n = l$ as:

$$\begin{aligned} &p(X_{n-c:n+c} = x_{n-c:n+c}, A_n = l|Q_n) \\ &\quad = \frac{p(X_{n-c:n+c} = x_{n-c:n+c}, A_n = l) \cdot P(Q_n|X_{n-c:n+c} = x_{n-c:n+c}, A_n = l)}{P(Q_n)} \\ &\quad \propto \frac{P(Q_n|X_{n-c:n+c} = x_{n-c:n+c}, A_n = l)}{P(Q_n)}. \end{aligned} \tag{4.16}$$

Note that the prior of the states $P(Q_n)$ is over all the training data, independent of the distribution of $A_n$; future work would involve having a different prior for each ANN for the discrete value $l$ of $A_n$: $P(Q_n|A_n = l)$. There are $L$ ANNs for each of the $L$ discrete values of $A_n$; each ANN for $A_n = l$ is trained using the features $X_n$ whose corresponding auxiliary value is $A_n = l$. This results in $L$ ANNs whose structure is just like the BASELINE structure described above except that the window size for the input layer is smaller and that the number of parameters in each of the $L$ ANNs is approximately $\frac{1}{L}$ that of the number of parameters in the regular BASELINE so as to keep the total number of parameters between the two types of system approximately the same. Each of the $L$ ANNs is, thus, tailored to a specific type of speech, based on the value of $A_n$; but as each receives, on average, only $\frac{1}{L}$ of the training data that the BASELINE has, there is the risk of having poorly trained ANNs, particularly for large $L$.

$X_n \perp\!\!\!\perp A_n \,|\, Q_n$

Conversely, to have $A_n$ affect $Q_n$ but not $X_n$, the continuous $A_n$ needs to have its layers separated from $X_n$'s layers. The only point where information through which $A_n$ and $X_n$ can share information would be their would be the output layer, just like with the BN in Figure 4.5 (d), where $X_n$ and $A_n$ are only connected via $Q_n$. Like with $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$ above, this is a subject of research; the approach taken here to this research question is to have an input and a hidden layer for $X_n$ but only an input layer for $A_n$; furthermore, no context is taken for $A_n$ in the input layer. By connecting $X_n$'s hidden layer and $A_n$'s input layer both to the output layer, $X_n$ is modeled in a non-linear manner while $A_n$ is modeled in a linear manner; that is, the hidden layer can do non-linear modeling of $X_n$ while having $A_n$'s input layer directly connected to the output layer, the (weighted) values of $A_n$ and used directly in the output layer's softmax function. It is modeled as

$$
\begin{aligned}
& p(X_{n-c:n+c} = x_{n-c:n+c}, A_n = a_n | Q_n) \\
&= \frac{p(X_{n-c:n+c} = x_{n-c:n+c}, A_n = a_n) \cdot P(Q_n | X_{n-c:n+c} = x_{n-c:n+c}, A_n = a_n)}{P(Q_n)} \\
&\propto \frac{P(Q_n | X_{n-c:n+c} = x_{n-c:n+c}, A_n = a_n)}{P(Q_n)}.
\end{aligned} \tag{4.17}
$$

As in the other HMM/ANN setups, there is a window size of nine frames for $X_n$; however, no time frame is used for $A_n$ as this was an initial investigation into this type of system setup. The modeling of (4.17) resembles that of (4.16) except that $A_n$ is continuous and that the value of $A_n$ directly changes $P(Q_n | X_{n-c:n+c} = x_{n-c:n+c}, A_n = a_n)$ in a continuous fashion. Investigating a linear dependency for $A_n$ was motivated by the use of $A_n$'s linear dependency upon $X_n$ in the conditional Gaussians in the DBN framework (though here with the ANNs, the linear dependency is not between $X_n$ and $A_n$ but between $Q_n$ and $A_n$). Modeling $A_n$ in a such linear fashion is illustrated in Figure 4.7 (d). Extensions of this model (not presented in this thesis) include having a separate hidden layer, and, hence, non-linear modeling for $A_n$, as well as having a time window for $A_n$. For example, an alterate approach to modeling $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$, which is not pursued in this thesis, would be to use the multi-stream approach, were a separate ANN is modeled for both $X_n$ and for $A_n$ (Hagen, 2001).

### 4.4.3 Hiding Auxiliary Variables

The auxiliary information considered may not always be available or may not always be reliable due to noise. In such a case, we do not want to use it in probabilistic inference as it would corrupt the modeling. In such cases, we hide it and, instead, use its inferred value, which is computed using both its prior distribution and the observations of the other variables. In the case of a continuous auxiliary variable, this is accomplished through integration; in the case of a discrete auxiliary variable with values $1, \ldots, L$, this is accomplished through a sum:

$$p(X_n | Q_n) = \int_{-\infty}^{\infty} p(X_n, A_n | Q_n) \, \mathrm{d}A_n, A_n \text{ continuous} \tag{4.18}$$

$$p(X_n | Q_n) = \sum_{a_n=1}^{L} p(X_n, A_n = a_n | Q_n), A_n \text{ discrete} \tag{4.19}$$

In the case of using mixture distributions, we would have:

$$p(X_n|Q_n) = \sum_{j=1}^{J} \int_{-\infty}^{\infty} P(J_n = j|Q_n) \cdot p(X_n, A_n|Q_n, J_n = j)\, \mathrm{d}A_n, A_n \text{ continuous} \quad (4.20)$$

$$p(X_n|Q_n) = \sum_{j=1}^{J} \sum_{a_n=1}^{L} P(J_n = j|Q_n) \cdot p(X_n, A_n = a_n|Q_n, J_n = j), A_n \text{ discrete} \quad (4.21)$$

This is straightforward to do in the context of DBNs, where an integration over a continuous $A_n$ involves merely removing its dimension from within a Gaussian and a sum over a discrete $A_n$ is simply defined if there are no conditional Gaussians in the distribution. This is very similar to the work done in missing feature theory (Morris *et al.*, 1998), where certain elements of $X_n$ are integrated out when they are suspected of being corrupted by noise. However, in missing feature theory where $X_n$ contains energy values, the bounds of the integral are restricted to $[0, a_n]$, where $a_n$ is the corrupted value and, hence, the maximum possible uncorrupted value (since the "true" clean signal is $a_n$ minus the unknown amount of energy in the added noise) and where 0 is the minimum possible uncorrupted value (representing silence) of the missing feature.

Note that if $X_n$ has a conditional Gaussian dependent upon a hidden $A_n$, there would be additional computation involved (all simple multiplications, divisions, and additions, as to be explained in Section 4.4.5). This is due to the fact that all the elements are correlated with each other via this hidden $A_n$. So, an updated distribution of $A_n$ needs to be computed given the observations $x_n$, and this updated distribution of $A_n$ is needed in the computation of the likelihood of $x_n$.

Regarding the case with ANNs, the hiding can easily be done in the case of discrete $A_n$; as there is an ANN for each value of $A_n$, we do a weighted sum of the posteriors from each ANN (where the weights are the prior distribution of $A_n$ in the training data). In the case of continuous $A_n$, it is not straightforward how to hide the $A_n$, which is a part of the input layer. Hence, for our HMM/ANN systems, I only present systems with hidden $A_n$ where $A_n$ has been discretized.

### 4.4.4 Parameter Estimation

#### Gaussians

In machine learning, the Gaussian distribution is the basis for modeling the distributions of a wide variety of data (where it is used in mixture distributions, as discussed later on page 56 ff). It consists of two parameters: the mean vector $\mu$ and the covariance matrix $\Sigma$, which are referred to as the first moment and the second moment, respectively. If data $X$ is distributed in such a manner, we note it as:

$$X \sim \mathcal{N}(\mu_X, \Sigma_X) \quad (4.22)$$

If $\mu_X$ and $\Sigma_X$ are not known but if we do have data $X = \{x_1, \ldots, x_N\}$ drawn from this distribution, we can estimate their values (these were already introduced in Section 2.1.2 on page 11 in the context of training HMMs):

$$\mu_X = E(X) \approx \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \quad (4.23)$$

$$\Sigma_X = E((X - E(X))^2) \approx S_X = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T, \quad (4.24)$$

where (4.24) is a biased estimate of $\Sigma_X$. Alternatively, we can have an unbiased estimator of $\Sigma_X$:

$$S_X = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T, \tag{4.25}$$

where (4.25) divides by $(N-1)$ instead of by $N$ and takes into account that $\bar{x}$ is only an estimate (Papoulis, 1991). The same result for $S_X$ in (4.24) can be obtained by:

$$\begin{aligned} S_X &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T \\ &= \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T - \frac{1}{N}\sum_{i=1}^{N} 2x_i\bar{x}^T + \frac{1}{N}\sum_{i=1}^{N}\bar{x}\bar{x}^T \\ &= \frac{1}{N}\sum_{i=1}^{N} x_i x_i^T - \bar{x}\bar{x}^T. \end{aligned} \tag{4.26}$$

The advantage of (4.26) over (4.24) is that both $\mu_X$ and $\Sigma_X$ can be estimated in the same pass over the data, instead of two.

### Conditional Gaussians

The conditional Gaussian (Lauritzen and Wermuth, 1989) is a less common distribution used for modeling the dependency between data $X$ and data $A$. If we defined $X$ as being dependent upon $A$, then we have a distribution whose first moment is no longer the mean of $X$ but, rather, a function $u_X$. In my work, this is represented by the linear function $u_X = \mu_X + B_X A$, using regressions in $B_X$ upon $A$. Its second moment is still the variance $\Sigma_X$ (albeit, estimated differently, as explained below):

$$\begin{aligned} X &\sim \mathcal{N}(u_X, \Sigma_X) \\ &\sim \mathcal{N}(\mu_X + B_X A, \Sigma_X). \end{aligned} \tag{4.27}$$

Even though not directly dependent upon $A$, $\Sigma_X$ is indirectly conditioned upon it as when we estimate $\Sigma_X$ for this conditional Gaussian, the value of $A$ accounts for some the variance; hence, the variance will be lower, as illustrated in Figure 2.5 on page 22. Furthermore, it is only when $X$ and $A$ are factored into two distributions that the variance appears to not be dependent upon $A$; if a multivariate distribution were formed from (4.22) and (4.27) for the vector $(X, A)$ such that we are modeling

$$(X, A) \sim \mathcal{N}(\mu_{(X,A)}, \Sigma_{(X,A)}), \tag{4.28}$$

then the variance for $X$ within the $\Sigma_{(X,A)}$ would be $\Sigma_X + B_X\,\Sigma_A\,B_X^T$, where $\Sigma_X$ and $B_X$ are taken from (4.27) and $\Sigma_A$ is taken from $A$'s distribution (Lauritzen and Jensen, 2001, Section 4.4):

$$A \sim \mathcal{N}(\mu_A, \Sigma_A). \tag{4.29}$$

Giving the second moment a direct dependence upon $A$ in the conditional Gaussian for $X$ is a subject of research.

Hence, whereas the Gaussian (4.22) has two parameters to estimate, $\mu$ and $\Sigma$, the conditional Gaussian (4.27) has three parameters: $\mu$, $B$, and $\Sigma$. These parameters are learned in a two-stage procedure. First, we need to estimate the joint mean and joint covariance of $X$ and $A$. Let $W$ be the combination of data $X$ and $A$ (that is, for each pair of data vectors $x$ and $a$, $w$ is the data vector which is the concatenation of the two). Then we estimate $\mu_W$ and $\Sigma_W$:

$$\mu_W \approx \bar{w} = \frac{1}{N}\sum_{i=1}^{N} w_i \tag{4.30}$$

$$\Sigma_W \approx S_W = \frac{1}{N}\sum_{i=1}^{N}(w_i - \bar{w})(w_i - \bar{w})^T = \frac{1}{N}\sum_{i=1}^{N} w_i w_i^T - \bar{w}\bar{w}^T \tag{4.31}$$

(4.30) and (4.31) are the maximum likelihood estimates of the mean and variance, respectively, for the joint distribution of $W = (X, A)$. If we are then interested in having the individual distributions for $X$ and for $A$ (which, in the context of DBNs, we want — see Section 2.4.1), we can factor the multivariate Gaussian distribution using the parameters (4.30) and (4.31). To do so, let us partition $\bar{w}$ according to the portion obtained from $X$ and that obtained from $A$:

$$\bar{w} = \begin{bmatrix} \bar{w}\{X\} \\ \bar{w}\{A\} \end{bmatrix} \tag{4.32}$$

Let us also partition $S_W$, according to that related only to $X$, that related only to $A$ and those related jointly to $X$ and $A$, as follows

$$S_W = \begin{bmatrix} S_W\{X^2\} & S_W\{XA\} \\ S_W\{AX\} & S_W\{A^2\} \end{bmatrix} \tag{4.33}$$

and let $[S_W\{A^2\}]^-$ be the pseudo-inverse of $S_W\{A^2\}$. Then the parameters $\mu_X$, $B_X$, and $\Sigma_X$ for the conditional Gaussian are estimated as (see Lauritzen and Jensen (2001, Section 4.5):

$$B_X \approx \hat{B}_X = S_W\{XA\}[S_W\{A^2\}]^- \tag{4.34}$$

$$\mu_X \approx \bar{x}_X = \bar{w}\{X\} - \hat{B}_X \bar{w}\{A\} \tag{4.35}$$

$$\Sigma_X \approx S_X = S_X - \hat{B}_X S_W\{AX\}. \tag{4.36}$$

The parameters for $A$'s distribution remain as $\bar{w}\{A\}$ and $S_W\{A^2\}$ for its mean and variance, respectively (Lauritzen and Jensen, 2001, Section 4.3); they are not modified as they are not conditioned upon anything.

Note that while only the first moment, $u_X$, is conditioned upon $A$, the estimates for both the first and second moments, $u_X$ and $\Sigma_X$, respectively, are dependent upon $A$. Specifically, $S_W\{XA\}$ and $[S_W\{A^2\}]^-$ are used for calculating all three of the estimates; additionally, $\bar{w}\{A\}$ is used in the estimate of $\mu_X$. In summary, we obtain the conditional distribution of $X$ upon $A$ by first computing the maximum likelihood estimates of the multivariate Gaussian for $(X, A)$ and then factor this into the conditional Gaussian of $X$ upon $A$ and into the Gaussian of $A$.

### Mixture Distributions

For many problems, the type of distribution is not known. While it is simple to estimate the mean and covariance for a single Gaussian distribution, the resulting distribution may not truly represent the distribution. Now, certain mixture models, if set up correctly, can be used to model any type of distribution (Bishop, 1995). Therefore, one type of mixture model, the Gaussian mixture model (GMM), is sometimes used so as to better represent the true underlying distribution of the data. So, for $J$ Gaussians with respective means $\mu_{X_1}, \ldots, \mu_{X_J}$; covariances $\Sigma_{X_1}, \ldots, \Sigma_{X_J}$; and weights $w_{X_1}, \ldots, w_{X_J}$, where $\sum_{j=1}^{J} w_{X_j} = 1$ and $w_{X_j} \geq 0$ (the notation $w_{X_j}$, $\mu_{X_j}$, $\Sigma_{X_j}$, etc., refers to the parameters of $X$ when it is distributed according to mixture component $j$), GMMs are:

$$\sum_{j=1}^{J} w_{X_j} \mathcal{N}(\mu_{X_j}, \Sigma_{X_j}) \tag{4.37}$$

Similarly, we can have conditional GMMs:

$$\sum_{j=1}^{J} w_{X_j} \mathcal{N}(u_{X_j}, \Sigma_{X_j}), \text{where } u_{X_j} = \mu_{X_j} + B_j\, a, \tag{4.38}$$

with a different regression matrix $B_j$ as well for each mixture component. The parameters of (conditional) GMMs can be estimated by algorithms such as $K$-means clustering (Rabiner and Juang, 1993, Section 3.4.4), where there is a cluster for each of the $J$ mixtures. In $K$-means clustering, the resulting vectors of $X$ for each cluster will be used to estimate that mixture's means and variances; for conditional GMMs, the clustering will be done by the concatenated vector $(X, A)$ and the resulting vectors of $(X, A)$ for each cluster will be used to estimate the mixture's multivariate means and covariances, which will then be factored as explained above. The weights of the (conditional) GMM will be the distribution of the vectors amongst the clusters.

### 4.4.5 Likelihoods

In this section I make the assumption that the elements of the covariance matrix for each mixture component of $X$ are zero off of its diagonal. In other words, the dimensions of the data $X$ are uncorrelated with each other, given the state, mixture component, and auxiliary variables. Therefore, the covariance matrix $\Sigma_{X_j}$ can be represented by its diagonal elements, contained in the variance vector $\sigma^2_{X_j}$. Put more simply, each dimension $p$ of $X$ in mixture component $j$ has its own one-dimensional Gaussian.

#### Gaussian Mixture Models

**Observed** $X$ Given a data sample $x$ from $X$, denote the $P$ elements in the vector as $x[1], \ldots, x[P]$. Likewise, the elements of the mean vector and of the variance vector of $\mathcal{N}(\mu_{X_j}, \sigma^2_{X_j})$, the Gaussian of mixture component $j$ of $X$, are denoted, respectively, as $\mu_{X_j}[1], \ldots, \mu_{X_j}[P]$ and $\sigma^2_{X_j}[1], \ldots, \sigma^2_{X_j}[P]$. As each dimension is independent of each other, the likelihood is computed independently for each dimension:

$$\mathcal{L}(X[p] = x[p]|j) = \frac{\exp\left[\frac{-0.5(x[p]-\mu_{X_j[p]})^2}{\sigma^2_{X_j[p]}}\right]}{\sqrt{2\pi\sigma^2_{X_j[p]}}}. \tag{4.39}$$

The likelihoods for each dimension are then multiplied together:

$$\mathcal{L}(X = x|j) = \prod_{p=1}^{P} \mathcal{L}(X[p] = x[p]|j) \tag{4.40}$$

If there are GMMs, then we use a weighted sum of the likelihoods using (4.40) and $w_j$ for each mixture component $j$:

$$\mathcal{L}(X = x) = \sum_{j=1}^{J} w_j \mathcal{L}(X = x|j). \tag{4.41}$$

**Partially-observed** $X$ If any dimensions of $x$ are missing, those dimensions need to be integrated out. Here we take advantage of the property of the Gaussian that, since it is a probability distribution, the integral over all of its domain is 1:

$$\int_{-\infty}^{\infty} \mathcal{L}(X[p]|j)\,\mathrm{d}X[p] = 1, \tag{4.42}$$

hence, expanding (4.41), using (4.40), as:

$$\begin{aligned} \mathcal{L}(x) &= \sum_{j=1}^{J} w_j \prod_{p=1}^{P} f(x[p], j), \\ \text{where } f(x[p], j) &= \begin{cases} 1 & , x[p] \text{ missing} \\ \mathcal{L}(X[p] = x[p]|j) & , x[p] \text{ observed} \end{cases} \end{aligned} \tag{4.43}$$

Hence, in the product over the $P$ dimensions, a missing dimension $p$ will have no impact on computing the likelihood $\mathcal{L}(X = x|j)$ for each mixture component $j$, as its contribution to the product is 1, that is, the integral over all possible values for dimension $p$.

Attempting to use the expected value (i.e., the mean) of the missing dimensions instead of integrating over all values would not be as appropriate, as we want to take into account all of the possible values for the missing dimension. Furthermore, if we had used the expected value instead, those dimensions with a low variance would give a high likelihood ($> 1$) at the mean while those with a high variance would give a low likelihood ($< 1$) at the mean; hence, different dimensions would weight the overall likelihood differently, if they were missing, merely because of their difference in their respective variances — this could potentially lead to undesired effects.

### Conditional Gaussian Mixture Models

**Observed** $A$ Given data samples $x$ and $a$ for mixture component $j$ of a conditional GMM, we compute the likelihood as in (4.41) except that each mean $\mu_{X_j}$ is offset, or shifted, according to the value $a$ and the regression weights $B_{X_j}$:

$$\hat{\mu}_{X_j} = \mu_{X_j} + B_{X_j}\, a. \tag{4.44}$$

$\hat{\mu}_{X_j}$ (referred to as $u_X$ earlier) is then substituted for $\mu_{X_j}$ in (4.39) to compute the conditional likelihood for a single dimension, single mixture component, and all mixture components of a conditional Gaussian, respectively:

$$\mathcal{L}(X[p] = x[p]|A = a, j) = \frac{\exp\left[\frac{-0.5(x[p]-\hat{\mu}_{X_j}[p])^2}{\sigma^2_{X_j}[p]}\right]}{\sqrt{2\pi\sigma^2_{X_j}[p]}} \tag{4.45}$$

$$\mathcal{L}(X = x|A = a, j) = \prod_{p=1}^{P} \mathcal{L}(X[p] = x[p]|A = a, j) \tag{4.46}$$

$$\mathcal{L}(X = x|A = a) = \sum_{j=1}^{J} w_j \mathcal{L}(X = x|A = a, j). \tag{4.47}$$

These are analogous to (4.39), (4.40), and (4.41), respectively. Thus, computing a likelihood with a conditional Gaussian with given data sample $x$ and $a$ is only slightly more complicated per mixture component than having a regular Gaussian: there is an additional (matrix) multiplication and (vector) addition to compute each $\hat{\mu}_{X_j}[p]$.

If any dimensions of $x$ are missing in the conditional Gaussian, those dimensions can easily be integrated out. Regardless of whether $a$ is observed or missing, we also have, similarly to (4.42):

$$\int_{-\infty}^{\infty} \mathcal{L}(X[p]|A, j)\,\mathrm{d}X[p] = 1, \tag{4.48}$$

hence, expanding (4.47), in the case of observed $a$, as:

$$\begin{aligned} \mathcal{L}(x|A = a) &= \sum_{j=1}^{J} w_j \prod_{p=1}^{P} f(x[p], j, A = a), \qquad (4.49)\\ \text{where } f(x[p], A = a, j) &= \begin{cases} 1 & , x[p] \text{ missing} \\ \mathcal{L}(X[p] = x[p]|A = a, j) & , x[p] \text{ observed} \end{cases} \end{aligned}$$

**Non-observed** $A$ However, suppose that we only have data sample $x$ for a conditional GMM and that the sample $a$ is missing. In other words, we do not know what the distribution is to be conditioned upon. The only items we have for $A$ are its mean $\mu_A$ and variance $\sigma_A^2$, that is, its prior distribution. In Lauritzen and Jensen (2001), if we have the observations for $X[1], \ldots, X[P]$ but not for $A$ we need to "push" (Lauritzen and Jensen, 2001, Section 6.2) the distributions $p(X[1]|A), \ldots, p(X[P]|A)$ up so to as to combine them with the distribution $p(A)$ (or $p(A|Q)$, as appropriate) because the observation $x[p]$ can not be incorporated into the distribution $p(X[p]|A)$ if the continuous conditioning variable $A$ is hidden. With the push, we can form the joint, non-conditional distribution $p(X[1], \ldots, X[P], A)$; after marginalizing $A$ out of this distribution, we then incorporate the observations upon the resulting multivariate distribution, as it has no conditioning variable.

Since we are using a diagonal covariance matrices for $X$ in each mixture component of each state, we do not need to form the whole distribution $p(X[1], \ldots, X[P], A)$ but, rather, can compute the likelihood in steps, one dimension at a time (e.g., for $p(X[p], A)$). Let us, then, consider the likelihood for a given dimension $p$ of $x$ in a given mixture component $j$. The following discussion is based upon an analysis of the DBN inference algorithm of Lauritzen and Jensen (2001). First before processing any dimensions of $X_j$, we initialize the distribution $\hat{A}$:

$$\hat{A} \sim \mathcal{N}\left(\mu_A, \sigma_A^2\right). \tag{4.50}$$

Consider that we have an arbitrary ordering $\mathcal{P}$ of the dimensions. For the first dimension $p$ in $\mathcal{P}$ of $X_n$, in incorporating the observation for $X_j[p]$ in $p(X_j[p]|A)$, we can not do so with the continuous conditioning variable $A$. So, we "push" this distribution onto that of $p(A)$ using the combination operation of Lauritzen and Jensen (2001, Section 4.4) (as briefly introduced in Section 3.3.1 on page 35), obtaining the multivariate (non-conditional) Gaussian:

$$\begin{aligned} \begin{bmatrix} X_j[p] \\ \hat{A} \end{bmatrix}_{|\mathcal{P},j,x} &\sim \mathcal{N}\left( \begin{bmatrix} \mu_{X_j[p]} + B_{X_j[p]}\,\hat{\mu}_A \\ \hat{\mu}_A \end{bmatrix}, \begin{bmatrix} \sigma^2_{X_j[p]} + B_{X_j[p]}\,\hat{\sigma}^2_A\,B^T_{X_j[p]} & \hat{\sigma}^2_A\,B^T_{X_j[p]} \\ B_{X_j[p]}\,\hat{\sigma}^2_A & \hat{\sigma}^2_A \end{bmatrix} \right) \\ &\sim \mathcal{N}\left( \begin{bmatrix} \hat{\mu}_{X_j[p]} \\ \hat{\mu}_A \end{bmatrix}, \begin{bmatrix} \hat{\sigma}^2_{X_j[p]} & \hat{\sigma}_{X_j[p],A} \\ \hat{\sigma}_{A,X_j[p]} & \hat{\sigma}^2_A \end{bmatrix} \right), \end{aligned} \tag{4.51}$$

using the respective abbreviations in the second line of (4.51) in further discussion and with this distribution being determined by $\mathcal{P}$ (indicated by $_{|j,\mathcal{P}}$).

This is the effect of "push"ing the distribution of $X_j[p]$ upon that of $A$. We then use this distribution to calculate the likelihood of only $X[p]$ ($A$ is left missing/hidden) using the standard formula:

$$\mathcal{L}(X[p] = x[p]|\hat{A}, \mathcal{P}, j, x) = \frac{\exp\left[\frac{-0.5(x[p]-\hat{\mu}_{X_j[p]})^2}{\hat{\sigma}^2_{X_j[p]}}\right]}{\sqrt{2\pi\hat{\sigma}^2_{X_j[p]}}}. \tag{4.52}$$

After the likelihood for $x[p]$ is calculated from (4.51) using (4.52), we are then left with the distribution only for $\hat{A}$, which, when is updated to take $x[p]$ account (Lauritzen and Jensen, 2001, Section 7):

$$\begin{aligned}
\hat{A}_{|j,\mathcal{P},p,x} &\sim \mathcal{N}\left(\hat{\mu}_A + \hat{\sigma}_{A,X_j[p]}\frac{(x[p]-\hat{\mu}_{X_j[p]})}{\hat{\sigma}^2_{X_j[p]}}, \hat{\sigma}^2_A - \hat{\sigma}_{A,X_j[p]}\frac{\hat{\sigma}_{X_j[p],A}}{\hat{\sigma}^2_{X_j[p]}}\right) \\
&\sim \mathcal{N}\left(\hat{\mu}^{\dagger}_A, \hat{\sigma}^{2,\dagger}_A\right). \qquad (4.53)
\end{aligned}$$

We then continue iteratively through (4.51), (4.52), and (4.53), processing all of the dimensions of $X = x$ in turn (using (4.53) instead of (4.50) for each subsequent iteration).

The likelihood of the mixture component, using (4.52) is then

$$\mathcal{L}(X = x|\hat{A}, j) = \prod_{p=1}^{P} \mathcal{L}(X[p] = x[p]|\hat{A}, \mathcal{P}, j, x), \qquad (4.54)$$

and the likelihood of all the mixture components is

$$\mathcal{L}(X = x|\hat{A}) = \sum_{j=1}^{J} w_j \mathcal{L}(X = x|\hat{A}, j). \qquad (4.55)$$

Hence, computing a likelihood with a conditional Gaussian with $x$ given but $a$ hidden involves even more computations due to inferring $\hat{A}$'s updated distribution given dimensions $1, \ldots, P-1$ of the observed $x$ (inferring the distribution after dimension $P$ is not necessary for the likelihood computation).

If any elements of $X$ are hidden in addition to $A$ being hidden, then (4.55) is expanded.

$$\begin{aligned}
\mathcal{L}(X = x|\hat{A}) &= \sum_{j=1}^{J} w_j \prod_{p=1}^{P} f(x[p], \hat{A}, \mathcal{P}, j), \qquad (4.56) \\
\text{where } f(x[p], \hat{A}, \mathcal{P}, j) &= \begin{cases} 1 & , x[p] \text{ missing} \\ \mathcal{L}(X[p] = x[p]|\hat{A}, \mathcal{P}, j) & , x[p] \text{ observed} \end{cases}
\end{aligned}$$

Note that if $x[p]$ is missing, we do not do the combinations in (4.51) and (4.53) as these are intended to account for observations.

To illustrate this process, consider a two-dimensional $X$ and processing the dimensions in the order $\mathcal{P} = \{1, 2\}$ (it can be shown that the final result in (4.66) is the same if $\mathcal{P} = \{2, 1\}$). Consider the following prior distributions for a mixture $j$:

$$\begin{aligned}
X[1]_{|A,j} &\sim \mathcal{N}(-2 + 0.2 \cdot A, 2) \qquad (4.57) \\
X[2]_{|A,j} &\sim \mathcal{N}(0 - 0.1 \cdot A, 1) \qquad (4.58) \\
A &\sim \mathcal{N}(3, 5). \qquad (4.59)
\end{aligned}$$

Let $x[1] = 1.5$ and $x[2] = -1$. Proceeding $\mathcal{P} = 1, 2$, we have, with respect to (4.50), (4.51), (4.52), and

(4.53), above for dimension 1:

$$\hat{A} \sim \mathcal{N}(3,5) \tag{4.60}$$

$$\begin{aligned}
\left[\begin{array}{c} X[1] \\ A \end{array}\right]_{|\mathcal{P},j} &\sim \mathcal{N}\left(\left[\begin{array}{l} -2+0.2\cdot 3 \\ 3 \end{array}\right], \left[\begin{array}{lr} 2+0.2\cdot 5\cdot 0.2 & 5\cdot 0.2 \\ 0.2\cdot 5 & 5 \end{array}\right]\right) \\
&\sim \mathcal{N}\left(\left[\begin{array}{l} -1.4 \\ 3 \end{array}\right], \left[\begin{array}{ll} 2.2 & 1 \\ 1 & 5 \end{array}\right]\right)
\end{aligned} \tag{4.61}$$

$$\begin{aligned}
\mathcal{L}(X[1]=1.5|\hat{A},j) &= \frac{\exp\left[\frac{-0.5(1.5+1.4)^2}{2.2}\right]}{\sqrt{2\pi 2.2}} \\
&= 0.0397744
\end{aligned} \tag{4.62}$$

$$\begin{aligned}
A_{|j,\mathcal{P},p,x} &\sim \mathcal{N}\left(3+1\cdot\frac{(1.5+1.4)}{2.2}, 5-1\cdot\frac{1}{2.2}\right) \\
&\sim \mathcal{N}(4.31818, 4.54545)
\end{aligned} \tag{4.63}$$

Continuing with dimension 2, we have, with respect to (4.51), and (4.52):

$$\begin{aligned}
\left[\begin{array}{c} X[2] \\ \hat{A} \end{array}\right]_{|\mathcal{P},j,x} &\sim \mathcal{N}\left(\left[\begin{array}{l} 0-0.1\cdot 4.31818 \\ 4.31818 \end{array}\right], \left[\begin{array}{lr} 1+(-0.1)\cdot 4.54545\cdot(-0.1) & 4.54545\cdot(-0.1) \\ (-0.1)\cdot 4.54545 & 4.54545 \end{array}\right]\right) \\
&\sim \mathcal{N}\left(\left[\begin{array}{l} -0.431818 \\ 4.31818 \end{array}\right], \left[\begin{array}{rr} 1.0454545 & -0.454545 \\ -0.454545 & 4.54545 \end{array}\right]\right)
\end{aligned} \tag{4.64}$$

$$\begin{aligned}
\mathcal{L}(X[2]=-1|\hat{A},j) &= \frac{\exp\left[\frac{-0.5(-1+0.431818)^2}{1.0454545}\right]}{\sqrt{2\pi 1.0454545}} \\
&= 0.334352
\end{aligned} \tag{4.65}$$

Using (4.54), the likelihood of mixture $j$ is:

$$\mathcal{L}(X=\left[\begin{array}{c} 1.5 \\ -1 \end{array}\right]|\hat{A},j) = 0.0397744\cdot 0.334352 = .0132987 \tag{4.66}$$

## 4.5 Auxiliary chain information (factorial HMMs)

An alternative method of modeling discrete auxiliary information involves having two state spaces represented by two (conditionally) independent Markov chains. This can be represented as a factorial HMM (Ghahramani, 1997). The two chains are represented by $Q_{1:N} = \{Q_1, \ldots, Q_n, \ldots, Q_N\}$ and $L_{1:N} = \{L_1, \ldots, L_n, \ldots, L_N\}$ with joint observations $X_{1:N}$, as represented by Figure 4.8. The $Q_n$ chain represents, as elsewhere in this thesis, the phonemes (the sounds in spoken language) while the $L_n$ chain represents the graphemes (the letters, punctuation, etc. used in written language).

In training such a system, the labeling of the speech with respect to $Q_{1:N}$ and the labeling of the speech with respect to $L_{1:N}$ are used in the respective control layers. In recognition, such a system can be used in different ways. This has to do with the fact that, in recognition, we want to refer to the labels with respect to only one of the chains; this is because in recognition, we typically only need either the sequence of graphemes or the sequence of phonemes (but not both) to determine what was spoken. This then gives us the option to break the chain which does not have any associated labels and to leave it as hidden and with no restrictions on its sequence of values through time. For example, in breaking the $L_n$ chain, we might recognize the sequence of phonemes /k/-/æ/-/t/ (in the $Q_n$ chain) for a given utterance of "cat" while, in breaking the $Q_n$ chain, we might recognize the sequence of graphemes 'c'-'a'-'t' (in the $L_n$ chain) for the same utterance; in recognizing the

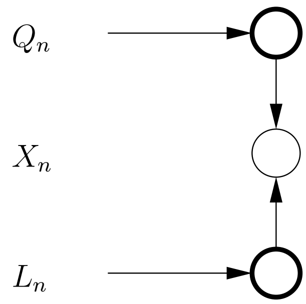


Figure 4.8: Factorial HMM, with chains for $Q_n$ and $L_n$ and with observations $x_n$. This is used for training.

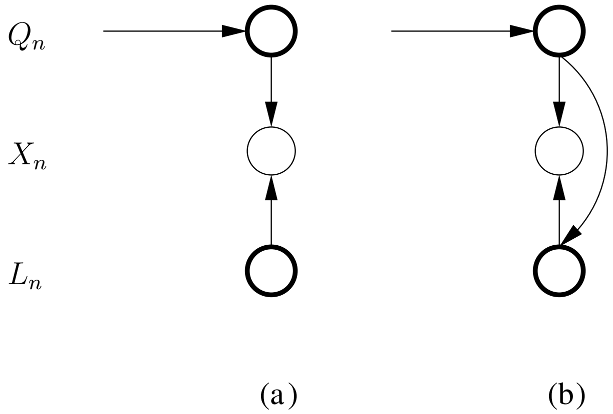


Figure 4.9: Two possibilities for breaking one of the chains of the DBN in Figure 4.8. (these examples are for breaking the chain of $L_n$). (a) one chain ($L_n$) is made independent of all variables and (b) one chain ($L_n$) is made dependent only upon the corresponding variable in the other chains ($Q_n$).

series of phonemes (or graphemes), we leave the series of graphemes (or phonemes) hidden and marginalize out the graphemes (or phonemes). One option is to just break the connection in the case of having labels for $Q_n$, between $L_{n-1}$ and $L_n$, leaving $L_n$ independent of any other variable, as shown in Figure 4.9 (a). Another option, and that used in the experiments, is that, in addition to breaking the time-dependency, to add a dependency between the two chains; this is illustrated for the case of breaking the $L_n$ chain in Figure 4.9 (b). This is the same as used in having Gaussian mixture models (GMMs) for $X_n$ except that, in this case, the "mixture" component variable can have a meaning of grapheme (in the case of 4.9 (b)) or of phoneme.

## 4.6 Conclusion

In this chapter I have argued for the need for auxiliary information in the acoustic models so as to better handle the wide variation in the data that the models will encounter. I have discussed that this auxiliary information is typically higher-level information that is independent of the discrete model state. Furthermore, this auxiliary information may be such that it is not available or not reliable in recognition; in such case, the models can still be used, in which case the inferred distributions of the auxiliary variable are used.

While some of the aspects of auxiliary information have been used in ASR previously, I have not only given a general framework encompassing them but have also introduced additional methods for handling it, including the aforementioned need to hide it at times as well; for handling it in both HMM/ANN and DBN speech recognition; and for incorporating it in an "auxiliary chain." These methods for handling auxiliary information will be incorporated using the actual auxiliary information in Chapter 5 with the experimental results presented in Chapter 6.

# Chapter 5

# DBNs with auxiliary information for ASR

In Chapter 2 I discussed in general how time series modeling is done in the context of ASR. Time series modeling makes statistical assumptions that make the modeling feasible; without these assumptions, it would have otherwise been infeasible, given the limited training data and the high-dimensionality of the models. I then presented in Chapter 4 how “auxiliary information” can be incorporated, in general, to this time series modeling so as to reduce the variation in the modeling and, in so doing, provide more robust models. In this chapter, I now show more practical aspects of using auxiliary information for ASR. First, in Section 5.1, I show how DBNs can be structured to do ASR modeling. While Sections 4.3, 4.4,& 4.5 gave illustrative DBNs, I here give the actual corresponding, full DBNs that are used; the difference is the incorporation of the “control” layer. Second, in Section 5.2, I present the actual types of auxiliary information that I am examining: articulator positions, pitch, rate-of-speech, energy, and graphemes. Third, in Section 5.3, I explain the software as well as the methodology that was used for some of the experiments (those done in the latter part of my studies, with mixed DBNs). In Chapter 6 I will then show the results of doing experiments with these DBNs and types of auxiliary information so as to show how they help the ASR to be more robust to variation in having a higher recognition rate in various conditions.

## 5.1 DBNs for ASR

Zweig (1998) introduced in detail how to do ASR with DBNs. The DBNs perform two simultaneous tasks:

1. Stochastic modeling of the hidden states
2. Deterministic modeling of legal sequences of states

In HMM-based modeling, there were only two stochastic/random variables, modeling the transitions and the emissions:

$$P(Q_n|Q_{n-1}) \tag{5.1}$$

$$p(X_n|Q_n) \tag{5.2}$$

The DBN for doing isolated word recognition is then constructed as in Figure 5.1. $TRANS_n$ and $X_n$ are stochastic variables that model equations (5.2) and (5.1), respectively while $POS_n$ and $Q_n$

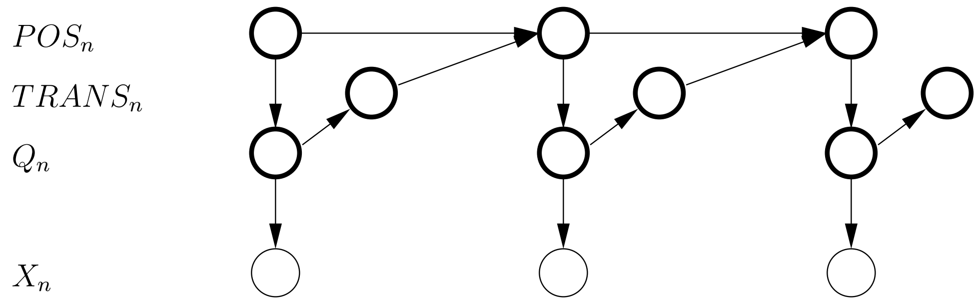


Figure 5.1: DBN for ASR, illustrated for three time frames (N=3). For any value of $N$ ($N > 1$), the same variables and edges are replicated for each time frame. For example, if $N = 4$, then we would add the variables $POS_4$, $TRANS_4$, $Q_4$, and $X_4$ on the right edge of the above DBN with edges added between time frames 3 and 4 ($< POS_3, POS_4 >, < TRANS_3, POS_4 >$) and with edges added within time frame 4 ($< POS_4, Q_4 >, < Q_4, X_4 >, < Q_4, TRANS_4 >$). The DBN is extended as such for any time length $N$.

are deterministic variables. The distribution $TRANS_n$ makes one assumption related to equation (5.1): that $P(Q_n = k'|Q_{n-1} = k) = P(Q_n = k''|Q_{n-1} = k)$ for any states $k', k''$. In other words, only the exit transition from a state is modeled, with $TRANS_n$ itself not putting any restriction modeling the probability of which phone can follow the current phone. The modeling of the sequence of phones is handled by the deterministic variables $POS_n$ and $Q_n$, which, along with $TRANS_n$, are part of the "control" layer of Zweig (1998). The control layer in the DBN framework accomplishes the same task as concatenating sub-models in the HMM framework: assuming a left-to-right modeling of phonemes, it specifies which sequence of phonemes the DBN is modeling. $POS_n$ will have a different index for each sub-model that is being used ("concatenated" in the HMM framework). The time dependency between $POS_{n-1}$ and $POS_n$ is used to force the indices to come in order (assuming a left-to-right sub-models). Each index in $POS_n$ maps to a single phoneme state in $Q_n$, via $POS_n$'s conditioning of $Q_n$. During the training phase, where the utterance labeling is known, the $Q_n$ distribution (i.e., mappings) needs to be changed from utterance to utterance; hence, we change the "controlling" so as to allow a different sequence of states. That is, as there is a different sequence of sub-models from one word to the next, the distribution (i.e., mapping) from $POS_n$ to $Q_n$ needs to be changed. Furthermore, during training, the model needs to end with a transition out of the final index/phoneme state; one way this can be done is to observe the final position, $POS_N$, to the highest index in the utterance and the final transition, $TRANS_N$, to 'true' so as to force the model to pass through all of the models when doing inference.

For example, if the DBN represents the word "cat", pronounced using three states: /k/-/æ/-/t/, $POS$ would have the following distribution:

$$P(POS_n = 1) = 1, n = 0 \tag{5.3}$$
$$P(POS_n = d+1|POS_{n-1} = d, TRANS_{n-1} = \text{'true'}) = 1, \forall d, 1 \le d < 3, n > 0 \tag{5.4}$$
$$P(POS_n = d|POS_{n-1} = d, TRANS_{n-1} = \text{'false'}) = 1, \forall d, 1 \le d \le 3, n > 0 \tag{5.5}$$

$Q_n$, furthermore, would have the following distribution:

$$P(Q_n = /k/|POS_n = 1) = 1, n \ge 0 \tag{5.6}$$
$$P(Q_n = /æ/|POS_n = 2) = 1, n > 0 \tag{5.7}$$
$$P(Q_n = /t/|POS_n = 3) = 1, n > 0 \tag{5.8}$$

Zweig (1998) also addresses another requirement upon the DBN: that it must end with a transition out of the last sub-model. He uses a static deterministic variable called 'End-of-sequence observation,' which is conditioned by $POS_N$ and $TRANS_N$, where $N$ is the last time frame. Its distribution is defined as follows:

$$P(\text{End-of-sequence Observation} = 1 | POS_N = \text{Last-Position}, TRANS_N = \text{'true'}) = 1 \quad (5.9)$$

It is then instantiated as having value '1,' thus forcing $POS_N$ to have value 'Last-Position' and $TRANS_N$ to have value 'true.'

In the implementation I used for my work, I have not used an 'End-of-sequence Observation.' Rather, I first observed $TRANS_N$ as having value 'true'. I then followed one of the following for fixing the value of $POS_N$:

1. observe it as having the 'Last-Position'

2. give it its own distribution specifying it must end in 'Last-Position.' The advantage of this second approach over just observing it is that this gives more flexibility in having multiple valid values for 'Last-Position,' which is reflected in the distribution given to 'Last-Position'. For example, there can be multiple valid positions in the last frame during decoding. One place where this is useful is in isolated word recognition, where there are, for example, 75 possible word models to classify an utterance as. By having 75 different paths through $POS_n$ with 75 respective values for the 'Last-Position', we can do inference over all of the models with one call to the DBN inference algorithm; this is in contrast to building a different DBN for each of the 75 possible word models (each with only one path through $POS_n$) and having to call the DBN inference algorithm 75 times, in this example. Another place this can be useful is in doing decoding in connected word recognition: an utterance can end with different words and, hence, in different last states; we can then indicate all of the valid states that we can end an utterance in.

In setting up DBNs for ASR, I have based the topologies upon those presented in Zweig (1998); Zweig and Padmanabhan (1999). In summary, the DBNs contain the following random and deterministic variables:

- $POS_n$ - deterministic (in training); for the model that the DBN represents and its sequence of sub-models (states), this is the sequence index ("position") of the sub-model. In recognition, when there are many candidate word models, it will be deterministic within a word model but random when transitioning between candidate word models.
- $TRANS_n$ - random; probability of exiting from a state.
- $Q_n$ - deterministic; for each value of $POS_n$, this is the phoneme state that it maps to.
- $J_n$ - random; the Gaussian mixture component for $X_n$ (not used in the discrete DBNs).
- $A_n$ - random; the auxiliary features.
- $X_n$ - random; the standard features.

Here I present the different base topologies actually used in the experiments: for discrete DBNs, for mixed DBNs, and for two Markov-chain DBNs. Those pictured are the "base" topologies; topologies with certain edge(s) removed are also used, as indicated. In addition to the variables already introduced, the mixed DBNs also contain the mixture component variable $J_n$ used for Gaussian mixture models.

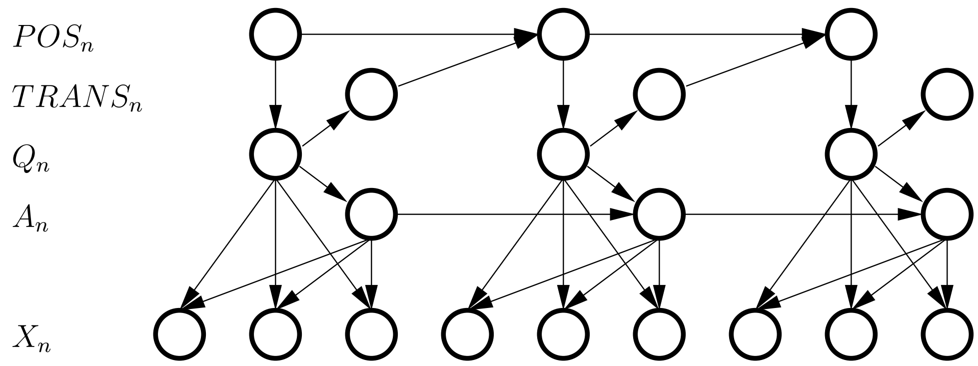


Figure 5.2: Discrete DBN with discrete time-dependent $A_n$. The edges $Q_n \rightarrow A_n$ are removed in certain experiments in Table 6.4.

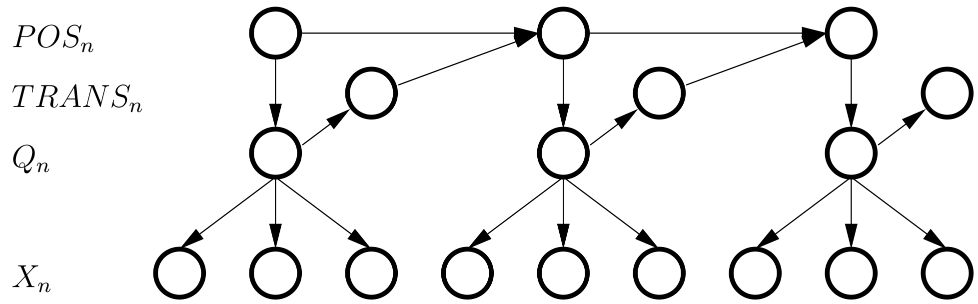


Figure 5.3: Discrete DBN, equivalent to standard discrete HMM. This DBN is used as the "BASELINE" system for the DBN experiments with discretized features $X_n$ and performs exactly the same as if an HMM had been used (there is no mixture component variable $J_n$ — see, for example, Figure 5.5 — because the features $X_n$ are discretized).

The discrete DBNs are looking at the effect of having a time-dependent discrete auxiliary variable $A_n$ (See Figure 5.2). That is, the topologies examined always contained a dependency $A_{n-1} \rightarrow A_n$. The dependency that was removed in some of the tests was $Q_n \rightarrow A_n$. These DBNs with $A_n$ were compared against a baseline approach (without $A_n$), equivalent to a standard HMM that has discrete features, as represented by Figure 5.3.

The mixed DBNs are looking at the effect of having a time-independent continuous auxiliary variable $A_n$ (see Figure 5.4). As with the discrete DBNs, I also look at the necessity of having the edge $Q_n \rightarrow A_n$. Furthermore, I look at whether $A_n$ should be conditioning $X_n$ by having the edge $A_n \rightarrow X_n$ removed. The dependency $A_{n-1} \rightarrow A_n$ was not looked at with mixed DBNs due, in part, to complexities in hiding it during inference (cf. Section 2.4.3); therefore, at each time frame, $A_n$ was considered conditionally independent of the previous time frames, given the state $Q_n$. These DBNs with $A_n$ were compared against a baseline approach (without $A_n$), equivalent to a standard HMM that has continuous features, as represented by Figure 5.5.

The mixed, two-chain DBNs (the factorial HMMs), as explained in Section 4.5, are looking at the effect of having a Markov chain of (discrete) auxiliary information during training. These two chains are treated as independent processes, given the observations, as represented in Figure 5.6 on page 70. In recognition, I only want to have one chain which I do decoding on–the other is to be left hidden–as explained in Section 4.5. It would be preferred that this hidden chain does retain

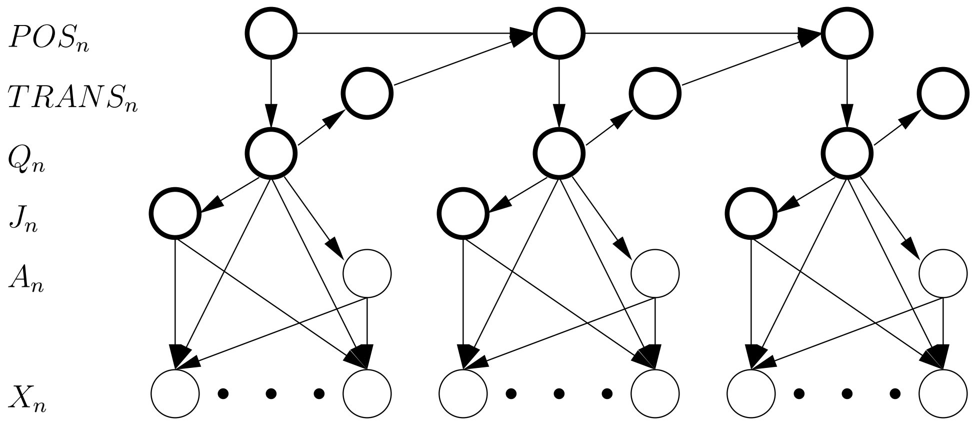


Figure 5.4: Mixed DBN with continuous, time-independent $A_n$. The edges $Q_n \rightarrow A_n$ are removed in certain experiments, $A_n \rightarrow X_n$ in certain other experiments.

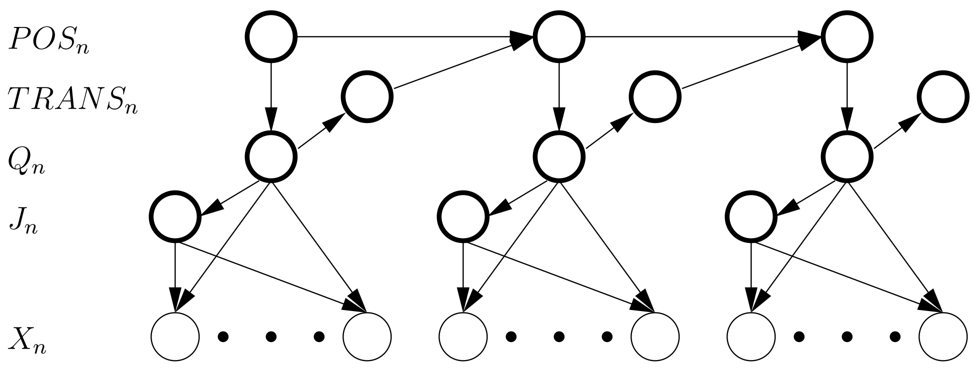


Figure 5.5: Mixed DBN, equivalent to standard HMM with Gaussian mixture models. This DBN is used as the "BASELINE" system for the DBN experiments with continuous features $X_n$ and performs exactly the same as if an HMM had been used.

time-dependency, which would show the amount of preference that the DBN has for the hidden chain's staying in the same state. I have chosen simpler, however less ideal, topologies which, while ignoring the time-dependency, do take into account the dependency of $Q_n$ and $L_n$ upon each other, as shown in Figure 5.7 on page 70.

## 5.2 Auxiliary Information Examined

I am looking at five types of auxiliary information: pitch (i.e., the fundamental frequency $F_0$), rate-of-speech, and short-term energy (in the logarithm domain), articulator positions, and graphemes. They are all fundamental features of speech that are speaker-dependent but which can also change within a given speaker according to prosodic conditions and the environment.

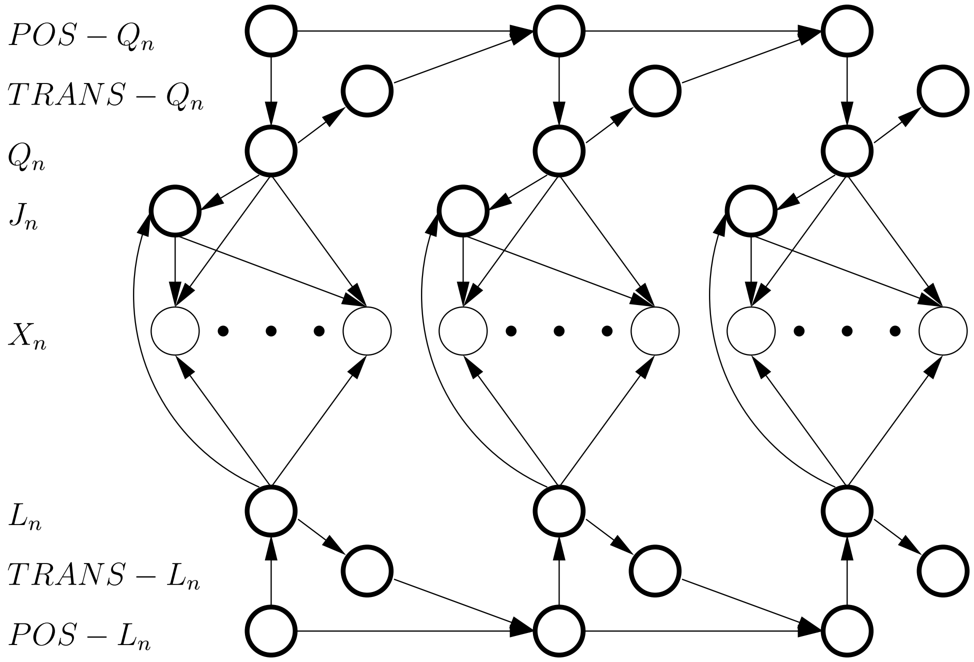


Figure 5.6: Mixed, two-chain, DBN, with phoneme chain $Q_n$ and grapheme chain $L_n$ for training. In recognition, one of the chains is broken and its variable is conditioned upon the other chain, as illustrated in Figure 5.7.

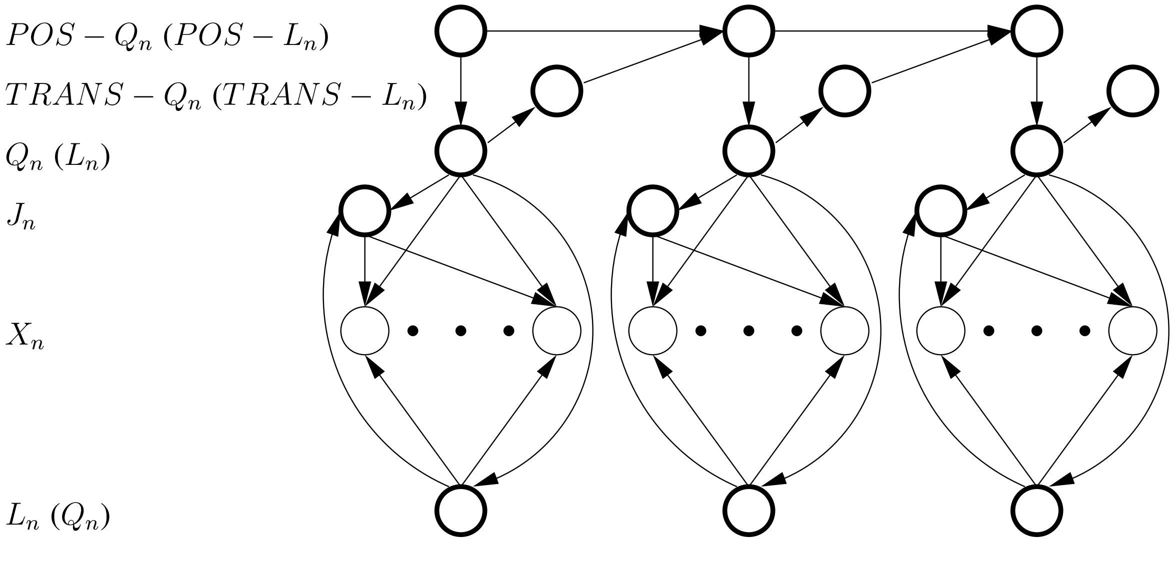


Figure 5.7: Mixed, single-chain version of the DBN in Figure 5.6, where the chain of $L_n$ is broken and conditioned by $Q_n$. Names given in parenthesis for some of the variables are alternate variable names; with the alternate variable names this DBN represents a mixed, single-chain version of the DBN in Figure 5.6, where the chain of $Q_n$ is broken and conditioned by $L_n$.

### 5.2.1 Pitch

Pitch (which I use to refer to the fundamental frequency $F_0$) is one of the most fundamental properties of speech but also one of the most difficult to extract (Hess, 1983). In principle, it is absent in the features usually used in ASR. However, in reality, it may still have be present in some form in these features. Given that pitch is difficult to estimate, there is going to be a lot of noise in whatever estimate I use. Therefore, these noisy estimates need to be incorporated carefully into the ASR models so as not to degrade the modeling. The estimates may be reliable enough over a large sampling of them so as to train good models but may be too noisy over a single utterance to be reliable in recognition.

Whether the pitch is non-zero or not (that is, if there is voicing) is highly correlated with the phonemes. Hence, there is some relation between the states and this auxiliary feature of pitch; however, this relation is between groups of states (voiced vs. unvoiced) and the auxiliary feature. The auxiliary feature for pitch (in units of Hertz) was estimated[1] using the simple inverse filter tracking (SIFT) algorithm (Markel, 1972), also with a window size of 25 ms. It was computed at the same frame rate as the standard features. The SIFT algorithm lowpass filters the speech at 300 Hz so as to have the band where the pitch is expected to be found; using the LP coefficients of the filtered speech, the same filtered speech is passed through an inverse filter. The second peak (the first peak is the energy) of the autocorrelation of the resulting excitation signal gives the period (and hence the pitch) of the speech. For simplicity, it was not transformed any further (i.e., no logarithm was taken of it) for these initial investigations into using pitch; pitch has been used in Fujinaga *et al.* (2001) in its logarithm form.

In this thesis, Pitch has been examined both in a discretized form and in the continuous form. The initial experiments with Pitch in Chapter 6 in isolated word recognition are done with both forms. The experiments with Pitch in spontaneous, noisy speech (Section 6.3) are done with continuous speech, though investigating discretized pitch in this context is a valid research path (though I have not investigated it in this thesis).

### 5.2.2 Rate-of-speech (ROS)

One obvious way the speaking rate changes the ASR models is in the transition probabilities between states. In fast speech, the probability for transitioning out of a state can increase. Further, the probability for skipping a state entirely increases, whereas in normal speech, we may have insisted on a zero probability of skipping a state. Another way that the speaking rate could change the ASR models is in the emission distributions for $X_n$. The articulators may be following a different trajectory in fast speech and may also have more coarticulatory effects. Both factors could change the underlying acoustic models.

I investigate the second correction for speaking rate changes by using conditional Gaussians for the acoustic modeling. By conditioning the Gaussians upon the speaking rate, the means of the models can adapt themselves to the speaking rate. Note that for a more complete study, I also look at using ROS in regular Gaussians and in ANNs.

There are different approaches for determining the ROS of an utterance. If a hand transcription is available, then it can be used to calculate an accurate ROS (Mirghafori *et al.*, 1995). Without a hand labeling (which would not be available in a deployed system), a forced alignment of the data needs to be done if the transcription is available (Siegler and Stern, 1995). The phone rate, or the normalized phone rate (Martínez *et al.*, 1997), are common units to use in rate-of-speech dependent ASR. In contrast to using a phone rate ROS dependent upon a transcription, more recent work has looked into using not only a *syllable* ROS but also having it calculated directly from the speech

[1] Thanks to Mathew Magimai-Doss.

signal (not using any transcription) (Morgan and Fosler-Lussier, 1998). Its benefit is that syllables appear to have a more uniform rate from one to the next than phones do and that there is no need to do a preliminary segmentation of the speech in order to compute the ROS. The auxiliary feature for rate-of-speech was estimated using *mrate* (Morgan and Fosler-Lussier, 1998), using a window size of 1 s. *mrate* estimates ROS by combining ROS estimates from three different signal processing algorithms; these algorithms look how the energy is changing. It has a correlation of .6 with the actual ROS (Morgan and Fosler-Lussier, 1998).

### 5.2.3 Short-term energy

Like pitch, energy is a fundamental element of speech, and it is often excluded from the features that ASR models. In clean speech, it is straightforward to calculate and does not have all of the difficulties that estimating pitch does. Nevertheless, care needs to be used in incorporating it into the ASR models. The short-term energy is correlated with the pitch. Indeed, the presence of voicing (the presence of non-zero pitch) in the signal adds much energy to it. Unlike pitch and ROS, the short-term energy can often be found as a standard feature in normal ASR systems but its benefit as a standard feature is questionable, as suggested by its use in Tables 6.8 & 6.11.

There are different approaches to calculating energy. First, the logarithm of the energy is commonly used instead of the plain energy so as to reflect the behavior of the human auditory system. Second, the energy of just a certain frequency band can be used; for example, (Fujinaga *et al.*, 2001) used low frequency energy as an estimate of voicing. Third, and more generally, energies for several frequency bands can be used (as standard features), as done in some of the work on multi-band speech recognition (Bourlard and Dupont, 1996). An energy feature will be directly affected by any noise found in the speech. However, if frequency bands are used, then only the frequency bands where the noise is found will be affected; this is taken advantage of in Morris *et al.* (1998), where the dimensions for the noisy features (energies, in their case) are marginalized out. Fourth, short, medium, or long-term energies can all potentially be used. Short-term energies may be useful for capturing the transitions between phoneme states, which are important to correctly modeling; long-term energies, on the other hand, may capture more prosodic information.

In this thesis, the auxiliary feature for energy was computed by taking the logarithm of the short-term energy of 25 ms of the signal:[2]

$$\log \left[ \frac{1}{C} \sum_{t=1}^{T} (\omega(t) s_n(t))^2 \right], \tag{5.10}$$

where $C$ is a normalizing constant used for numerical reasons, $s_n(t)$ is the $t^{th}$ sample of the extracted window of speech for time frame $n$, $\omega(t)$ is the $t^{th}$ element of a Hamming window, and $T$ is the length of both the speech sample and the Hamming window.

### 5.2.4 Articulators

Many have been interested in incorporating information about the articulators (principally, the lips, tongue, jaw, and velum) into ASR. There are various problems, however, in doing so. The main problem is the lack of training data with accurate measurements of the articulator positions. A secondary problem is how to properly incorporate any articulatory data.

Some people have addressed the problem of lack of data. Wrench (2000) has been recording continuous speech with simultaneous speech and articulator recordings. Zlokarnik (1995); Frankel

[2]Work on energy was done in collaboration with Jaume Escofet.

*et al.* (2000); Frankel and King (2001); Krstulović (2001) have dealt with estimating the articulator positions from the speech. Kirchhoff (1999) has used the phonetic labeling and the known relation between phonemes and articulators in order to estimate features that resemble articulator positions.

One issue in incorporating articulator information concerns whether the measurements will always be observed or not. If they are always observed, then they can be used as observations in Gaussians or ANNs. If they are hidden, they can not be used easily in ANNs but can be used (with certain restrictions) in Gaussians. Alternately, they can be incorporated into the state space (Richardson *et al.*, 2000). Also, the articulators in this work have been discretized for the initial investigations that this work does in investigating their use as auxiliary information; using the articulators in the continuous space is of interest but is not presented in this thesis.

### 5.2.5 Graphemes

One discrete source of auxiliary information is the graphemes of the word being spoken. The graphemes are the characters (letters and punctuation) that are used in written language. While some languages have a high correspondence between the written grapheme and the spoken phoneme (e.g., Spanish and Japanese), others have a more erratic correspondence between the written grapheme and the spoken phoneme (e.g., English). Furthermore, even if there is a simple correspondence between the graphemes and phonemes in carefully read speech of a given language, this may become more erratic in spontaneous speech. Therefore, one way to potentially deal with this pronunciation variation is to use the grapheme as auxiliary information to condition the acoustics of the phonemes. Conversely, the phoneme can be used as auxiliary information to condition the acoustics of grapheme-based states.

## 5.3 Software and Experimental Method

DBNs were used in ASR with software that was written primarily by myself. I present here some of the different modules used during the training and testing of the various DBNs in this thesis. Note that the discrete DBNs used here, as presented in Stephenson *et al.* (2001, 2000), actually used an earlier version of the software and a modified methodology. The experimental method outlined here is based on methods commonly used elsewhere (e.g., (Young *et al.*, 1999)).

### 5.3.1 DBNEXPECT

DBNEXPECT collects the "counts" for the 'E' step of EM training. It is given the list(s) of feature files along with their corresponding phoneme (or grapheme) labels. There can be multiple feature file lists as the auxiliary feature may be stored in a different file than the standard features. For each feature(s)/label set, it then performs the following:

1. "Splice" the DBN to the correct length, according to the number of frames in the feature file(s). This involves repeating each of the the variables being modeled for each point in time, along with their associated edges, as explained in Figure 5.1. In order to not have to rebuild the clique tree each time the length of the DBN is changed, Zweig (1998, Section 3.6) gives details of how to build one clique tree and how to change the length of the clique tree itself; in a similar manner to changing the length of the underlying DBN, this involves repeating certain cliques over time along with having the appropriate connections, with certain conditions, as explained in more detail in Zweig (1998).

2. Load the feature observations for the current utterance into the DBN. Also, change the "control layer" deterministic probability distributions according to the current labels (Zweig, 1998, Section 6.1); this will enforce that the DBN goes through the correct labels in order, over time.

3. Do inference on the DBN.

4. For each frame, collect the posterior distributions for each hidden discrete variable (typically, the transition variable and the mixture component variable) as well as for each hidden continuous variable (none of the work presented in this thesis happened to have hidden continuous variables during training). Also, collect the observations for each frame. Add all of this to the sub-total collected from previous feature(s)/label sets.

As also allowed in HTK-based HMM training (Young *et al.*, 1999), it is easy to incorporate simple parallel processing in the DBN training. The training lists are partitioned into multiple sub-training lists, each of which is used for a different processor. After collecting all of the counts of its given lists of files, the (sub-)total is written to a file.

### 5.3.2 DBNMAX

DBNMAX uses all of the count files from DBNEXPECT, each containing the (sub-)totals for their respective training lists, for the 'M' step of EM training. After summing them all together, it then normalizes them to obtain joint distributions; that is, a variable is not conditioned by its parents — rather, the distribution of the variable distribution *jointly* with its parents is maximized according to all of the counts. These maximized joint distributions are then factored so that at this point we obtain the *conditional* distribution of the each variable, given its parents. Before saving the resulting distributions in a file, the variances of the continuous variables are floored so that they are at least 0.10 the global variance of the training data. The choice of global variance can be changed — Young *et al.* (1999) gives a suggested value of 0.01 the global variance; I used 0.10 the global variance as it seemed to be better to err in having too large a variance than in having too small a variance, especially if there is a limited amount of training data for estimating a Gaussian mixture component for a given state. The results of DBNMAX can then be used again in DBNEXPECT for another iteration of EM training. For the systems with mixed DBNs, I continued training the DBNs until the difference between the log likelihoods outputted by DBNMAX between two successive training iterations decreased by less than 0.1% (note that the likelihood given by DBNMAX is the data likelihood of the inputted DBN–not of the outputted, maximized DBN — so we, in effect, do one more iteration of EM after reaching the criterion of reducing the log likelihood by less than 0.1% over the previous iteration); this convergence criterion was chosen as a balance between having DBNs that were reasonably trained and having the training finished in a reasonable amount of time.

### 5.3.3 DBNSPLIT

DBNSPLIT is used for splitting Gaussians. It is only used in this thesis with the experiments related to graphemes/phonemes. With the grapheme system and combined graphemes/phonemes system, I did not have any segmentation available to use for providing initial estimates of GMMs to the DBNs. Therefore, I chose to start the training with single Gaussians for every state, all of them with the same parameters. To avoid having to split the data, I could have done $K$-means clustering over all the training data and then use the single, resulting GMM as the initial estimate for all states. In my case I initialized all Gaussians with a mean of 0 and a variance of 1; this could equally have been the global mean and variance of the training data, though this would not make any difference, in theory. This does not make a difference because if each state value is initialized with the same Gaussians, they will each give the same likelihood for a given frame $n$ (though each time frame can have a different likelihood), i.e., $p(X_n = x_n|Q_n = q'_n) = p(X_n = x_n|Q_n = q''_n)$ for all state values $q'_n, q''_n$; it can be shown that the posterior distribution of the hidden variables during the first iteration EM training will be the same, regardless of that common likelihood that each state gives at a given time frame $n$.

After a given number of iterations of EM training, I then split some (typically, about 60%) of the trained Gaussians. It is the Gaussians with the highest mixture component weights that were split by replacing them with Gaussian with the same variance but with the means shift 0.1 times the standard deviation to the left and right of the original Gaussian and with mixture component weights 0.5 times the original weight. Young *et al.* (1999) has the same algorithm except for shifting the means by 0.2 times the standard deviation — the shift amount can be changed; a shift of 0.1 times the standard deviation gives Gaussians that are not as far away from the original but, hopefully, separated enough so as to each be learned well in further EM training. Two more iterations (the number of iterations was chosen so as to do some training with the split Gaussians but to avoid getting stuck in a local minimum before a possible additional round of Gaussian splitting) of EM training then occurred before optional additional splitting; the splitting was continued until the desired number of mixture components was reached. After all rounds of Gaussian splitting were done and if the two iterations of EM after the last splitting did not reach the convergence criterion (as given in Section 5.3.2), EM training occurred until the convergence criterion was reached.

### 5.3.4 DBNRECOG

DBNRECOG is used for performing "best-of-$N$" recognition on a trained DBN. That is, a list of $N$ potential utterances (words) is given (along with their lexicon) and the DBN is used to determine the most likely utterance. DBNRECOG has only been used for isolated word recognition. It finds the "full" likelihood of each potential utterance by taking into account all possible assignments of state values to the frames.

### 5.3.5 DBNVITE

DBNVITE is used for performing Viterbi decoding (see, in the HMM framework, (2.28) and (2.41) on pages 15 & 18, respectively, and, in the DBN framework, (3.9) on page 36) through the DBNs in continuous speech. As Viterbi decoding in the DBN context implies doing a maximization over every discrete variable in the DBN, I make one modification to Viterbi coding in DBNs so as to make it equivalent to what is done in HMMs: in dealing with the values of the mixture component variable $J_n$, a summation is done — in dealing with all other discrete variables in the DBN, a maximization is done. For a given lexicon, one or more of the words in the lexicon will appear in a given utterance and need to be "decoded." DBNVITE uses a very simple language model by stating that there is an equal probability of changing from one word to the next. Furthermore, it does not incorporate factors such as a word insertion penalty or a language model scaling, as used in software such as HTK (Young *et al.*, 1999); so, the recognition accuracy results could potentially have been raised had DBNVITE used these factors and if they had been appropriately tuned from data. Note that no pruning is done during decoding.

### 5.3.6 MAXDISCOND

MAXDISCOND is used for the grapheme/phoneme system, where there are two independent Markov chains emitting the same features. As I then modify the topology to create two DBNs for recognition such that the grapheme variable $Q_n$ or the phoneme variable $L_n$ is no longer a deterministic variable, I need to provide a probability distribution for this variable which is now random. As both $Q_n$ and $L_n$ are parents of the features $X_n$, they appear together in the counts files for learning the distribution of $X_n$. I can therefore extract the counts for one of these child features, remove the distribution of the features, and then factor the resulting discrete distribution twice: once for the graphemes conditioned by the phonemes and once for the phonemes conditioned by the graphemes. For example, during DBNEXPECT, we are collecting counts for maximizing the joint distribution $p(X_n, Q_n, L_n, J_n)$. These counts are maximized and factored in DBNMAX so as to obtain the emission distribution $p(X_n|Q_n, L_n, J_n)$:

$$p(X_n|Q_n, L_n, J_n) = \frac{p(X_n, Q_n, L_n, J_n)}{p(Q_n, L_n, J_n)} \tag{5.11}$$

However, in the new topologies we also need a distribution for $Q_n$ (or for $L_n$ if $L_n$'s chain is being broken); we can use the same counts used for $p(X_n, Q_n, L_n, J_n)$ and factor it again in a different way in MAXDISCOND so as to obtain the conditional distribution $P(Q_n|L_n, J_n)$ (or $P(L_n|Q_n, J_n)$ if $L_n$'s chain is being broken). For example, in breaking $Q_n$'s chain, we would have:

$$P(Q_n, L_n, J_n) = \int_{-\infty}^{\infty} p(X_n, Q_n, L_n, J_n)\, \mathrm{d}X_n \tag{5.12}$$

$$P(Q_n|L_n, J_n) = \frac{P(Q_n, L_n, J_n)}{\sum_{k=1}^{K} P(Q_n = k, L_n, J_n)} \tag{5.13}$$

The discrete conditional distribution from MAXDISCOND will be used in addition to the maximized distributions typically provided by DBNMAX in the recognition. As it is only needed for the DBNs with the new topologies used in recognition, MAXDISCOND is called only at the end of EM training.

## 5.4 Conclusion

In this chapter I have presented more of the practical aspects of my work, from the actual topologies used for implementing DBNs for auxiliary information based ASR in Section 5.1 to the actual types of auxiliary information that I use in Section 5.2. Various topologies of the actual DBNs used in my work were presented in Section 5.1: those for discrete-only DBNs, followed by those for mixed DBNs. The discrete-only DBNs are for investigating time-dependent auxiliary information while the mixed DBNs are for investigating various dependencies within a single time frame. Various types of auxiliary information were presented in Section 5.2: discrete information (discretized articulators and pitch as well as graphemes) and continuous information (pitch, rate-of-speech, and energy). A review of some of the software that I developed for this work was also given in Section 5.3. Using this software (and others) with these various types of auxiliary information as well as with these various DBNs, I will show the usefulness of using auxiliary information in DBNs (as well as in HMM/ANN hybrids) in Chapter 6.

# Chapter 6

# Experiments

The experimental foundation for the DBN topologies and types of auxiliary information being laid in Chapter 5, I now proceed in this chapter to present the experiments performed on such topologies and data in two parts. First, in Section 6.2 I discuss experiments on isolated word recognition by looking discretized auxiliary information, auxiliary chain information (factorial HMMs), and continuous auxiliary information. Then, in Section 6.3, I move onto the more complicated scenario of spontaneous, noisy speech recognition, this time using only continuous auxiliary information.

## 6.1 Preliminaries

All significance tests in this thesis are done with a standard proportion test, assuming a binomial distribution for the targets, and using a normal approximation. All DBN models (and their HMM baselines) were trained using expectation-maximization (EM) training. In the models with discretized features $X_n$, after the log likelihood increased by less then 1% from the previous iteration, one more maximization step was done before termination; in the models with continuous features $X_n$, after the log likelihood increased by less then 0.1% from the previous iteration, one more maximization step was done before termination. With both cases (discretized $X_n$ vs. continuous $X_n$), the respective convergence criterion was chosen arbitrarily according to what I considered in the circumstances to be reasonable. No cross-validation set was used in the EM training nor in the recognition (except as noted with the articulator experiments)

All DBN systems with discretized features $X_n$ were initialized with a uniform distribution for simplicity reasons. All DBN systems (except those in Table 6.5) with continuous features $X_n$ were initialized using segmentations computed using a forced-alignment over the training data.[1] These segmentations were used to compute the transition probabilities (with a floor/ceiling of 0.20/0.80 for these initial probabilities so as to give the training more flexibility) and the (conditional) GMMs for each state; the GMMs used with the PhoneBook database were computed using binary splitting (Rabiner and Juang, 1993, Section 4.4.4) on the segmented data while the GMMs for the OGI Numbers database were computed using a modification where only one mixture component was split per iteration. This modification of the splitting for OGI Numbers was due to problems in estimating the GMMS with splitting each mixture component at each iteration; while this modification was not done with the PhoneBook data, such a modification might also be advantageous in future work with PhoneBook as it takes a more cautious approach to estimating the mixture components. For simplicity, in estimating the initial parameters for conditional GMMs, the $B$ parameters were always set to 0 and the weights, means, and variances were initialized as for normal GMMs; during

[1] Thanks to the Faculté Polytechnique de Mons, Belgium for their help with this.

the training, the $B$ parameters would be learned. The systems in Table 6.5 (involving graphemes and/or phonemes) were all trained with a uniform distribution. This is because while a segmentation is available for the phoneme states, no segmentation is available for the grapheme states; so, to have systems that were trained in a similar fashion, all systems were trained with the initial uniform distribution. After each two iterations of EM, approximately 60% of the mixture components were split as explained in Section 5.3.3. Such types of initializations as described above can also be done in the framework of normal HMM systems (except those related to conditional GMMs, as normal HMMs can not handle conditional GMMs).

## 6.2 Isolated Word Recognition

To investigate the use of auxiliary information in ASR, I first investigated its use in isolated word recognition, which has the advantage of not having to worry about the decoding issues that arise with continuous speech recognition. However, it has the disadvantage of not being able to test out the effectiveness of the auxiliary information on continuous, spontaneous speech. To do recognition, the likelihood of each of the candidate models was used, with equal priors over all models; no approximations, such as Viterbi, were used in computing the likelihoods.

### 6.2.1 Discretized Auxiliary Information

I used two different databases for ASR with discretized auxiliary information: PhoneBook (Pitrelli *et al.*, 1995) and the University of Wisconsin X-ray Microbeam Speech Production Database (Westbury *et al.*, 1994) (hereafter referred to as the "Microbeam" database). PhoneBook was specifically designed for research into isolated word recognition. The Microbeam database, however, was not designed with speech recognition research in mind; specifically, it lacks both the variety of data and the amount of data that PhoneBook was specifically designed to have. As a result, I have used PhoneBook by default for my experiments except for when examining the auxiliary feature of articulatory positions that is absent in PhoneBook. The results on discrete articulators and discrete pitch appeared, in part, in Stephenson *et al.* (2000) and Stephenson *et al.* (2001), respectively. Note that all variables used in these DBNs are discretized; so as the emission distributions for $X_n$ are represented by tables, no Gaussian distributions and, hence, no mixture component variables, are used (as mixtures are generally used in ASR with continuous emissions).

#### Articulators

Using the University of Wisconsin X-ray Microbeam Speech Production database (Westbury *et al.*, 1994), I did experiments on speaker-independent, task-dependent, isolated word recognition with articulator information as auxiliary information. The speech is recorded at 21739 Hz with a recording of selected articulator positions (lower lip, upper lip, four tongue positions, lower front tooth, and lower back tooth) at approximately 146 Hz (6.866 ms between samples). Of the 48 speakers in the database, eight were randomly selected to be in the test set; of the remaining 40 speakers, eight were randomly selected to be in the validation set with the remaining 32 speakers comprising the training set. All three lists were constructed as to be gender-balanced (the test and validation sets each have an equal number of males and females while the training set has four more females than males). There are different tasks that the speakers were asked to do. For this work, I chose to use the "Citation Words" tasks, where the speaker reads a list of single words, separated by pauses. Using a segmentation[2] produced by a forced alignment with an HMM based on an HTK system

[2]Provided by Sacha Krstulović

| | # Param. | WER |
|---|---|---|
| Baseline (HMM, no Articulatory Variable) | 31k | 9.8% |
| 2 Discrete Articulatory Values | 63k | 8.5% |
| 4 Discrete Articulatory Values | 127k | **7.7%** |
| 8 Discrete Articulatory Values | 257k | ***8.4%*** |

Table 6.1: Discrete articulators. Recognition results, given as Word Error Rate (WER), for models trained on the training set (with observed articulators), with recognition performed on the *validation* set of 2155 words (with hidden articulators). The number of free parameters is given. The DBNs in Figure 5.3 & 5.2 were used (without any edges removed). Results in **bold** are significantly better than the baseline with 95% confidence; results in ***bold italics*** are significantly better than the baseline with 99% confidence.

(Young *et al.*, 1999), the set of words for each Citation Words task was cut into individual files with some surrounding silence. The lexicon size was 106 words; some of the words were repeated multiple times by the same speaker, giving an average, across all of the data, of about 260 utterances per speaker. There was a total of 8468 utterances in the train set, 2155 in the validation set, and 2151 in the test set. Thirty-nine monophone models (as used in the earlier work on using this database for ASR in Krstulović (2001, Chapter 5)) were used in addition to beginning and ending silence. Three states were used for each monophone and silence being modeled.

Twelve mel-frequency cepstral coefficients (MFCCs) plus $C_0$, the energy coefficient, were extracted per window from the speech, using a Hamming window of 20.6 ms with successive windows shifted by 6.9 ms. This shift rate was chosen so as to have one articulatory observation per window. There were 26 filters in the filterbanks with a preemphasis coefficient of 0.97. Energy normalization as well as cepstral mean subtraction were performed. The delta (i.e., first derivative) coefficients for all 13 MFCC coefficients were used as well.

The cepstral coefficients were then quantized. Using K-means clustering, four codebooks are generated from the training data, as done in Zweig (1998): a 256 value codebook for the 12 MFCC coefficients, a 256 value codebook for the 12 MFCC delta coefficients, a 16 value codebook for the $C_0$ coefficient, and a 16 value codebook for the $C_0$ delta coefficient. The $C_0$ and the $C_0$ delta values are concatenated bitwise in the DBN to give a single 256 value variable.

Likewise, the articulatory values are also quantized, using K-means clustering. The measurements of the eight articulators are used for the codebook. For certain frames (22% of the frames, across all of the data), an articulator value was not recorded for some time slices; in these cases, the whole vector for the concerned time slices was thrown out and not used in any part of the experiments. One codebook was generated to represent all eight articulator positions. Various values for the size of the codebook are presented in this paper: two, four, and eight. The baseline DBN system did not use an articulatory variable; with such a configuration, it was theoretically equivalent to a standard discrete HMM for ASR.

During EM training, Dirichlet priors (Cooper and Herskovits, 1992) of 0.1 were used on all probabilities to prevent any from becoming 0; this prior is added to all counts obtained from EM:

$$N^*(\cdot) = N(\cdot) + 0.1. \tag{6.1}$$

$N^*(\cdot)$ is then used instead of $N(\cdot)$ during maximization. The effect is to give greater weight to variable values that did not occur very often in the training data. Except where noted, recognition was then performed using only the acoustics from the validation set (the articulators were ignored and thus treated as hidden). Results are given in Table 6.1 for a system trained on the training set with recognition on the validation set. As can be seen, the word error rate is improved when articulatory information is added.

| | # Param. | WER |
|---|---|---|
| Baseline (HMM, no Articulatory Variable) | 31k | 8.6% |
| 4 Discrete Articulatory Values | 127k | 7.8% |

Table 6.2: Discrete articulators. Recognition results, given as Word Error Rate (WER), for models trained on both the training set and the validation test (with observed articulators) with recognition performed on the *test* set (with hidden articulators). The number of free parameters is given. Only the best acoustics/articulatory system from Table 6.1 was used. The DBN in Figure 5.3 & 5.2 were used (without any edges removed). These two results are not significantly different, in contrast to the corresponding results presented in Table 6.1.

| | # Param. | WER |
|---|---|---|
| 4 Discrete Articulatory Values | 127k | 7.6% |

Table 6.3: Discrete articulators. Using *observed* articulator values, recognition results, given as Word Error Rate (WER), for models trained on both the training set and the validation test (with observed articulators) with recognition performed on the test set (with observed articulators). This uses the same trained acoustics/articulatory system from Table 6.2. The number of free parameters is given. The DBN in Figure 5.2 was used (without any edges removed). This result is not significantly better than the baseline in Table 6.2.

Using the optimal number of discrete articulatory values on the validation set given in Table 6.1, I then started the experiments over using only the baseline system (acoustics only) and the best acoustics/articulatory system (with four articulatory values). However, this time all codebook generation and DBN training were done on the combination of the training set and the validation set. Recognition was then done on the test set. The results are given in Table 6.2. The results of these recognition tests are the true estimates of the two systems' performances on new data as the test set was not used previously to select any parameters for either system.

Finally, note the results in Table 6.3, where I used the articulator values in recognition. While having measured articulator values in recognition is an unrealistic scenario, in comparing the results in Table 6.3 with that in Table 6.2, we see the ability of the DBNs to cope with hidden articulators. That is, in Table 6.2, the DBN used only inferred articulator values and obtained a WER of 7.8%, which is close to the result of 7.6% obtained using the actual, observed articulator values. Also, while the tests on the validation set in Table 6.1 gave a significant improvement with an articulator variable, the tests on the test set in Table 6.2 did not give a significant improvement; therefore, tests on a more extensive database need to be done in future work.

**Pitch**

I have selected PhoneBook (Pitrelli *et al.*, 1995) as the corpus to use for training and testing my models for using discrete estimated pitch as auxiliary information. The bulk of this corpus contains isolated word utterances collected from a large number of speakers with a large variety of words, spoken over telephone lines. I have chosen the data partition used in Dupont *et al.* (1997): I used their "small" training set of 19421 utterances (over 283 speakers) for training my models and their "cross-validation" set of 6598 utterances (over 106 speakers) for testing my models. The small training set contains 21 lists of words while the cross-validation set contains 8 lists of words; the "test" set, which will be used in later experiments, itself contains 8 lists of words with a total of 6598 utterances (over 96 speakers). Each of these lists contains 75 or 76 words that are unique from the other lists used and is read by a different set of speakers.

| | # Param. | Num. Pitch Values | Obs. Pitch | Hid. Pitch |
|---|---|---|---|---|
| Baseline | 33k | - | 7.8% | |
| Pitch Baseline | 66k | 2 | 8.5% | 7.6% |
| Pitch Baseline | 133k | 4 | 7.9% | 7.1% |
| Pitch Baseline | 270k | 8 | 8.6% | 7.2% |
| Pitch ($A_n \perp\!\!\!\perp Q_n \mid A_{n-1}$) | 66k | 2 | 8.5% | 7.7% |
| Pitch ($A_n \perp\!\!\!\perp Q_n \mid A_{n-1}$) | 133k | 4 | 8.0% | 7.2% |
| Pitch ($A_n \perp\!\!\!\perp Q_n \mid A_{n-1}$) | 270k | 8 | 8.9% | 7.2% |

Table 6.4: Discrete pitch. Word Error Rates. Both results within the same given line are from the same system, which was trained on estimated pitch data (discretized). The number of free parameters is given. The DBN in Figure 5.3 was used for the "Baseline" (identical to a standard discrete HMM); the DBN in Figure 5.2 was used for the "Pitch Baseline"; and the DBN in Figure 5.2 was used, except with the connections for $Q_n \rightarrow A_n$ removed, for the "Pitch ($A_n \perp\!\!\!\perp Q_n \mid A_{n-1}$)". The best "Pitch Baseline" and the best "Pitch ($A_n \perp\!\!\!\perp Q_n \mid A_{n-1}$)", both of which have 4 discrete prototypes, perform statistically the same as the "Baseline". All six results with hidden discretized pitch are statistically better (with at least 95% confidence) than the respective results with observed discretized pitch. Furthermore, the best system with hidden pitch, giving a result of 7.1%, performs statistically better than the baseline result (with 95% confidence).

Sampled at 8 kHz, the speech signal (initially recorded $\mu$-law format but transformed into pcm format during the feature computation) was parameterized using mel-frequency cepstral coefficients (MFCC's) with a Hamming window of 25 ms in width, shifting 8.3 ms per frame. Ten MFCC's as well as $C_0$, the energy coefficient, were retained for the models (ten MFCCs being retained here so as to have similar features to the related experiments done in Zweig (1998)). I then created three codebooks for the acoustic data (using K-means clustering on the training data), each of 256 values, as done with the Microbeam database acoustics.

Codebooks of size one, three, and seven were generated for the data where the pitch (as estimated by the Simple Inverse Filter Tracking algorithm (SIFT) (Hess, 1983)) was non-zero. With an additional entry in each reserved for the case for unvoiced speech (where the pitch is zero), this resulted in codebooks of size two, four, and eight (as appearing in the column "Num. Pitch Values" in Table 6.4). The codebooks were each reserved for a different set of experiments.

All systems used three hidden states for each monophone model, as is often done with ASR (though, similar work with DBNs and the PhoneBook database also did some experiments with four hidden states per monophone (Zweig, 1998)); there were 41 monophone sub-word models plus models for beginning silence and ending silence (the same set of 41+2 models is based on earlier work using DBNs on this database in Zweig (1998, Chapter 7)). A baseline system, with no auxiliary information was trained using the baseline model and data described above. It is theoretically equivalent to a discrete HMM. The two sets of DBNs (see Figure 5.2, with and without the connection $Q_n \rightarrow A_n$.) with auxiliary information were each trained using the different sized codebooks for the auxiliary information, as explained above.

Using these trained DBNs, I tested their performance with the auxiliary information observed and also with it left hidden. Results are given in Table 6.4. In all cases, the DBNs with $A_n$ performed significantly better when the auxiliary information was left hidden than when the auxiliary information was observed. Furthermore, the best performing DBN with $A_n$ is significantly better than the Baseline DBN: 7.1% verses 7.8%.

The significant improvement obtained in hiding the pitch could be due to the noise inherent in its estimation. That is, the SIFT estimator will not be correct all of the time. While training with the estimates over the large amount of data in the training set, we can assume that the erroneous estimates are infrequent enough that the resulting learned distributions related to the pitch are fairly accurate. However, in recognition, erroneous pitch estimates for an individual utterance can not as easily be overcome. That is, there may be many utterances in the recognition evaluation that have been corrupted by noise and, hence, misrecognized; the effect of these errors would then overcome any potential benefit of better recognizing those utterances with accurate pitch estimates. Therefore, using the inferred (hidden) pitch estimates in recognition provides more robust recognition than using the estimates.

### 6.2.2 Auxiliary chain information (factorial HMMs)

Auxiliary information in the form of a second auxiliary chain is also discrete information and could have been included in the section above on discrete auxiliary information. However, since it is used differently and since the acoustic vector itself was not discrete in these experiments, I include it here in its own section. ASR is typically performed using Markov chains whose state represents phonemes; recent work has investigated using graphemes (the actual letters and punctuation used in writing) as the states (Kanthak and Ney, 2002). In using auxiliary chains, I am looking here training models having both a chain for the phonemes and a chain for the graphemes. Recognition is then performed with the time-dependency broken on whichever chain is chosen as the "auxiliary" chain; recognition is done on the units of the "primary" chain with the aid of the auxiliary chain.

These experiments were done using the PhoneBook database, as already introduced in Section 6.2.1. The same MFCC coefficients as used with the PhoneBook experiments in Section 6.2.1 are used; however, in these experiments I did not use the coefficient $C_0$ but did use its delta coefficient. $C_0$ was not used here as other work on the PhoneBook database using continuous features (Dupont *et al.*, 1997) uses only the first (and second) derivatives of energy (which resembles $C_0$) but not the energy itself. The advantage that this database has for these experiments is that there is a wide variety of words available in it. In training a DBN with two chains (i.e., a factorial HMM), there is to be a Gaussian (or GMM) for each combined instantiation of the two chains. That is, given $K$ states in the phoneme chain and $L$ states in the grapheme chain, there are $K \cdot L$ Gaussians (or GMMs) to be learned in the observation space. In Figure 4.8 on page 62, we see that $X_n$ has two parents $Q_n$ and $L_N$ (here I note that they have $K$ values and $L$ values, respectively); so a Gaussian (or GMM) needs to be learned for each joint instantiation of the two parents. So, a database where the phonemes occur in a wide variety of grapheme contexts (and vice-versa) is advantageous for this. Note that, for example, the phoneme /æ/ will probably occur more often with the grapheme 'a' than with the grapheme 'z'; however, we would want the phoneme /æ/ to occur with a wide variety of graphemes in its *context*; conversely, we would also want each grapheme to occur with a wide variety of phonemes in its context.

Results are shown in Table 6.5. As with other experiments with PhoneBook in this thesis, there were 43 phoneme states (but with a different number of sub-states for the non-silence phonemes, as indicated); there were 28 grapheme states (the 26 letters of the Latin alphabet, begin silence, end silence, and the '+' symbol which is used in the PhoneBook dictionary between certain affixes and the root word). With the two Markov chains, two (time-dependent) chains were trained, both simultaneously conditioning the acoustic observation. For recognition, a given two-chain DBN is broken in two different ways to make a one-chain DBN: once for "$Q_n$ mode" (where $L_n$ is made a time-independent auxiliary variable) and once for "$L_n$ mode" (where $Q_n$ is made a time-independent auxiliary variable). In "$Q_n$ mode", we are doing recognition using phonemes and with a phoneme lexicon; at each time frame all of the graphemes in $L_n$ will be summed over during inference In "$L_n$ mode", we are doing recognition using graphemes and with a grapheme lexicon; at each time frame all of the phonemes in $Q_n$ will be summed over during inference Different numbers of states were tried (i.e., each phoneme and/or grapheme, except for the begin and end silence models, was divided

| #-chains | # Param. | | # sub-states $L_n$ | # sub-states $Q_n$ | Mixes | WER |
|---|---|---|---|---|---|---|
| 2 | 52k | $Q_n$ mode (with auxiliary $L_n$) (Figure 5.7) | 1 | 1 | 1 | 21.5% |
| 2 | 52k | $L_n$ mode (with auxiliary $Q_n$) (Figure 5.7, with alt. var. names) | 1 | 1 | 1 | 41.4% |
| 1 | 52k | $Q_n$ chain (Figure 5.5) | - | 1 | 28 | 9.9% |
| 1 | 52k | $L_n$ chain (Figure 5.5, with $Q_n$ changed to $L_n$) | 1 | - | 42 | 20.5% |
| 2 | 198k | $Q_n$ mode (with auxiliary $L_n$) (Figure 5.7) | 2 | 2 | 1 | 13.6% |
| 2 | 198k | $L_n$ mode (with auxiliary $Q_n$) (Figure 5.7, with alt. var. names) | 2 | 2 | 1 | 40.5% |

Table 6.5: Phoneme ($Q_n$) and Grapheme ($L_n$) Markov chains, using auxiliary chain information as two Markov chain DBNs (factorial HMMs) and, for comparison, one Markov chain DBNs. The number of sub-states (in the non-silence states) per phoneme and per grapheme (as applicable) is given. "#-chains" indicates how many chains were used in training. When one of the chains is broken, it is no longer a factorial HMM. The models trained with one-chain have many mixture components so as to give them approximately the same total number of Gaussians as the two-chain DBNs (after accounting for Gaussian components with zero weight); the results are grouped according to DBNs that have the same total number of Gaussians. Results are stated in word-error-rate (WER).

into multiple sub-states, such as 'a' being represented by a[1] and a[2]). These results show that, in the current framework, graphemes can not be used to aid the modeling of the phoneme states. However, I would expect a better performance if a time-dependency were to be added between the "auxiliary chain" (that is, if we did not break a chain during inference); however, this would add complexity to the inference. The auxiliary chain DBN with twice the number of states per phoneme/grapheme performs worse (13.6 WER) than a traditional phoneme DBN with only one sub-state per phoneme (9.9 WER) which has one-quarter of the parameters. Future work would be to look properly the effect of using two or more sub-states per grapheme in a single chain.

### 6.2.3 Continuous Auxiliary Information

Work on incorporating continuous auxiliary information into isolated word recognition was performed exclusively on the PhoneBook database. The same MFCC coefficients as used with the PhoneBook experiments in Section 6.2.2, coefficients are used. The same set of discrete states was used here as was used in the case of discretized pitch. For further background discussion on the following auxiliary variables, please refer to Section 5.2, as well as Section 6.2.1. The experiments with continuous auxiliary information on the PhoneBook database were trained on the small training set and were tested on the test set of Dupont *et al.* (1997) (in contrast to the systems with discrete pitch in Section 6.2.1 being tested on the validation set of Dupont *et al.* (1997)).

These results are based on those that appeared in Stephenson *et al.* (2002a,c) and, in an earlier form, in Stephenson *et al.* (2002b). Notable differences between these results and those in Stephenson *et al.* (2002a,c) are that, first, I give all systems here the same number of mixture components (which is in line with the methodology used in Section 6.3) and, second, I use a higher variance floor here so as to have more robust Gaussians. The number of mixture components was chosen so as to

| | # Param. | Mix. | Obs. ROS | Hid. ROS |
|---|---|---|---|---|
| Baseline | 32k | 6 | 4.3% | |
| ROS Baseline | 48k | 6 | 4.4% | 4.3% |
| ROS ($X_n \perp\!\!\!\perp A_n \mid Q_n$) | 32k | 6 | 4.3% | 4.3% |
| ROS ($A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$) | 48k | 6 | 4.4% | 4.3% |

Table 6.6: Word error rate for the Baseline (non-ROS) DBN and for the three ROS DBNs on the PhoneBook test set. Results for the ROS DBNs are given with observed and hidden ROS. The number of parameters and of mixture components is given as well. The DBN in Figure 5.5 was used for the "Baseline" (identical to a standard HMM); the DBN in Figure 5.4 was used for the "ROS Baseline"; the DBN in Figure 5.4, except with the connections $A_n \to X_n$ removed, was used for the "ROS ($X_n \perp\!\!\!\perp A_n \mid Q_n$)"; the DBN in Figure 5.4, except with the connections $Q_n \to A_n$ removed, was used for the "ROS ($A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$)". There is not statistical difference between any of the results.

be large enough to get reasonable recognition but to be small enough so as to reduce the computation complexity. The variance floor was chosen to be 0.10 of the variance of the variables over all of the training data; while the HMM training tool HTK (Young *et al.*, 1999) suggests using 0.01 of the global variance, 0.10 was used so as to take a more conservative approach by having broader Gaussians. I have not repeated the earlier results of Stephenson *et al.* (2002a,c) here in this thesis since the results given in this thesis cover the same tasks but with a better training setup in using the variance floors. (I also note here an errata for the "Figure 1 (b)" result in Table 1 of Stephenson *et al.* (2002a) as well as for the "Pitch $x_n \perp\!\!\!\perp a_n \mid q_n$" result in Table 1 of Stephenson *et al.* (2002c): in both tables in those works, the word error rate should be 52.0 for Obs. Pitch and 6.7 for Hid. Pitch; this big difference in performance, appears to be because of not having a variance floor in those systems).

### Speaking Rate

Experiments using speaking rate, as explained in Section 5.2.2 on page 71, are presented in Table 6.6. The speaking rate is calculated at the same frame rate as the MFCCs. Using the speaking rate in isolated word recognition does not bring any change, for better or for worse, to the recognition performance. This could be due to the fact the ROS is not a significant factor in recognizing isolated words in clean speech. That is, the DBNs can match themselves to isolated-word utterances equally well whether it is a fast or a slow utterance. It is possible that this approach to using ROS in ASR would have a better impact if used in continuous speech recognition, where there is a continuous stream of words, not separated by silence. In this case, an ROS measure may then make a difference in matching the DBN word models to the individual component words. Other possible explanations include there being an unreliable estimate of ROS or there not being a linear relation between the MFCCs and the estimated ROS.

### Pitch

Experiments using estimated pitch, as explained in Section 5.2.1 on page 71, are presented in Table 6.7. The pitch was calculated using the SIFT algorithm with smoothing, as explained in Section 6.2.1.

The results with continuous pitch in Table 6.7 do not give any statistical improvement over that of the baseline (HMM) approach; furthermore, in the case of the system "Pitch ($A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$)", there is a significant drop in performance if we hide the pitch instead of observing it. This is in contrast to using (time-dependent) discrete pitch in Table 6.4 on page 81, where not only was there

| | # Param. | Mix. | Obs. Pitch | Hid. Pitch |
|---|---|---|---|---|
| Baseline | 32k | 6 | 4.3% | |
| Pitch Baseline | 48k | 6 | 4.1% | 4.4% |
| Pitch ($X_n \perp\!\!\!\perp A_n \mid Q_n$) | 32k | 6 | 4.6% | 4.4% |
| Pitch ($A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$) | 48k | 6 | 4.0% | 4.4% |

Table 6.7: Word error rate for the Baseline (non-Pitch) DBN and for the three Pitch DBNs on the PhoneBook test set. Results are presented as in Table 6.6. With 90% confidence, there is a statistical different between the observed and hidden cases of the system "Pitch ($A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$)." With respect to the Baseline, all systems perform statistically the same.

| | # Param. | Mix. | Obs. Energy | Hid. Energy |
|---|---|---|---|---|
| Baseline | 32k | 6 | 4.3% | |
| Energy Baseline | 48k | 6 | 5.1% | 4.4% |
| Energy ($X_n \perp\!\!\!\perp A_n \mid Q_n$) | 32k | 6 | 5.4% | 4.3% |
| Energy ($A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$) | 48k | 6 | 4.1% | 4.4% |

Table 6.8: Word error rate for the Baseline (non-Energy) DBN and for the three Energy DBNs on the PhoneBook test set. Results are presented as in Table 6.6. The "Energy Baseline" and "Energy ($X_n \perp\!\!\!\perp A_n \mid Q_n$)" systems perform statistically worse than the baseline (with 99% confidence). "Energy Baseline" and "Energy ($X_n \perp\!\!\!\perp A_n \mid Q_n$)" both perform statistically better with $A_n$ hidden than with it observed (with 99% and 95% confidence, respectively); however, even with hidden energy, they are both statistically equivalent to the baseline.

a significant improvement in performance if we observe the pitch instead of hiding it, but there was also a significant improvement of the best discrete pitch system over the baseline system. This suggests that there is a need to have a time-dependency between the hidden, continuous pitch variables; as discussed in Section 2.4.3, this is not computationally feasible in the current framework, and is therefore recommended as an area of future research in Section 7.2.5.

**Energy**

Experiments using energy, as explained in Section 5.2.3 on page 72, are presented in Table 6.8. The energy was calculated with windowed frames of speech, of the same length and frame rate as the MFCCs. It was then further transformed by taking the logarithm of this short-term energy. See equation (5.10) on page 72.

Unlike with ROS and pitch, incorporating the energy in a state-dependent way ("Energy Baseline" and "Energy ($X_n \perp\!\!\!\perp A_n \mid Q_n$)") increases the error rate significantly with respect to the baseline system (from 4.3% to 5.1% and 5.4%, respectively). However, as with using pitch, the error rate remains statistically the same as the baseline when modeling the $X_n$ dependent upon $A_n$, with $A_n$ being state-independent. Unlike with pitch, the energy must be modeled independently of the state; modeling the energy dependent upon the state $Q_n$ increases the error rate if $X_n$ is conditioned itself by the energy. Nevertheless the systems with $A_n$ dependent upon $Q_n$ perform significantly better with hidden energy than with observed energy. Thus, by hiding and, hence, marginalizing out the energy and using its inferred distribution, these systems "recover" the baseline system's performance of 4.3%. While not useful here in the context of isolated word recognition in clean speech, the energy will be shown to be of use in noisy speech in Section 6.3.

## 6.3 Spontaneous, Noisy Speech Recognition

I have done (in collaboration with Mathew Magimai-Doss for the HMM/ANNs) my experimentation using the Numbers data (Cole *et al.*, 1994), which contains free format numbers spoken over the telephone. As done in certain other work (Mirghafori and Morgan, 1998), I used 3233 utterances, covering 92 minutes, from the database for training; this is a subset of the training list suggested by the database. I used a subset of 1206 utterances of the suggested development set for evaluating the performance of the system. The validation set contained 357 utterances; it was used for training the ANNs but not used with the DBNs as no tuning was done with the DBNs. As done in certain other work with the Numbers database at IDIAP, there were thirty words (including silence) in the lexicon, composed of 27 monophone models (the three stops in the lexicon, /t/, /d/, and '/k/' were each represented as two monophones: for the closure and release of each respective stop). In the case of the DBNS, the silence and closure models were each modeled with one state models, the releases were modeled with two state models, and all of the other monophones were modeled with three state models, as done in related work at IDIAP. In the case of the HMM/ANNs, every monophone was modeled, in recognition, with a minimum state duration of three frames, reflecting the constraint of the decoder used that all states have the same minimum state duration; in training the HMM/ANNs, the minimum state duration follows that of the DBNs, as the training of both types of models is based on the same (initial) segmentation of Numbers used in-house at IDIAP (this segmentation, in the case of the DBNs is only used to determine the initial parameters to be used in expectation-maximization training). With the DBNs, except if noted, 12 components were used in the (conditional) Gaussian mixture models (GMMs); similarly to the choice of number of mixture components in isolated word recognition in Section 6.2.3, this number of components was chosen so as to provide reasonable recognition results while not being too computationally complex. To do recognition, the Viterbi score (see Section 2.1.3 on page 15) was used by calculating the single most likely instantiation of all of the discrete variables in the models. In the case of the DBNs the one variable that was the exception to this was the discrete mixture component $J_n$: a sum (instead of a max) was always over it. In doing decoding with the Viterbi algorithm, a uniform distribution was used over all of the word models.

For all features I used a frame rate of 12.5 ms. The standard features used are 13 perceptual linear prediction (PLP) coefficients in addition to their approximate first and second derivatives; the analysis window size was 25 ms. These features included the $0^{th}$ coefficient; future work would involve removing this coefficient (but not its first or second derivative), so as to, first, resemble more the work done with continuous features in isolated word recognition in Section 6.2.3 (which did not use the $0^{th}$ MFCC but did use its first derivative) and to, second, investigate the effects of not having the $0^{th}$, energy-like coefficient as a standard feature when the short-term energy is an auxiliary feature (to compare with the "$A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$" system presented in Table 6.11). Some of these results have appeared in Stephenson *et al.* (2003).

### 6.3.1 Gaussians in DBNs

Results using the auxiliary features of pitch, ROS, and energy in DBNs are given in Tables 6.9, 6.10, & 6.11, respectively, along with the baseline performance. BASELINE uses the DBN in Figure 5.5, thus being equivalent to a standard HMM. "$X_n, A_n$" makes no statistical assumptions between $X_n$, $A_n$, and $Q_n$ (using the DBN in Figure 5.4 as is); "$A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$" assumes $A_n$ and $Q_n$ are independent of each other, given the observation $X_n = x_n$ (using the DBN in Figure 5.4 with the $Q_n \rightarrow A_n$ connections removed); and "$X_n \perp\!\!\!\perp A_n \,|\, Q_n$" assumes that $X_n$ and $A_n$ are conditionally independent of each other, given $Q_n$ (using the DBN in Figure 5.4 with the $A_n \rightarrow X_n$ connections removed), thus resembling an approach to incorporating auxiliary information where it is treated as an emission of the state that is not directly correlated with the other emissions and

| | | | NOISE TYPE | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | CLEAN | CAR SNR (dB) | | LYNX SNR (dB) | | FACTORY SNR (dB) | |
| Topology | $A_n$ | Param. | | 0 | 12 | 0 | 12 | 0 | 12 |
| BASELINE ($J = 12$) | | 66 k | 9.3 | 23.0 | 10.1 | 65.6 | 30.7 | 76.9 | 30.7 |
| BASELINE ($J = 18$) | | 100 k | 8.9 | **20.7** | 9.9 | 68.1 | 33.9 | 79.1 | 34.9 |
| $X_n, A_n$ | O | 99 k | 8.9 | 32.5 | 13.9 | 74.2 | 42.2 | 79.5 | 29.7 |
| $X_n, A_n$ | H | 99 k | 8.9 | 27.3 | 11.3 | 66.8 | 31.8 | 77.1 | 29.4 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | O | 99 k | 9.6 | **20.6** | 9.9 | 63.6 | **26.6** | 79.3 | 31.1 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | H | 99 k | 9.4 | **20.2** | 10.0 | **62.7** | **26.6** | 76.3 | †**25.1** |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | O | 67 k | 9.4 | 28.6 | 14.5 | 72.3 | 39.7 | 76.5 | 28.9 |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | H | 67 k | 9.2 | 23.5 | 11.0 | 66.1 | 30.9 | **74.8** | 29.7 |

Table 6.9: **Pitch** DBN word error rate on the OGI Numbers development set. $X_n$ = PLP features, $A_n$ = Pitch feature, $Q_n$ = discrete state, O = "Observed", H = "Hidden." The number of free parameters is also given for each system. Results which are marked with **bold** are significantly better (with 99% confidence) than the corresponding 12-mixture component BASELINE result in the same column. Results with hidden (or observed) $A_n$ which are marked with † (only one occurrence in this table) are significantly better (with 99% confidence) than both the corresponding 12-mixture component BASELINE ($J = 12$) result and the corresponding result with the same DBN but using observed (or hidden) $A_n$ immediately above (or below) it in the same column. Unless otherwise indicated, there are 12 mixture components for all systems. All WERs are in percentage terms.

where the auxiliary information is not even part of a GMM. Each system models $X_n$ with 12 mixture components in the Gaussians/conditional Gaussians (except for the second Baseline with 100 $k$ parameters, which uses 18 mixture components and is given to show that increasing the number of mixture components is typically of no benefit for increasing the model's robustness to noise). Under the '$A_n$' column, systems with auxiliary information that have observed $A_n$ in recognition are marked with an 'O' while those with hidden $A_n$ in recognition are marked with 'H'; in both cases, $A_n$ was observed during training. Auxiliary systems using $X_n, A_n$ (meaning there are no assumptions on $A_n$) or $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ have conditional Gaussians while all other systems have Gaussians. Results are reported on clean data (SNR $= \infty$) as well as for SNRs of 0 dB and 12 dB for three types of added noises.

Better results would be expected with further refinements, such as context-dependent submodels, word insertion penalties, a trained language model, etc (the language model used is described in Section 5.3.5 on page 75). However, my goal was not to have the best performances possible, as I wanted to concentrate on new modeling techniques and not spend all my time implementing the fine details of already established ones. So my goal was to have a reasonable training and recognition methodology that gives acceptable performances and to apply the same methodology (as described in Chapters 5 & 6) to all of the systems within a given set of experiments.

### Clean speech

Being trained on clean speech, all of the systems should perform their best in clean conditions. Compared to the baseline, HMM equivalent performance of 9.3 word-error-rate (WER), the systems with auxiliary information do not perform significantly better with pitch, ROS, and energy; this confirms the studies with continuous auxiliary information with isolated word recognition in Section 6.2.3. Moreover, using energy as an auxiliary variable in the "$A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$" topology actually degrades the performance significantly (with 99% confidence), with respect to the base-

| | | | NOISE TYPE | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | CLEAN | CAR SNR (dB) | | LYNX SNR (dB) | | FACTORY SNR (dB) | |
| Topology | $A_n$ | Param. | | 0 | 12 | 0 | 12 | 0 | 12 |
| BASELINE ($J = 12$) | | 66 k | 9.3 | 23.0 | 10.1 | 65.6 | 30.7 | 76.9 | 30.7 |
| BASELINE ($J = 18$) | | 100 k | 8.9 | **20.7** | 9.9 | 68.1 | 33.9 | 79.1 | 34.9 |
| $X_n, A_n$ | O | 99 k | 9.3 | 33.1 | 12.9 | 66.6 | 28.8 | 81.3 | 30.6 |
| $X_n, A_n$ | H | 99 k | 8.8 | 22.9 | 10.4 | 66.0 | 30.6 | 78.3 | 31.5 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | O | 99 k | 9.3 | 25.0 | 10.6 | 65.6 | $^\dagger$**26.3** | 79.0 | **28.2** |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | H | 99 k | 9.0 | 21.2 | 10.0 | 65.3 | 28.6 | 76.6 | **28.5** |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | O | 67 k | 9.3 | 26.2 | 11.3 | 65.8 | 31.3 | 80.3 | 31.9 |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | H | 67 k | 9.4 | 23.5 | 10.2 | 66.3 | 31.2 | 77.4 | 30.7 |

Table 6.10: **ROS** DBN word error rate on the OGI Numbers development set, with a similar setup to that of Table 6.9 except that $A_n$ = ROS. $X_n$ = PLP features, $Q_n$ = discrete state, O = "Observed", H = "Hidden."

| | | | NOISE TYPE | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | CLEAN | CAR SNR (dB) | | LYNX SNR (dB) | | FACTORY SNR (dB) | |
| Topology | $A_n$ | Param. | | 0 | 12 | 0 | 12 | 0 | 12 |
| BASELINE ($J = 12$) | | 66 k | 9.3 | 23.0 | 10.1 | 65.6 | 30.7 | 76.9 | 30.7 |
| BASELINE ($J = 18$) | | 100 k | 8.9 | **20.7** | 9.9 | 68.1 | 33.9 | 79.1 | 34.9 |
| $X_n, A_n$ | O | 99 k | 10.2 | 21.9 | 13.9 | **62.4** | **24.1** | **66.9** | **17.4** |
| $X_n, A_n$ | H | 99 k | 10.0 | 23.4 | 12.9 | $^\dagger$**47.9** | $^\dagger$**17.9** | $^\dagger$**62.2** | $^\dagger$**15.5** |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | O | 99 k | 11.3 | $^\dagger$**15.9** | 12.9 | **41.0** | **16.0** | **69.3** | **15.4** |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | H | 99 k | 11.0 | 28.2 | 16.0 | **39.5** | $^\dagger$**14.2** | $^\dagger$**64.2** | **14.9** |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | O | 67 k | 9.0 | 30.7 | 13.0 | 70.1 | 37.0 | 75.8 | 35.1 |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | H | 67 k | 9.0 | 23.7 | 11.3 | 65.1 | 34.7 | 79.9 | 33.2 |

Table 6.11: **Energy** DBN word error rate on the OGI Numbers development set, with a similar setup to that of Table 6.9 except that $A_n$ = energy. $X_n$ = PLP features, $Q_n$ = discrete state, O = "Observed", H = "Hidden."

line's performance. Hence, in these contexts, continuous auxiliary information can not help with recognition in clean environments. This is in contrast to the discrete auxiliary information of Section 6.2.1 which helped recognition; the difference may lie in that the discrete auxiliary information was time-dependent information whereas continuous auxiliary information is being modeled here as conditionally time-independent.

#### Noisy speech

Using systems that were trained in clean speech, I tested their ability to handle recognition in noisy environments. Three different types of additive noise from Varga *et al.* (1992) were added to the speech signal in separate tests in recognition: one set of tests was with stationary CAR noise; another set of tests was with stationary LYNX helicopter noise; and a third set of tests was done with non-stationary FACTORY noise. These noises were added to the signal so as to have a certain SNR ratio; I present results showing the results of adding these noises both with an SNR of 12 dB and with an SNR of 0 dB.

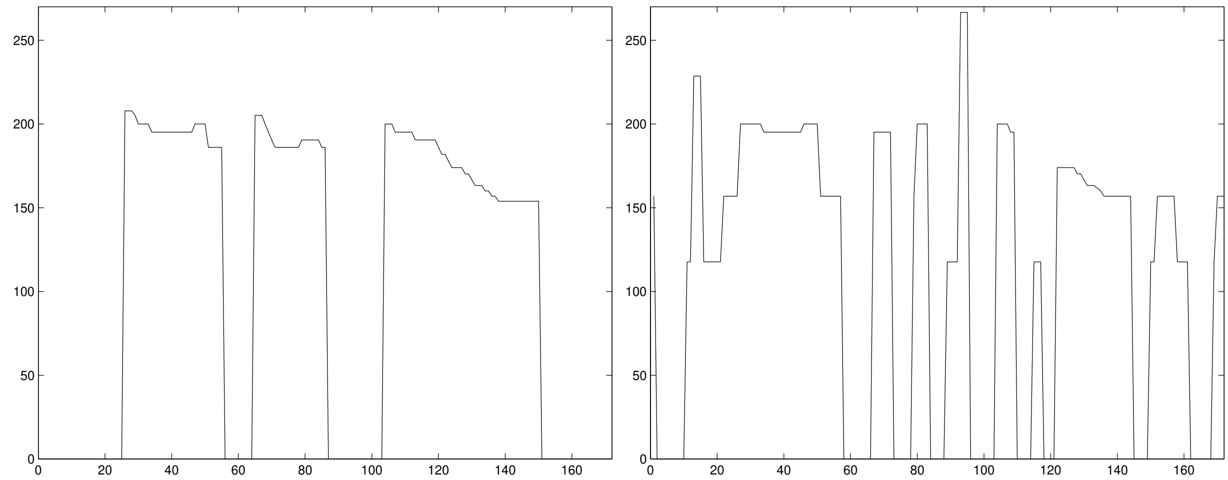


Figure 6.1: Pitch observations for a sample utterance. Plot of pitch observations in clean conditions on left. Plot of pitch observations in noise (LYNX noise added at 0 dB SNR) on right. The "error" between these two is representative of the estimates with this type of noise.

Table 6.10 shows that auxiliary information of ROS only performs marginally, though significantly, better than the baseline system in certain noise conditions. The improvement came in the system using ROS as a state-independent conditioning variable to $X_n$. This shows the ability to use a continuous ROS information to change the distributions of $X_n$, thus furthering the work done with discrete ROS in acoustic modeling, as discussed in Section 5.2.2 on page 71.

As shown in Table 6.9, the auxiliary information of pitch, provides a better example than ROS, in certain noise conditions, of the benefits of using auxiliary information as I have outlined here. The systems with $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$ perform significantly better than the baseline system in certain noise conditions of CAR, LYNX, and FACTORY. This is the system where the value of $A_n$ conditions the distribution of $X_n$ and where $A_n$ is modeled independently of $Q_n$. Furthermore, hiding the pitch in recognition, having used it in training, can be of potential benefit, as this system with state-independent pitch performs significantly better without pitch (25.1 WER) than with pitch estimates (31.1 WER) in the case of 12 dB SNR FACTORY noise; this indicates that, at least in this case, the pitch estimates were unreliable in noise (see Figure 6.1).

Finally, Table 6.11 shows that the auxiliary information of the logarithm of the short-term energy provides a big reduction in WER, with respect to the performance of the baseline, in many of the noisy conditions. All of the performances of systems with $A_n$ that did significantly better than the baseline used conditional Gaussians, thus showing the utility of having energy condition the distributions of $X_n$. While the $X_n, A_n$ system (with a dependency between $Q_n$ and $A_n$) did perform better than the baseline as well, the $A_n \perp\!\!\!\perp Q_n \,|\, X_n = x_n$ (with no dependency between $Q_n$ and $A_n$) tend to perform even better (at least in the case of LYNX noise), thus suggesting that energy is better modeled independently of the state. There is often a significant improvement in performance when hiding $A_n$ in recognition, as indicated in the table, thus indicating that sometimes it is better to let the DBN infer $A_n$'s distribution instead of providing it.

### 6.3.2 HMM/ANNs

Results using HMM/ANNs and the auxiliary features are given in Tables 6.12, 6.13, & 6.14, along with the baseline performance. The various systems used for HMM/ANN hybrids (note that no

| ANN Type | $A_n$ | Param. | CLEAN | CAR SNR (dB) 0 | CAR SNR (dB) 12 | LYNX SNR (dB) 0 | LYNX SNR (dB) 12 | FACTORY SNR (dB) 0 | FACTORY SNR (dB) 12 |
|---|---|---|---|---|---|---|---|---|---|
| BASELINE | | 455 k | 12.1 | 23.1 | 13.7 | 84.6 | 29.3 | 81.0 | 29.3 |
| $X_n, A_n$ | O | 465 k | 11.1 | 21.4 | 13.4 | **81.3** | 30.3 | 82.8 | 27.4 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | O | 465 k | 10.6 | 22.5 | 12.5 | †**79.0** | †**24.6** | 84.2 | 27.9 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | H | 465 k | 13.7 | 25.5 | 14.9 | 91.0 | 28.7 | 93.1 | 32.8 |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | O | 455 k | 13.1 | 27.2 | 16.1 | **79.6** | 28.4 | 80.2 | 30.5 |

Table 6.12: **Pitch** ANN word error rate on the OGI Numbers development set. $X_n$ = PLP features, $A_n$ = Pitch feature, $Q_n$ = discrete state, O = "Observed", H = "Hidden." The number of parameters for each system is given (in the case of "$A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$", which has multiple ANNs, this is the sum of the parameters in all of its ANNs). Under the '$A_n$' column, systems with auxiliary information that have observed $A_n$ in recognition are marked with an 'O' while those with hidden $A_n$ in recognition are marked with 'H'; in both cases, $A_n$ was observed during training. Hidden auxiliary information is only performed on the system with $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ (which has three discrete values for $A_n$) for theoretical reasons. Results are reported on clean data (SNR $= \infty$) as well as for SNRs of 0 and 12 for three types of added noises. Results which are marked with **bold** are significantly better than the corresponding BASELINE result in the same column. Results with hidden (or observed) $A_n$ which are marked with † are significantly better than both the corresponding BASELINE result and the corresponding result with the same ANN but using observed (or hidden) $A_n$ immediately above (or below) it in the same column. All WERs are in percentage terms.

DBNs were involved with the ANNs) were discussed in 4.4.2 and illustrated in Figure 4.7 on page 51. The BASELINE system uses inputs only of the standard features $X_n$ with a time window, using equation (2.47) to estimate its scaled likelihoods. The $X_n, A_n$ systems uses an augmented feature vectors of the two sets of features into a standard ANN, using equation (4.15) to calculate its scaled likelihoods. The $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ systems use multiple ANNs, one for each discrete value of $A_n$, using equation (4.16) to estimate its scaled likelihoods. Finally, $X_n \perp\!\!\!\perp A_n \mid Q_n$ systems separate $A_n$'s inputs from $X_n$'s hidden layer, using equation 4.17 to estimate its scaled likelihoods.

The ANNs used the same 39-dimension PLP features for the DBNs in Section 6.3.1. They compute the scaled likelihoods using single state monophone models for the same monophones explained earlier in Section 6.3. During the training phase, it is only the ANNs which are trained (using back propagation with cross-validation over the validation set), which is done using the previously supplied segmentation. The HMMs are not trained themselves (their emission distributions are handled by the ANNs and their transition probabilities are set to a uniform distribution); their role is to take these scaled likelihoods for doing Viterbi decoding.

#### Clean speech

As with the DBNs, all of the HMM/ANN systems were trained on clean speech and have their best performance when presented with clean speech. With reference to the baseline HMM/ANN, which uses only standard features, only one of the systems with observed auxiliary information presented in the tables perform better. The attempts to use hidden (discrete) auxiliary information, done as explained in Section 4.4.3, in each table did not help recognition The system that did well used energy as a conditioning variable, thus showing the potential for using ANNs tailored to different values of a discrete $A_n$.

| ANN Type | $A_n$ | Param. | CLEAN | CAR SNR (dB) 0 | CAR SNR (dB) 12 | LYNX SNR (dB) 0 | LYNX SNR (dB) 12 | FACTORY SNR (dB) 0 | FACTORY SNR (dB) 12 |
|---|---|---|---|---|---|---|---|---|---|
| BASELINE | | 455 k | 12.1 | 23.1 | 13.7 | 84.6 | 29.3 | 81.0 | 29.3 |
| $X_n, A_n$ | O | 465 k | 11.3 | 21.6 | 12.5 | **66.3** | 27.4 | 85.0 | 30.3 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | O | 465 k | 12.2 | 28.1 | 14.7 | 86.1 | 30.9 | 89.0 | 36.3 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | H | 465 k | 14.4 | 26.1 | 16.8 | 91.8 | 36.4 | 94.3 | 39.8 |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | O | 455 k | 13.3 | 24.7 | 15.4 | **78.5** | 27.6 | 84.9 | 31.3 |

Table 6.13: **ROS** ANN word error rate on the OGI Numbers development set, with a similar setup to that of Table 6.12 except that $A_n$ = ROS. $X_n$ = PLP features, $Q_n$ = discrete state, O = "Observed", H = "Hidden."

| ANN Type | $A_n$ | Param. | CLEAN | CAR SNR (dB) 0 | CAR SNR (dB) 12 | LYNX SNR (dB) 0 | LYNX SNR (dB) 12 | FACTORY SNR (dB) 0 | FACTORY SNR (dB) 12 |
|---|---|---|---|---|---|---|---|---|---|
| BASELINE | | 455 k | 12.1 | 23.1 | 13.7 | 84.6 | 29.3 | 81.0 | 29.3 |
| $X_n, A_n$ | O | 465 k | 10.6 | **19.9** | **11.8** | **69.5** | **23.6** | 84.3 | 31.7 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | O | 465 k | $^{\dagger}$**10.2** | 22.9 | $^{\dagger}$**12.1** | 84.6 | 31.2 | 87.4 | 33.4 |
| $A_n \perp\!\!\!\perp Q_n \mid X_n = x_n$ | H | 465 k | 13.3 | 23.7 | 15.4 | 85.6 | 26.7 | 87.6 | 31.7 |
| $X_n \perp\!\!\!\perp A_n \mid Q_n$ | O | 455 k | 12.7 | 30.5 | 15.7 | 87.6 | 31.3 | 82.0 | 30.7 |

Table 6.14: **Energy** ANN word error rate on the OGI Numbers development set, with a similar setup to that of Table 6.12 except that $A_n$ = energy. $X_n$ = PLP features, $Q_n$ = discrete state, O = "Observed", H = "Hidden."

#### Noisy speech

Auxiliary information in HMM/ANNs has mixed results with noisy speech. None of the systems with auxiliary information performs statistically better than the baseline in FACTORY noise. Pitch was of some benefit, but only under LYNX noise conditions. As with the DBNs, the best auxiliary information of the three tested in spontaneous, noisy speech was energy. Of the various systems tested with energy, the $X_n, A_n$ system typically did well in noise, at least in CAR and LYNX noise, where it performed significantly better than the baseline system.

## 6.4 Conclusion

The most important conclusions from this chapter are that auxiliary information appears to be best implemented when it is time-dependent upon its previous value; that auxiliary information often needs to be left hidden in recognition; and that the best source, of those test, of auxiliary information in noisy speech is the short-term energy.

With the discretized auxiliary information, I showed the ability of the DBNs to have the auxiliary information hidden during recognition, having used it during training. At least with the case of pitch in isolated word recognition, hiding it with a time-dependent $A_n$ was effective (Table 6.4) whereas hiding it when it was a time-independent $A_n$ was not effective (Table 6.7) . This implies that it is important to have $A_n$ time-dependent if it is going to be effectively hidden in recognition; however, more experimentation needs to be done as those two experiments where done with different representations of $A_n$ (discrete versus continuous, respectively), and this comparison involves only the auxiliary information of pitch.

With hidden auxiliary information, I showed how it can be effective in modeling various types of speech when it is hidden during recognition. This was shown not only in the case of discrete pitch but also in the case of noisy speech and, to a lesser extent, in the case of clean spontaneous speech. I note that, in certain cases, such as with rate-of-speech and continuous pitch on isolated words, it was not helpful to hide it in recognition.

Finally, HMM/ANNs show some ability to be able to incorporate auxiliary information. The auxiliary information can easily be treated as standard information by appending it to the standard feature vector. However, to treat it as conditioning information, it had to be discretized. It was only when the auxiliary information was discretized were the HMM/ANN systems able to treat the auxiliary information as hidden by summing over the likelihoods when using the different possible instantiations of the auxiliary information. I was not able to show the effect of hiding continuous auxiliary information in HMM/ANNs due to the limitations of ANNs of having continuous input features be hidden. Overall, the performance of HMM/ANNs with auxiliary information in noisy speech trails that of the DBNs.

# Chapter 7

# Conclusion

## 7.1 Review

The single, most important contribution of this thesis has been to demonstrate the need for a time-dependent, auxiliary variable that conditions the standard features and which can be hidden, if noisy or missing, in recognition. I have laid this in the context of dynamic Bayesian networks (DBNs) as they provide the framework to do probabilistic inference in a wide variety of conditions. As examples, I showed both discrete auxiliary information and continuous auxiliary information.

In Chapter 1 I gave an overview of automatic speech recognition (ASR) in terms of its three principal components: feature extraction, acoustic modeling, and language modeling. I explained how much of ASR research is concerned with the adaptation of these aspects to the wide variety of ASR environments. It is the varying acoustics in particular that I have investigated in this thesis.

In Chapter 2 I presented a discussion on times-series modeling for ASR, as related to this thesis. I reviewed hidden Markov model (HMM) theory, which is of prime importance in many time (or sequence) modeling problems, and discussed its use in ASR; this included a discussion of the expectation-maximization (EM) algorithm. I also reviewed dynamic Bayesian network (DBN) theory, which is a newer development since HMMs that generalizes the HMM framework; this included my own presentation of an important complexity issue, explained from the viewpoint of "$d$-separation."

After presenting DBNs in Chapter 2, I showed in Chapter 3 how to do inference in mixed DBNs. This included contributions, from my own experience, as to how the DBN inference algorithm could be implemented, in the context of ASR, so as to increase the computational efficiency. Having implemented the theory without any optimizations would have resulted in very slow inference.

In Chapter 4 I presented a detailed discussion on auxiliary information. The idea of an auxiliary variable was not original to this thesis. For example, Zweig (1998, Chapter 7) investigated the use of an auxiliary variable which was both discrete and latent (that is, while given certain initial parameters, it was not given any observations during training). So, my contribution to using auxiliary information in ASR, as explained in this and other chapters, involves the following:

- using real, "high-level" data in training the auxiliary variable instead of training it in an unsupervised manner;
- arguing the need for time-dependent auxiliary information;
- showing the effect and, sometimes, the benefit of hiding the auxiliary variable's value during the recognition process;

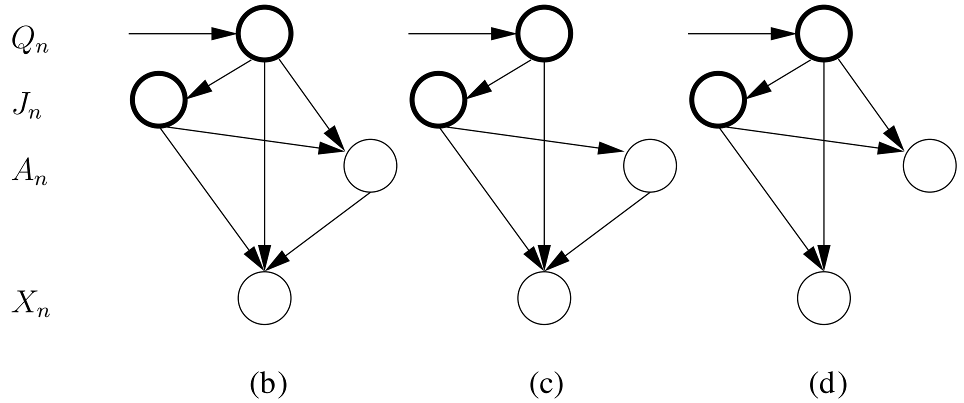


Figure 7.1: BNs for ASR with auxiliary variable $A_n$ using the same mixture component $J_n$ as the standard variable $X_n$. Compare with Figure 4.5 on page 49.

- using a variety of continuous-valued auxiliary information (which can be state-dependent or state-independent) to condition the distributions of the continuous valued standard features (certain types of continuous-valued auxiliary information had already been implemented as conditioning information in a state-independent way in (Fujinaga *et al.*, 2001)).

In Chapter 5, I discussed the various DBNs as well as the various types of auxiliary information that I have investigated in the course of this work. Here I showed the types of information that my auxiliary variables represented: speaking rate, pitch, energy, articulator positions, and graphemes. This chapter included what, to my knowledge, is the first application of factorial HMMs (a type of DBN) to ASR for the case where the two Markov chains represent two distinct different processes and where these processes have defined, precise meanings; this factorial HMM built upon the work of Logan and Moreno (1998), whose two Markov chains were different parameterizations of the same process.

In Chapter 6, I presented the experimental studies which take into account the DBNs of Chapter 2 and the auxiliary information of Chapter 4. In addition to showing the experimental effect of having auxiliary variables (discrete and continuous) trained on real data and hidden in recognition, a particular contribution of this chapter is that I have applied auxiliary variables to spontaneous speech in noisy conditions.

Auxiliary information makes acoustic modeling in ASR more robust to noise in many cases with DBNs and, to a lesser extent, with HMM/ANNs as well. In DBNs with continuous auxiliary information $A_n$, it is important to use $A_n$ to "shift" the conditional Gaussians that model the standard information $X_n$; for a small amount of additional computation, this allows the modeling of the correlation between $A_n$ and $X_n$ and some of the correlation within $X_n$ itself that is captured via $A_n$. Hiding the $A_n$ in the conditional Gaussian sometimes makes the DBNs even more robust–though this is done through even more computational cost. Finally, the logarithm of the short-term energy proves very promising as an auxiliary variable while pitch also shows some potential.

## 7.2 Future Directions

There are still research paths that can be followed in investigating auxiliary information in DBN-based ASR. Here I present the following: modeling of the auxiliary variable, choice(s) of auxiliary variable, latency of the auxiliary variable, missing feature ASR, and use of approximate inference.

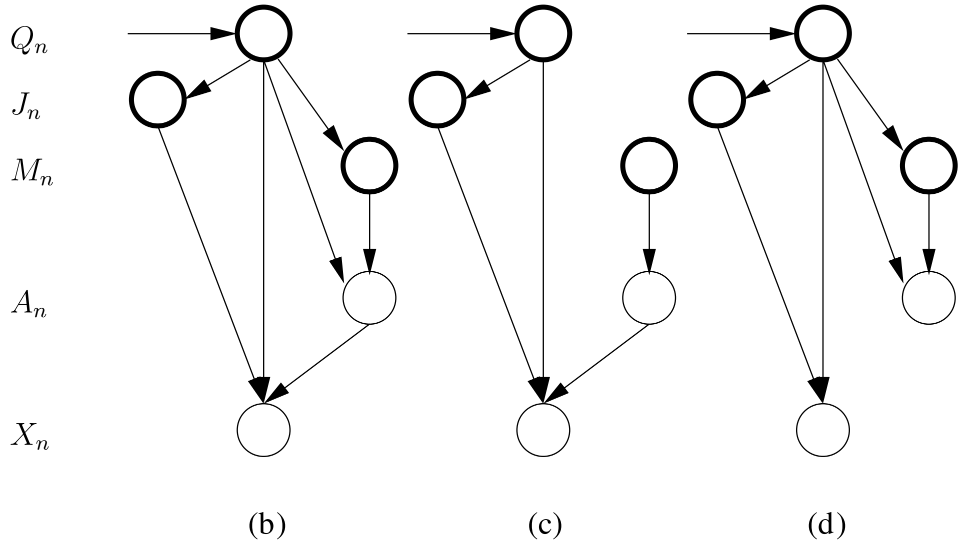


Figure 7.2: BNs for ASR with auxiliary variable $A_n$ using its own mixture component $M_n$. Compare with the respective BNs (b)-(d) in Figure 4.5 on page 49.

### 7.2.1 Modeling of the auxiliary variable

All of the systems presented in this thesis modeled the continuous-valued auxiliary variable $A_n$ using a single Gaussian (in the case of state-dependent $A_n$, it was a single Gaussian for each value of the state $Q_n$). As GMMs are known to better model the standard features $X_n$, they may be of help as well in modeling $A_n$. In applying GMMs to $A_n$, the variables $X_n$ and $A_n$ could share the same mixture component variable, as in Figure 7.1. The potential advantage of $A_n$ and $X_n$ having the same mixture component $J_n$ is that it allows additional correlation between $A_n$ and $X_n$ to be modeled via $J_n$. Hence, in the case of Figures 7.1 (b) & (c), the connected $A_n$ and $J_n$ can together be viewed as a hybrid continuous/discrete mixture component variable (see the discussion on "continuous mixture" component variable in Section 4.4.1 on page 49). However, it may be desirable to model $A_n$ with a different number of mixture components than what $X_n$ is modeled with. So, alternatively, $A_n$ can have its own mixture component variable, as in Figure 7.2. For example, we may want to model an auxiliary variable 'pitch' with only two Gaussians, such as $\mathcal{N}(0, 0)$ and $\mathcal{N}(200, 2500)$; that is, there is a single Gaussian representing a pitch only of zero (i.e., voiceless) and a Gaussian representing voiced regions (where there might be a mean of 200 Hz, with a standard deviation of of $\sqrt{2500}\ Hz = 50\ Hz$). The inference algorithm of Lauritzen and Jensen (2001) can handle Gaussians with a covariance of 0.

The systems with discrete-valued $A_n$ provided some of the most interesting results regarding $A_n$ that is hidden during recognition. These topologies with discrete $A_n$ also used the time-dependency of $A_{n-1} \rightarrow A_n$. As the mixed DBNs did not improve as much as the discrete DBNs did when $A_n$ was hidden, this could be due to not having the time-dependency with a continuous $A_n$. However, I was restricted from having a time-dependent, continuous, hidden $A_n$, as explained in Section 2.4.3 on pages 23 ff. Therefore, one of the other research directions, as explained below, is using approximate inference.

A final approach to modeling $A_n$ is a hybrid between that of modeling it independent of $Q_n$ and that of modeling it dependent upon $Q_n$. That is, $A_n$ is dependent on broad classes of $Q_n$. So, given that $Q_n$ has $K$ classes, we define a discrete function $g(Q_n)$ which has $H$ "broad" classes, where $H < K$. We then model the distribution of $A_n$ as being dependent upon these broad classes and,

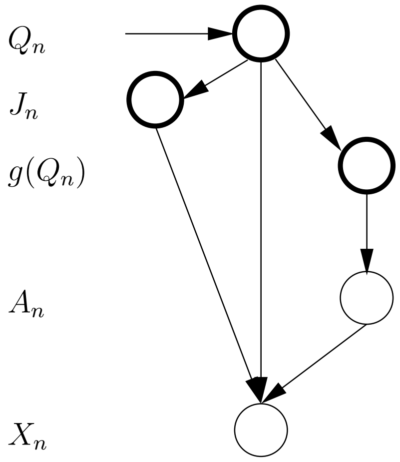


Figure 7.3: BN for ASR where there are broad classes of $Q_n$, $g(Q_n)$, that condition $A_n$.

hence, indirectly dependent upon $Q_n$ (See Figure 7.3):

$$p(A_n|g(Q_n)). \tag{7.1}$$

This falls within the framework of the equivalence classes proposed in Zweig (1998) and has already been experimented with, in the case of different types of energy, in Escofet Carmona and Stephenson (2003). Note that having state-independent $A_n$ is equivalent to having one broad class ($H = 1$). To illustrate why using broad classes might be advantageous, consider an auxiliary variable of energy. Voiced sounds tend to have more energy than unvoiced sounds. However, individual voiced sounds tend to have roughly the same energy as other voiced sounds. So, we may have more robust modeling of the auxiliary variable if we condition it on appropriate broad classes of $Q_n$.

### 7.2.2 Choice of auxiliary variable(s)

In this thesis I have presented a selection of auxiliary variables of speaking rate, pitch, log energy, articulators, and graphemes. Furthermore, I worked with each individually, as a single auxiliary variable. Further factors can be investigated regarding these auxiliary features already presented. First, they may need to be mapped to another domain (e.g., using the logarithm of the pitch frequency instead of the plain frequency). As the conditional Gaussians assume a linear relationship between the value of $A_n$ and $X_n$, it may be advantageous to map $A_n$ to a space $f(A_n)$ where there is a more linear relationship with $X_n$; how to do this would be a subject of research. Second, we would hope to have even better modeling in using multiple auxiliary variables in the same system. In doing so, we may or may not want to add dependencies between the multiple auxiliary variables themselves (see Figure 7.4).

### 7.2.3 Latency of the auxiliary variable

The systems were trained with maximum likelihood (ML) training. So, the most likely parameters were learned, given the observations $X_{1:N} = x_{1:N}, A_{1:N} = a_{1:N}$. That is, in the context of equation (3.2) on page 34, $\mathbf{e} = \{X_{1:N} = x_{1:N}, A_{1:N} = a_{1:N}\}$, with the maximized parameters $\lambda^i = \lambda^\dagger$. Now, recognition worked well in many cases when $A_n$ was hidden when using parameters $\lambda^\dagger$. So, I suggest that after maximizing the likelihood of the parameters given the observations for $X_n$ and

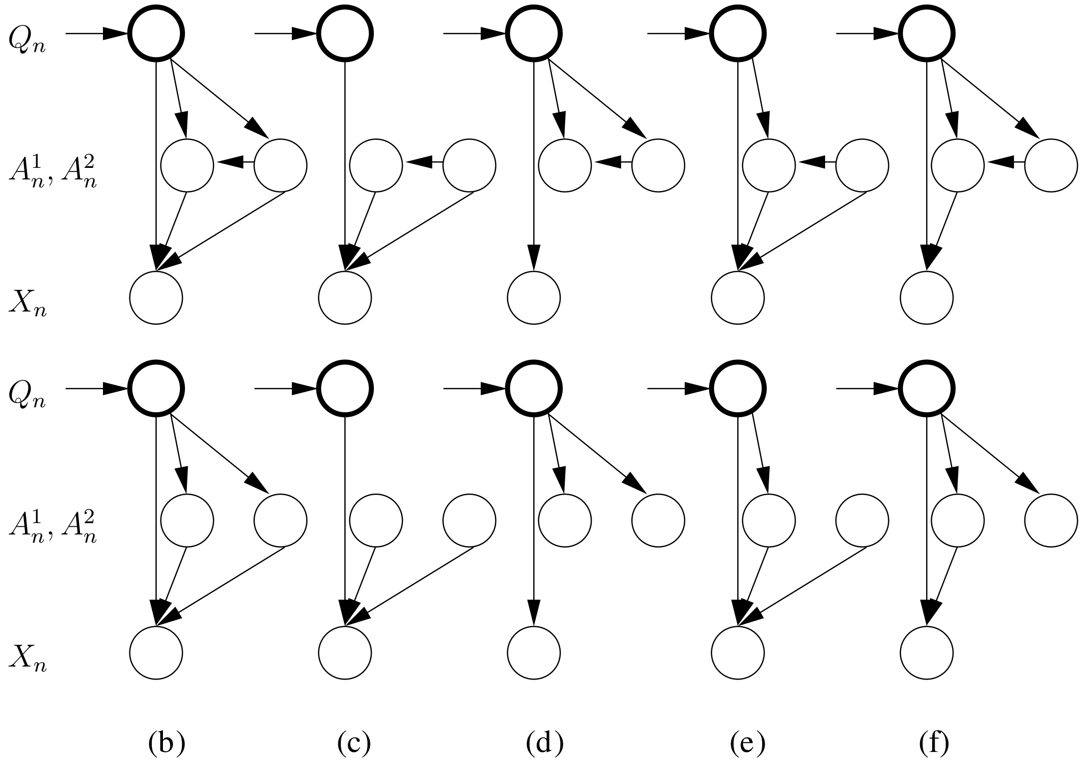


Figure 7.4: Different possible topologies for incorporating two time-independent auxiliary variables $A_n^1$ and $A_n^2$. (b)-(d) are based directly on Figure 4.5 (b)-(d), respectively, on page 49 (with the mixture component variable $J_n$ removed for simplicity). (e)-(f) are hybrids of (c) and (d) in that one of the two auxiliary variables is state-independent (in the case of (e)) or that $X_n$ is conditionally independent of one of the two auxiliary variables (in the case of (f)). The upper row is the cases for the two auxiliary variables directly dependent upon each other; the lower row is the case with this dependency removed.

$A_n$, that a second phase of training begin with $A_n$ being hidden and with $\lambda^\dagger$ as the (new) initial parameters. That is, we can use the trained parameters from the first phase as the initial parameters for a system where $A_n$ is always a hidden variable in both training and recognition so that we only have $\mathbf{e} = \{X_{1:N} = x_{1:N}\}$. After this second phase of training, $A_n$ would then no longer be guaranteed to have a meaning related to the original data $A_{1:N} = a_{1:N}$ used to obtain the parameters $\lambda^\dagger$; hence, I refer to it as a "latent" variable, as it is no longer known what it represents. It is hoped, then, that this system with a latent $A_n$ would perform even better than a system using observed $A_n$ in training but hidden $A_n$ in recognition. Note that, if training with a latent $A_n$, using the original observations for $A_{1:N} = a_{1:N}$ in recognition would introduce a lot of errors.

As an alternative to using a latent $A_n$ in the second stage of training, $A_n$ could be considered as latent from the initial stages of training. Some investigation could then be done into the best way to initialize the regression coefficients $B$ relating $X_n$ to $A_n$. I have not done any investigations in this thesis about how the effects of different initializations; so, if an investigation is made into the initialization of a latent $A_n$, the initialization of the other variables can also be looked at concurrently. This is an extension in the continuous domain of the latent discrete auxiliary variables used in Zweig (1998).

### 7.2.4 Missing Feature ASR

Missing feature ASR (Morris *et al.*, 1998) is concerned with integrating out the noisy features at a time frame $n$. As I also deal with integrating out (supposedly) noisy auxiliary features, there is a connection between this thesis and established missing feature theory. One avenue which can be explored is to only marginalize out the auxiliary feature when there is noise–though determining when there is noise is itself an area of research. Additionally, I have shown that it can be advantageous to train with an auxiliary feature but to leave it hidden in recognition; therefore, I propose that some of the elements in the standard feature may themselves be of use in training but need to be marginalized out in recognition.

### 7.2.5 Approximate Inference

All of the inference used in this thesis used exact inference (or the Viterbi-like simplification of it). However, to use a hidden, time-dependent continuous auxiliary variable (e.g., Figure 2.6) as illustrated in this thesis would require an approximate inference method. One such approximate inference algorithm, which uses stochastic methods, is Rao-Blackwellised Particle Filtering (RBPF) (Murphy, 2002, Section 5.3). With observed variables $X_{1:N} = x_{1:N}$, discrete hidden variables $Q_{1:N}$, and continuous hidden variables $A_{1:N}$, we use sampling upon the discrete hidden variable (i.e., $Q_{1:N}$) so that the only hidden variable is the continuous hidden variables (i.e., $A_{1:N}$). Given the $i^{th}$ set of samples $Q_{1:N} = q_{1:N}$, "exact" inference can be done so as to obtain the distribution of the remaining hidden variables (i.e., $A_{1:N}$). Multiple samples are used along with their computed likelihood so as to get an estimate of the likelihood $p(X_{1:N} = x_{1:N}|A_{1:N}, Q_{1:N})$. Another approximate inference algorithm, which uses deterministic methods, is the Generalized Pseudo Bayesian (GPB) algorithm (Murphy, 2002, Section 4.3.1). This would involve the list of $K$ values on the right side of equation (2.54) on page 26 being replaced by a single distribution:

$$\sum_{\forall q_N} p(Q_N = q_N, Q_{N-1}, X_N|A_{N-1}) \approx p^*(Q_{N-1}, X_N|A_{N-1}), \tag{7.2}$$

where $p^*(Q_{N-1}, X_N|A_{N-1})$ is obtained using moment matching (Murphy, 2002, Appendix 5.2). This allows an approximation of marginalizing out discrete variables in a distribution with conditional Gaussians.

## 7.3 Conclusion

This thesis has shown how auxiliary information $A_n$ is of benefit to increasing the modeling capability of acoustic models by allowing them to use $A_n$ to explain variation in the data. $A_n$ is typically high-level information that is speaker-dependent or utterance-dependent. Since this auxiliary information explains variation in the data, the distributions for the standard features $X_n$ will be more compact and, hopefully, more robust and discriminative. It is often important to model this auxiliary information as conditioning the distributions of $X_n$. Using the auxiliary information as such, we can take advantage of its values for constraining the training of the systems; we can then sometimes leave it hidden in recognition and still have more robust ASR.

This auxiliary information will quite possibly be of even better use if further research can be done into modeling distribution of $A_n$ itself better (this thesis concentrated more on the distribution of $X_n$, in relation to the value of $A_n$). Particularly, modeling $A_n$ with a dependency upon $A_{n-1}$, where $A_n$ and $A_{n-1}$ are hidden would involve an interesting and challenging study. Having this time-dependency with hidden values is of use in cases such as with $A_n$ representing articulatory positions, where the positions have a heavy time-dependency but which are not available in recognition.

# Appendix A

# Graph Theory Terminology

Bayesian networks combine probability theory and graph theory; as readers of this thesis may not be as familiar with graph theory, I provide here some of the basics of graph theory here. For more information, please see Neapolitan (1989, chap. 3 & 7), which provides a good background into the graph theory from a Bayesian networks' perspective.

Brualdi (1992, Section 11.1) defines a graph as (1) a set of **nodes** (also called vertices) and (2) a set of **edges** (also called arcs), each edge being a pair of nodes. This is also called an undirected graph. If the two vertices within each edge $(X_i, X_j)$ are ordered, then the edges have a direction assigned to them (that is, they lead specifically from node $X_i$ to node $X_j$ and not in the other direction); this is called a digraph (or, a **directed** graph). Brualdi (1992) also presents multigraphs and general digraphs, which allow multiple connections between any two nodes in graphs and digraphs, respectively. The undirected graph is, therefore, the special case of a directed graph where for every edge from node $X_i$ to node $X_j$, there is also an edge from $X_j$ to node $X_i$ (See Figure A.1). The following properties of a graph deal with what arrangement of edges appear among the nodes in a graph or digraph.

A **chain** is a series of nodes where each successive node in the chain is connected to the previous node by an edge (regardless of the direction, if any, of the edge). A **path** is a chain with the further constraint for digraphs that each connecting edge in the chain has a directionality going in the same direction as the chain. A **cycle** is a path that starts and ends at the same node. A **simple path** is a path with unique nodes. A **simple cycle** is a cycle where, except for the start/end node, all nodes are unique (See Figure A.2). A **directed acyclic graph**, or **DAG**, is a directed graph that has no

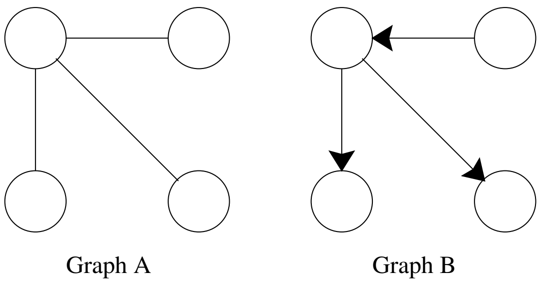


Figure A.1: Directionality. While graphs $A$ and $B$ have connections between the same nodes, they are not the same graph. Graph $A$ has undirected edges while graph $B$ has directed edges which point in certain directions.

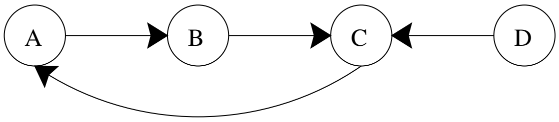


Figure A.2: Chains and paths. The vertex sequence $A - B - C - D$ is a chain of length 3 (but it is not a path). The vertex sequence $A - B - C - A - B$ is a path of length 4. The vertex sequence $A - B - C$ is a simple path of length 2. The vertex sequence $A - B - C - A - B - C - A$ is a cycle of length 6. The vertex sequence $A - B - C - A$ is a simple cycle of length 3.

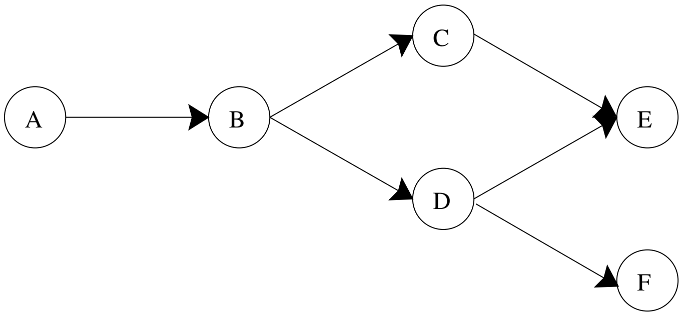


Figure A.3: DAGs, parents/children, ancestors/descendants, family. The above graph is a DAG. $A$ is the parent of $B$ while $B$ is the child of $A$; $B$ is the parent of $C$ and $D$ while $C$ and $D$ are the children of $B$; and so on. $A$ is the ancestor of $B$, $C$, $D$, $E$, and $F$ while $B$, $C$, $D$, $E$, and $F$ are the descendants of $A$; and so on. A possible ancestral ordering for this graph is $A, B, C, D, E, F$; another acceptable ancestral ordering is $A, B, D, F, C, E$. Some sample families from this graph are $A, B$ as the family of $B$; $B, C$ as the family of $C$; and $C, D, E$ as the family of $E$.

cycles (See Figure A.3).

A **parent/child** relationship in a directed graph occurs when there is an edge ($X_1$,$X_2$), from $X_1$ to $X_2$; $X_1$ is called the parent of $X_2$ and $X_2$ is called the child of $X_1$. In other words, the edge points from the parent to the child. An **ancestor/descendant** relationship, furthermore, is the extension of the parent/child relationship. For example, if $X_1$ is the parent of $X_2$ and if $X_2$ is the parent of $X_3$, then $X_1$ is an ancestor of $X_3$ and $X_3$ is a descendant of $X_1$. Like in human genealogies, this relationship extends further than just these two sets of parent/child relationships. An **ancestral ordering** is an ordering of nodes where each ancestor comes before its respective descendants. This is always and only possible in DAGs. (See Figure A.3). A **family** is the set of vertices composed of $X$ and the parents of $X$. For example, in Figure A.3, the vertices $\{C, D, E\}$ are the family of the vertex $E$. Whereas the terms **parent** and **child** define the relationship between two vertices connected by a directed edge, the term **adjacent** (or **neighbor**) describes the relationship between two vertices connected by an undirected edge; the two nodes are said to be adjacent.

A **forest** is a DAG where each node has either one parent or none at all. A **tree** is a forest where only one node (called the root) has no parent; in other words, every node but the root has exactly one parent. (See Figures A.4 & A.5). It should be noted that while books in graphical modeling define trees as above (Cowell *et al.*, 1999; Neapolitan, 1989), others define a tree as being a connected, *undirected*, acyclic graph (Brualdi, 1992; Boffey, 1982). This brings about the term **directed tree** versus regular trees. I am only concerned with directed trees, and I will, therefore, use the directed definition from Cowell *et al.* (1999) and Neapolitan (1989).

A **moral graph** is made from a DAG. For a DAG, we marry the parents of each node; this means

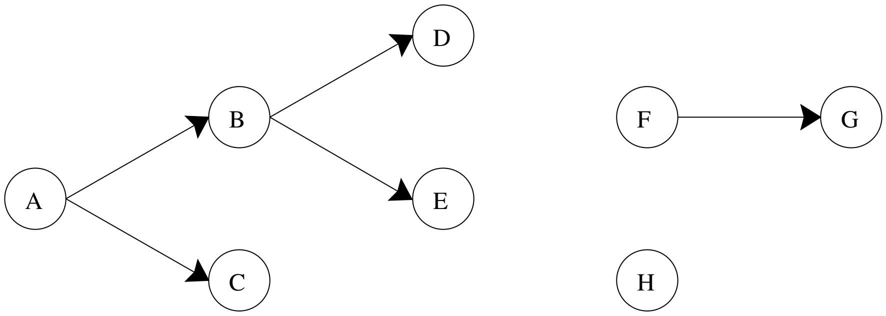


Figure A.4: Forest. The above graph, consisting of all nodes $A, B, \ldots, H$ is a forest. Every node has either one parent or none at all. $B, C, D, E, G$ have one parent each while $A, F, H$ have no parents. This graph is not a tree because it does not meet the requirement of only one of its nodes having no parent.

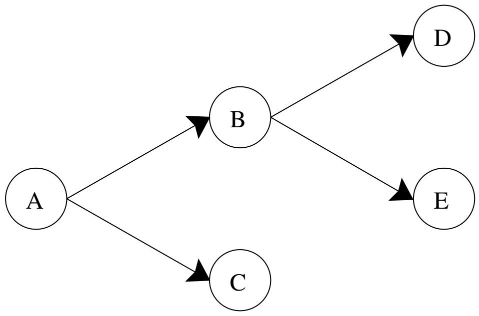


Figure A.5: Trees. The above graph is a tree. It is a DAG where every node has exactly one parent, except for the root node $A$ which has no parent.

that we add an undirected edge between each parent. After doing this, we remove the directionality from all of the original edges, resulting in a undirected graph. (See Figure A.6).

For a given path (or cycle), a **chord** is an edge that does not appear in the path but which is between two nodes that occur in the path. The term **chordless** describes a simple path or (simple cycle) for which no chords exist. For example, consider the cycle $B-C-E-D-B$ from the graph in Figure A.6. A chord for this cycle is the edge $(C, D)$. However, the cycle $B-C-D-B$ is chordless as there are no edges between non-adjacent nodes in this cycle.

The term **triangulated**, or decomposable, describes an *undirected* graph where any simple cycle with at least four nodes also has at least one chord. Note that it is not possible to have any chords in a cycle with three nodes. (See Figure A.6).

To make a graph triangulated we first need an **elimination order**. Any arbitrary numbering of the nodes will work, unless there are requirements that specify otherwise. The resultant ordering, $1, \ldots, n$, can then be used to make the triangulated graph. The following algorithm, taken from Pearl (1988) but originally from Tarjan and Yannakakis (1984), is how to use this numbering to construct the triangulated graph. Proceeding from node $n$, decreasing to node 1:

1. Determine the lower-numbered nodes which are adjacent to the current node, including those which may have been made adjacent to this node earlier in this algorithm.

2. Connect these nodes to each other.

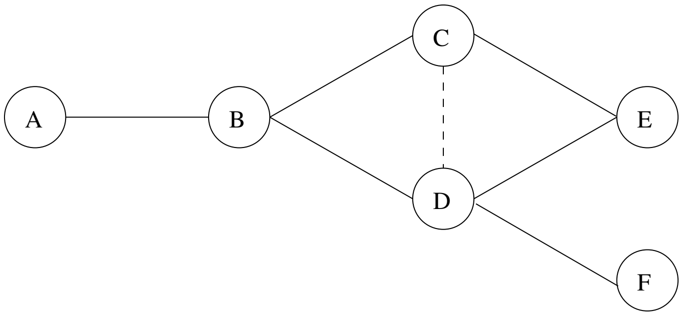


Figure A.6: Moral and triangulated graphs. The above graph is the moralized form of the graph in Figure A.3 on page 102. The only node with "unwed" parents in that graph is $E$. In moralizing the graph, $C$ and $D$, the parents of $E$, are connected. Then, the directions are removed from all the edges. This also happens to be a triangulated graph because its only simple cycle with a length of at least 4, $B-C-E-D-B$, does have a chord ($C-D$).

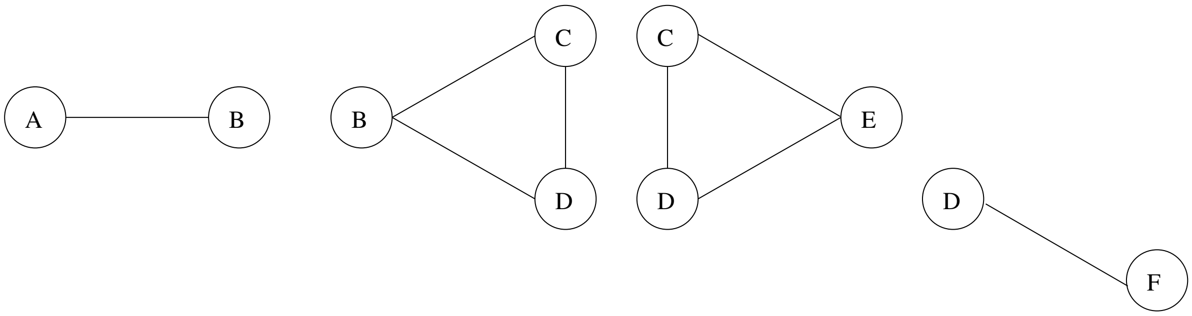


Figure A.7: Cliques. These subgraphs are all of the (maximal) cliques from the graph in Figure A.6. These cliques are complete subgraphs (that is, they are fully connected).

The term **complete** describes an *undirected* graph where every node is connected to all other nodes.

A **clique** is a subset of nodes which is complete and can not be made any larger while still being complete. For example, say that the subset of nodes $X_1, X_2, X_3$ is complete; if they form a clique, this means that there is no other node $X_i$ that can be added this subset with it still being complete. While Whittaker (1990, Section 3.1) specifies that a clique is maximal, this is not always strictly enforced. For example, Golumbic (1980, Section 1.1) makes a distinction between a clique and a maximal clique. (See Figure A.7).

A **clique tree** is a DAG that has been moralized and triangulated and then had its cliques become the nodes of a tree. See Figure A.8 on page 105. It needs to have the characteristic known as the **running intersection property**, which means that any vertex (or vertex set) that is found in any two cliques $C_i$ and $C_j$ in the tree will also be found in all of the cliques found on the chain between $C_i$ and $C_j$.

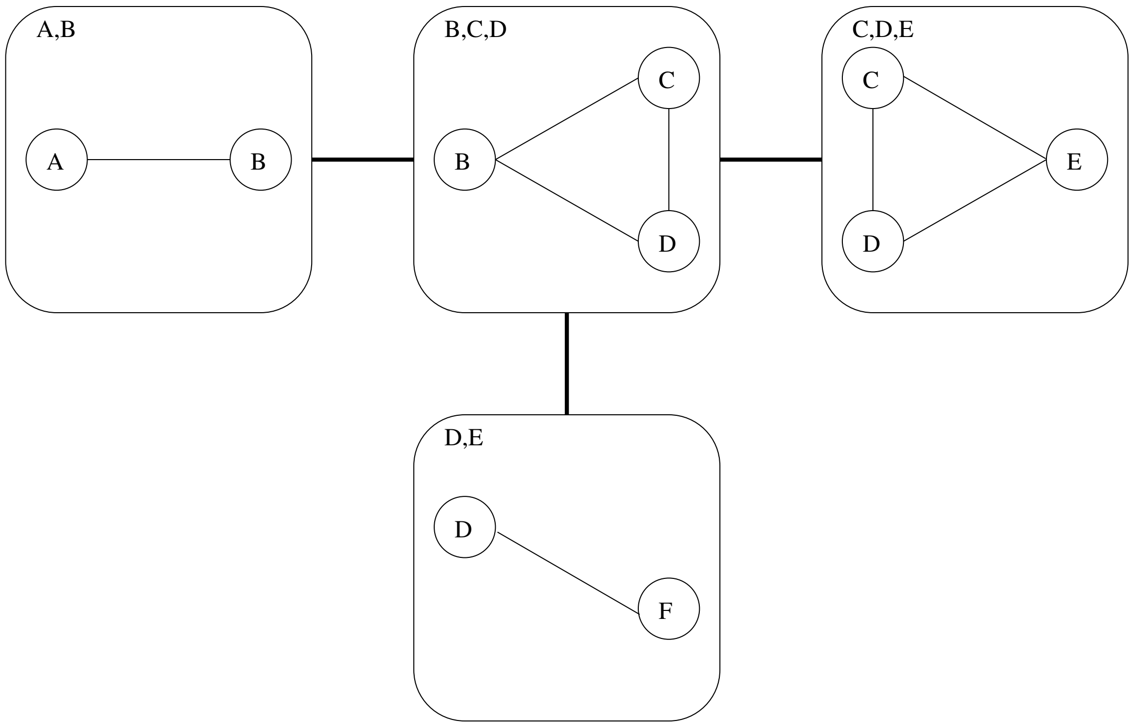


Figure A.8: A Join Tree. This is a clique tree for the cliques in Figure A.7. There are four nodes in this tree: (A,B), (B,C,D), (C,D,E), and (D,F).

# Appendix B

# An Introduction to Inference

## B.1 Example

Heckerman (1999) gives an example of a Bayesian network that models credit card fraud, as illustrated in Figure B.1. From this BN we can see, for example, that if there is a case of credit card fraud, then the chances of gas or jewelry being bought is affected; also, the chance of jewelry being bought is further affected by the age and sex of the purchaser. Table B.1 indicates the probabilities associated with each of the variables in this Bayesian network. For example, we can see that if 'Fraud' is 'Yes', then there is a probability of 0.2 that 'Gas' will be 'Yes' but if 'Fraud' is 'No', then there is a much smaller probability of 0.01 that 'Gas' will be 'Yes'. (That is, a person who is fraudulently using a credit card is twenty times more likely to buy gas than someone who is legitimately using a credit card). These different probabilities are commonly referred to as *beliefs* (Heckerman, 1999, Section 2).

Based on the discussion given in Heckerman (1999), inference in this Bayesian network proceeds as follows. Suppose that we notice a certain value for one or more of the variables in the network. If one variable has a definite (i.e., observed) value, our beliefs (i.e., probabilities) for the other variables need to be revised. This is what inference is: determining the updated (posterior) probability distribution for a variable based on the known values of the other variables. By itself, the prior probability for fraud is given as $P(Fraud = yes) = 0.00001$. However, we notice that a young man is using a credit card to buy jewelry (but not any gas). That is, $Sex = Male$, $Age =< 30$, $Jewelry = Yes$, $Gas = No$ (Table B.2). We then want to infer whether he is using the card fraudulently. In other words, we need to calculate $P(F|J, G, S, A)$ (each letter is the first letter of its respective variable). We proceed as follows:

By Bayes' rule,

$$P(F|J,G,S,A) = \frac{P(J,G,S,A,F)}{P(J,G,S,A)} \tag{B.1}$$

Because the states of $F$ are mutually exclusive and exhaustive, we can transform the denominator of (B.1) to get:

$$P(F|J,G,S,A) = \frac{P(J,G,S,A,F)}{\sum_{f\in(Yes,No)} P(J,G,S,A,F=f)} \tag{B.2}$$

Now, using the product rule of probability, both the numerator and denominator of equation B.2

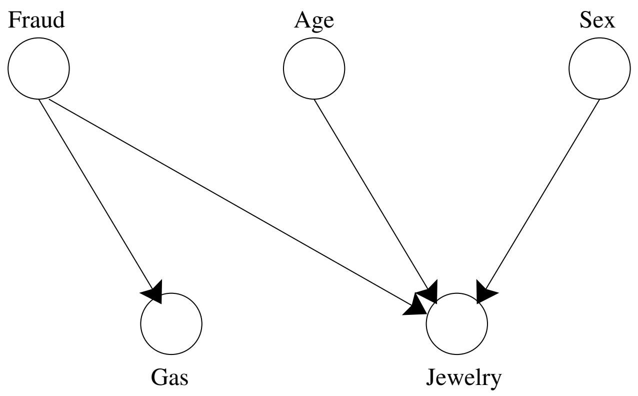


Figure B.1: A Bayesian network from Heckerman (1999) illustrating credit card fraud. There are five variables in this network. The probability of the Jewelry variable is dependent upon the values of the Fraud, Age, and Sex variables. Likewise, the value of the Gas variable is dependent upon the value of the Fraud variable. The Fraud, Age, and Sex variables themselves are not conditioned on any variable. See Table B.1 for the local probability distributions for each variable.

| Probability | | | Conditions | | |
|---|---|---|---|---|---|
| | | | Fraud | Age | Sex |
| Fraud = Yes | Fraud = No | | | | |
| 0.00001 | 0.99999 | | – | – | – |
| Age $< 30$ | Age = 30-50 | Age $> 50$ | | | |
| 0.25 | 0.40 | 0.35 | – | – | – |
| Sex = Male | Sex = Female | | | | |
| 0.5 | 0.5 | | – | – | – |
| Gas = Yes | Gas = No | | | | |
| 0.2 | 0.8 | | Yes | – | – |
| 0.01 | 0.99 | | No | – | – |
| Jewelry = Yes | Jewelry = No | | | | |
| 0.05 | 0.95 | | yes | * | * |
| 0.0001 | 0.9999 | | no | <30 | male |
| 0.0004 | 0.9996 | | no | 30-50 | male |
| 0.0002 | 0.9998 | | no | >50 | male |
| 0.0005 | 0.9995 | | no | <30 | female |
| 0.002 | 0.998 | | no | 30-50 | female |
| 0.001 | 0.999 | | no | >50 | female |

Table B.1: Probabilities for the variables in Figure B.1

| Variable | $Fraud$ | $Jewelry$ | $Gas$ | $Sex$ | $Age$ |
|---|---|---|---|---|---|
| Value | ? | $Yes$ | $No$ | $Male$ | $< 30$ |

Table B.2: An example set of one unknown and four observations from the credit card fraud network in Figure B.1. Values are observed for every variable except $Fraud$. These observed values can then be used to give an updated probability for the belief of $Fraud$ being $Yes$.

can be factored as follows:

$$P(F|J,G,S,A) = \frac{P(J|G,S,A,F)\,P(G|S,A,F)\,P(S|A,F)\,P(A|F)\,P(F)}{\sum_{f\in(Yes,No)} P(J|G,S,A,F=f)\,P(G|S,A,F=f)\,P(S|A,F=f)\,P(A|F=f)\,P(F=f)} \tag{B.3}$$

Due to the conditional independencies, we can remove certain variables from the conditional lists in (B.3) (see Section 2.4.1). That is, if the variable $X_0$ is not a child of $X_n$ in the graph, then

$$P(X_0|X_1,\ldots,X_n,\ldots,X_k) = P(X_0|X_1,\ldots,X_{n-1},X_{n+1},\ldots,X_k).$$

Using this simplification provides the following:

$$P(F|J,G,S,A) = \frac{P(J|S,A,F)\,P(G|F)\,P(S)\,P(A)\,P(F)}{\sum_{f\in(Yes,No)} P(J|S,A,F=f)\,P(G|F=f)\,P(S)\,P(A)\,P(F=f)} \tag{B.4}$$

At this point, we can then simplify equation B.4 by cancelling common factors in the numerator and denominator. (Doing so indicates that the prior probabilities for both age and sex have no direct impact on the probability for fraud being computed).

$$P(F|J,G,S,A) = \frac{P(J|S,A,F)\,P(G|F)\,P(F)}{\sum_{f\in(Yes,No)} P(J|S,A,F=f)\,P(G|F=f)\,P(F=f)} \tag{B.5}$$

(B.5) can then be computed by inserting the values for the variables and reading the probabilities from Table B.1.

$$\begin{aligned} &P(F=Yes|J=Yes,G=No,S=Male,A=<30)\\ &= \frac{P(J=Yes|S=Male,A=<30,F=Yes)\,P(G=No|F=Yes)\,P(F=Yes)}{\sum_{f\in(Yes,No)} P(J=Yes|S=Male,A=<30,F=f)\,P(G=No|F=f)\,P(F=f)} \end{aligned} \tag{B.6}$$

By substituting of the actual probabilities, we get:

$$\begin{aligned} &P(F=Yes|J=Yes,G=No,S=Male,A=<30)\\ &= \frac{0.05\cdot 0.8\cdot 0.00001}{(0.05\cdot 0.8\cdot 0.00001)+(0.0001\cdot 0.99\cdot 0.99999)}\\ &= 0.00402 \end{aligned}$$

Thus, while the prior probability of fraud is 0.00001, the inferred probability of fraud, given the other variables, is 0.00402 (over 400 times as probable).

## B.2 Implementation

Probabilistic inference in BNs is typically done on BNs which are viewed graphically as trees; a tree structure to the graph (instead of the more general DAG) is necessary as the propagation assumes certain conditional independence assumptions between variables (Pearl, 1988, Section 4.4). However, if the BN does not meet this requirement, it can be transformed from a DAG into a tree whose vertices are actually clusters of vertices (i.e., cliques) from the original DAG. This is done by moralizing and triangulating the original BN (Cowell *et al.*, 1999) and by forming the required tree from the cliques in this transformed graph (Golumbic, 1980). One of the goals in of this process is to

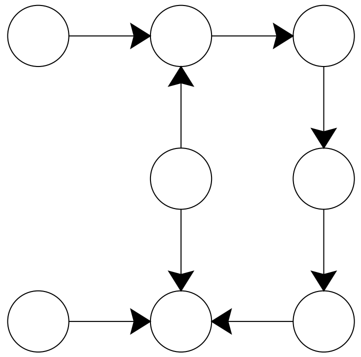

Figure B.2: A DAG

put together related variables (e.g., a given variable's parent variables) upon which the local computations can be performed; another goal is for neighboring cliques to have certain dependencies so that propagation can be done.

In the algorithms, a topological ordering of the vertices is needed. One algorithm for getting a topological ordering is to do a depth-first search of the DAG, based on Golumbic (1980, Algorithm 2.4):

```
Init:        S = {}
             Order = []
for each v ∈ V
             if v ∉ S
                          TopSort(v)

TopSort(v):  N = children(v)
             S = S ∪ {v}
             for each n ∈ N
                          if n ∉ S
                                       TopSort(n)
                          else if n ∉ Order
                                       return 'not a DAG'
             Order = [v Order]
```

As this algorithm does not use the markings of discrete versus continuous, a post-sort needs to be done where, while maintaining the ordering of discrete variables with respect to other discrete variables and of continuous variables to other continuous variables, the discrete variables occur before the continuous variables. As discrete variables are required to occur before continuous ones in the DAG, the resulting order will still be topological. See (Lauritzen and Jensen, 2001, Algorithm 7.10) (note that there they put continuous vertices before discrete but that this is accounted for in the ordering that their algorithms process the vertices).

### B.2.1 Constructing the clique tree

#### Moralizing the Bayesian network

Moralizing (as defined on page 102) a Bayesian network is concerned with "marrying" the parents of each child. The procedure is as follows. For every child in the initial graph (such as the graph in

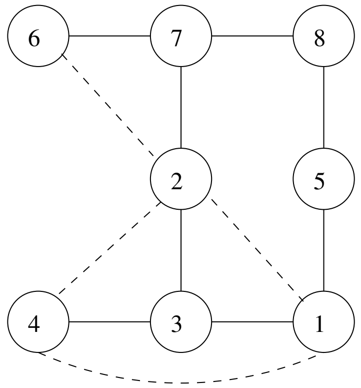


Figure B.3: The moralized version of the DAG in Figure B.2 (with the vertices numbered according to a maximum cardinality search after the graph was moralized)

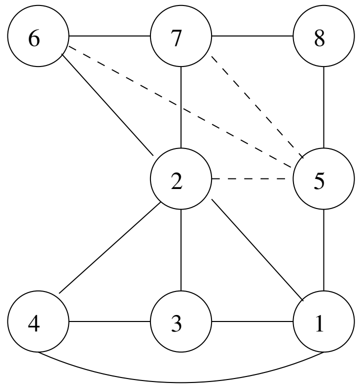


Figure B.4: The triangulated graph from Figure B.3

Figure B.2), we add undirected edges between all of its parents from the initial graph. Each parent of a given node should then be directly connected to each other parent of that given node (Jensen, 1996). Then we make all of the original edges in the entire graph undirected. So, we now end up with an undirected graph (such as in Figure B.3).

### Triangulating the graph

Triangulation is the process of making a graph triangulated (as defined on page 103). To do so, we first need to number the vertices of the graph. Any arbitrary numbering of the nodes will work. The resultant ordering, $1, \ldots, n$, can then be used to make the triangulated graph. The following algorithm, taken from Pearl (1988) but originally from Tarjan and Yannakakis (1984), is how to use this numbering to construct the triangulated graph (See Figure B.4). Proceeding from node $n$, decreasing to node 1:

1. Determine all the lower-numbered nodes which are adjacent to the current node, including those which may have been made adjacent to this node earlier in this algorithm.

2. Connect these nodes to each other.

Instead of choosing an arbitrary numbering of the nodes in order to do the triangulation, we can also use a maximum cardinality search (Neapolitan, 1989) (see Figure B.4):

1. Give any node an index of 1
2. For each subsequent number, pick an unnumbered node that neighbors the most already numbered nodes (if there is a tie, we can pick any of the nodes in the tie).
3. Give this node the next highest index and, if any nodes remain, go to Step 1.

One advantage of using the maximum cardinality search is that it can also be used as a test to determine if a graph is already triangulated. That is, if we use a numbering from a maximum cardinality search to do the triangulation, we will only have to add an edge if the graph is not already triangulated.

### Constructing the clique tree

Having a triangulated graph, we can now construct the clique tree. As a preliminary step, we need an algorithm for producing a perfect vertex elimination scheme from a triangulated graph. A perfect vertex elimination scheme is where each vertex subset $X_i = \{v_j \in Adj(v_i) | j < i\}$ ($Adj(v_i)$ is the set of vertices that neighbor $v_i$) is complete (as defined on page 104 (Golumbic, 1980, Section 4.2)). This means that the lower numbered neighbors of any given vertex form a complete subgraph (the numbering used in the triangulation step meets this criterion). This definition from Golumbic (1980) is modified so as to conform with the conventions that Neapolitan (1989) uses, who numbers the nodes from $1, 2, \ldots, n$ as opposed to from $n, \ldots, 2, 1$.

The following algorithm for producing the clique tree is based on Golumbic (1980, Section 4.7). It produces a list of maximal cliques.

```
Given: Graph G = (V, E)
Given: ordering of vertices v_1, ..., v_n: σ (a perfect elimination order)
Initialize: χ = 1; S[1, ..., n] = [0, ..., 0], Cliques = []
for each decreasing vertex v in σ
    if v has neighbors
        if any of v's neighbors occur earlier than v in σ
            X = v's earlier occurring neighbors in σ
            u = the last occurring element of X
            S[u] = the greater of either S[u] or |X| − 1
            if S[v] < |X|
                Cliques = Cliques + X, v
                χ = the greater of either χ or 1 + |X|
        else
            break out of for loop
    else
        Cliques = Cliques + v
return Cliques, χ
```

Having a list of the cliques in the graph, construct the clique tree (Pearl, 1988, Section 3.2.4):

1. Sort the clique list according to the highest numbered node (using the *maximum cardinality* ordering) in the clique.
2. Connect each clique to a previous clique in the sorted list with which it shares the most number of nodes.

An example of a clique tree for the cliques in Figure B.5 is given in Figure B.6 on page 114.

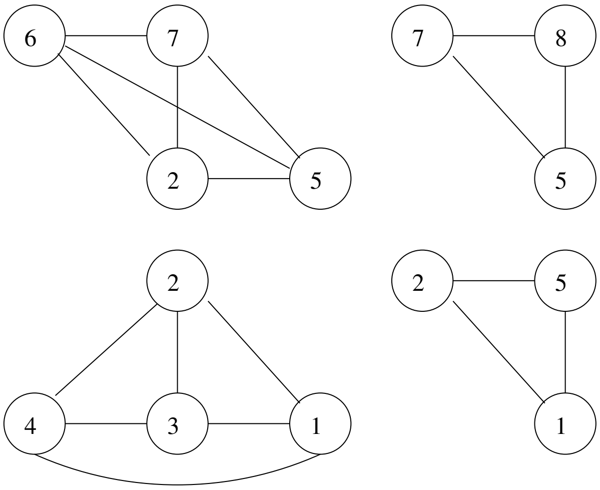


Figure B.5: The cliques from Figure B.4. Note that it is common for variables to occur in multiple cliques.

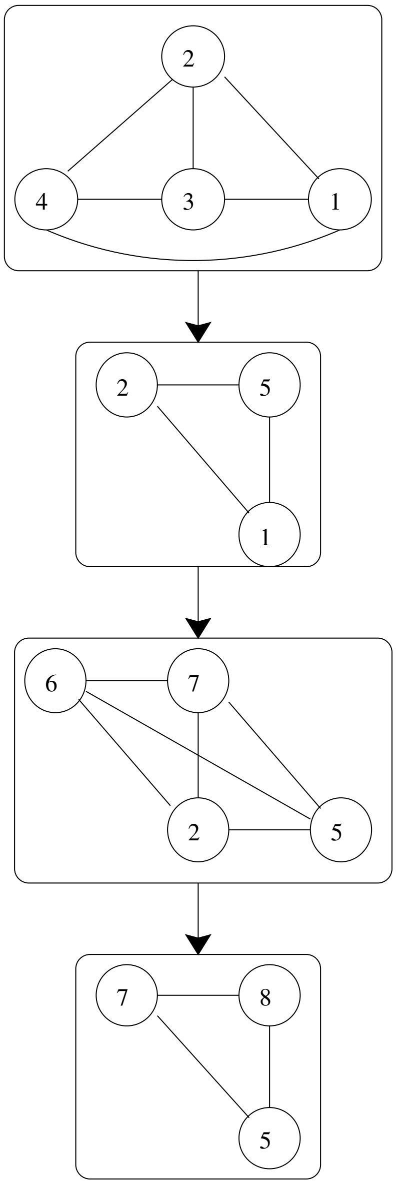


Figure B.6: The cliques from Figure B.5 formed into a clique tree. Note that for any given variable $x$ in the tree, that all of the cliques that it occurs in are connected together (a.k.a., running intersection property). While this clique tree is a chain, nodes in a tree in general are allowed to have more than one child.

# Bibliography


Atal, B. S. (1974). Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *The Journal of the Acoustical Society of America*, **55**, 1304–1312.

Baum, L. E. (1972). An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, **3**, 1–8.

Bilmes, J. A. (1998). Data-driven extensions to HMM statistical dependencies. In ICSLP '98 (1998).

Bilmes, J. A. (1999). *Natural Statistical Models for Automatic Speech Recognition*. Ph.D. thesis, University of California, Berkeley.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, UK.

Boffey, T. B. (1982). *Graph Theory in Operations Research*. Macmillan Computer Science Series. The Macmillan Press, Ltd.

Boll, S. F. (1979). Suppression of acoustic noise in using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **ASSP-27**(2), 113–120.

Bourlard, H. and Dupont, S. (1996). A new ASR approach based on independent processing and recombination of partial fruequency bands. In *Proceedings ICSLP 96: Fourth International Confence on Spoken Language Processing*, volume 1, pages 426–429, Philadelphia.

Bourlard, H. and Morgan, N. (1993). *Connectionist Speech Recognition: A Hybrid Approach*, volume 247 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston.

Bourlard, H., Adali, T., Bengio, S., Larsen, J., and Douglas, S., editors (2002). *Neural Networks for Signal Processing XII–Proceedings of the 2002 IEEE Signal Processing Society Workshop (NNSP 2002)*, Martigny, Switzerland.

Boyen, X. and Koller, D. (1998). Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI–98)*, pages 33–42, Madison, WI. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Brualdi, R. A. (1992). *Introductory Combinatorics*. Elsevier Science Publishing Co., Inc., New York, second edition.

Cole, R. A., Fanty, M., and Lander, T. (1994). Telephone speech corpus at CSLU. In *Proc. of Intl. Spoken Language Processing*, Yokohama, Japan.

Cooper, G. F. and Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, **9**, 309–347.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (1999). *Probabilistic Networks and Expert Systems*. Statistics for Engineering and Information Science. Springer-Verlag New York, Inc.

Dean, T. and Kanazawa, K. (1988). Probabilistic temporal reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 524–528, St. Paul, MN.

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

Deviren, M. and Daoudi, K. (2001). Structural learning of dynamic Bayesian networks in automatic speech recognition. In Eurospeech '01 (2001), pages 1669–1672.

Dupont, S. (2000). *Etude et développement d'architectures multi-bandes et multi-modales pour la reconnaissance robuste de la parole*. Ph.D. thesis, Factulté Polytechnique de Mons, Mons, Belgium.

Dupont, S., Bourlard, H., Deroo, O., Fontaine, V., and Boite, J.-M. (1997). Hybrid HMM/ANN systems for training independent tasks: Experiments on phonebook and related improvements. In *Proceedings of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-97)*, volume 3, pages 1767–1770, Munich.

Escofet Carmona, J. and Stephenson, T. A. (2003). Automatic speech recognition using dynamic Bayesian networks with the energy as an auxiliary variable. IDIAP-RR 18, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2003/rr03-18.ps.gz.

Eurospeech '01 (2001). *7th European Conference on Speech Communication and Technology (Eurospeech '01)*, Aalborg, Denmark.

Eurospeech '97 (1997). *5th European Conference on Speech Communication and Technology (Eurospeech '97)*, Rhodes, Greece.

Frankel, J. and King, S. (2001). ASR - articulatory speech recognition. In Eurospeech '01 (2001), pages 599–602.

Frankel, J., Richmond, K., King, S., and Taylor, P. (2000). An automatic speech recognition system using neural networks and linear dynamic models to recover and model articulatory traces. In ICSLP '00 (2000), pages 254–257.

Fujinaga, K., Nakai, M., Shimodaira, H., and Sagayama, S. (2001). Multiple-regression hidden Markov model. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-01)*, volume 1, pages 513–516, Salt Lake City, Utah, USA.

Ghahramani, Z. (1997). Factorial hidden Markov models. *Machine Learning*, **29**, 245–275.

Golumbic, M. C. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York.

Hagen, A. (2001). *Robust speech recognition based on multi-stream processing*. Ph.D. thesis, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland.

Heckerman, D. (1999). A tutorial on learning with Bayesian networks. In Jordan (1999), pages 301–354.

Hermansky, H. and Morgan, N. (1994). Rasta processing of speech. *IEEE Transancations Speech and Audio Processing*, **2**(4), 578–589.

Hermansky, H., Hanson, B. A., and Wakita, H. (1985). Perceptually based linear predictive analysis of speech. In *Proceedings of the 1985 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-85)*, pages 509–512, Tampa, FL.

Hermansky, H., Morgan, N., Bayya, A., and Kohn, P. (1992). Rasta-plp speech analysis technique. In ICASSP '92 (1992), pages 121–124.

Hess, W. (1983). *Pitch Determination of Speech Signals: Algorithms and Devices*, volume 3 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin.

ICASSP '92 (1992). *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92)*, San Francisco, CA.

ICASSP '95 (1995). *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-95)*, Detroit, MI.

ICASSP '98 (1998). *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-98)*, Seattle, Washington, USA.

ICSLP '00 (2000). *6th International Conference on Spoken Language Processing: ICSLP 2000 (Interspeech 2000)*, Beijing.

ICSLP '98 (1998). *Proceedings ICSLP 98: 5th International Confence on Spoken Language Processing*, Sydney.

Jelinek, F. (1969). A fast sequential decoding algorithm using a stack. *IBJ J. Res. Develop.*, **13**.

Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of Markov souce parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397. North-Holland Pub. Co., Amsterdam.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*. UCL Press Ltd., London.

Jordan, M. I., editor (1999). *Learning in Graphical Models*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, first MIT press edition.

Josuttis, N. M. (1999). *The C++ Standard Library: A Tutorial and Reference*. Addison-Wesley, Boston.

Kanthak, S. and Ney, H. (2002). Context-dependent acoustic modeling using graphemes for large vocabulary speech recognition. In *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-02)*, volume I, pages 845–848, Orlando, Florida.

Kirchhoff, K. (1999). *Robust Speech Recognition Using Articulatory Information*. Ph.D. thesis, Universität Bielefeld.

Koller, D., Lerner, U., and Angelov, D. (1999). A general algorithm for approximate inference and its application to hybrid Bayes nets. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI–99)*, pages 324–333, Stockholm, Sweden. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Konig, Y. and Morgan, N. (1992). GDNN: A gender-dependent neural network for continuous speech recognition. In *Proceedings of the 1992 International Joint Conference on Neural Networks (IJCNN)*, pages 332–337, Baltimore, MD.

Krstulović, S. (2001). *Speech Analysis with Production Constraints*. Ph.D. thesis, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland.

Lagus, K. and Kurimo, M. (2002). Language model adaptation in speech recognition using document maps. In Bourlard *et al.* (2002), pages 627–636.

Lauritzen, S. L. (1992). Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, **87**(420), 1098–1108. Theory and Methods.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, **19**, 191–201.

Lauritzen, S. L. (1996). *Graphical Models*. Oxford Statistical Science Series, 17. Clarendon Press, Oxford.

Lauritzen, S. L. and Jensen, F. (2001). Stable local computations with conditional Gaussian distributions. *Statistics and Computing*, **11**(2), 191–203.

Lauritzen, S. L. and Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, **17**(1), 31–57.

Lerner, U. and Parr, R. (2001). Inference in hybrid networks: Theoretical limits and practial algorithms. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventeenth Conference (UAI-2001)*, pages 310–318, Seattle, WA. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Li, S. Z. (1995). *Markov random field modeling in computer vision*. Number XVI in Computer Science Workbench. Springer-Verlag, Tokyo.

Logan, B. and Moreno, P. (1998). Factorial HMMs for acoustic modeling. In ICASSP '98 (1998), pages 813–816.

Lowerre, B. and Reddy, R. (1980). The harpy speech understanding system. In W. A. Lea, editor, *Trends in Speech Recognition*. Prentice-Hall, Englewood Cliffs, NJ.

Markel, J. D. (1972). The SIFT algorithm for fundamental frequency estimation. *IEEE Trans. Audio and Electroacoustics*, **20**, 367–377.

Martínez, F., Tapias, D., Álvarez, J., and León, P. (1997). Characteristics of slow, average and fast speech and their effects in large vocabulary continuous speech recognition. In Eurospeech '97 (1997), pages 469–472.

Martínez, F., Tapias, D., and Álvarez, J. (1998). Towards speech rate independence in large vocabulary continuous speech recognition. In ICASSP '98 (1998), pages 725–728.

Mirghafori, N. and Morgan, N. (1998). Combining connectionist multi-band and full-band probability streams for speech recognition of natural numbers. In ICSLP '98 (1998), pages 743–746.

Mirghafori, N., Fosler, E., and Morgan, N. (1995). Fast speakers in large vocabulary continuous speech recognition: analysis & antidotes. In *4th European Conference on Speech Communication and Technology (Eurospeech '95)*, volume 1, pages 491–494, Madrid.

Monti, S. and Cooper, G. F. (1999). Learning hybrid Bayesian networks from data. In Jordan (1999), pages 521–540.

Morgan, N. and Fosler-Lussier, E. (1998). Combining multiple estimators of speaking rate. In ICASSP '98 (1998), pages 729–732.

Morgan, N., Fosler, E., and Mirghafori, N. (1997). Speech recognition using on-line estimation of speaking rate. In Eurospeech '97 (1997), pages 2079–2082.

Morris, A. C., Cooke, M. P., and Green, P. D. (1998). Some solutions to the missing feature problem in data classification, with application to noise robust ASR. In ICASSP '98 (1998), pages 737–740.

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California, Berkeley.

Neapolitan, R. E. (1989). *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York.

Owens, F. J. (1993). *Signal processing of speech*. Macmillan New Electronics: Introductions to Advanced Topics. The Macmillan Press Ltd., Houndmills, Basingstoke, UK.

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Inc., New York, third edition.

Paul, D. B. (1992). An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model. In ICASSP '92 (1992), pages 25 – 28.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc.

Pitrelli, J. F., Fong, C., Wong, S. H., Spitz, J. R., and Leung, H. C. (1995). PhoneBook: A phonetically-rich isolated-word telephone-speech database. In ICASSP '95 (1995), pages 101–104.

Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition*. PTR Prentice-Hall, Inc., Englewood Cliffs, NJ.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**(2), 257–286.

Renals, S. and Hochberb, M. M. (1999). Start-synchronous search for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, **7**(5), 542–553.

Richardson, M., Bilmes, J., and Diorio, C. (2000). Hidden-articulator Markov models for speech recognition. In *Proceedings of the ISCA ITRW ASR2000. Automatic Speech Recognition: Challenges for the new Millenium*, pages 133–139, Paris.

Roweis, S. and Ghahramani, Z. (1999). A unifying review of linear Gaussian models. *Neural Computation*, **11**(2).

Siegler, M. A. and Stern, R. M. (1995). On the effects of speech rate in large vocabulary speech recognition systems. In ICASSP '95 (1995), pages 612–615.

Smith, P. W. F. and Whittaker, J. (1999). Edge exclusion tests for graphical Gaussian models. In Jordan (1999), pages 555–574.

Smyth, P., Heckerman, D., and Jordan, M. I. (1997). Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, **9**(2), 227–269.

Stephenson, T. A., Bourlard, H., Bengio, S., and Morris, A. C. (2000). Automatic speech recognition using dynamic Bayesian networks with both acoustic and articulatory variables. In ICSLP '00 (2000), pages 951–954.

Stephenson, T. A., Mathew, M., and Bourlard, H. (2001). Modeling auxiliary information in Bayesian network based ASR. In Eurospeech '01 (2001), pages 2765–2768.

Stephenson, T. A., Escofet, J., Magimai-Doss, M., and Bourlard, H. (2002a). Dynamic Bayesian network based speech recognition with pitch and energy as auxiliary variables. In Bourlard *et al.* (2002), pages 637–646.

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. (2002b). Mixed Bayesian networks with auxiliary variables for automatic speech recognition. In *International Conference on Pattern Recognition (ICPR 2002)*, volume 4, pages 293–296, Quebec City, PQ, Canada.

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. (2002c). Auxiliary variables in conditional Gaussian mixtures for automatic speech recognition. In *7th International Conference on Spoken Language Processing: ICSLP 2002 (Interspeech 2002)*, volume 4, pages 2665–2668, Denver.

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. (2003). Speech recognition of spontaneous, noisy speech using auxiliary information in Bayesian networks. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03)*, volume 1, pages 20–23, Hong Kong.

Tarjan, R. E. and Yannakakis, M. (1984). Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM J. Computing*, **13**, 566–79.

Varga, A., Steeneken, H., Tomlinson, M., and Jones, D. (1992). The NOISEX-92 study on the effect of additive noise on automatic speech recognition. Technical report, DRA Speech Research Unit, Malvern, England.

Wellekens, C. J. (1987). Explicit time correlation in hidden Markov models for speech recognition. In *Proceedings of the 1987 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-87)*, volume 1, pages 384–386, Dallas, Texas.

Westbury, J. R., Turner, G., and Dembowski, J. (1994). *X-ray Microbeam Speech Production Database User's Handbook*. Waisman Center on Mental Retardation & Human Development, University of Wisconsin, Madison, WI, first edition.

Whittaker, J. (1990). *Graphical Models in Applied Multivariate Statistics*. John Wiley & Sons Ltd., Chichester, UK.

Wrench, A. (2000). Multichannel/multispeaker articulatory database for continuous speech recognition research. In *PHONUS*, Saarbrücken, Germany. Institute of Phonetics, Saarland University.

Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. (1999). *The htk book*. Entropic, Ltd., Cambridge, UK, HTK version 2.2 edition.

Zlokarnik, I. (1995). Adding articulatory features to acoustic features for automatic speech recognition. *The Journal of the Acoustical Society of America*, **97**(5), 3246. Abstract 1aSC38.

Zweig, G. and Padmanabhan, M. (1999). Dependency modeling with Bayesian networks in a voice-mail transcription system. In *6th European Conference on Speech Communication and Technology (Eurospeech '99)*, volume 3, pages 1135–1138, Budapest, Hungary.

Zweig, G. G. (1998). *Speech Recognition with Dynamic Bayesian Networks*. Ph.D. thesis, University of California, Berkeley.

# Curriculum Vitae

# Todd A. Stephenson



| | | | |
|---|---|---|---|
| Permanent address: | 1140 Silver Maple Dr | Phone: | +1 570 586 0803 |
| | Clarks Summit, PA 18411-9780 | email: | Todd.Stephenson@idiap.ch |
| | USA | Citizenship: | USA |



## Work Experience

| | |
|---|---|
| 1999– | Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP), Switzerland<br>Speech Processing Group<br>Research Assistant |
| winter 2001 | AT&T Labs-Research, New Jersey<br>Information Sciences Research Division<br>Intern |
| 1995–1997 | AT&T Corp., New Jersey<br>Consumer Markets Division<br>Computer Programmer and System Administrator |
| 1993–1995 | The Pennsylvania State University, Pennsylvania (part-time)<br>University Learning Resource Center<br>Supplemental Instruction Leader and Mathematics Tutor |

## Education

| | |
|---|---|
| 2000– | Docteur ès Sciences (anticipated May 2003)<br>The Swiss Federal Institute of Technology Lausanne (EPFL), Switzerland<br>Dissertation: *Speech Recognition with Auxiliary Information*. |
| 1997–1998 | Master of Science in Cognitive Science and Natural Language<br>The University of Edinburgh, United Kingdom<br>Dissertation: *Speech Recognition using Phonetically Featured Syllables*. |
| 1991–1995 | Bachelor of Science in Mathematics, with minor in Computer Science<br>The Pennsylvania State University, Pennsylvania |

## Computer Experience

| | |
|---|---|
| Operating Systems: | Solaris, UNIX/Linux, Windows, FTX 2.1, MacOS |
| Computer Applications: | HTK 2.1, esps/waves+, NICO Artificial Neural Network Toolkit, Conversant Voice Information System |
| Languages: | C++, C, Pro-C, PERL, Matlab, ksh, HTML, SQL, REXX, Pascal |

## Languages

English (native), French (good), Spanish (moderate), Japanese (basic)

## Published/Accepted Papers

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. (2003). Speech recognition of spontaneous, noisy speech using auxiliary information in Bayesian networks. In *Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03)*, volume 1, pages 20–23, Hong Kong.

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. (2002). Auxiliary variables in conditional Gaussian mixtures for automatic speech recognition. In *7th International Conference on Spoken Language Processing: ICSLP 2002 (Interspeech 2002)*, volume 4, pages 2665–2668, Denver.

Stephenson, T. A., Escofet, J., Magimai-Doss, M., and Bourlard, H. (2002). Dynamic Bayesian network based speech recognition with pitch and energy as auxiliary variables. In H. Bourlard, T. Adali, S. Bengio, J. Larsen, and S. Douglas, editors, *Neural Networks for Signal Processing XII–Proceedings of the 2002 IEEE Signal Processing Society Workshop (NNSP 2002)*, pages 637–646, Martigny, Switzerland.

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. (2002). Mixed Bayesian networks with auxiliary variables for automatic speech recognition. In *International Conference on Pattern Recognition (ICPR 2002)*, volume 4, pages 293–296, Quebec City, PQ, Canada.

Stephenson, T. A., Mathew, M., and Bourlard, H. (2001). Modeling auxiliary information in Bayesian network based ASR. In *7th European Conference on Speech Communication and Technology (Eurospeech '01)*, volume 4, pages 2765–2768, Aalborg, Denmark.

Stephenson, T. A., Bourlard, H., Bengio, S., and Morris, A. C. (2000). Automatic speech recognition using dynamic Bayesian networks with both acoustic and articulatory variables. In *6th International Conference on Spoken Language Processing: ICSLP 2000 (Interspeech 2000)*, volume 2, pages 951–954, Beijing.

King, S., Stephenson, T., Isard, S., Taylor, P., and Strachan, A. (1998). Speech recognition via phonetically featured syllables. In *Proceedings ICSLP 98: 5th International Confence on Spoken Language Processing*, volume 3, pages 1031–1034, Sydney.

## Submitted Papers

Stephenson, T. A., Magimai-Doss, M., and Bourlard, H. Speech recognition with auxiliary information. *IEEE Transactions on Speech and Audio Processing*. Submitted and completed one round of reviews, current status of "revise and resubmit."

## Unpublished Papers

Magimai.-Doss, M., Stephenson, T. A., and Bourlard, H. (2003). Using pitch frequency information in speech recognition. IDIAP-RR 23, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2003/rr03-23.ps.gz.

Escofet Carmona, J. and Stephenson, T. A. (2003). Automatic speech recognition using dynamic Bayesian networks with the energy as an auxiliary variable. IDIAP-RR 18, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2003/rr03-18.ps.gz.

Stephenson, T. A. (2003). Conditional Gaussian mixtures. IDIAP-RR 11, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2003/rr03-11.ps.gz.

Magimai.-Doss, M., Stephenson, T. A., and Bourlard, H. (2002). Modelling auxiliary information (pitch frequency) in hybrid HMM/ANN based ASR systems. IDIAP-RR 62, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2002/rr02-62.ps.gz.

Stephenson, T. A., Magimai Doss, M., and Bourlard, H. (2000). Automatic speech recognition using pitch information in dynamic Bayesian networks. IDIAP-RR 41, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2000/rr00-41.ps.gz.

Stephenson, T. A. (2000). An introduction to Bayesian network theory and usage. IDIAP-RR 03, IDIAP, Martigny, Switzerland. Available at ftp://ftp.idiap.ch/pub/reports/2000/rr00-03.ps.gz.

Stephenson, T. A. (1998). *Speech Recognition using Phonetically Featured Syllables*. Master's thesis, University of Edinburgh, United Kingdom. Available at http://www.cstr.ed.ac.uk/projects/espresso.

Stephenson, T. A. (1998). Artificial neural networks in recognition of phonetic features in speech. Project report for Neural Computation module during M.Sc. course. The University of Edinburgh, United Kingdom. Available at http://www.cstr.ed.ac.uk/projects/espresso.

Stephenson, T. A. (1998). Speech recognition of phones using feature streams. Project report for Speech Recognition module during M.Sc. course. The University of Edinburgh, United Kingdom. Available at http://www.cstr.ed.ac.uk/projects/espresso.