# Audio-Visual Automatic Speech Recognition using Dynamic Bayesian Networks

by

Helge Reikeras

*Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Applied Mathematics at Stellenbosch University*

Department of Applied Mathematics,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisors:

Prof. Ben Herbst   Prof. Johan du Preez,   Dr. Herman Engelbrecht

January 2011

# Declaration

# Abstract

## Audio-Visual Automatic Speech Recognition using Dynamic Bayesian Networks

Helge Reikeras
*Department of Applied Mathematics,*
*University of Stellenbosch,*
*Private Bag X1, Matieland 7602, South Africa.*

Thesis: MSc (AM)
January 2011

In audio-visual automatic speech recognition (AVASR) both acoustic and visual modalities of speech are used for speech recognition. The use of the visual speech modality for speech recognition is motivation by the ability of hearing-impaired listener to understand speech from visual cues only through so-called lip-reading. Perceptual phenomena such as the McGurk effect (McGurk and MacDonald, 1976) also suggests that there is indeed information in visual speech that is useful for recognition purposes. AVASR is in particular expected to perform better than traditional audio-only speech recognition systems as the visual channel is not affected by acoustic noise.

The components comprising an AVASR system are acoustic and visual feature extraction, feature stream weighting, feature stream integration, model learning, and classification. In this thesis we mainly focus on the feature stream weighting, feature stream integration, model learning, and classification problems. The topics of acoustic and visual feature extraction are briefly discussed for completeness.

Feature stream weighting is the process of weighting the influence that the acoustic and visual feature streams have on the recognition decision according to some measure of the reliability of each stream. Typically, we would expect the acoustic stream to contain more information than the visual stream and as such we would give the acoustic stream more weight. However, acoustic noise may render the acoustic stream less reliable. Thus, in noisy acoustic environments we would like to weight the visual stream more as it is not affected by acoustic noise.

Feature stream integration is the strategy that is used when integrating speech information extracted from acoustic and visual speech samples. A particularly interesting problem in AVASR is that of audio-visual asynchrony. When speaking the motion of visible articulators such a lips, tongue and jaw occurs prior to the actual sound being uttered. Thus, there is a slight delay between the acoustic and visual feature streams. This delay is not constant, but depends on the particular sound that is being uttered as well as on the speaker. We propose several different audio-visual models that model audio-visual asynchrony differently.

Probability theory, with its inherent notions of uncertainty and confidence, is a natural approach to solving these problems. We have chosen to focus on the specific class of probabilistic models that can be formulated as Bayesian networks (BNs) (Pearl, 1988). This model class is particularly well-suited to modelling the causalities inherent to audio-visual speech (Nefian *et al.*, 2002). In particular we discuss dynamic Bayesian networks (DBNs). The DBN is an extension to BNs which allow for modelling variable-length sequences of observed and unobserved random variables such as sequences of features extracted from speech samples. We show that the hidden Markov model, which is the most popular model in the speech recognition literature, is a special case of the DBN framework. We discuss general learning and inference methods for BNs and DBNs in detail, giving a complete and self-contained presentation of literature that is otherwise only available in different works.

Learning DBNs in the context of AVASR is performed through estimating parameters of audio-visual DBN models from sample data. We discuss maximum likelihood learning in the form of the expectation maximisation (EM) algorithm, and variational learning in the form of the variational Bayes (VB) algorithm. Given a set of learned models, for instance representing different words or phonemes, we wish to determine which model a previously unobserved audio-visual sample is most similar to (in some appropriate sense). The latter is known as the classification problem. We show that learning and classification can be performed using efficient general algorithms in the BN and DBN frameworks. In particular we discuss the junction tree algorithm for BNs and the interface junction tree algorithm for DBNs. The interface junction tree algorithm is a generalisation of the classic forward-backwards algorithm used in HMMs.

As a part of the research we propose and implement a full-featured AVASR system. The system is used to perform a set of experiments where we evaluate and compare the performance of the different models and learning algorithms that we have proposed for AVASR. In the experiments we use the Clemson University audio-visual experiments (CUAVE) data corpus for learning and testing the models and algorithms. The CUAVE data corpus consists of multiple speakers uttering the digits from zero to nine. As a result the experiments conducted are multiple-speaker digit recognition experiments. As we are particularly interested in the performance of AVASR at different levels of acoustic noise we add artificial Gaussian noise to the acoustic feature stream and evaluate performance at different signal-to-noise (SNR) levels. The level of noise ranges from −6 to 18 dB SNR in steps of 4 dB.

The results of the experiments show that there is indeed information in the visual speech modality useful for speech recognition. In particular we find that for the digit recognition experiment using only the visual speech information we achieve a 25.6% misclassification rate. In comparison, the misclassification rate when using only the audio features at the most severe noise level (−6 dB) is 84.2%. We also find that models that combine acoustic and visual features in general perform better than models that only use the acoustic features. The performance increase is more pronounced for smaller SNR levels as in this region the visual observations compensate for the acoustic noise. However, even at large SNR levels we are able to show a statistically significant difference between the audio-visual and audio-only classifiers. At 18 dB SNR the error rate of the audio-visual classifier is 1.1% versus 2.7% for audio-only, which is more than a halving of the error rate when using the audio-visual classifier.

We are able to reproduce the results presented in Glotin *et al.* (2001) which shows that stream weighting is beneficial to AVASR. We are also able to reproduce the results in Liu *et al.* (2002) which show that the feature stream integration scheme modelled by the audio-visual coupled HMM (AV-CHMM) is superior to less sophisticated integration schemes modelled by the audio-visual product HMM (AV-PHMM) and audio-visual independent HMM (AV-IHMM). The AV-CHMM, AV-PHMM and AV-IHMM are all DBN models and as such the general learning and inference framework applies.

The novel contribution of the research is the application of variational Bayesian (VB) learning to audio-visual DBNs. The VB learning method is an alternative to the classic expectation maximisation (EM) algorithm. Variational learning leads to automatic model complexity selection and avoids the singularities and overfitting problems associated with EM. In the experiments we find that audio-visual DBN models trained using VB are more robust to noise, likely due to its more compact form. However, there is not sufficient evidence to support that VB in general performs better than EM when learning audio-visual DBN models. In particular, for certain levels of acoustic noise VB performs worse than EM illustrating the possibility of over-smoothing when using variational learning. We do, however, find that models trained using variational learning appears to perform equally well as models trained using maximum likelihood estimation at the smallest levels of acoustic noise while the automatic model selection property yields a more sparse representation. Thus, by using variational methods we are able to do “just as well with less”. This is always a desirable property, in particular in resource-critical applications such as in handheld devices.

# Uittreksel

**Audio-Visual Automatic Speech Recognition using Dynamic Bayesian Networks**

Helge Reikeras

*Department of Applied Mathematics,*
*University of Stellenbosch,*
*Private Bag X1, Matieland 7602, South Africa.*

Tesis: MSc (AM)

Januarie 2011

In oudio-visuele outomatiese spraakherkenning (OVOSR) word sowel akoestiese en visuele modusse van spraak gebruik vir spraakherkenning. Die gebruik van die visuele spraak modaliteit vir spraakherkenning word gemotiveer deur gehoorgestremde luisteraars se vermoë om spraak te verstaan deur slegs van visuele leidrade gebruik te maak (sogenaamde liplesing). Perseptuele verskynsels soos die McGurk effek (McGurk en MacDonald, 1976) impliseer ook dat daar inderdaad inligting vervat is in die visuele spraak wat nuttig is vir die herkenning van spraak. Daar word verwag dat OVOSR beter sal presteer as tradisionele spraakherkenningsstelsels gebaseer op klank aleen, omdat die visuele kanaal nie geaffekteer word deur akoestiese ruis nie.

Die komponente waaruit 'n OVOSR stelsel bestaan is akoestiese en visuele kenmerkontrekking, kenmerkstroomweging, kenmerkstroomintegrasie, modelafrigting, en klassifikasie. Ons fokus in hierdie tesis hoofsaaklik op die probleme van kenmerkstroomweging, kenmerkstroomintegrasie, modelafrigting en klassifikasie. Die onderwerp van akoestiese en visuele kenmerkonttrekking word kortliks bespreek vir volledigheid.

Kenmerkstroomweging is die weging van die invloed wat die onderskeie akoestiese en visuele kenmerkstrome het op die herkenningsbesluit volgens 'n maatstaf van die betroubaarheid van die onderskeie kenmerkstroom. Ons sou tipies verwag dat die akoestiese kenmerkstroom meer inligting bevat as die visuele kenmerkstroom, en sou dus die akoestiese kenmerkstroom 'n swaarder gewig toeken. Akoestiese ruis mag egter veroorsaak dat die akoestiese kenmerkstroom minder betroubaar is en dus in raserige omgewing wil ons graag die visuele kenmerkstroom swaarder wil laat weeg, siende dat dit nie geaffekteer word deur akoestiese ruis nie.

Kenmerkstroomintegrasie is die strategie wat gevolg word vir die kombinering van die inligting onttrek uit die akoestiese en visuele spraakmonsters. 'n Besondere interessante probleem in OVOSR is dié van oudio-visuele asinkronisasie. Wanneer daar gepraat word vind die beweging van sigbare artikulators soos die lippe, tong en kaak plaas voordat die werklike klank geuiter word. Daar is dus 'n effense vertraging tussen die akoestiese en visuele kenmerkstrome. Hierdie vertraging is egter nie konstant nie, maar is afhanklik van die spesifieke klank wat geuiter word, sowel as die spesifieke spreker. Ons stel verskillende oudiovisuele modelle voor wat oudio-visuele asinkronisasie verskillend modelleer.

Waarskynlikheidsleer, met sy inherente begrippe van onsekerheid en vertroue, is 'n natuurlike benadering tot die oplossing van hierdie probleme. Ons fokus op die spesifieke klas van probabilistiese modelle wat formuleer kan word as Bayesiaanse netwerke (BNe) (Pearl, 1988). Hierdie klas model is besonder goed geskik vir die modellering van die kousaliteite inherent tot oudio-visuele spraak (Nefian *et al.*, 2002). In die besonder bespreek ons dinamiese Bayesiaanse netwerke (DBNe). Die DBN is 'n uitbreiding van BNe wat voorsiening maak vir die modellering van veranderlike lengte sekwensies van waargeneemde en onwaargeneemde toevalsveranderlikes soos sekwensies van kenmerke onttrek uit spraakmonsters. Ons bespreek dat die verskuilde Markov model, wat tot op hede die mees gewilde model in die spraakherkenningsliteratuur is, 'n spesiale geval van die DBN raamwerk is. Ons bespreek algemene afrigting en inferensie metodes vir BNe en DBNe in detail, en gee 'n volledige en selfstandige aanbieding van die literatuur wat andersins slegs beskikbaar is in verskeie publikasies.

In die konteks van OVOSR word die afrigting van DBNe uitgevoer d.m.v. die afskatting van parameters van oudio-visuele DBN modelle vanaf gemonsterde data. Ons bespreek maksimum waarskynlikheid afskatting in die vorm van die verwagting maksimering (VM) algoritme, en variasie-leer in die vorm van die variasie Bayes (VB) algoritme. Gegee 'n versameling van afgerigte modelle, wat byvoorbeeld verteenwoordigend is van verskillende woorde en foneme, wil ons bepaal watter model is mees verteenwoordigend van 'n onbekende oudiovisuele spraakmonster. Hierdie staan bekend as die klassifikasie probleem. Ons toon aan dat afrigting en klassifikasie uitgevoer kan word m.b.v. doeltreffende algemene algoritmes in die BN en DBN raamwerk. In die besonder bespreek ons die aansluiting boom ("junction tree") algoritme vir die BNe en die koppelvlak aansluiting boom ("interface junction tree") algoritme vir DBNe. Die koppelvlak aansluiting boom algoritme is 'n veralgemening van die klassieke vorentoe-agtertoe algoritme wat gebruik word in HMMs.

Ons implementeer 'n volledig funksionele OVOSR stelsels as deel van die navorsing. Die stelsel word gebruik om 'n stel eksperimente uit te voer waar ons die prestasie van 'n aantal voorgestelde modelle en afrigtingsalgoritmes vir OVOSR evalueer en vergelyk. In die eksperimente gebruik ons die Clemson Universiteit oudio-visuele eksperimente (CUAVE) data korpus vir die afrig en evaluasie van die modelle en algoritmes. Die CUAVE data korpus bestaan uit opnames van verskeie sprekers wat die syfers nul tot nege uitspreek. Die eksperimente is dus multispreker syferherkenningseksperimente. Ons is veral geïnteresseerd in die werking van die OVOSR by verskillende vlakke van ruisop verskillende vlakke van ruis en voeg dus kunsmatig Gaussiese ruis tot die akoestiese kenmerkstroom. Ons evalueer dan die prestasie van die stelsel by verskillende sein-tot-ruis (SNR) vlakke. Die vlak van ruis wissel vanaf -6  tot 18 dB SNR in stappe van 4 dB.

Die resultate van die eksperimente toon dat daar inderdaad inligting is in die visuele spraak modaliteit wat nuttig is vir spraakherkenning. Ons vind spesifiek vir die syferherkenningseksperiment wat slegs die visuele spraakinligting gebruik 'n misklassifikasie tempo van $25,6\%$. Wanneer slegs die akoestiese spraakinligting gebruik word, is die misklassifikasie tempo $84,2\%$ by die ergste ruisvlak van -6 dB. Ons vind ook dat modelle wat akoestiese en visuele kenmerke kombineer in die algemeen beter presteer as modelle wat net die akoestiese eienskappe gebruik. Die verbetering is duideliker by kleiner SNR vlakke, siende dat in hierdie gebied die visuele inligting kompenseer vir die akoestiese ruis. Selfs by groot SNR vlakke vind ons nogsteeds 'n statistiese beduidende verskil tussen die oudio-visuele klassifiseerder en die slegs-oudio klassifiseerder. Die persentasie fout verkry teen 18dB SNR met die oudio-visuele klassifiseerder is 1.1% teenoor die 2.7% verkry met oudio alleenlik. Die persentasie fout verkry deur die oudio-visuele klassifiseerder is dus meer as helfde minder as die van audio alleenlik.

Ons is in staat om die resultate van Glotin *et al.* Glotin *et al.* (2001) te herproduseer wat aangetoon het dat kenmerkstroomweging voordelig is vir OVOSR. Ons is ook in staat om die resultate van Liu *et al.* Liu *et al.* (2002) te herproduseer wat wys dat die kenmerkstroomintegrasie tegniek wat gemodeleer word deur die oudio-visueelgekoppelde VMM (OV-KVMM) beter is as die minder gesofistikeerde integrasie tegniek wat gemodeler word deur die oudio-visueelproduk VMM (OV-PVMM) en die oudio-visueelonafhanklike HMM (OV-OVMM). Die OV-KVMM, OV-PVMM en OV-OVMM is almal DBN modelle;dus is die algemene afrigting en inferense raamwerk van toepassing.

Die unieke bydrae van die navorsing is die toepassing van variasie Bayesiaanse (VB) leer tot oudio-visuele DBNe. Die VB leer metode is 'n alternatief vir die klassieke verwagting maksimering (VM) algoritme. Variasie leer lei tot outomatiese modelkompleksiteitseleksie en vermy die singulariteite en oorafrigtingsprobleme wat met EM met gepaard gaan. In die eksperimente vind ons dat oudio-visuele DBN modelle afgerig met VB is meer robuust tot ruis, waarskynlik a.g.v. hul meer kompakte vorm. Daar is egter nie voldoende bewyse om bewyse om aan te toon dat VB algoritme in die algemeen beter presteer beter as die VM wanneer oudio-visuele DBN modelle afgerig word nie. Inteendeel, vir seker ruisvlakke vaar die VB algoritme slegter as die VM algoritme wat die moontlikheid van oorgladde afrigting van die modelle met VB afrigting uitbeeld. Ons vind egter dat modelle wat afgerig is met VB afskatting ewe goed presteer as modelle wat afgerig is met VM afskatting wanneer die ruisvlakke laag is. Die outomatiese modelseleksie eienskap van die VB algoritme lei dan tot 'n meer kompakte model. So, deur die gebruik van variasionele metodes is ons in staat om net so goed te doen met minder. Dit is altyd 'n kosbare eienskap, veral in hulpbron-kritiese toepassings vir byvoorbeeld draagbare toestelle soos selfone.

# Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- My advisor, Professor Ben Herbst, for giving me the freedom to explore the research topic while always guiding me in the right direction whenever necessary. My co-advisor, Professor Johan du Preez, for many valuable discussions around the theory of speech recognition and HMMs, the design of experiments, and interpretation of results. My co-advisor, Dr. Herman Engelbrecht, for discussions around acoustic feature extraction, HMMs, and speech recognition. I also greatly appreciate the trust that was invested in my judgement and decisions during the course of the research by all advisors.
- The students and staff at the MIH Electronic Media Lab at Stellenbosch University and the members of the Vision and Learning at Stellenbosch group for ideas, comments and inspiration during the course of the research.
- MIH Holdings for providing funding for the research and for creating such an inspirational research environment in the Media Lab.

# Dedications

*This thesis is dedicated to to my parents for their patience, support, and love while pursing my dreams so far away from home for so long.*

# Contents

# List of Figures

# List of Tables

# List of acronyms

| | |
|---|---|
| CPD | Conditional probability density |
| PDF | Probability density function |
| BN | Bayesian network |
| DBN | Dynamic Bayesian network |
| DAG | Directed acyclic graph |
| MLE | Maximum likelihood estimation |
| EM | Expectation maximisation |
| VB | Variational Bayes |
| GMM | Gaussian mixture model |
| VB-GMM | Variational Bayes GMM |
| HMM | Hidden Markov model |
| AV-HMM | Audio-visual hidden Markov model |
| AV-PHMM | Audio-visual product HMM |
| AV-IHMM | Audio-visual independent HMM |
| AV-CHMM | Audio-visual coupled HMM |
| MFCC | Mel-frequency cepstrum coefficient |
| AAM | Active appearance model |

# Chapter 1

# Introduction

Dave : Hello, HAL. Do you read me, HAL?

HAL: Affirmative, Dave. I read you.

Dave : Open the pod bay doors, HAL.

HAL: I'm sorry, Dave. I'm afraid I can't do that.

Dave : What's the problem?

HAL: I think you know what the problem is just as well as I do.

Dave : What are you talking about, HAL?

HAL: This mission is too important for me to allow you to jeopardize it.

Dave : I don't know what you're talking about, HAL.

HAL: I know that you and Frank were planning to disconnect me, and I'm afraid that's something I cannot allow to happen.

Dave : Where the hell'd you get that idea, HAL?

HAL: Dave, although you took very thorough precautions in the pod against my hearing you, I could see your lips move.

*From the movie "2001: A Space Odyssey"*

## 1.1 Background

It has been documented since the 17th century that there is useful information conveyed about speech in the facial movements of a speaker (?). This is in particular manifested in the ability of hearing-impaired listeners to be able to understand speech from visual cues only. This ability is known as *lip-reading* or *speech-reading*. However, even for listeners with normal hearing, being able to see the face of the speaker is known to significantly improve speech intelligibility, especially under noisy conditions. Taylor (1987) shows that some speech sounds which are easily confused in the audio domain (for instance /b/ and /v/,/m/ and /n/) are distinct in the visual domain. In addition, there is evidence that visual information is used to compensate for elements in the audio signal that are vulnerable to acoustic noise.

Motivated by this multi-modal manner in which humans perceive their environment, research in audio-visual automatic speech recognition (AVASR) is focused on the integration of the acoustic and visual speech modalities with the purpose of improving accuracy and robustness of automatic speech recognition systems. The objective of AVASR is to combine acoustic and visual speech cues such that recognition performance is better than what is possible using either modality alone.

Early evidence that vision can improve automatic speech recognition was presented by Petajan (1984), who used the then current technology of dynamic time-warping with visual features derived from mouth opening and showed that the audio-visual system was better than either speech or vision alone. In the 1980s, the development of hidden Markov models (HMMs) (Rabiner, 1989) improved speech recognition accuracy and made possible large-vocabulary continuous speech recognition (LVCSR). HMMs were first applied to visual speech recognition by Goldschen (1993) using an extension of Petajan's mouth blob extraction method. Various approaches have since been tried with visual and audio-visual speech recognition. Some notable reviews may be found in Chen and Rao (1998), Neti *et al.* (2000), Potamianos *et al.* (2003) and Potamianos *et al.* (2004).

Acoustic speech recognition is now "solved" to the extent that we have speech recognition systems that run on personal computers, mobile phones and in vehicles, to name a few. However, there is much room for improvement regarding robustness to such factors as different speakers, accents, microphones, and environmental noise. The fundamental requirements for any acoustic speech recognition system is feature extraction, model learning, and classification (Rabiner and Juang, 1993). In addition to these requirements that are inherited from classic acoustic speech recognition, the following problems need to be addressed in AVASR.

- Identification and extraction of informative visual features;
- Optimal integration of the acoustic and visual features.

The first problem, that of extracting visual speech features, has naturally received much less attention than its acoustic counterpart. A major problem of extracting visual features is the enormous amount of data in video sequences, a problem that is common to most computer vision systems. Each video frame contains thousands of pixels whereas in most application we are only able to handle feature vectors of sizes between 10 and 100 elements. Additionally, these features should be robust to such variables as different speakers, head poses and lighting conditions.

There are many possible approaches to reducing speech image data to lower-dimensional feature vectors. At one end of the spectrum are approaches where features are estimated directly from the image, for example statistical analysis of pixel intensities such as the *eigenlips* technique (Bregler and Konig, 1994). At the other end are approaches where a priori information, assumptions and expert knowledge are encapsulated by a model and the features extracted by fitting the model parameters to images (Matthews *et al.*, 2002). We would expect the first approach to avoid systematic model errors and the second approach to be more resistant to noise. However, between these extremes lie many possibilities. State-of-the art solutions are based on active appearance models (Cootes *et al.*, 2001), where there has recently been much work on optimising the models for visual speech (Papandreou *et al.*, 2009).

The second problem, integration of audio and visual features, is the main focus of the research presented in this thesis. By *optimal stream integration* we mean that we are looking for the best strategy for combining the two feature streams so that the resulting system is superior to any system that could be built from any one of the feature streams by itself. Probability theory, with its inherent notions of uncertainty and confidence, is a natural approach to this problem.

We have chosen to focus on the specific class of probabilistic models that can be formulated as Bayesian networks (BNs) (Pearl, 1988). Bayesian networks are particularly well-suited to modelling the causalities inherent to audio-visual speech (Nefian *et al.*, 2002) and allows for the development of general and efficient inference and learning algorithms to be developed. We shall in particular see that an extension to Bayesian networks called dynamic Bayesian networks (DBNs) (Murphy, 2002) allows us to model variable-length sequences of observations. This is a fundamental requirement for speech recognition applications where recorded speech samples (audio and video) typically are of variable length. A classical example of a DBN is the hidden Markov model (HMM) (Rabiner, 1989) which is used in many state-of-the-art speech recognition systems.

The DBN models that we propose for solving the stream integration problem of AVASR are essentially extensions of the classic HMM model. One such extension is the *audio-visual product HMM* (AV-PHMM). The AV-PHMM allows for weighting individual feature streams according to some measure of the reliability in each stream. Stream weighting is necessary as the acoustic and visual modalities typically differ in information content and noise. However, the classic HMM formulation requires us to concatenate acoustic and visual features into a single feature vector, therefore giving little opportunity of weighting the individual streams. The classic forwards-backwards HMM (Rabiner, 1989) applied to AVASR using feature vector concatenation is referred to as the *audio-visual HMM* (AV-HMM). The AV-PHMM, however, treats the two observations streams as separate observation models that can be weighted independently. In the AV-PHMM the two observation models share the same HMM state space.

In addition to stream weighting, DBNs allow for modelling asynchrony between acoustic and visual feature streams. This is necessary since, when speaking, the motion of visible articulators such lips, tongue and jaw precedes the actual sound being uttered. Thus, there is a slight delay between the visual and acoustic speech modalities. This delay is referred to as *audio-visual synchrony.* The delay is not constant, but depends on the particular sound that is being uttered as well as the speaker. In part this offset is caused by different channel delays, but is also affected by forward-articulation as described in Benoit (1992). In Bregler and Konig (1994) this delay is estimated to be approximately 120 ms on average.

In order to allow asynchrony between the two feature streams we can consider using two separate HMMs for the acoustic and visual feature streams. We can then weight the outputs of each model separately, according to measured stream reliability. This method of stream integration is referred to as *decision fusion* (Nefian *et al.*, 2002) and the resulting model as the *audio-visual independent HMM* (AV-IHMM)

The decision fusion approach may fail to capture the natural correlation that exists between the acoustic and visual observations as the modalities are considered independent by the model. However, in the DBN framework it is possible to model the acoustic and visual feature streams independently using two separate HMMs but *couple* the two HMMs at the state-level. In this way, we are able to allow audio-visual asynchrony in the model while controlling the level of asynchrony and maintaining the natural correlation between the two feature streams. The resulting model is called the *audio-visual coupled HMM* (AV-CHMM).

## 1.2 Speech recognition fundamentals

We shall briefly review some of the fundamental concepts in speech recognition.

Most large-vocabulary continuous speech recognition (LVCSR) systems consist of a language model, a pronunciation model, and an acoustic model. Language models capture regularities in spoken language and are used in speech recognition to estimate the probability of isolated words or word sequences. The most popular statistical method in use is the $n$-gram model, which attempts to capture the syntactic and semantic constraints of the language by estimating the frequencies of sequences of $n$ words. Language models are usually trained from manually transcribed speech corpora and general text corpora. The required size and domain of such a corpus depends on the actual speech recognition task, for instance small-vocabulary or large-vocabulary or application domain.

For large vocabularies, many of the possible words seldom or never occur in the transcribed training corpus. As a result, there is not enough training data describing a large portion of words. As such, most speech recognition systems do not learn acoustic models for every word in the vocabulary. Instead, words are divided into smaller linguistic units such as phonemes or phones (Rabiner and Juang, 1993). A phoneme is defined as the smallest segmental unit of sound that can change the meaning of a utterance. For example, the /t/ and /d/ sounds in the words **t**ip and **d**ip are different phonemes, as substituting one with the other alters the meaning of the word. Phones also represent different sounds, but may not necessarily alter the meaning of the word. For example, the /k/ sound in **k**it and s**k**ill are different sounds. However, substituting one for the other would not alter how a listener would perceive the meaning of the word although the pronouncing might sound strange. There is no standard phoneme or phone set size as it varies depending on the threshold set for what constitutes a separate phoneme or phone. Common phoneme set sizes used for English range from 40 to 48 phonemes.

Modern speech recognition systems typically use HMMs as acoustic models of sub-word units such as phonemes. Each sub-word unit has a separate HMM with its own set of parameters. There are many variations on this concept. For instance, it is common to model phoneme context instead of isolated phonemes as certain phonemes sound different depending on the context in which the phoneme appears (Rabiner and Juang, 1993). It also common to tie parameters between models to reduce the total number of parameters required by the system.

In most LVCSR systems, the recognised sub-word units are used by the pronunciation model to determine which words are spoken. The recognised words are in turn used by the language model to determine the most likely word sequence given the individual words. Thus, LVCSR systems are typically hierarchical in nature. In small-vocabulary speech recognition systems it is common to model each word in the vocabulary directly as an acoustic model instead of using a pronunciation model. In an isolated word recognition system the $n$-gram model is replaced with a single discrete probability distribution $p(w_i)$ over the words in the vocabulary. If all words are equally likely to occur, for instance in the case of digit recognition, the language model can be ignored all together.

## 1.3 Visual speech

The field of visual speech has naturally received significantly less attention than its acoustic counterpart. Nevertheless, some studies have been performed in order to discover the characteristics of visual speech (McGurk and MacDonald, 1976), (Taylor, 1987), (Benoit, 1992), (Liew and Wang, 2009).

Just as phonemes are said to represent the smallest unit of sound that can be said to change the meaning of an utterance, a similar concept exists for visual speech. These smallest units of visual speech are referred to as

**Table 1.1:** Phoneme to viseme mapping (Lucey *et al.*, 2004).

| Phoneme | Viseme | Phoneme | Viseme |
|---|---|---|---|
| P | /p/ | K | /k/ |
| B | | G | |
| M | | N | |
| EM | | L | |
| F | /f/ | NX | |
| V | | HH | |
| T | /t/ | Y | |
| D | | EL | |
| S | | EN | |
| Z | | IY | /iy/ |
| TH | | IH | |
| DH | | AA | /aa/ |
| DX | | AH | /ah/ |
| W | /w/ | AX | |
| WH | | AY | |
| R | | ER | /er/ |
| CH | /ch | AO | /ao/ |
| JH | | OY | |
| SH | | IX | |
| ZH | | OW | |
| EH | /ey | UH | /uh/ |
| EY | | UW | |
| AE | | SIL | |
| AW | | SP | |

visemes. Table 1.1 shows an example phoneme set of 48 phonemes grouped by their respective viseme classes, taken from Lucey *et al.* (2004). As we might have expected, there are far fewer visemes than phonemes. This many-to-one relationship between phonemes and visemes reflects the fact that visual speech in general contains less information than acoustic speech. Interestingly we also note that phonemes that are easily confusable acoustically such as “D” and “B” belong to separate viseme classes and therefore should be more distinguishable in the visual domain. This is also the case for other confusable sets of phonemes (Chen, 2001) and provides an additional motivation for AVASR research.

The multimodal nature of speech is also illustrated by the McGurk effect (McGurk and MacDonald, 1976). The McGurk effect refers to a perceptual phenomenon where a person who is hearing the sound /ba/, but watching the sound /ga/, instead perceives the sound /da/. The McGurk effect clearly demonstrates that human speech perception is indeed multimodal and thus justifies the pursuit of developing automatic audio-visual speech recognition systems.

## 1.4 Applications

Application areas for AVASR include those of traditional automatic speech recognition (ASR) some of which are:

**Speech-to-text processing.** Speech-to-text processing is a classic application of speech recognition. With the ability to transcribe speech directly into text, computer users can for instance perform word processing or send emails without using a keyboard. As many modern PCs are equipped with a user-facing camera, speech-to-text is a natural application of AVASR, in particular in noisy work environments where traditional acoustic-only ASR systems may fail.

**Accessibility.** Speech recognition is a popular human-computer interface for people with disabilities who often find traditional computer interfaces such as a keyboard and a mouse difficult or impossible to use. AVASR can improve existing human-computer speech recognition interfaces, in particular in noisy acoustic environments.

**Automatic captioning.** An interesting application of AVASR is to automatically generate captions of video content where the speaker's face is visible. Applications include video blogs, news broadcasts and YouTube clips. Using automatic translation technologies the transcriptions may also be translated into different languages.

**Data mining and search.** With the exponential growth of multimedia content on the Internet, searching and organising this content becomes increasingly important. AVASR is well-suited for digital content that combines speech and video such as video blogs, online news broadcasts, and YouTube clips.

**Mobile and handheld devices.** With the increased popularity and capability of advanced mobile handsets and handheld devices, overcoming the user interface limitation of such small devices is an crucial problem. As most high-end devices come equipped with camera and microphone, speech and visual-speech is an interesting way of interfacing with such a device. For instance, Google Voice Search allows the user to perform searches in Google's search engine through voice only. For AVASR to be used with such devices it would be desirable for the device to have a camera that is facing the user. In mobile applications there is typically little control over the acoustic environment, justifying the application of AVASR.

**Vehicles** The advent of intelligent transportation systems has brought new technologies inside vehicles. Vehicle operators in the future will be able to access greater information than is provided by current instrument panel displays and controls. Navigation, route guidance, traffic management information, collision avoidance, communication systems, and alternative methods for displaying and controlling vehicle information (speed, audio, climate control, engine status, and warning tell-tales) are just some examples of the new technologies proposed to improve driving performance, comfort, and convenience. However, conventional system interfaces are impractical as the driver's arms are required for operating the vehicle. As such, there is a great interest in in-vehicular voice interfaces. Automotive cockpit environment are often severely contaminated by acoustic noise from the surrounding environment, rendering AVASR a particularly interesting option.

## 1.5 Objectives of the study

The objectives of the research are as follows:

- Provide a comprehensive review of the theory of Bayesian networks and dynamic Bayesian networks and inference and learning algorithms in such networks. Integrate theory from multiple sources in the literature and providing additional detail as necessary.
- Provide enough details such that an implementation is straightforward.
- Apply the theory of Bayesian networks to solving the stream integration, learning, and classification problems in AVASR. In particular derive models that allows modelling stream weighting and audio-visual asynchrony.
- Provide a conceptual overview of the components comprising an AVASR system, and outline how such a system is implemented in practice.
- Implement a full-featured AVASR system.
- Use the AVASR system to perform the following experiments:
  - Compare the performance of an AVASR model where there is no stream weighting (AV-HMM) versus a model that features stream weighting (AV-PHMM).
  - Compare the performance of AVASR models with different asynchrony properties (AV-PHMM, AV-IHMM and AV-CHMM).
  - Evaluate the performance of AVASR compared to **(a)** classic audio-only speech recognition and **(b)** visual-only speech recognition (automatic lip-reading)

- Compare the performance of AVASR models learned using maximum likelihood estimation versus models learned using variational learning.

## 1.6 Contribution

The main contributions of the research is a comprehensive treatment of Bayesian network theory in Chapter 3, that covers literature otherwise only available from multiple different sources, and the experimental results presented in Chapter 5.

The treatment of Bayesian network theory in Chapter 3 provides a coherent presentation of theory found in Murphy (2002), Jordan (2003) and Bishop (2007). These references focus on the general theory of machine learning and Bayesian networks and as such leaves much for the reader to "figure out". Most of the examples given in literature are provided for illustration purposes only, and as such do not typically provide sufficient detail for solving complex real-life problems. In this thesis we focus on the particular application of AVASR and thus we are able to provide more detail in the derivation of the theoretical results while maintaining a coherent relationship between theory and a non-trivial real-life application. In particular the details provided in the derivation of the expectation maximisation (EM) algorithm and VB algorithm for Bayesian networks of latent multinomial variables and observed Gaussian variables in Section 3.3 and 3.5, respectively, is to the best of our knowledge not available from other resources.

The Bayesian network framework allows us to develop general inference and learning algorithms. Thus, when we later propose several audio-visual DBN models for solving problems in AVASR we do not have to derive learning and inference algorithms for each model; it is all taken care of by the general framework.

In the experiment chapter (Chapter 5) we reproduce the results presented in Neti *et al.* (2000) that shows that the visual channel indeed contains valuable speech information and that by combining acoustic and visual speech it is possible to build speech recognition systems that perform better than classical audio-only ASR systems in noisy acoustic environments.

We are also able to reproduce the results presented in Glotin *et al.* (2001) which shows that weighting the acoustic and visual feature streams according to measured stream reliability is essential to AVASR.

Finally, we are able to reproduce the results presented in Nefian *et al.* (2002) which show that the AV-CHMM performs better than the AV-PHMM and AV-IHMM models confirming our intuition that audio-visual asynchrony should be allowed while restricting the amount of asynchrony in order to exploit the natural correlation between the acoustic and visual feature streams.

The novel contribution of the research is the application of variational learning to AVASR models. Variational learning has been shown to avoid problems associated with maximum likelihood learning such as overfitting and singularities and also leads to automatic model complexity selection by giving preference to models with fewer parameters that equally well approximates the training data. As such variational learning adheres to the principle of *Occam's razor* (Sober, 1996).

Variational learning has been successfully applied to speech recognition in Somervuo (2002), Valente and Wellekens (2003) and Watanabe *et al.* (2003). However, these studies are limited to the application of variational learning to acoustic speech modelled as Gaussian Mixture Models (GMMs) (Bishop, 2007). GMMs assume that the observed features are *independent and identically distributed* (i.i.d.) which in general is not the case for speech data where features are typically highly context-dependent. This context dependency is modelled in HMMs while still allowing efficient inference algorithms to be used, which has made HMMs so popular for speech recognition applications where performance is often crucial. Attias (2000) shows that, in theory, variational learning can be performed efficiently in general Bayesian networks. McGrory and Titterington (2006) applies variational learning in HMMs and observe the same model complexity selection properties that is observed in GMMs. However, McGrory and Titterington (2006) only consider artificial data and one-dimensional real-world time-series data. The high dimensionality of speech features may yield entirely different results.

This research provides the first application of variational learning to speech recognition with more sophisticated models than GMMs. In particular we perform variational learning in the AV-CHMM model. The results show that we are not able to reproduce the superior performance of variational learning as reported by Valente and Wellekens (2003) and Watanabe *et al.* (2003) when applying variational learning to the more sophisticated AV-CHMM model used in AVASR. However, we did find the AV-CHMM model learned using variational methods appear to be more robust to noise than the same model learned using maximum likelihood learning. This is possibly a result of the more compact form of the variational AV-CHMM model. We also find that the model learned using variational learning perform equally well as models learning using maximum likelihood estimation for the smallest levels of acoustic noise while the automatic model selection property yields a more sparse representation. Thus, by using variational methods we are able to do "just as well with less". This is always a desirable property, in particular in resource critical applications such as in handheld devices.

As part of the research a complete AVASR system was implemented. The tools needed to perform visual feature extraction was implemented by us from scratch (Reikeras *et al.*, 2010*a*) and is available at

`https://bitbucket.org/helger/pyaam`

under the GPL license. The remaining main components of the system are

DBN inference and learning and acoustic feature extraction. The audio-visual DBNs are implemented using the Bayesian network toolbox (Murphy, 2001) by Kevin Murphy. MFCCs are calculated using Talkbox (Cournapeau, 2008) by David Cournapeau.

The research for this thesis resulted in the submission of two peer-reviewed papers. Reikeras *et al.* (2010*b*) passed the review and has been published. Reikeras *et al.* (2010*a*) is still under review at the time of writing.

## 1.7 Overview of the rest of the thesis

In Chapter 2 we review existing literature in the field of AVASR and set the context for our research within the research field.

In Chapter 3 we present the theoretical framework of Bayesian networks and dynamic Bayesian networks. We discuss representation, inference and learning, and give several examples of Bayesian networks including Gaussian mixture models and hidden Markov models.

In Chapter 4 we describe the various components comprising the proposed AVASR system including feature extraction, feature stream integration, learning and classification. The system is used to conduct the experiments presented in Chapter 5. In the experiments the performance of several audio-visual probabilistic models are evaluated and compared. We also compare the performance of models learned using maximum likelihood and variational learning methods. Performance is tested using the Clemson University audio-visual experiments (CUAVE) database which consists of 36 speakers uttering the digits from zero to nine. Thus the experiments are multi-speaker digit recognition experiments.

Finally, in Chapter 6 we summarise the main contributions of the research, present our conclusions, and propose interesting directions for future research in AVASR. In addition to the main chapters there are three appendices. Appendix A lists important results from probability theory and the probability distributions that we have used. In Appendix B we list tables of results from significance tests performed on the data from our experimental results. In Appendix C we briefly present the software that was developed as a part of the research.

# Chapter 2

# Literature review

The idea of using visual information in speech recognition has been around for a long time. It has been documented since the 17th century (?) that there is useful information conveyed about speech in the facial movements of a speaker. McGurk and MacDonald (1976) first described the *McGurk effect*. The McGurk effect is a perceptual phenomenon whereby a listener who is hearing the sound /ba/ but watching the sound /ga/ instead perceives the sound as /da/. The McGurk effect is a core motivational factor for AVASR research as it clearly demonstrates that human speech perception is indeed multimodal. The emergence of AVASR as an active research field is relatively recent and has to a large extent been enabled by the significant increase in computational resources available to researchers and industry.

Petajan (1984) and Petajan *et al.* (1988) present an automatic lip-reading method that uses vector quantisation, dynamic time warping, and a heuristic distance measure. The system uses a codebook of prior images that are used to translate novel images into corresponding symbols. The symbol strings are then compared to stored sequences representing different words in the vocabulary. Results from combined acoustic and visual speech recognition are also presented that show improved performance compared to an acoustic recognition system alone on a small-vocabulary speech recognition task. Others mapped power spectra from static images (Yuhas *et al.*, 1989), or used optic flow (Mase and Pentland, 1991) as visual features and achieved similar results.

Yuhas *et al.* (1989) use artificial neural networks (ANNs) as an improvement to the early encoding scheme used by Petajan *et al.* (1988). The proposed system is evaluated on a vowel recognition task. Results of integrating the visual and auditory signals for vowel recognition in the presence of acoustic noise is presented. The results show that the proposed system performs significantly better in noisy acoustic environments than a corresponding audio-only system. The performance increase in clean acoustic environments is marginal. In the 1980's, the development of hidden Markov models (HMMs) (Rabiner, 1989) improved speech recognition accuracy and enabled large-vocabulary continuous speech recognition (LVCSR). HMMs are first applied to visual speech recognition by Goldschen (1993) using an extension of the mouth blob extraction method of Petajan (1984).

Another ANN based system proposed by Bregler and Konig (1994) uses active contour models or "snakes" (Kass *et al.*, 1988) to perform automatic lip-tracking. PCA models trained from manually annotated images are used to constrain the contour search space, improving the robustness of the tracker. A technique named *eigenlips*, which is similar to eigenfaces (**?**) used in facial recognition, is used to model image texture over the region of interest. A hybrid multilayer perceptron (MLP) and hidden Markov model (HMM) method is used to integrate visual and acoustic speech modalities. The proposed model accounts for asynchrony between the acoustic and visual modalities. Maximum mutual information is used to measure the average delay between the two modalities. The delay is found to be approximately 120ms. The performance of the system is evaluated on a German multi-speaker database consisting of connected letters. Experiments are performed in clean and nosy acoustic environments. The system showed no performance increase over audio-only speech recognition in the clean acoustic environment. However, in the noisy environment the audio-visual system performed significantly better than the audio-only one.

Neti *et al.* (2000) contains the most comprehensive review of AVASR to date. The report is a result of a workshop held at the Center for Language and Speech Processing at the John Hopkins University. It demonstrates for the first time that LVCSR performance can be improved by use of visual information in the case of non-noisy audio. The report discusses state-synchronous and phone-synchronous decision fusion using the multi-stream and product HMM, respectively. The report also shows that for speech contaminated by "babble" noise at 10 dB SNR, the recognition performance can be improved by 27% in relative word error rate reduction compared to an equivalent audio-only recogniser subject to the same noise model.

There is an extensive amount of literature available on the theory of Bayesian networks and graphical models. An introduction can be found in Jordan (2003). The most comprehensive reference to date regarding DBNs in particular is available in Murphy (2002).

DBNs for AVASR is introduced in Nefian *et al.* (2002). Several DBN models are proposed including both synchronous and asynchronous models. Some of the models are used in earlier AVASR applications although independent from the framework of DBNs. Nefian *et al.* (2002) unifies these models in the DBN framework and proposes several novel models inspired by the framework. Experimental results are presented comparing audio-visual to audio-only and video-only models on a small-vocabulary recognition task with different levels of acoustic noise. Stream exponents are used to weight the acoustic and visual modalities according to the level of acoustic noise. The stream exponents are estimated discriminatively. It is found that overall the audio-visual model performs better than audio-only and visual-only models with a significant performance increase in the presence of acoustic noise. However, even in reasonably clean audio the audio-visual model performs marginally better than the audio-only. As the visual-only model is not affected by acoustic noise its performance is fixed at 33.1% misclassification rate on the small-vocabulary task. At a 10dB SNR level the audio-visual model has 34.3% and the audio-only model 85% misclassification rate. In clean audio the audio-visual model has 1.9% and the audio-only 3.1% misclassification rate. A second experiment compares the performance of different audio-visual DBNs. It is found that the AV-CHMM performs best amongst the audio-visual DBN models considered (including AV-HMM, AV-IHMM and AV-PHMM).

An updated review of the state of AVASR is presented in Potamianos *et al.* (2004). The DBN framework has continued to attract the interest of AVASR researchers Gowdy *et al.* (2004), Hershey *et al.* (2004), Saenko and Livescu (2006), Lv *et al.* (2007), and Chu and Huang (2007).

Variational learning for Bayesian networks is first introduced in Attias (2000) who discusses the variational Gaussian mixture model (VB-GMM) as an example. A comprehensive treatment of variational methods in general is given in Wainwright and Jordan (2008). Somervuo (2002) and Valente and Wellekens (2003) applies variational learning to Gaussian mixture models for acoustic speech recognition. In their experiments VB-GMM is shown to perform better than the standard EM algorithm, its convergence is faster, and it avoids the problem of overfitting associated with standard EM. Watanabe *et al.* (2003) applies variational learning to construct shared-state triphones used in LVCSR. McGrory and Titterington (2006) applies variational learning in HMMs and observe the same model complexity selection properties that is observed in GMMs. However, McGrory and Titterington (2006) only consider artificial data and one-dimensional real-world time-series data. The high dimensionality of speech features may yield entirely different results. To date, no one has applied variational learning to HMMs with application to speech recognition or any of the models that are used in AVASR. Thus the performance of variational learning in more complex models than the GMM for high-dimensional speech data remains unknown.

AVASR still remains an active area of research with many challenging problems yet to be solved. Some issues that have been addressed recently are as follows. Marcheret and Libal (2007) introduces dynamic stream weighting and shows that their dynamic weighting scheme is superior to a static weighting scheme. This idea is further developed in Gurban *et al.* (2008). Terry and Katsaggelos (2008) proposes a hierarchical DBN that models every aspect of speech recognition including language, words, phonemes, visemes, acoustics and vision. The book of Liew and Wang (2009) present a collection of selected papers on visual-only speech recognition. Most of this material is also highly relevant in the context of audio-visual speech recognition.

An interesting area of research is visual feature extraction. There are many ways of reducing speech image data to feature vectors. At one end of the spectrum is the approach where features are estimated directly from the image, for example the statistical analysis of pixel intensities such as *eigenlips* (Bregler and Konig, 1994). At the other end of the spectrum is the analytical approach where a priori information, assumptions and expert knowledge are encapsulated into a model and features are extracted by fitting model parameters to images. We would expect the first approach to avoid systematic model errors and the second approach to be more resistant to noise. However, between these extremes lie many possibilities. Based on their work on AAMs Matthews *et al.* (2002) proposes to use AAMs for extracting visual speech features. The experiments yield encouraging results and establish the validity of using AAMs as a method of visual speech feature extraction.

Papandreou *et al.* (2009) make several novel improvements to existing AVASR techniques. A dynamic adaptation method for multimodal fusion schemes in changing environmental conditions is presented. The adaptive fusion method is based on measuring the reliability of individual feature streams. The method can be integrated with any of the common DBN models found in AVASR. The paper also introduce the *visemic* AAM that addresses the problem of non-speech related information and speaker dependency associated with traditional AAMs.

Due to the lack of a standard and publicly available AVASR data corpus, open source software, and benchmarking results, it is difficult to compare the performance of different approaches to AVASR found in the literature. As a result there is also no extensive comparison of different methods available. However, the results presented in Papandreou *et al.* (2009) suggests that their system based on visemic AAMs, DBNs and dynamic stream weighting represents the state of the art in AVASR research.

# Chapter 3

# Theoretical framework

## 3.1 Introduction

The main focus of our research is the investigation of how the framework of Bayesian networks can be used to solve problems in AVASR. In particular, we are interested in the stream weighting, stream integration, learning, and classification problems. In this chapter we discuss how Bayesian networks are represented using graph theory and probability theory and how we can estimate model parameters of Bayesian networks from data. In Chapter 4 we use the results from this chapter to solve the stream weighting, stream integration, and classification problems.

We first discuss Bayesian networks and then we describe dynamic Bayesian networks (DBNs). A DBN is an extension to Bayesian networks (static Bayesian networks) that models dynamic systems such as speech. We first focus on the theory of static Bayesian networks. Once we have the theoretical foundation and intuition from static Bayesian networks in place we shall move on to discussing DBNs. We shall see that many of the results from the analysis of static Bayesian networks are transferable to DBNs.

## 3.2 Bayesian networks

### 3.2.1 Representation

We now discuss how to represent a Bayesian network, following the approach in Bishop (2007, Chapter 8).

A Bayesian network (Pearl, 1988) consists of a set of nodes together with a set of directed edges connecting the nodes. The direction of an edge is indicated by an arrowhead when drawing the graph. This situation is shown in Figure 3.1.

A Bayesian network must constitute a *directed acyclic graph* (DAG) meaning that, in addition to arcs having direction, there must be no cycles within

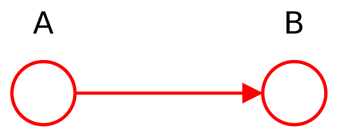

**Figure 3.1:** A simple Bayesian network consisting of two nodes. The direction of the arrow is from $A$ to $B$ representing the statement that $A$ and $B$ are dependent.

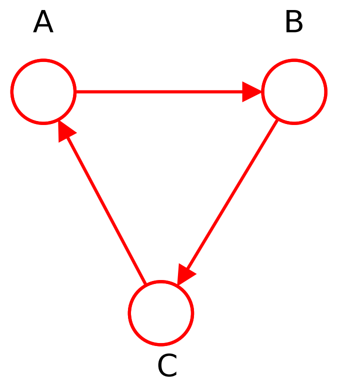

**Figure 3.2:** This graph is cyclic and therefore does not constitute a Bayesian network.

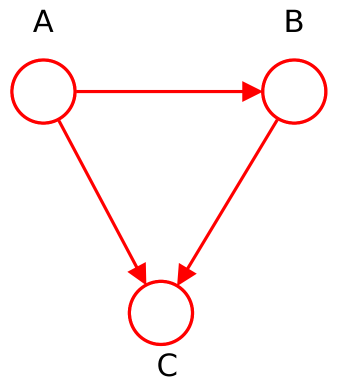

**Figure 3.3:** This graph is an acyclic directed graph and thus represents a Bayesian network.

the graph. For instance, the graph in Figure 3.2 is not a DAG and therefore not a Bayesian network. The graph shown in Figure 3.3, however, is both acyclic and directed, and therefore satisfies the conditions for a Bayesian network.

More formally, we represent a Bayesian network as a DAG denoted by $G(V, E)$, where $V$ is a set of nodes and $E$ is the set of edges connecting the nodes, with an associated probability distribution. Let $\mathbf{x}_V = \{\mathbf{x}_v : v \in V\}$ be a set of random variables indexed by the nodes on the graph. Then $p(\mathbf{x}_V)$ is the joint probability distribution over these random variables. For instance, for the Bayesian network in Figure 3.3 we have

$$p(\mathbf{x}) = p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) \tag{3.2.1}$$

where $V = \{A, B, C\}$.

We denote the set of parents of a node $v$ as $\pi_v$ and we shall allow sets of indices to be used wherever a single index is used. Thus, $\mathbf{x}_{\pi_v}$ denotes the set of random variables indexed by the parents of $v$. For each random variable $\mathbf{x}_v$ we associate a *conditional probability distribution* (CPD). The conditioning set of a CPD associated with $\mathbf{x}_v$, i.e. the set of variables on which the CPD is conditioned, is defined as the set of parents $\mathbf{x}_{\pi_v}$ of $\mathbf{x}_v$.

The joint probability distribution over all variables $p(\mathbf{x}_V)$ is defined as

$$p(\mathbf{x}_V) = \prod_{v \in V} p(\mathbf{x}_v | \mathbf{x}_{\pi_v}). \tag{3.2.2}$$

That is, the joint probability distribution over all random variables in the Bayesian network is the product of CPDs. For instance, the joint distribution for the Bayesian network in Figure 3.3 is

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_C | \mathbf{x}_A, \mathbf{x}_B) p(\mathbf{x}_B | \mathbf{x}_A) p(\mathbf{x}_A). \tag{3.2.3}$$

We often make explicit reference to the dependency of the model on a set of model parameters $\boldsymbol{\theta}$. We then write the joint probability distribution as $p(\mathbf{x}_V | \boldsymbol{\theta})$. Assuming that each CPD has an individual set of parameters (3.2.2) becomes

$$p(\mathbf{x}_V | \boldsymbol{\theta}) = \prod_{v \in V} p(\mathbf{x}_v | \mathbf{x}_{\pi_v}, \boldsymbol{\theta}_v) \tag{3.2.4}$$

where $\boldsymbol{\theta}_v$ are the parameters associated with the conditional probability distribution $p(\mathbf{x}_v)$. In many applications we are interested in models with CPDs that *share the same parameters*. We shall see examples of this situation later in the chapter.

### 3.2.2 Conditional independence

Central to the theory of graphical models and Bayesian networks is the concept of conditional independence. The discussion follows that of Bishop (2007). We say that a variable $\mathbf{x}_A$ is conditionally independent of $\mathbf{x}_B$ given $\mathbf{x}_C$ if

$$p(\mathbf{x}_A, \mathbf{x}_B | \mathbf{x}_C) = p(\mathbf{x}_A | \mathbf{x}_C) p(\mathbf{x}_B | \mathbf{x}_C). \tag{3.2.5}$$

Consider the simple Bayesian network shown in Figure 3.4. From (3.2.2) we have that

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_A | \mathbf{x}_C) p(\mathbf{x}_B | \mathbf{x}_C) p(\mathbf{x}_C). \tag{3.2.6}$$

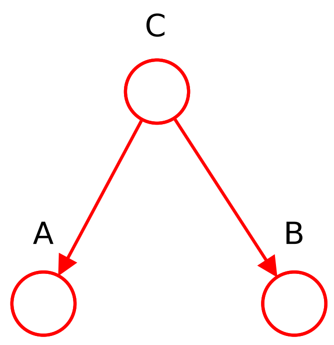

**Figure 3.4:** Example of conditional independence in a simple Bayesian network.

From the product rule of probability (A.1.2),

$$p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) = \frac{p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)}{p(\mathbf{x}_C)} \tag{3.2.7}$$

which, using (3.2.6), gives

$$p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) = p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_B|\mathbf{x}_C). \tag{3.2.8}$$

That is, $\mathbf{x}_A$ and $\mathbf{x}_B$ are conditionally independent given $\mathbf{x}_C$.

Conditional independence is denoted by the $\perp\!\!\!\perp$ symbol. For the above example we have

$$\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C. \tag{3.2.9}$$

We are often interested in the posterior distribution of the variables in a Bayesian network given that the values of a subset of variables have been observed. The posterior distribution is obtained by conditioning the joint distribution on the observed variables. For instance, if $\mathbf{x}_C$ is observed in Figure 3.4 the posterior distribution is given by (3.2.8). We often indicate the observed variable when drawing the graph by shading the conditioned variables as shown in Figure 3.5 for the case when $\mathbf{x}_C$ is observed.

#### 3.2.2.1 D-separation

Conditional independence plays an important role when using probabilistic models for machine learning as it often significantly simplifies the structure of the model and allows efficient inference algorithms to be used. An elegant and important feature of Bayesian networks is that conditional independence properties can be read directly from the graph. To describe how, we first need to understand the concept of *d-separation.*

The concept of d-separation is most easily understood from a few examples. In the first example we again consider the graph in Figure 3.4. Recall that the joint probability distribution represented by this graph is given by

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_B|\mathbf{x}_C)p(\mathbf{x}_C). \tag{3.2.10}$$

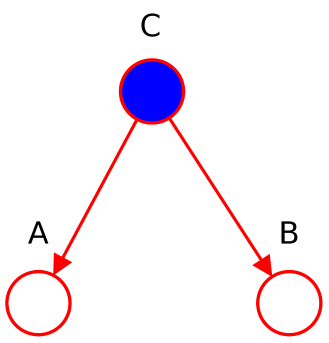

**Figure 3.5:** In this graph $\mathbf{x}_A$ is conditionally independent of $\mathbf{x}_B$ once $\mathbf{x}_C$ is observed.

First we consider the case where the conditioning set is the empty set $\emptyset$. Using the above expression we investigate whether $\mathbf{x}_A$ and $\mathbf{x}_B$ are independent given the empty set by marginalising both sides of (3.2.10) with respect to $\mathbf{x}_C$ to give

$$p(\mathbf{x}_A, \mathbf{x}_B) = \sum_{\mathbf{x}_C} p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_B|\mathbf{x}_C)p(\mathbf{x}_C). \tag{3.2.11}$$

In general, this expression does not factorise into a product of the form $p(\mathbf{x}_A)p(\mathbf{x}_B)$ hence

$$\mathbf{x}_A \not\perp\!\!\!\perp \mathbf{x}_B \mid \emptyset \tag{3.2.12}$$

where $\not\perp\!\!\!\perp$ means that the conditional independence property does not hold.

Next we consider the case where we condition on the variable $\mathbf{x}_C$. The corresponding distribution is

$$\begin{aligned} p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) &= \frac{p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)}{p(\mathbf{x}_C)} \\ &= p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_B|\mathbf{x}_C) \end{aligned} \tag{3.2.13}$$

from which we get

$$\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C, \tag{3.2.14}$$

and hence $\mathbf{x}_A$ is conditionally independent of $\mathbf{x}_B$ given $\mathbf{x}_C$.

It is possible to give a graphical interpretation of these results by considering the path from $A$ to $B$ via $C$. The node $C$ is said to be *tail-to-tail* with respect to this path because the node is connected to the tails of the two connecting arrows. When we condition on $\mathbf{x}_C$ as in Figure 3.5 the node corresponding to the conditioned variable *blocks* the path from $A$ to $B$ causing $\mathbf{x}_A$ and $\mathbf{x}_B$ to become (conditionally) independent. When we do not condition on $\mathbf{x}_C$ the path is *unblocked* causing $\mathbf{x}_A$ and $\mathbf{x}_B$ to become dependent.

In the next example we consider the graph shown in Figure 3.6. The corresponding joint distribution is

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_A)p(\mathbf{x}_C|\mathbf{x}_A)p(\mathbf{x}_B|\mathbf{x}_C). \tag{3.2.15}$$

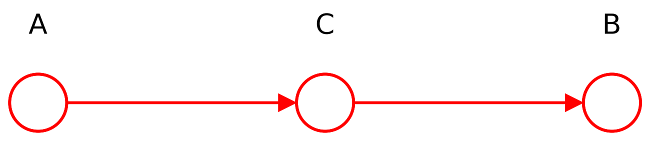

**Figure 3.6:** In this graph $\mathbf{x}_A$ depends on $\mathbf{x}_B$ since $\mathbf{x}_C$ is unknown.

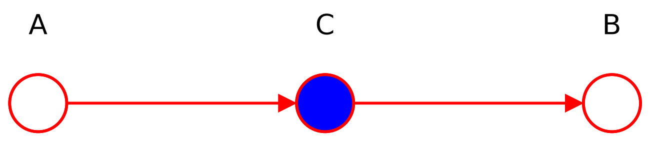

**Figure 3.7:** In this graph $\mathbf{x}_A$ is conditionally independent of $\mathbf{x}_B$ once $\mathbf{x}_C$ is observed.

Again, we start by considering the case where none of the variables are observed. Marginalising over $\mathbf{x}_C$ gives

$$p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_A) \sum_{\mathbf{x}_C} p(\mathbf{x}_C|\mathbf{x}_A) p(\mathbf{x}_B|\mathbf{x}_C) = p(\mathbf{x}_A) p(\mathbf{x}_B|\mathbf{x}_A) \tag{3.2.16}$$

which in general does not factorise into $p(\mathbf{x}_A)p(\mathbf{x}_B)$. Hence, again we have

$$\mathbf{x}_A \not\perp\!\!\!\perp \mathbf{x}_B \mid \emptyset \tag{3.2.17}$$

which we recognise as the same result we found in our previous example.

Next we condition on node $\mathbf{x}_C$ as shown in Figure 3.7. Using Bayes' theorem (A.1.3), together with (3.2.15), we have

$$\begin{aligned} p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) &= \frac{p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)}{p(\mathbf{x}_C)} \\ &= \frac{p(\mathbf{x}_A)p(\mathbf{x}_C|\mathbf{x}_A)p(\mathbf{x}_B|\mathbf{x}_C)}{p(\mathbf{x}_C)} \\ &= \frac{p(\mathbf{x}_C)p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_B|\mathbf{x}_C)}{p(\mathbf{x}_C)} \\ &= p(\mathbf{x}_A|\mathbf{x}_C)p(\mathbf{x}_B|\mathbf{x}_C). \end{aligned} \tag{3.2.18}$$

Hence, we have

$$\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C. \tag{3.2.19}$$

Again, we can interpret these results graphically. The node $C$ is said to be *head-to-tail* with respect to the path from $A$ to $B$. This path connects $A$ and $B$ and renders the nodes dependent. If we now observe $\mathbf{x}_C$, then this observation *blocks* the path from $A$ to $B$ and we obtain the conditional independence statement $\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C$.

The final example is shown in Figure 3.8. The joint distribution is given

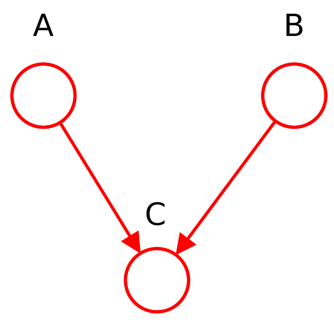

**Figure 3.8:** In this graph $\mathbf{x}_A$ is independent of $\mathbf{x}_B$ as long as $\mathbf{x}_C$ remain unobserved.

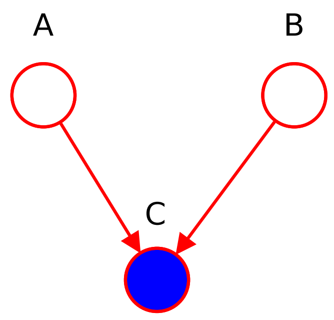

**Figure 3.9:** In this graph $\mathbf{x}_A$ depends on $\mathbf{x}_B$ through the observed variable $\mathbf{x}_C$.

by

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_A)p(\mathbf{x}_B)p(\mathbf{x}_C|\mathbf{x}_A, \mathbf{x}_B). \tag{3.2.20}$$

Consider first the case where none of the variables are observed. Marginalising both sides of (3.2.20) with respect to $\mathbf{x}_C$ we get

$$p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_A)p(\mathbf{x}_B) \tag{3.2.21}$$

and thus $\mathbf{x}_A$ and $\mathbf{x}_B$ are *independent* when no variables are observed.

$$\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \emptyset. \tag{3.2.22}$$

Note that this is the opposite behaviour of what we observed in our first two examples.

Now suppose we condition on $\mathbf{x}_C$ as shown in Figure 3.9. The conditional distribution of $\mathbf{x}_A$ and $\mathbf{x}_B$ is given by

$$\begin{aligned} p(\mathbf{x}_A, \mathbf{x}_B|\mathbf{x}_C) &= \frac{p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)}{p(\mathbf{x}_C)} \\ &= \frac{p(\mathbf{x}_A)p(\mathbf{x}_B)p(\mathbf{x}_C|\mathbf{x}_A, \mathbf{x}_C)}{p(\mathbf{x}_B)} \end{aligned} \tag{3.2.23}$$

which in general does not factorise into the product $p(\mathbf{x}_A)p(\mathbf{x}_B)$, and so

$$\mathbf{x}_A \not\perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C. \tag{3.2.24}$$

As we can see, the final example has the exact opposite behaviour of the first two. Graphically, we say that the node $C$ is *head-to-head* with respect to the path from $A$ to $B$ as it connect the heads of the two arrows. When node $c$ is unobserved, it *blocks* the path, and the variables $\mathbf{x}_A$ and $\mathbf{x}_B$ are independent. When conditioning on $\mathbf{x}_C$ the path is *unblocked* which results in $\mathbf{x}_A$ and $\mathbf{x}_B$ becoming dependent.

There is one more subtlety associated with the final example that needs to be addressed. First we introduce some more terminology. We say that node $N$ is a *descendant* of a node $M$ if there is a path from $N$ to $M$ in which each step of the path follows the direction of the arrows. It can be shown that a head-to-head path will become unblocked if either the node, or any of its descendants, is observed.

In summary, a tail-to-tail node or a head-to-tail node leaves a path unblocked unless it is observed in which case the path becomes blocked. A head-to-head node blocks a path if it is unobserved, but once the node, and/or at least one its descendants, is observed the path becomes unblocked.

We now summarise the concept of d-separation in more general terms. Consider a general DAG in which $A$, $B$ and $C$ are arbitrary non-intersecting subsets of $V$ such that $A \cup B \cup C \subseteq V$. We wish to ascertain whether a particular conditional independence statement $A \perp\!\!\!\perp B \mid C$ holds. We can achieve this by considering all paths from any node in $A$ to any node in $B$. Any such path is said to be *blocked* if it includes a node where

- the arrows on the path meet either head-to-tail or tail-to-tail at the node, and the node is in the set $C$, or
- the arrows meet head-to-head at the node and neither the node, nor any of its descendants, is in the set $C$.

If all paths from $A$ to $B$ are blocked, then $A$ is said to be d-separated from $B$ by $C$ and the joint distribution over all the variables in the graph will satisfy $A \perp\!\!\!\perp B \mid C$.

The concept of d-separation is illustrated further in Figure 3.10. In **a)** the path from $A$ to $B$ is not blocked by node $F$ because it is a tail-to-tail node for this path and is unobserved. It is also not blocked by node $E$ because, although the latter is a head-to-head node, it has a descendant $C$ that is in the conditioning set. As such, we have that $A$ is not d-separated from $B$ and the conditional independence statement $\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C$ does not follow from this graph. In **b)** the path from $A$ to $B$ is blocked by node $F$ because this is a tail-to-tail node that is observed, and so $A$ is d-separated from $B$ and the conditional independence property $\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B \mid \mathbf{x}_C$ is satisfied by this graph.

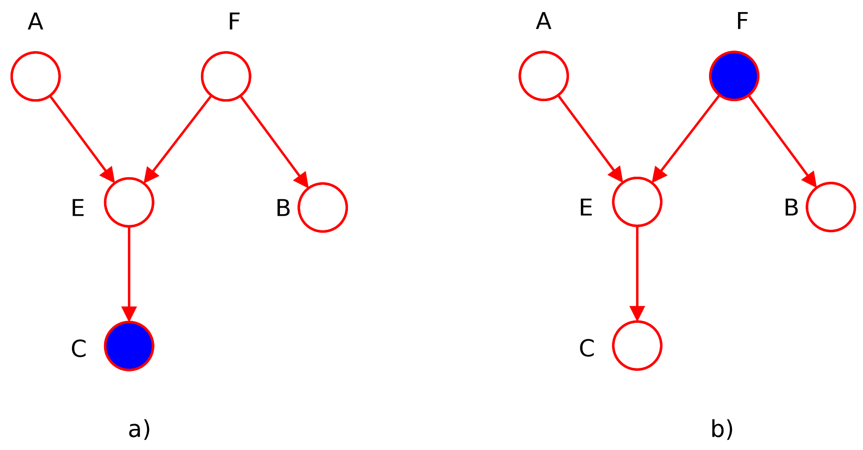

**Figure 3.10:** In graph a) $\mathbf{x}_A$ depends of $\mathbf{x}_B$ even though $\mathbf{x}_B$ is observed. In graph b) the observations of $\mathbf{x}_F$ causes A and B to be d-separated.

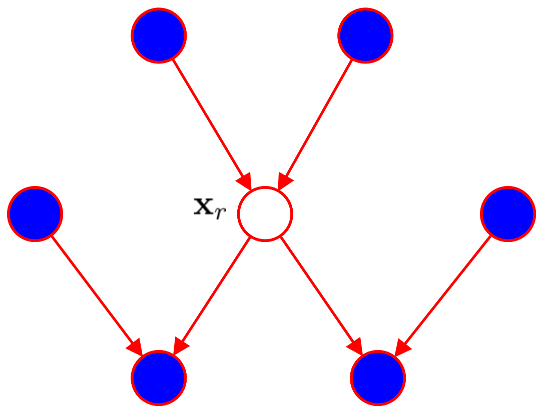

**Figure 3.11:** The Markov blanket of a node $\mathbf{x}_r$.

Note that this path is also blocked by node $E$ because $E$ is head-to-head and neither it nor any of its descendants is in the conditioning set.

The *Markov blanket* of a variable $\mathbf{x}_r$ is illustrated in Figure 3.11. The Markov blanket of a node $\mathbf{x}_v$ comprises the set of parents, children and co-parents (parents of children) of the node. It has the property that the conditional distribution of $\mathbf{x}_v$, conditioned on all the remaining variables in the graph, dependents only on the variables in the Markov blanket.

Consider a joint distribution $p(\mathbf{x}_V)$ represented by a Bayesian network. Now consider the conditional distribution of a particular node $\mathbf{x}_r$ conditioned on all of the remaining variables $\mathbf{x}_S$ where $S = V \backslash \{r\}$. Using (3.2.2) we get

$$\begin{aligned} p(\mathbf{x}_r|\mathbf{x}_S) &= \frac{p(\mathbf{x}_V)}{\int p(\mathbf{x}_V)\mathrm{d}\mathbf{x}_r} \\ &= \frac{\prod_v p(\mathbf{x}_v|\mathbf{x}_{\pi_v})}{\int \prod_v p(\mathbf{x}_v|\mathbf{x}_{\pi_v})\mathrm{d}\mathbf{x}_r} \end{aligned} \tag{3.2.25}$$

in which the integral is replaced by a summation where necessary if the variables are discrete. We observe that the CPD of any variable $\mathbf{x}_v$ that does not have a functional dependence on $\mathbf{x}_r$ can be taken outside the integral and as a result will cancel between the numerator and denominator. The only factors that remain are the conditional distribution $p(\mathbf{x}_r|\mathbf{x}_{\pi_r})$ for the node $\mathbf{x}_r$ itself and any conditional distributions for child nodes $s \in S$ with random variables $\mathbf{x}_s$ for which $\mathbf{x}_r$ is in the conditioning set of $p(\mathbf{x}_s|\mathbf{x}_{\pi_s})$, that is for which $r \in \pi_s$. The conditional distribution $p(\mathbf{x}_r|\mathbf{x}_{\pi_r})$ depends on the parents of node $\mathbf{x}_r$ whereas the conditionals $p(\mathbf{x}_s|\mathbf{x}_{\pi_s})$ depend on the children of $\mathbf{x}_r$ as well as the *co-parents* of $\mathbf{x}_r$. The co-parents of $\mathbf{x}_r$ are the parents of the nodes in $\mathbf{x}_S$ other than $\mathbf{x}_r$. The set of nodes comprising the parents, children, and co-parents is called the Markov blanket of node $\mathbf{x}_r$. From (3.2.25) we see that the Markov blanket d-separates $\mathbf{x}_r$ from the rest of the nodes in the network. Intuitively, the Markov blanket is the minimum set of nodes that "isolates" $\mathbf{x}_r$ from the rest of the network.

## 3.3 Maximum likelihood estimation

In maximum likelihood estimation (MLE) model parameters are estimated by maximising the likelihood function with respect to the parameters given a set of observations. Suppose we have a Bayesian network consisting of a set of nodes $V$ and a set of corresponding random variables $\mathbf{x}_V$. Now suppose we have $N$ independent and identically distributed (i.i.d.) observations of each variable denoted by $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$ where $\mathbf{X}_n$ is a single observation of all $\mathbf{x}_v$ for $v \in V$. Then the likelihood of the data as a function of model parameters is given by

$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{n=1}^{N} p(\mathbf{X}_n|\boldsymbol{\theta}) \tag{3.3.1}$$

where $\boldsymbol{\theta}$ is the set of model parameters. It is common to work with the log likelihood function instead of the likelihood function directly. Using the log likelihood function simplifies the analysis and during computations avoids issues with numerical underflow as a result of taking the product of small probabilities. The log likelihood function is defined as

$$\begin{aligned}\ln p(\mathbf{X}|\boldsymbol{\theta}) &= \ln \prod_{n=1}^{N} p(\mathbf{X}_n|\boldsymbol{\theta}) \\ &= \sum_{n=1}^{N} \ln p(\mathbf{X}_n|\boldsymbol{\theta}). \end{aligned} \tag{3.3.2}$$

Note carefully that the i.i.d. assumption is made for the observations, i.e. that $\mathbf{X}_n \perp\!\!\!\perp \mathbf{X}_m$ for $n \neq m$ where $\perp\!\!\!\perp$ denotes that the two variables are statistically independent. This does not mean that observations for separate nodes within the network are independent. Thus, in general we have that $\mathbf{x}_v \not\perp\!\!\!\perp \mathbf{x}_w$ for $v \neq w$.

Equation (3.3.2) is referred to as the *complete-data log likelihood* as the assumption is that we have a complete set of observations for all the nodes in the network. Letting $\mathbf{x}_v^n$ denote the $n$-th observation for the node $v$ and using (3.2.4) we get

$$\begin{aligned}\ln p(\mathbf{X}|\boldsymbol{\theta}) &= \sum_{n=1}^{N} \ln \prod_{v \in V} p(\mathbf{x}_v^n|\mathbf{x}_{\pi_v}^n, \boldsymbol{\theta}_v) \\ &= \sum_{n=1}^{N} \sum_{v \in V} \ln p(\mathbf{x}_v^n|\mathbf{x}_{\pi_v}^n, \boldsymbol{\theta}_v). \end{aligned} \tag{3.3.3}$$

We see that the likelihood decouples into local terms involving a node $v$ and its parents $\pi_v$ only, thereby significantly simplifying the MLE problem as we can optimise with respect to each $\boldsymbol{\theta}_v$ independently. In particular, for node $\mathbf{x}_v$ in the Bayesian network we can estimate the model parameters $\boldsymbol{\theta}_v$ by only considering observations of $\mathbf{x}_v^n$ for $n = 1, \dots N$ and the corresponding observations of the parent nodes $\mathbf{x}_{\pi_v}^n$.

The maximum likelihood estimation of the parameters is typically calculated by setting the derivative with respect to $\boldsymbol{\theta}_v$ equal to zero while taking parameter constraints into consideration for instance by using Lagrange multipliers (Bishop, 2007). The exact form of the optimal parameters depends on the nature of the individual CPDs. In the complete-data case we are in most instances able to find a closed-form solution to the MLE problem.

Until now we have assumed that we have observations for all variables in the Bayesian network. However, in many applications there are unobserved variables, for instance the nodes representing phonemes in speech recognition. We therefore separate the set of random variables $\mathbf{x}_V$ into hidden and observed variables. We denote hidden variables as $\mathbf{x}_H$ and observed variables as $\mathbf{x}_E$ such that $V = H \cup E$. We often use the terms *latent* and *evidence* for the hidden and observed variables, respectively.

The corresponding set of variables in the case of $N$ observations are $\mathbf{X}_H = \{\mathbf{Z}_1, \dots, \mathbf{Z}_N\}$ and $\mathbf{X}_E = \{\mathbf{X}_1, \dots, \mathbf{X}_N\}$. That is, for each observed set of variables $\mathbf{X}_n$ in the network there will be a corresponding set of latent variables $\mathbf{Z}_n$.

We now form the *incomplete-data likelihood*

$$p(\mathbf{X}_E|\boldsymbol{\theta}) = \prod_{n=1}^{N} \sum_{\mathbf{Z}_n} p(\mathbf{Z}_n, \mathbf{X}_n|\boldsymbol{\theta}) \tag{3.3.4}$$

where $\sum_{\mathbf{Z}_n}$ means that we marginalise over all variables in $\mathbf{Z}_n$. The motivation for this marginalisation shall become apparent in the next section when we discuss the EM algorithm. The corresponding incomplete-data log likelihood becomes

$$\ln p(\mathbf{X}_E|\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \sum_{\mathbf{Z}_n} p(\mathbf{Z}_n, \mathbf{X}_n|\boldsymbol{\theta}) \tag{3.3.5}$$

and using (3.2.4) we obtain

$$\ln p(\mathbf{X}_E|\boldsymbol{\theta}) = \sum_{n=1}^{N} \ln \sum_{\mathbf{Z}_n} \prod_{v \in V} p(\mathbf{x}_v^n|\mathbf{x}_{\pi_v}^n, \boldsymbol{\theta}_v)$$

where $\mathbf{x}_v^n, \mathbf{x}_{\pi_v}^n \in \mathbf{Z}_n \cup \mathbf{X}_n$. We see that, due to the presence of the marginalisation over $\mathbf{Z}_n$, this time the likelihood function does not decouple over the CPDs thereby substantially complicating the MLE of the model parameters. In general, we are not able to find a closed-form solution to the MLE problem in the case of an incomplete-data log likelihood. Instead, we turn to the EM algorithm.

## 3.3.1 The EM algorithm

The *expectation maximisation* (EM) algorithm is an iterative method for finding maximum likelihood estimates of model parameters from observed data in the case of incomplete-data. We start by introducing a probability distribution $q(\mathbf{X}_H)$ over the latent variables $\mathbf{X}_H$. We can then show that the following decomposition holds

$$\ln p(\mathbf{X}_E|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q||p) \tag{3.3.6}$$

where

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{X}_H} q(\mathbf{X}_H) \ln \left\{ \frac{p(\mathbf{X}_E, \mathbf{X}_H|\boldsymbol{\theta})}{q(\mathbf{X}_H)} \right\} &(3.3.7) \\ \mathrm{KL}(q||p) &= -\sum_{\mathbf{X}_H} q(\mathbf{X}_H) \ln \left\{ \frac{p(\mathbf{X}_H|\mathbf{X}_E, \boldsymbol{\theta})}{q(\mathbf{X}_H)} \right\}. &(3.3.8) \end{aligned}$$

We can verify (3.3.6) using the product rule of probability which gives

$$\ln p(\mathbf{X}_E, \mathbf{X}_H|\boldsymbol{\theta}) = \ln p(\mathbf{X}_H|\mathbf{X}_E, \boldsymbol{\theta}) + \ln p(\mathbf{X}_E|\boldsymbol{\theta}) \tag{3.3.9}$$

which we substitute into $\mathcal{L}(q, \boldsymbol{\theta})$. This results in two terms, one of which cancels $\mathrm{KL}(q||p)$ and another that gives the required log likelihood $\ln p(\mathbf{X}_E|\boldsymbol{\theta})$ since $q(\mathbf{X}_H)$ is a normalised distribution that sums to 1.

Equation (3.3.8) is the Kullbach-Leibler (KL) divergence (Bishop, 2007, page 55) between $q(\mathbf{X}_H)$ and the posterior distribution $p(\mathbf{X}_H|\mathbf{X}_E, \boldsymbol{\theta})$. The KL divergence satisfies

$$\mathrm{KL}(q||p) \geq 0, \tag{3.3.10}$$

with equality if, and only if,

$$q(\mathbf{X}_H) = p(\mathbf{X}_H|\mathbf{X}_E, \boldsymbol{\theta}). \tag{3.3.11}$$

It therefore follows from (3.3.6) that

$$\mathcal{L}(q, \boldsymbol{\theta}) \leq \ln p(\mathbf{X}_E|\boldsymbol{\theta}). \tag{3.3.12}$$

That is, $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound of $\ln p(\mathbf{X}_E|\boldsymbol{\theta})$. Note that $\mathcal{L}(q, \theta)$ is a *functional* of the distribution $q(\mathbf{X}_H)$ and a function of the parameters $\boldsymbol{\theta}$ (Bishop, 2007, Chapter 9).

The EM algorithm proceeds as follows. Suppose that the current value of the parameter vector is $\boldsymbol{\theta}$. In the E step, the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximised with respect to $q(\mathbf{X}_H)$ while holding $\boldsymbol{\theta}$ fixed to give

$$q^*(\mathbf{X}_H) = \underset{q(\mathbf{X}_H)}{\mathrm{argmax}}\, \mathcal{L}(q, \boldsymbol{\theta}). \tag{3.3.13}$$

Equation (3.3.13) is solved by noting that the value of $\ln p(\mathbf{X}_E|\boldsymbol{\theta})$ does not depend on $q(\mathbf{X}_H)$ and hence from (3.3.6) and (3.3.12) we get that the largest value of $\mathcal{L}(q, \boldsymbol{\theta})$ will occur when the KL divergence vanishes. This happens when $q(\mathbf{X}_H)$ is equal to the posterior distribution $p(\mathbf{X}_H|\mathbf{X}_E, \boldsymbol{\theta})$. In this case, the KL divergence vanishes and the lower bound equals the log likelihood.

In the subsequent M step, the distribution $q(\mathbf{X}_H)$ is held fixed and the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximised with respect to $\boldsymbol{\theta}$ to give some new value

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\mathrm{argmax}}\, \mathcal{L}(q, \boldsymbol{\theta}). \tag{3.3.14}$$

From (3.3.7) we see that this step will cause the lower bound to increase in the next E-step, except for the case where it is already at its maximum, which will necessarily cause the corresponding log likelihood function to increase since with $q(\mathbf{X}_H) = p(\mathbf{X}_H|\mathbf{X}_E)$ the lower bound is equal to the likelihood

Note that the distribution $q(\mathbf{X}_H)$ is determined using the old parameter values rather than the new ones and is held fixed during the M step. As such, $q(\mathbf{X}_H)$ will not equal the new posterior distribution $p(\mathbf{X}_H|\mathbf{X}_E, \boldsymbol{\theta}^*)$ and hence there will be a non-zero KL divergence. Thus, again the log likelihood can be increased by fixing the new set of parameters (which now become the old parameters), and maximising with respect to $q(\mathbf{X}_H)$ by setting $q(\mathbf{X}_H) =$

$p(\mathbf{X}_H|\mathbf{X}_E,\boldsymbol{\theta})$. The EM algorithm continues iterating between the E step and the M step until convergence.

In the E step we calculate $q(\mathbf{X}_H) = p(\mathbf{X}_H|\mathbf{X}_E,\boldsymbol{\theta})$. Substituting into (3.3.7) we get

$$\begin{aligned}\mathcal{L}(q,\boldsymbol{\theta}) &= \sum_{\mathbf{X}_H} p(\mathbf{X}_H|\mathbf{X}_E,\boldsymbol{\theta}) \ln p(\mathbf{X}_E,\mathbf{X}_H|\boldsymbol{\theta}^*) - \sum_{\mathbf{X}_H} p(\mathbf{X}_H,\mathbf{X}_E,\boldsymbol{\theta}) \\ &= Q(\boldsymbol{\theta}^*,\boldsymbol{\theta}) + \text{const.} \end{aligned} \tag{3.3.15}$$

We see that the second term on the right-hand side is independent of $\boldsymbol{\theta}^*$ and thus a constant when maximising the lower bound in the M step.

Note that the quantity that is being maximised is actually the expectation of $\ln p(\mathbf{X}_E,\mathbf{X}_H|\boldsymbol{\theta}^*)$ with respect to the posterior distribution over the latent variables given the previous parameters, $q(\mathbf{X}_E) = p(\mathbf{X}_H|\mathbf{X}_E,\boldsymbol{\theta})$. That is, we have

$$Q(\boldsymbol{\theta}^*,\boldsymbol{\theta}) = \mathbb{E}[\ln p(\mathbf{X}_E,\mathbf{X}_H|\boldsymbol{\theta}^*)]. \tag{3.3.16}$$

where the expectation is taken with respect to the posterior $p(\mathbf{X}_H|\mathbf{X}_E,\boldsymbol{\theta})$.

The intuition behind the EM algorithm is thus as follows. If we had the complete data we could estimate $\boldsymbol{\theta}$ directly by maximising the complete-data log likelihood in (3.3.2) using (3.3.3). However, since we only have incomplete data we instead maximise the *expectation* of the complete-data log likelihood given the observed data and the current estimate of $\boldsymbol{\theta}$. Consequently, the quantities required for the M step are the expected values of the same quantities required to estimate the parameters the complete-data case. In the complete-data case these quantities are called *sufficient statistics* whose precise form depends on the type of CPDs used. Thus, in the case of incomplete-data these quantities are referred to as the *expected sufficient statistics.*

In this research we only consider random variables whose CPDs are either multinomial (A.2.6) or Gaussian (A.2.1). We further assume that all multinomial variables are latent, all Gaussian variables are observed, and that all parent nodes of any variable are latent-multinomial. We shall denote a variable $\mathbf{x}_v^n$ as $\mathbf{z}_n$ if the variable is latent-multinomial and simply $\mathbf{x}_n$ if the variable is observed-Gaussian. For the multinomial variables we use the $1-$of$-K$ representation (Bishop, 2007) in which a particular element $z_k$ is equal to 1 and all other elements equal 0. The values of $z_k$ satisfy $z_k \in \{0,1\}$ and $\sum_k z_k = 1$. Thus, there are $K$ possible states the vector $\mathbf{z}$ can assume according to which element is non-zero. Note that we omit the dependencies of the variables on the node $v$ in order to keep the notation uncluttered. The CPD parameters $\boldsymbol{\theta}_v$ are $\boldsymbol{\rho}$ if the variable is multinomial and the pair $(\boldsymbol{\mu},\boldsymbol{\Sigma})$ if the variable is Gaussian. Each node has a separate set of parameters for each possible configuration of parent nodes.

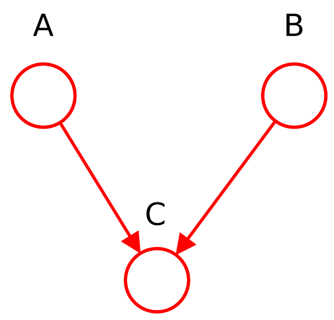

**Figure 3.12:** In the case that all parent nodes are multinomial we may treat the parents as a single multinomial variable with entries for all possible configurations of the parent nodes.

In the case that we have $N$ observations $\mathbf{X}_E = (\mathbf{X}_1, \ldots, \mathbf{X}_N)$ and corresponding latent variables $\mathbf{X}_H = (\mathbf{Z}_1, \ldots, \mathbf{Z}_N)$ we get that $Q(\boldsymbol{\theta}^*, \boldsymbol{\theta})$ becomes

$$\begin{aligned} Q(\boldsymbol{\theta}^*, \boldsymbol{\theta}) &= \sum_{n=1}^{N} \sum_{\mathbf{Z}_n} q(\mathbf{Z}_n) \ln \prod_{v \in V} \ln p(\mathbf{x}_v^n | \mathbf{x}_{\pi_v}^n, \boldsymbol{\theta}_v) \\ &= \sum_{n=1}^{N} \sum_{\mathbf{Z}_n} \sum_{v \in V} q(\mathbf{Z}_n) \ln p(\mathbf{x}_v^n | \mathbf{x}_{\pi_v}^n, \boldsymbol{\theta}_v), \end{aligned} \tag{3.3.17}$$

where again $\mathbf{x}_v^n, \mathbf{x}_{\pi_v}^n \in \mathbf{Z}_n \cup \mathbf{X}_N$. Recalling that $q(\mathbf{Z}_n) = p(\mathbf{Z}_n | \mathbf{X}_n)$ from the E step we wish to optimise $Q(\boldsymbol{\theta}^*, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}_v \in \boldsymbol{\theta}^*$.

We note that the sum over the indices $v$ in Equation (3.3.17) decouples over the parameters $\boldsymbol{\theta}_v$. Thus, taking the derivative with respect to $\boldsymbol{\theta}_v$ causes all other terms not dependent on $v$ to disappear and as such we can optimise with respect to each $\boldsymbol{\theta}_v$ individually.

Since the set of parent variables are discrete it is possible to treat these variable as a single parent $\mathbf{z}_\pi$ with elements $z_{\pi_k}$ corresponding to the different configurations of the parent variables. For instance, consider the graph in Figure 3.8. Suppose $\mathbf{z}_A$ is a multinomial variable that can assume one of two possible values $z_{A1}$ and $z_{A2}$ and similarly $\mathbf{z}_B$ can assume one of two possible values $z_{B1}$ and $z_{B2}$. Then jointly the variables $\mathbf{z}_A$ and $\mathbf{z}_B$ can assume one of four possible combinations of values $(z_{A1}, z_{B1})$, $(z_{A1}, z_{B2})$, $(z_{A2}, z_{B1})$,and $(z_{A2}, z_{B2})$. From the four configurations we can create a single multinomial parent "mega-variable" $\mathbf{z}_\pi$ that can assume one of four possible states. This idea generalises to any number of parents of potentially different dimensionality. Thus, in practice we shall treat the parent nodes $\mathbf{x}_{\pi_v}$ of a variable $\mathbf{x}_v$ as a single parent variable.

We now consider how to perform parameter updates in the case $\mathbf{x}_v$ is a multinomial variable and subsequently in the case that $\mathbf{x}_v$ is a Gaussian variable.

**Multinomial.** In the case the variable is a multinomial variable $\mathbf{z}_n$ with elements $z_{nl} \in \{0,1\}$ such that $\sum_{l=1}^{L} z_{nl} = 1$ the CPD is given by

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{z}_{\pi_n}, \boldsymbol{\rho}) &= \prod_{k=1}^{K} \left[ \prod_{l=1}^{L} \rho_{kl}^{z_{nl}} \right]^{z_{\pi_{nk}}} \\ &= \prod_{k=1}^{K} \prod_{l=1}^{L} \rho_{kl}^{z_{nl} z_{\pi_{nk}}} \end{aligned} \tag{3.3.18}$$

where $\boldsymbol{\rho}$ are the model parameters with $0 \leq \rho_{kl} \leq 1$, $\sum_{l=1}^{L} \rho_{kl} = 1$ and $\rho_{kl} = p(z_l = 1|z_{\pi_k} = 1)$.

Taking the derivative of $Q(\boldsymbol{\theta}^*, \boldsymbol{\theta})$ with respect to $\boldsymbol{\rho}$ we get

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\rho}} Q(\boldsymbol{\theta}^*, \boldsymbol{\theta}) &= \sum_{i=1}^{N} \sum_{\mathbf{Z}_n} q(\mathbf{Z}_n) \frac{\partial}{\partial \boldsymbol{\rho}} \ln \left\{ \prod_{k=1}^{K} \prod_{l=1}^{L} \rho_{kl}^{z_{nl} z_{\pi_{nk}}} \right\} \\ &= \sum_{i=1}^{N} \sum_{\mathbf{Z}_n} q(\mathbf{Z}_n) \frac{\partial}{\partial \boldsymbol{\rho}} \sum_{k=1}^{K} \sum_{l=1}^{L} z_{nl} z_{\pi_{nk}} \ln \rho_{kl}. \end{aligned} \tag{3.3.19}$$

We recall that from the E step $q(\mathbf{Z}_n) = p(\mathbf{Z}_n|\mathbf{X}_n, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the old parameter set. Thus, the effect of taking the sum over $\mathbf{Z}_n$ is to marginalise out all variables from $p(\mathbf{Z}_n|\mathbf{X}_n, \boldsymbol{\theta})$ except $\mathbf{z}_n$ and $\mathbf{z}_{\pi_n}$ that also appear outside the posterior on the right-hand side of (3.3.19). Thus, (3.3.19) can be rewritten as

$$\frac{\partial}{\partial \boldsymbol{\rho}} Q(\boldsymbol{\theta}^*, \boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{\mathbf{z}_n, \mathbf{z}_{\pi_n}} p(\mathbf{z}_n, \mathbf{z}_{\pi_n}|\mathbf{X}_n, \boldsymbol{\theta}) \frac{\partial}{\partial \boldsymbol{\rho}} \sum_{k=1}^{K} \sum_{l=1}^{L} z_{nl} z_{\pi_{nk}} \ln \rho_{kl}. \tag{3.3.20}$$

where we see that we only need to calculate the local posterior $p(\mathbf{z}_n, \mathbf{z}_{\pi_n}|\mathbf{X}_n, \boldsymbol{\theta})$. We shall discuss in detail how to efficiently calculate this posterior in the next section when we discuss the junction tree algorithm.

We note from (3.3.20) that the sums on the right-hand side decouple over the parameters $\rho_{kl}$. Thus, we can take partial derivatives with respect to $\rho_{kl}$ giving

$$\begin{aligned} \frac{\partial}{\partial \rho_{kl}} Q(\boldsymbol{\theta}^*, \boldsymbol{\theta}) &= \sum_{n=1}^{N} \sum_{\mathbf{z}_n, \mathbf{z}_{\pi_n}} p(\mathbf{z}_n, \mathbf{z}_{\pi_n}|\mathbf{X}_n, \boldsymbol{\theta}) z_{nl} z_{\pi_{nk}} \frac{\partial}{\partial \rho_{kl}} \ln \rho_k \\ &= \sum_{n=1}^{N} r_{nkl} \left( \frac{1}{\rho_{kl}} \right) \end{aligned} \tag{3.3.21}$$

where we have defined the quantity

$$\begin{aligned} r_{nkl} &= \sum_{\mathbf{z}_n, \mathbf{z}_{\pi_n}} p(\mathbf{z}_n, \mathbf{z}_{\pi_n}|\mathbf{X}_n, \boldsymbol{\theta}) z_{nl} z_{\pi_{nk}} \\ &= \mathbb{E}[z_{nl} z_{\pi_{nk}}] \end{aligned} \tag{3.3.22}$$

where the expectation is taken with respect to the distribution $p(\mathbf{z}_n, \mathbf{z}_{\pi_n}|\mathbf{X}_n, \boldsymbol{\theta})$. The quantity $r_{nkl}$ represents the *responsibility* that the parent configuration $z_{\pi_{nk}}$ takes for the explaining the value of $z_{nl}$ given the parameters $\boldsymbol{\theta}$.

Defining $N_{kl}$ as

$$N_{kl} = \sum_{n=1}^{N} r_{nkl} \tag{3.3.23}$$

and setting the derivative equal to zero we get

$$\frac{N_{kl}}{\rho_{kl}} + \lambda = 0 \tag{3.3.24}$$

where $\lambda$ is a Lagrange multiplier whose role is to ensure he constraint $\sum_{l=1}^{L} \rho_{kl} = 1$. The quantity $N_{kl}$ is the expected sufficient statistics for the multinomial variable $\mathbf{z}$ and represents the *expected* number of times $\mathbf{z}$ is in state $k$ while its parents are in state $l$ where the expectation is taken with respect to the variational distribution $q(\mathbf{z}_n)$ over the latent variables.

Upon rearranging equation (3.3.24)

$$\rho_{kl}\lambda_k + N_{kl} = 0. \tag{3.3.25}$$

and using the constraint $\sum_{l=1}^{L} \rho_{kl} = 1$ we get

$$\lambda_k = -\sum_{l=1}^{L} N_{kl}. \tag{3.3.26}$$

Finally, this gives us

$$\rho_{kl} = \frac{N_{kl}}{\sum_{l=1}^{L} N_{kl}}, \tag{3.3.27}$$

which is the maximum likelihood estimate of $\rho_{kl}$ for a latent multinomial variable with latent multinomial parents in a Bayesian network. In the case that $\mathbf{z}_n$ or any of the parents $\mathbf{z}_{\pi_n}$ are observed the analysis carries through unchanged with the observed values replacing expected values as necessary.

**Gaussian.** In the case that $\mathbf{x}_n$ is a Gaussian variable with latent multinomial parents $\mathbf{z}_{\pi_n}$ the CPD is given by

$$p(\mathbf{x}_n|\mathbf{z}_{\pi_n}) = \prod_{k=1}^{K} \left[\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]^{z_{\pi_{nk}}}. \tag{3.3.28}$$

We maximise the lower bound with respect to the parameters by setting the partial derivative of the log likelihood function with respect to the parameters

$(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ equal to zero. Taking partial derivatives and following a similar line of reasoning as for the multinomial we get

$$\begin{aligned}\frac{\partial}{\partial(\boldsymbol{\mu}, \boldsymbol{\Sigma})} Q(\boldsymbol{\theta}^*, \boldsymbol{\theta}) &= \frac{\partial}{\partial(\boldsymbol{\mu}, \boldsymbol{\Sigma})} \sum_{n=1}^{N} \sum_{\mathbf{Z}_n} q(\mathbf{Z}_n) \ln \left\{ \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{\pi_{nk}}} \right\} \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} \sum_{\mathbf{z}_{\pi_n}} p(\mathbf{z}_{\pi_n} | \mathbf{X}_n, \boldsymbol{\theta}) z_{\pi_{nk}} \frac{\partial}{\partial(\boldsymbol{\mu}, \boldsymbol{\Sigma})} \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \frac{\partial}{\partial(\boldsymbol{\mu}, \boldsymbol{\Sigma})} \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad (3.3.29)\end{aligned}$$

where we have used the fact that $\mathbf{x}_n$ is observed. In (3.3.29) $r_{nk}$ is again the responsibility that parent configuration $k$ takes for the observations $\mathbf{x}_n$

$$\begin{aligned} r_{nk} &= \sum_{\mathbf{z}_{\pi_n}} p(\mathbf{z}_{\pi_n} | \mathbf{X}_n, \boldsymbol{\theta}) z_{\pi_{nk}} \\ &= \mathbb{E}[z_{\pi_{nk}}] \qquad (3.3.30)\end{aligned}$$

where the expectation is taken with respect to $p(\mathbf{z}_{\pi_n} | \mathbf{X}_n, \boldsymbol{\theta})$.

Note that the derivative with respect to the parameters decomposes with respect to $k$. Thus, we can take the derivative with respect to the parameter pair $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$. Setting the derivative with respect to $\boldsymbol{\mu}_k$ equal to zero we obtain

$$\sum_{n=1}^{N} r_{nk} \frac{\partial}{\partial \boldsymbol{\mu}_k} \left( -\frac{1}{2} \ln |\boldsymbol{\Sigma}_k| - \frac{1}{2} (\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_k) \right) = 0. \qquad (3.3.31)$$

Using standard vector calculus we get

$$-\sum_{n=1}^{N} r_{nk} \boldsymbol{\Sigma}_k (\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \qquad (3.3.32)$$

which after multiplying by $\boldsymbol{\Sigma}_k^{-1}$ and rearranging gives

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \qquad (3.3.33)$$

where

$$N_k = \sum_{n=1}^{N} r_{nk}. \qquad (3.3.34)$$

Setting the derivative of $Q(\boldsymbol{\theta}^*, \boldsymbol{\theta})$ with respect to $\boldsymbol{\Sigma}_k$ equal to zero and following a similar line of reasoning we get

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(\pi_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}. \qquad (3.3.35)$$

In speech recognition applications it is common to use *diagonal* covariance matrices. That is, off-diagonal elements of $\mathbf{\Sigma}_k$ are set to zero. Diagonal covariance matrices reduces computational cost when evaluating the Gaussian probability density function and reduces the number of parameters in the model. In the case of diagonal covariance matrices (3.3.35) remains the same but with any off-diagonal elements set to zero and hence these elements do not need to be calculated.

## 3.4 Inference in Bayesian networks

This section follows Jordan (2003, Chapter 17).

### 3.4.1 The junction tree algorithm

In order to find the expected sufficient statistics needed to update the parameters $\boldsymbol{\theta}_v$ of the CPD $p(\mathbf{x}_v|\mathbf{x}_{\pi_v}, \boldsymbol{\theta}_v)$ in the EM algorithm, we saw in the previous section that we need to find the joint posterior probability of $p(\mathbf{x}_v, \mathbf{x}_{\pi_v}|\mathbf{x}_E, \boldsymbol{\theta}_v)$ in the E step in the E step. That is the joint posterior over a node and its parents given the evidence for all the nodes in the network. Calculating this joint posterior is an example of an *inference problem.*

We shall give inference in Bayesian networks a slightly more general treatment and at the same time develop the machinery necessary to solve the classification problem in AVASR. As the set of model parameters are held fixed during the E step, we omit the explicit dependence on $\boldsymbol{\theta}$ in our notation. For example, we shall write $p(\mathbf{x}_H|\mathbf{x}_E)$ instead of $p(\mathbf{x}_H|\mathbf{x}_E)$.

Formally, the inference problem is defined as follows:

**Definition 1.** *Let $G(V, E)$ be a Bayesian network with hidden nodes $H$ and observed nodes $E$ such that $V = H \cup E$. Let $F \subseteq H$ be an arbitrary subset of the hidden nodes with corresponding random variables $\mathbf{x}_F$. We then wish to evaluate $p(\mathbf{x}_F|\mathbf{x}_E)$ for arbitrary $F$.*

This problem can in principle be solved by evaluating the joint posterior $p(\mathbf{x}_H|\mathbf{x}_E, \boldsymbol{\theta})$ explicitly and obtain $p(\mathbf{x}_F|\mathbf{x}_E)$ by marginalisation as

$$p(\mathbf{x}_F|\mathbf{x}_E) = \sum_{\mathbf{x}_{H\setminus F}} p(\mathbf{x}_H|\mathbf{x}_E). \tag{3.4.1}$$

Unfortunately this approach is intractable in most applications as the computational cost of taking the sum grows exponentially with the number of nodes in the network (Jordan, 2003). However, by exploiting sparse structures in Bayesian networks it is possible to derive an efficient inference algorithm that calculates the posterior distribution over smaller *cliques* of nodes while maintaining the joint distribution as a product of clique *potentials.* This algorithm is called the *junction tree algorithm.* The junction tree algorithm exploits conditional independence properties in Bayesian networks to perform computationally efficient inference.

### 3.4.2 Potentials

A clique $C \in V$ is defined as a completely connected subset of $V$. A completely connected subset is a subset of nodes where every two nodes are connected by an edge. The set of all possible cliques of $V$ is denoted by $\mathcal{C}$. Corresponding to each clique $C \in \mathcal{C}$ we have a set of random variables $\mathbf{x}_C$. For each clique $C$ we define a *potential* $\phi_C(\mathbf{x}_C)$ over the clique. A potential is a non-negative function on the realisations of $\mathbf{x}_C$. Note that we do allow the clique sets $C \in \mathcal{C}$ to overlap.

We now define the joint probability distribution over $\mathbf{x}_V$ as the normalised product of potential functions

$$p(\mathbf{x}_V) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_C). \tag{3.4.2}$$

The clique data structure is called a *hypergraph*; a set of subsets of the underlying graph. Equation (3.4.2) defines the joint probability distribution associated with the hypergraph.

We wish to keep the probability distribution over the hypergraph consistent with the joint probability (3.2.2) of the underlying graph. We can achieve this by initialising the potential functions from the underlying CPDs. One possibility is to initialise each clique potential from the CPDs of the nodes included in the clique. However, the local conditional probabilities need not necessarily be defined on cliques. For example, if the parents of node $\mathbf{x}_v$ are not connected then $p(\mathbf{x}_v|\mathbf{x}_{\pi_v})$ is not a function on a clique. The solution to this problem lies in *moralising* the underlying graph.

A moral graph $G^m$ corresponding to a directed graph $G$ is obtained by "marrying" (connecting) the parents of each node in the graph and dropping the direction of all edges. Figure 3.13 illustrates the process of moralisation in the Bayesian network shown in **a)** where **b)** shows the resulting moral graph. Note that we have "married" the parents of node 6. We note that in the moral graph the local conditional probabilities are indeed potential functions on cliques.

We associate each CPD in the underlying graph with one, and only one, potential function in the hypergraph. That is, a CPD can not be assigned to multiple potential functions. In the case that a node is an element of multiple cliques we choose one potential for which the CPD becomes a factor. We then have the desired result

$$\frac{1}{Z} \prod_{C \in \mathcal{C}} \phi_C(\mathbf{x}_C) = \prod_{v} p(\mathbf{x}_v|\mathbf{x}_{\pi_v}). \tag{3.4.3}$$

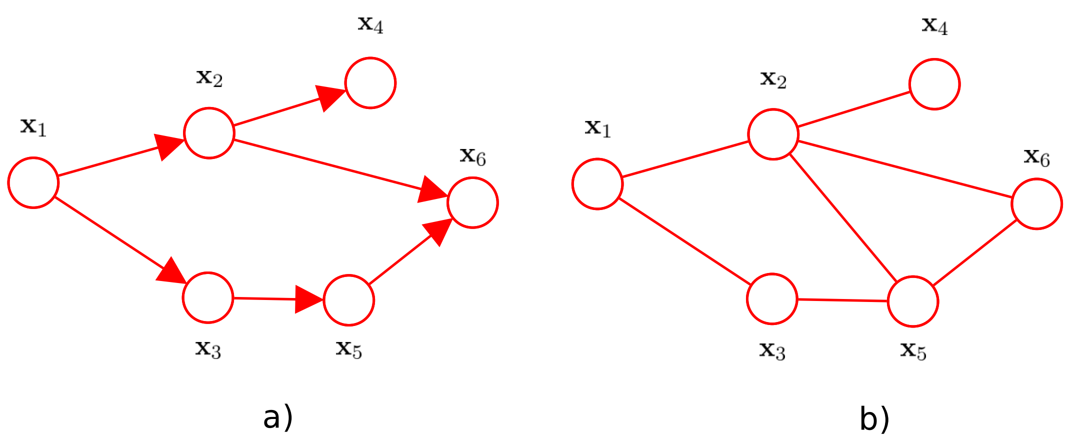

**Figure 3.13:** An example Bayesian network and its corresponding moral graph b). Note that the CPD $p(\mathbf{x}_6|\mathbf{x}_2, \mathbf{x}_5)$ has as arguments a subset of nodes that are not contained in any clique in the graph. By connecting $\mathbf{x}_2$ and $\mathbf{x}_5$ the arguments in the potential $p(\mathbf{x}_6|\mathbf{x}_2, \mathbf{x}_5)$ are contained in the clique $\{\mathbf{x}_2, \mathbf{x}_5, \mathbf{x}_6\}$.

That is, the product of potentials on the hypergraph is equivalent to the joint probability distribution of the underlying directed graph. Note that, if the potentials are initialised as CPDs, they are already normalised in which case the normalisation factor $Z$ is implicitly one.

### 3.4.3 Introducing evidence

We now return to the problem where some of the nodes in the network are observed and we wish to find the posterior distribution over subsets of latent variables (Definition 1).

Whenever some of the variables are observed, these variables are fixed at their observed values. We refer to this as *introducing evidence*. Introducing evidence will in general change the local distribution of all latent variables throughout the network.

For each clique $C \in \mathcal{C}$ we consider the intersections $C \cap E$ and $C \cap H$ where $C = (C \cap H) \cup (C \cap E)$ by the assumption that $H$ and $E$ partition $V$. The evidence nodes $C \cap E$ have been fixed to specific values and as a result the potential over $C$ now only ranges over the realisations of $C \cap H$. Thus, for a particular observed configuration $\bar{\mathbf{x}}_E$ of $\mathbf{x}_E$ we have

$$p(\mathbf{x}_H, \bar{\mathbf{x}}_E) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\mathbf{x}_{C \cap H}, \bar{\mathbf{x}}_{C \cap E}). \tag{3.4.4}$$

In the case of discrete CPDs, probabilities and potentials are represented as multidimensional tables, and thus conditionals are obtained by taking *slices* at the observed values of the potentials defining the joint probability distribution. Equation (3.4.4) then becomes a product of slices of potential functions.

A slice of a potential function is itself a potential function. Thus, we can also view (3.4.4) as a product of potential functions on subsets $\mathbf{x}_{C\cap H}$. We shall write

$$\widetilde{\psi}_{C\cap H}(\mathbf{x}_{C\cap H}) = \psi_C(\mathbf{x}_{C\cap H}, \bar{\mathbf{x}}_{C\cap E}) \tag{3.4.5}$$

to express the explicit reference to the fixed configuration $\bar{\mathbf{x}}_E$. Thus, we have

$$p(\mathbf{x}_H, \widetilde{\mathbf{x}}_E) = \frac{1}{Z}\prod_{C\in\mathcal{C}} \widetilde{\psi}_{C\cap H}(\mathbf{x}_{C\cap H}). \tag{3.4.6}$$

However, note that the normalisation factor $Z$ in (3.4.6) is obtained by summing over both $\mathbf{x}_H$ and $\mathbf{x}_E$, whereas the product is defined only over $\mathbf{x}_H$. In fact, $Z$ is not the normalisation factor for the product of potentials $\widetilde{\psi}_{C\cap H}$; indeed this product is not in general normalised due to the introduction of evidence. The normalisation constant can be obtained by summing $Z$ over $\mathbf{x}_H$

$$\begin{aligned}\widetilde{Z} &= \sum_{\mathbf{x}_H} p(\mathbf{x}_H, \bar{\mathbf{x}}_E) \\ &= \sum_{\mathbf{x}_H} \frac{1}{Z}\prod_{C\in\mathcal{C}} \widetilde{\psi}_{C\cap H}(\mathbf{x}_{C\cap H}). \end{aligned} \tag{3.4.7}$$

We know that $\sum_{\mathbf{x}_H} p(\mathbf{x}_H, \bar{\mathbf{x}}_E) = p(\bar{\mathbf{x}}_E)$ by definition, thus we get

$$\frac{p(\mathbf{x}_H, \bar{\mathbf{x}}_E)}{p(\bar{\mathbf{x}}_E)} = \frac{\prod_{C\in\mathcal{C}} \widetilde{\psi}_{C\cup H}(\mathbf{x}_{C\cup H})}{\sum_{\mathbf{x}_H}\prod_{C\in\mathcal{C}} \widetilde{\psi}_{C\cap H}(\mathbf{x}_{C\cap H})}. \tag{3.4.8}$$

From the above equation we see that the sliced potentials $\widetilde{\psi}_{C\cap H}$ provide a representation of the conditional probability $p(\mathbf{x}_H|\bar{\mathbf{x}}_E)$ in terms of the potential functions. The normalisation factor for this representation is the marginal probability $\widetilde{Z} = p(\bar{\mathbf{x}}_E)$.

Note that the original normalisation constant $Z$ cancels on the right-hand-side of (3.4.8). Thus, when calculating normalisation constants we do not need the normalisation constant associated with the original set of potentials. It suffices to compute the normalisation constant of the sliced potentials.

In order to illustrate the concept of slicing potentials we consider the example graph shown in Figure 3.14. Suppose the nodes represent two-dimensional binary random variables $\mathbf{x}_A$ and $\mathbf{x}_B$ with probabilities $p(\mathbf{x}_A = 1) = 0.8$, $p(\mathbf{x}_B = 1|\mathbf{x}_A = 1) = 0.6$ and $p(\mathbf{x}_B = 1|\mathbf{x}_A = 0) = 0.4$. These probabilities are sufficient for constructing the joint probability distribution $p(\mathbf{x}_A, \mathbf{x}_B)$.

Constructing the hypergraph by moralising we obtain a single clique $C = \{A, B\}$ with clique potential initialised as the product

$$\psi_{A,B} = p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_A)p(\mathbf{x}_B|\mathbf{x}_A) \tag{3.4.9}$$

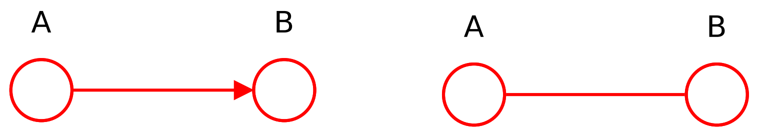

**Figure 3.14:** Example of moralising a simple two-node Bayesian network.

where we have used the short-hand notation $\psi_{A,B} = \psi_{A,B}(\mathbf{x}_A, \mathbf{x}_B)$. Substituting numerical values we get

$$\psi_{A,B} = \begin{bmatrix} 0.12 & 0.32 \\ 0.08 & 0.48 \end{bmatrix} \tag{3.4.10}$$

where $\psi_{A,B}(j,i) = p(\mathbf{x}_A = i, \mathbf{x}_B = j)$.

Note that the clique potential is normalised as a consequence of being initialised from the CPDs of the underlying graph. Suppose we now have observed evidence $\bar{\mathbf{x}}_B = 1$. We thus fix $\mathbf{x}_B$ at this value and obtain the slice

$$\psi_A = \begin{bmatrix} 0.08 \\ 0.48 \end{bmatrix}. \tag{3.4.11}$$

As expected, the new potential is not normalised. Normalising yields $\widetilde{Z} = 0.56$ which we recognise as $p(\bar{\mathbf{x}}_B) = p(\mathbf{x}_B = 1)$. The normalised potential is obtained by dividing $\widetilde{\psi}_A$ by $\widetilde{Z} = 0.56$.

$$\frac{1}{\widetilde{Z}}\widetilde{\psi}_{A,B} = \begin{bmatrix} 0.1428 \\ 0.8571 \end{bmatrix} \tag{3.4.12}$$

which is the conditional distribution $p(\mathbf{x}_A|\bar{\mathbf{x}}_B)$ where $\bar{\mathbf{x}}_B = 1$.

### 3.4.4 Clique trees

A fundamental assumption of the junction tree algorithm is that the cliques are arranged in a tree structure. We refer to this structure as a *clique tree*. We define a clique tree as a singly-connected graph whose nodes represent members of the clique set $\mathcal{C}$. Edges in this graph represent information flow between cliques. Intuitively, the junction tree algorithm is an algorithm that uses these information flows to manipulate the clique potentials to yield the desired marginal probabilities. In particular, after the junction tree algorithm has completed the potential $\psi_C$ will be equal to the marginal probabilities

$$p(\mathbf{x}_{C\cap H}, \bar{\mathbf{x}}_{C\cap E}). \tag{3.4.13}$$

This probability is a non-normalised version of the conditional probability $p(\mathbf{x}_{C\cap H}|\bar{\mathbf{x}}_{C\cap E})$, where the normalisation constant is obtained by summing or integrating $\psi_C$ over $\mathbf{x}_{C\cap H}$. Consequently, we get the important result that the

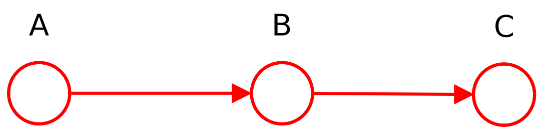

**Figure 3.15:** The cliques in this three-node Markov chain are $\{A, B\}$ and $\{B, C\}$.

desired marginal probabilities can be obtained via a local operation. From the clique marginals the desired marginals can be obtained at low computational cost.

In the previous section we showed how to initialise the clique potentials so as to obtain a representation of the joint probability of the underlying graph. This is a global representation, however, and the individual potentials do not necessarily correspond to local probabilities. Consider the *Markov chain* shown in Figure 3.15. The cliques of this graph are $\{A, B\}$ and $\{B, C\}$. The joint probability distribution is given by

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\mathbf{x}_A)p(\mathbf{x}_B|\mathbf{x}_A)p(\mathbf{x}_C|\mathbf{x}_B). \tag{3.4.14}$$

We find that $p(\mathbf{x}_A)$ and $p(\mathbf{x}_B|\mathbf{x}_A)$ can be grouped together to initialise the potential $\psi_{AB}$ as the marginal $p(\mathbf{x}_A, \mathbf{x}_B)$. However, the remaining factor $\psi_{BC} = p(\mathbf{x}_C|\mathbf{x}_B)$ is not a marginal. To convert this potential into a marginal, we marginalise $\psi_{AB}$ to obtain $p(\mathbf{x}_B)$ and multiply $\psi_{BC}$ by this factor

$$\psi^*_{BC} = p(\mathbf{x}_B)\psi_{BC} = p(\mathbf{x}_B)p(\mathbf{x}_C|\mathbf{x}_B) = p(\mathbf{x}_B, \mathbf{x}_C). \tag{3.4.15}$$

This "transfer of information" from the clique $\{A, B\}$ to the clique $\{B, C\}$ is an instance of the information flow between cliques mentioned above.

After adjusting $\psi_{BC}$ we have achieved the goal of obtaining marginal probabilities over both cliques. However, the joint probability on $p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)$ is no longer equal to the product of clique marginals since

$$\psi_{AB}\psi^*_{BC} = p(\mathbf{x}_A, \mathbf{x}_B)p(\mathbf{x}_B, \mathbf{x}_C). \tag{3.4.16}$$

This problem is addressed in the junction tree algorithm by using *separator sets*. With each edge in the clique tree we associate a set of nodes. This set contains the intersection of the two cliques that it connects, i.e. the set of nodes that "separates" the two sets. For example, in Figure 3.15 both edges will have $\{\mathbf{x}_B\}$ as a separator set. For a general clique tree of $N$ nodes, we have $N - 1$ separator sets. We shall often refer to a separator set as simply a *separator*.

With each separator there is also an associated potential function. Letting $\mathcal{S}$ denote the set of all separator sets, we introduce a potential function $\phi_S(\mathbf{x}_S)$ for each $S \in \mathcal{S}$. Then, given a clique tree with cliques $\mathcal{C}$ and separators $\mathcal{S}$ the joint probability is given by

$$p(\mathbf{x}) = \frac{\prod_C \psi_C(\mathbf{x}_C)}{\prod_S \phi_S(\mathbf{x}_S)}. \tag{3.4.17}$$

For convenience we have omitted the normalising constant $Z$. Instead we use the convention of including the empty set $\emptyset$ as one of the separators and letting the potential on this empty set be $Z$.

Continuing with the example in Figure 3.15 we expand the joint probability distribution as

$$\begin{aligned} p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) &= p(\mathbf{x}_A, \mathbf{x}_B)p(\mathbf{x}_C|\mathbf{x}_B) \\ &= \frac{p(\mathbf{x}_A, \mathbf{x}_B)p(\mathbf{x}_B, \mathbf{x}_C)}{p(\mathbf{x}_B)}. \end{aligned} \tag{3.4.18}$$

Note that this expression is of the same form as (3.4.17) where we define $\psi_{AB} = p(\mathbf{x}_A, \mathbf{x}_B)$, $\psi_{BC} = p(\mathbf{x}_B, \mathbf{x}_C)$ and $\phi_B = p(\mathbf{x}_B)$. Thus, making use of the separator potential, we are able to achieve a representation that is a product of marginals, and at the same time a representation of the joint probability distribution. It is possible to show that we can always find this kind of representation for a given probability distribution (Jordan, 2003).

The separator potentials are initialised to unity and the clique potentials are initialised from CPDs of the underlying Bayesian network as before. Thus, initially the product of clique potentials and separator potentials constitute a global representation of the joint probability distribution.

## 3.4.5 Local consistency

From the discussion of clique trees we note that it is possible for cliques to overlap. That is, the same node may appear in multiple cliques. If the potentials are to represent marginal probabilities, it is necessary that they are consistent across cliques. That is, we require potentials to give the same marginals for nodes that they have in common. However, it is not necessary to compare all pairs of cliques that intersect to ensure that this consistency holds. It will suffice to arrange the cliques into a constrained clique tree called a *junction tree*. In the junction tree we only require that the marginals of neighbouring cliques in the tree are consistent with respect to the nodes that they have in common.

We first consider how to achieve consistency between a pair of cliques. Suppose we have two cliques $V$ and $W$ and suppose that $V$ and $W$ have a non-empty intersection set $S$ as shown in Figure 3.16. The cliques $V$ and $W$ have associated potentials $\psi_V$ and $\psi_W$ and the separator $S$ has potential $\phi_S$. The separator potential is initialised to unity.

We now want to ensure that $\psi_W$ and $\psi_V$ are locally consistent. That is, we want the marginal distribution $\sum_{W\setminus S} \psi_W$ to equal the marginal $\sum_{V\setminus S} \psi_V$ such

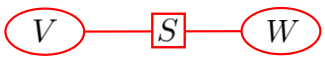

**Figure 3.16:** The basic data structures underlying the flow of information between cliques $V$ and $W$.

that the marginals over the shared nodes are the same with respect to both clique potentials. We can achieve this consistency through a *message passing* procedure. The message passing procedure consists of updating the potentials $\psi_W$, $\psi_V$ and $\phi_S$ in a way that achieves the desired local consistency. We first update $W$ from $V$ through

$$\phi_S^* = \sum_{V \setminus S} \psi_V \tag{3.4.19}$$

$$\psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W. \tag{3.4.20}$$

In (3.4.19) the potential $\psi_V$ is marginalised with respect to $S$. The result is an updated value of the separator potential $\phi_S^*$. In (3.4.20) we rescale the clique potential over $W$ through multiplying by the ratio of the new and old separator potentials.

This update has as important invariant property. Noting that $\psi_V$ is unchanged during the update and defining $\psi_V^* = \psi_V$ we get

$$\begin{aligned}\frac{\psi_V^*\psi_W^*}{\phi_S^*} &= \frac{\psi_V\psi_W\phi_S^*}{\phi_S\phi_S^*} \\ &= \frac{\psi_V\psi_W}{\phi_S}\end{aligned} \tag{3.4.21}$$

from which we conclude that the joint distribution defined by (3.4.17) remains unchanged.

Next we consider the case where we pass information from $W$ back to $V$ in which case we get

$$\psi_S^{**} = \sum_{W \setminus S} \psi_W^* \tag{3.4.22}$$

$$\psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*}\psi_V^*. \tag{3.4.23}$$

Note that $\psi_W^*$ is unchanged during this update. Defining $\psi_W^{**} = \psi_W^*$ we get

$$\begin{aligned}\frac{\psi_V^{**}\psi_W^{**}}{\phi_S^{**}} &= \frac{\psi_V^*\phi_S^{**}\psi_W^*}{\phi_S^*\phi_S^{**}} \\ &= \frac{\psi_V^*\psi_W^*}{\phi_S^*} \\ &= \frac{\psi_V\psi_W}{\phi_S}.\end{aligned} \tag{3.4.24}$$

Again we see that the joint distribution (3.4.17) remains unchanged after the update.

What have we achieved with this forward-backward message passing? Consider the marginal of $\psi_W^{**}$ over $S$

$$\begin{aligned}
\sum_{V\setminus S} \psi_V^{**} &= \sum_{V\setminus S} \frac{\phi_S^{**}}{\phi_S^{*}} \psi_V^{*} \\
&= \frac{\phi_S^{**}}{\phi_S^{*}} \sum_{V\setminus S} \psi_V^{*} \\
&= \frac{\phi_S^{**}}{\phi_S^{*}} \phi_S^{*} \\
&= \phi_S^{**} \\
&= \sum_{W\setminus S} \phi_W^{**}. \qquad (3.4.25)
\end{aligned}$$

We see that $\psi_V^{**}$ and $\psi_W^{**}$ are consistent with respect to their intersection $S$. Thus, we have indeed achieved local consistency while at the same time maintaining the underlying joint probability distribution after the updates.

In the forward pass, from $V$ to $W$, the algorithm stores the marginal of the $V$ potential in the separator potential. In the backward pass, from $W$ to $V$, the algorithm divides the $V$ potential by its stored marginal and multiplies the result by the new marginal $\phi_S^{**}$. This latter marginal is the marginal of the $W$ potential. The rescaling equation essentially substitutes one marginal for another, thus making the two clique potentials consistent. This is achieved by a symmetric algorithm that passes information in both directions, and leaves the joint probability distribution invariant.

As an example consider the Markov chain in Figure 3.15. Initially, the clique potential over $V = \{A, B\}$ is $p(\mathbf{x}_A, \mathbf{x}_B)$ and the clique potential over $W = \{B, C\}$ is $p(\mathbf{x}_C|\mathbf{x}_B)$. The first pair of update equations gives

$$\begin{aligned}
\phi_B^{*} &= \sum_{\mathbf{x}_A} p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_B) \qquad (3.4.26) \\
\psi_{B,C}^{*} &= \frac{p(\mathbf{x}_B)}{1} p(\mathbf{x}_C|\mathbf{x}_B) = p(\mathbf{x}_B, \mathbf{x}_C). \qquad (3.4.27)
\end{aligned}$$

We see that the clique potentials are now marginal probabilities. The backward phase in this case involves marginalising over $p(\mathbf{x}_B, \mathbf{x}_C)$ with respect to $\mathbf{x}_C$ which again yields $\phi_B^{*} = p(\mathbf{x}_B)$. Updating $\psi_{A,B}$ we get

$$\psi_{A,B}^{**} = \frac{p(\mathbf{x}_B)}{p(\mathbf{x}_B)} p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_A, \mathbf{x}_B). \qquad (3.4.28)$$

We now consider the case in which evidence is observed. Suppose for simplicity that the nodes in Figure 3.15 are two-dimensional binary random variables and that the evidence is $\bar{\mathbf{x}}_A = 1$. The first set of updates now gives

$$\begin{aligned} \phi_B^* &= p(\bar{\mathbf{x}}_A, \mathbf{x}_B) & (3.4.29) \\ \psi_{BC}^* &= p(\bar{\mathbf{x}}_A, \mathbf{x}_B)p(\mathbf{x}_C|\mathbf{x}_B) = p(\bar{\mathbf{x}}_A, \mathbf{x}_B, \mathbf{x}_C). & (3.4.30) \end{aligned}$$

Note that it is not possible to marginalise over $\mathbf{x}_A$ in the first step as this variable is observed. The second set of updates become

$$\begin{aligned} \phi_B^{**} &= \sum_{\mathbf{x}_C} p(\bar{\mathbf{x}}_A, \mathbf{x}_B, \mathbf{x}_C) = p(\bar{\mathbf{x}}_A, \mathbf{x}_B) & (3.4.31) \\ \psi_{AB}^{**} &= \frac{p(\bar{\mathbf{x}}_A, \mathbf{x}_B)}{p(\bar{\mathbf{x}}_A, \mathbf{x}_B)} p(\bar{\mathbf{x}}_A, \mathbf{x}_B) = p(\bar{\mathbf{x}}_A, \mathbf{x}_B). & (3.4.32) \end{aligned}$$

Thus we have

$$\begin{aligned} \psi_{AB}^* &= p(\bar{\mathbf{x}}_A, \mathbf{x}_B) & (3.4.33) \\ \phi_B^* &= p(\bar{\mathbf{x}}_A, \mathbf{x}_B) & (3.4.34) \\ \psi_{BC}^* &= p(\bar{\mathbf{x}}_A, \mathbf{x}_B, \mathbf{x}_C) & (3.4.35) \end{aligned}$$

from which we see that we have obtained *non-normalised* marginals. By normalising these marginals, we eventually get the posterior distributions conditioned on the observed evidence $p(\mathbf{x}_B|\bar{\mathbf{x}}_A)$, $p(\mathbf{x}_C|\bar{\mathbf{x}}_A)$ and $p(\mathbf{x}_B, \mathbf{x}_C|\bar{\mathbf{x}}_A)$.

### 3.4.6 Propagation in a clique tree

We now discuss how to perform local updates in a clique tree when we have multiple overlapping cliques. Consider the clique tree in Figure 3.17. Each edge in this tree is associated with a separator. Cliques that are neighbours in this tree are subject to the updating procedure described in the previous section.

We wish to find a set of update rules that ensures that any local consistencies that have been established between neighbouring cliques are not broken by subsequent updates between the clique and its other neighbours. Suppose that we have achieved local consistency between $V$ and $W$ using the pair of updates discussed in the previous section, and subsequently we update $W$ from its other neighbours. The latter would in general break the consistency we have achieved between $V$ and $W$. We can solve this problem using a *message-passing* protocol that ensures existing consistencies are not broken by subsequent updates.

We think of the update of one clique based on another as *passing a message.* That is, we pass a message from $V$ to $W$ by evaluating (3.4.19) and (3.4.20).

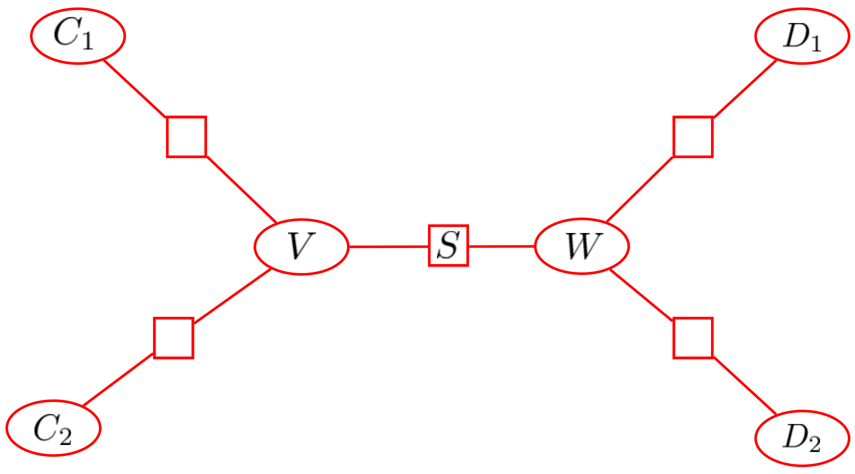

**Figure 3.17:** Example of a clique tree with separators represented as square nodes.

In general, as we saw in the previous section, we require a message in both directions in order to render a pair of cliques consistent with each other.

In the junction tree algorithm the desired consistency is obtained by constraining the order in which the updates are performed.

**Message-Passing Protocol.** *A clique can send a message to a neighbouring clique only when it has received messages from all of its other neighbours.*

For example, in Figure 3.17, we can only send a message from $W$ to $V$ when $W$ has received messages from its other neighbours $D_1$ and $D_2$. The following argument verifies the correctness of the protocol. Assume that $W$ has received all of the messages from its other neighbours, and is sending a message to $V$. There are two possible scenarios; either $V$ has not yet sent its message to $W$, or $V$ has already sent its message to $W$. In the latter case, we know that $V$ has already received messages from all of its other neighbours. The message from $W$ to $V$ renders the cliques consistent. Neither clique receives any additional messages, thus consistency is maintained. In the former case, $W$ sends a message to $V$, storing its marginal on $S$, and waits. At some later stage, $V$ will have received all of the messages from its other neighbours and will send a message to $W$. This message will utilise the stored marginal and render $W$ consistent with $V$. Neither clique will undergo any additional updates and consistency is maintained.

One way of implementing the message-passing protocol is using a recursive algorithm on a tree data structure. In a general clique tree we choose one of the cliques as the root. Once a root of the clique tree has been designated, the tree becomes an oriented tree with each leaf having a unique path to the root. Each leaf can send a message inward at any time. Interior nodes send a message toward the root once they have received messages from all their children. Once all messages have arrived at the root, we propagate messages outward to the leaves. This procedure is formalised in Algorithms 1 and 2.

**Algorithm 1** CollectEvidence(node)

```
for each child of node do
    Update(node, CollectEvidence(child))
end for
return node
```

**Algorithm 2** DistributeEvidence(node)

```
for each child of node do
    Update(child, node)
    DistributeEvidence(child)
end for
return node
```

In algorithms 1 and 2 the routine `Update(V,W)` invokes the pair of update equations (3.4.19) and (3.4.20). Calling `CollectEvidence(root)` followed by `DistributeEvidence(root)` causes the messages to propagate inward to the root and outward to the leaves. In order to see that the `CollectEvidence` and `DistributeEvidence` recursions respect the message-passing protocol consider the following argument.

When `CollectEvidence` is called at a node, the node calls all of its other neighbours and waits on return messages from those nodes before returning a message back to its caller. Thus, `CollectEvidence` obeys the protocol. After `CollectEvidence` has run, each node has received a message from all of its neighbours except its parent. Once it receives a message from its parents it is free to send messages to any other node. `DistributeEvidence` sends a message from its parent to its child before calling itself on that child. Thus, `DistributeEvidence` also respects the message-passing protocol.

### 3.4.7 The junction tree property

The final issue that needs to be addressed for the junction tree algorithm is illustrated by Figure 3.18. In particular, we note that the node $C$ appears in two different cliques that are not neighbours. Since our algorithm only guarantees local consistency, there is no guarantee that the two cliques containing $C$ will be consistent. In general, local consistency does not imply global consistency.

Note that the lack of global consistency does not imply that we have an incorrect representation of the joint probability distribution of the underlying graph. In fact, the junction tree algorithm does not alter the joint probability,

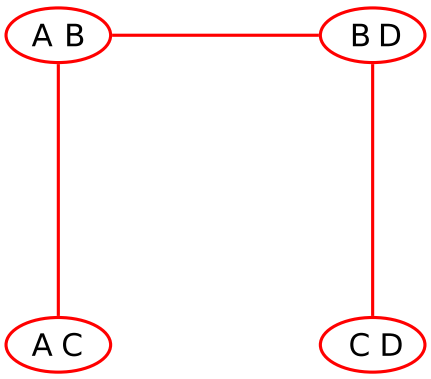

**Figure 3.18:** Example of a clique tree where the junction tree property does not hold.

and thus we maintain a correct representation of the joint throughout. However, we may still fail to achieve a global consistency. The solution to this problem is to impose the *junction tree property.*

**Definition 2.** *A clique tree possesses the* ***junction tree property*** *if for every pair of cliques $V$ and $W$, all cliques on the (unique) path between $V$ and $W$ contain $V \cap W$.*

A clique tree that possesses the junction tree property is referred to as a *junction tree.* It should be clear that the graph in Figure 3.18 is not a junction tree.

An intuitive interpretation of the junction tree property from the viewpoint of inference is as follows. If a node $A$ appears in two cliques in a junction tree, the $A$ is necessarily contained in every clique along the path between these two cliques. If the cliques along the path are *pairwise* consistent with respect to $A$ then they will be jointly consistent with respect to $A$. We therefore have that in a junction tree, *local consistency implies global consistency.* As a consequence, if we have a clique tree that is also a junction tree, and if we run the message-passing procedure as described in the previous section, we achieve not only local consistency but also global consistency; we can get the same answer for node $A$ from any potential that contains $A$.

However, recall that our goal is to obtain a set of potentials that are not only consistent, but that are also marginals. That is, each clique potential represents the marginal probability of the nodes in the clique. In fact, in a junction tree the message-passing algorithm not only achieves global consistency, but also yields the marginal distribution over cliques. Before we prove this result we need the following lemma.

**Lemma 3.4.1.** *Let $G = (V, E)$ be a Bayesian network with nodes $V$. Let $C$ be a leaf in the junction tree of the graph. Let $S$ be the (unique) separator associated with $C$ . Let $R = C \backslash S$ be the set of nodes in $C$ but not in the* *separator set, and let $U = V \backslash C$ be the set of nodes in $V$ but not in $C$. Then we have*

$$R \perp\!\!\!\perp U \mid S. \tag{3.4.36}$$

*Proof.* Proof by contradiction. Suppose that $A \in R$ has a neighbour $N \in U$. Consider the maximal complete subset containing both $A$ and $N$. This clique is not in $C$ because $N \notin C$. However, $A$ cannot be contained in any clique other than $C$ because $A$ would have to belong to $S$ as well, by the junction tree property, and nodes in $R$ are not in $S$ by definition. Thus, no such $N$ exists and $S$ therefore d-separates $A$ from $U$. Since $A$ is arbitrary we have that the separator set $S$ d-separates $R$ from $U$. □

We can now prove the main result of the analysis of the junction tree algorithm.

**Theorem 3.4.2.** *Let the joint probability $p(\mathbf{x}_H, \bar{\mathbf{x}}_E)$ be represented by the clique potentials $\psi_C$ and separator potentials $\phi_S$ of a junction tree. When the junction tree algorithm terminates, the clique potentials and separator potentials are proportional to local marginal probabilities. In particular,*

$$\phi_C = p(\mathbf{x}_C, \bar{\mathbf{x}}_E) \tag{3.4.37}$$

$$\phi_S = p(\mathbf{x}_S, \bar{\mathbf{x}}_E). \tag{3.4.38}$$

*Proof.* The separators are subsets of the cliques. That the separator potentials are proportional to marginals therefore follows from the fact that they are consistent with the clique potentials. Thus we need only prove the results for the clique potentials.

The proof is a proof by induction. The result holds for the base case of a BN consisting of a single clique by definition. Let us suppose that the result holds for junction trees of $N$ or fewer cliques and consider a junction tree with $N + 1$ cliques.

We choose a clique $\widetilde{C}$ that is a leaf of the junction tree. Let $\widetilde{S}$ be the corresponding separator, let $\widetilde{R} = \widetilde{C} \backslash \widetilde{S}$ and let $\widetilde{T} = V \backslash \widetilde{C}$. We also define analogous quantities in which the evidence variables are omitted. In particular, let $C = \widetilde{C} \backslash E$, $R = \widetilde{R} \backslash E$ and $T = \widetilde{T} \backslash E$. From Lemma 3.4.1 we have that

$$p(\mathbf{x}_H, \bar{\mathbf{x}}_E) = p(\mathbf{x}_R, \mathbf{x}_S, \mathbf{x}_T, \bar{\mathbf{x}}_E) = p(\mathbf{x}_R | \mathbf{x}_S, \bar{\mathbf{x}}_E) p(\mathbf{x}_S, \mathbf{x}_T, \bar{\mathbf{x}}_E). \tag{3.4.39}$$

Marginalising both sides over $\mathbf{x}_R$ we get

$$\begin{aligned} p(\mathbf{x}_S, \mathbf{x}_T, \bar{\mathbf{x}}_E) &= \sum_{\mathbf{x}_R} p(\mathbf{x}_H, \bar{\mathbf{x}}_E) \\ &= \sum_{\mathbf{x}_R} \frac{\prod_C \psi_C(\mathbf{x}_C)}{\prod_S \phi_S(\mathbf{x}_S)} \\ &= \sum_{\mathbf{x}_R} \frac{\psi_C}{\phi_S} \frac{\prod_{C' \neq C} \psi_{C'}(\mathbf{x}_{C'})}{\prod_{S \neq S'} \phi_{S'}(\mathbf{x}_{S'})} \\ &= \frac{\sum_{\mathbf{x}_R} \psi_C}{\phi_S} \frac{\prod_{C' \neq C} \psi_{C'}(\mathbf{x}_{C'})}{\prod_{S \neq S'} \phi_{S'}(\mathbf{x}_{S'})} \\ &= \frac{\prod_{C' \neq C} \phi_{C'}(\mathbf{x}_{C'})}{\prod_{S \neq S'} \phi_{S'}(\mathbf{x}_{S'})} \end{aligned} \tag{3.4.40}$$

where the last step follows from the fact that $C$ and $S$ are consistent and thus $\sum_R \psi_C = \phi_S$.

Equation (3.4.40) shows that $p(\mathbf{x}_S, \mathbf{x}_T, \bar{\mathbf{x}}_E)$ is represented by the clique potentials and separator potentials on the junction tree over $S \cup T$. By the induction hypothesis, after a full round of message passing the clique potentials on this junction tree are equal to marginals.

It remains to show that the clique potential on $C$ is a marginal. Let $D$ be the neighbour clique of $C$ in the junction tree. By consistency we have $\phi_S(\mathbf{x}_S) = \sum_{D \setminus S} \psi_D(\mathbf{x}_D)$. We have $\psi_D = p(\mathbf{x}_D, \bar{\mathbf{x}}_E)$ and thus $\psi_S(\mathbf{x}_S) = p(\mathbf{x}_S, \bar{\mathbf{x}}_E)$. Thus

$$\begin{aligned} p(\mathbf{x}_R | \mathbf{x}_S, \bar{\mathbf{x}}_E) &= \frac{\psi_C(\mathbf{x}_C)}{\phi_S(\mathbf{x}_S)} \\ &= \frac{\psi_C(\mathbf{x}_C)}{p(\mathbf{x}_S, \bar{\mathbf{x}}_E)} \end{aligned} \tag{3.4.41}$$

which implies that $\psi_C(\mathbf{x}_C) = p(\mathbf{x}_C, \bar{\mathbf{x}}_E)$. □

### 3.4.8 Summary of the junction tree algorithm

We now have most of the pieces that comprises the junction tree algorithm in place. There is still the issues of triangulation and how to construct a junction tree from a Bayesian network.

Triangulation is a necessary condition for a graph to have a junction tree. If the graph is not triangular it must *triangulated* using one of several possible algorithms (Jordan, 2003) before we run the junction tree algorithm.

Constructing a junction tree is an instance of the *maximal spanning tree* problem (Jordan, 2003). This problem can be solved using any one of a number of greedy algorithms, for instance Kruskal's algorithm (Kruskal, 1956).

We shall not discuss these two problems in any further detail. In summary, the steps involved in the junction tree algorithm are as follows.

**Moralisation** The moralisation step converts a directed graph into an undirected graph. Nodes that have a common child are linked, and directed edges are converted to undirected edges. The local probability of each node is multiplied onto the potential of a clique that contains the node and its parents.

**Introduction of evidence** Evidence is introduced by taking slices of the potentials.

**Triangulation** The graph is triangulated. The potential of each clique of the original graph is multiplied onto the potential of a clique that contains the clique.

**Construction of junction tree** A junction tree is constructed by forming a maximal spanning tree from the cliques of the triangulated graph. Separators are introduced and their potentials are initialised to unity.

**Propagation of probabilities** Computation proceeds in the junction tree using the pair of update equations

$$\phi_S^* = \sum_{V\setminus S} \psi_V \tag{3.4.42}$$

$$\psi_V^* = \frac{\phi_S^*}{\phi_S}\psi_V. \tag{3.4.43}$$

The updates must respect the Message-Passing Protocol. This is achieved by designating a root node in the clique graph and calling `CollectEvidence` and `DistributeEvidence` from the root node. Once the algorithm terminates, the clique potentials and separator potentials are proportional to marginal probabilities. Further marginalisation can be performed to obtain the probabilities of single nodes or subsets of nodes.

The junction tree inference algorithm is the most efficient method of performing exact inference in Bayesian networks. See Jordan (2003) for an analysis of the computational complexity of the junction tree algorithm.

### 3.4.9 Maximum probability configurations

We are often interested in finding the most likely configuration of the hidden nodes in the Bayesian network given the observed evidence, that is, the set of values of the hidden variables that maximises the joint probability over all the nodes in the network.

It turns out that we can simply replace the "sum" operators with "max" operators in the junction tree algorithm and all the results are still valid. In fact, the only step of the algorithm that explicitly refers to summation is the marginalisation step (3.4.42). This equation becomes

$$\phi_S^* = \max_{V \setminus S} \psi_V. \tag{3.4.44}$$

The key result of the junction tree algorithm is that each clique potential is equal to its marginal probability. In that case "marginal" means that the variables not contained in the clique have been *summed out*. If we replace summation by maximisation, we obtain a similar notion. The "marginal" means that the variables not contained in the clique have been *maxed out*. That is

$$\psi_C(\mathbf{x}_C) = \max_{V \setminus C} p(\mathbf{x}_C). \tag{3.4.45}$$

We can interpret the resulting entries in the clique potential as the values of maximal probability attainable for each possible configuration of the random variables $\mathbf{x}_C$. Maximising over these values, we obtain the actual configuration. This result is useful for classification where we are typically looking for the model which has the largest probability of having generated the observations.

## 3.5 Variational learning

We have yet to address how to choose a suitable model complexity. By model complexity we typically mean the number of model parameters that need to be estimated from data. Given an infinite amount of training data and assuming that the complexity of the model is greater than that of the underlying problem, the EM algorithm can be shown to converge to the correct model. However, the complexity of the problem is rarely known and the availability of data is in most cases not only finite, but severely limited. It is therefore essential to choose a model complexity that reflects the amount of available training data (see Bishop, 2007, Chapter 10). If the model complexity is too high there will not be enough data to accurately estimate the large number of parameters in complex models. The model tends to describe the training data exactly, but does not generalise well to new data. This phenomenon is referred to as *overfitting* and leads to decreased system performance.

Until now we have treated model parameters as deterministic variables. However, it is also possible to treat the model parameters themselves as random variables. In this case, the parameters become nodes in the Bayesian network similar to the latent and observed variables we have seen so far. We shall see how this approach in fact leads to automatic model complexity selection by choosing regularising priors and to an algorithm that is similar to EM and has the same computational complexity.

As before let $\mathbf{X}_H$ denote the set of latent variables, $\mathbf{X}_E$ the set of observed variables, and $\boldsymbol{\theta}$ the model parameters. We now choose a prior distribution $p(\boldsymbol{\theta})$ over the parameters. Following (3.3.6) the log marginal likelihood then decomposes as

$$\ln p(\mathbf{X}_E) = \mathcal{L}(q) + \mathrm{KL}(q||p) \tag{3.5.1}$$

where we have defined

$$\mathcal{L}(q) = \int \sum_{\mathbf{X}_H} q(\mathbf{X}_H, \boldsymbol{\theta}) \ln \left\{ \frac{p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})}{q(\mathbf{x}_H, \boldsymbol{\theta})} \right\} \mathrm{d}\boldsymbol{\theta} \tag{3.5.2}$$

$$\mathrm{KL}(q||p) = -\int \sum_{\mathbf{X}_H} q(\mathbf{X}_H, \boldsymbol{\theta}) \ln \left\{ \frac{p(\mathbf{X}_H, \boldsymbol{\theta}|\mathbf{X}_E)}{q(\mathbf{X}_H, \boldsymbol{\theta})} \right\} \mathrm{d}\boldsymbol{\theta}. \tag{3.5.3}$$

The above decomposition is verified using a similar approach as for (3.3.6). Note that $\mathcal{L}(q)$ is a functional of $q(\mathbf{X}_H, \boldsymbol{\theta})$. We have assumed that latent variables are discrete and therefore summed over, and that the parameters are continuous and therefore integrated over as this is the typical case. However, sums can easily be replaced with integrals or integrals with sums and the analysis will be unchanged.

We recognise the two equations (3.5.2) and (3.5.3) as (3.3.7) and (3.3.8) from the discussion of EM in Section 3.3.1. However, there is an important difference. The parameter vector $\boldsymbol{\theta}$ is no longer a deterministic variable, but forms part of the set of hidden random variables. Thus, there are no deterministic parameters to optimise with respect to. As we saw with EM, we can maximise the lower bound $\mathcal{L}(q)$ by optimising with respect to the distribution $q(\mathbf{X}_H, \boldsymbol{\theta})$, which is equivalent to minimising the KL divergence. If we allow any possible choice for $q(\mathbf{X}_H, \boldsymbol{\theta})$, then the maximum of the lower bound occurs when $q(\mathbf{X}_H, \boldsymbol{\theta})$ equals the posterior distribution $p(\mathbf{X}_H, \boldsymbol{\theta}|\mathbf{X}_E)$. Finding $p(\mathbf{X}_H, \boldsymbol{\theta}|\mathbf{X}_E)$ is intractable (Beal and Ghahramani, 2004), however, but we can approximate the posterior using variational calculus (Wainwright and Jordan, 2008).

The basic idea of variational learning is to simultaneously approximate the intractable joint distribution over both hidden states and parameters with a simpler distribution by assuming that the hidden states and parameters are independent given the observed data. Thus we constrain the posterior to have the form

$$q(\mathbf{X}_H, \boldsymbol{\theta}) = q(\mathbf{X}_H)q(\boldsymbol{\theta}) \tag{3.5.4}$$

which is called a *variational approximation.*

Note that we are omitting the subscripts on the $q$ distributions in the same way that we normally do with $p$ distributions. That is, different $q(\cdot)$ are in general different distributions and we are relying on the arguments to distinguish them. As in EM we omit the dependency of $q(\cdot)$ on $\mathbf{X}_E$ in our notation. It is understood that $q(\cdot)$ is always conditioned on the observed evidence.

It should be emphasised that we are making no further assumptions about the distributions other than the factorisation (3.5.4). In particular, we place no restriction on the functional forms of the individual $q(\cdot)$ factors. This approximation framework corresponds to a framework developed in physics called *mean field theory* (Parisi, 1988).

Amongst all distributions $q(\mathbf{X}_H, \boldsymbol{\theta})$ having the form (3.5.4) we seek the distribution for which the lower bound $\mathcal{L}(q)$ is largest. We therefore wish to make a free form (variational) optimisation of $\mathcal{L}(q)$ with respect to $q(\mathbf{X}_H)$ and $q(\boldsymbol{\theta})$ which we do by optimising with respect to each of the factors in turn in a procedure similar to EM.

Suppose we consider $q(\mathbf{X}_H)$ as fixed and optimise with respect to $q(\boldsymbol{\theta})$. Substituting (3.5.4) into (3.5.2), using the fact the $q(\boldsymbol{\theta})$ integrates to 1, and isolating the dependence on $q(\mathbf{X}_H)$ give

$$\begin{aligned}\mathcal{L}(q) &= \int \sum_{\mathbf{X}_H} q(\mathbf{X}_H) q(\boldsymbol{\theta}) \left\{ \ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta}) - \ln q(\mathbf{X}_H) - \ln q(\boldsymbol{\theta}) \right\} \mathrm{d}\boldsymbol{\theta} \\ &= \int q(\boldsymbol{\theta}) \left\{ \sum_{\mathbf{X}_H} q(\mathbf{X}_H) \ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta}) \right\} \mathrm{d}\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \ln q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} + \text{const} \\ &= \int q(\boldsymbol{\theta}) \ln \widetilde{p}(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \ln q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} + \text{const} \qquad (3.5.5)\end{aligned}$$

where we have defined

$$\begin{aligned}\ln \widetilde{p}(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta}) &= \sum_{\mathbf{X}_H} q(\mathbf{X}_H) \ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta}) \\ &= \mathbb{E}_H \left[ \ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta}) \right] \qquad (3.5.6)\end{aligned}$$

where the expectation is with respect to $q(\mathbf{X}_H)$.

The functional $\mathcal{L}(q)$ is optimised with respect to $q(\boldsymbol{\theta})$ by recognising that (3.5.5) is the negative Kullback-Liebler divergence between $q(\boldsymbol{\theta})$ and $\widetilde{p}(\mathbf{X}_E, \mathbf{X}_H)$ (Bishop, 2007). Thus maximising (3.5.5) is equivalent to minimising the KL divergence and the minimum occurs when $q(\boldsymbol{\theta}) = \widetilde{p}(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})$. Thus we obtain a general expression for the optimal solution $q^*(\boldsymbol{\theta})$ given by

$$\ln q^*(\boldsymbol{\theta}) = \mathbb{E}_H[\ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})] + \text{const} \qquad (3.5.7)$$

where the expectation is with respect to $q(\mathbf{X}_H)$. The additive constant is determined by normalising the distribution

$$q^*(\boldsymbol{\theta}) = \frac{\exp\{\mathbb{E}_H[\ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})]\}}{\int \exp\{\mathbb{E}_H[\ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})]\} \mathrm{d}\boldsymbol{\theta}}. \qquad (3.5.8)$$

Following a similar line of argument we can also show that

$$q^*(\mathbf{X}_H) = \frac{\exp\{\mathbb{E}_{\boldsymbol{\theta}}[\ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})]\}}{\sum_{\mathbf{X}_H} \exp\{\mathbb{E}_{\boldsymbol{\theta}}[\ln p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})]\}}. \qquad (3.5.9)$$

where the expectations are taken with respect to $q(\boldsymbol{\theta})$.

Attias (2000) shows that if the complete data likelihood is in the exponential family *and* the prior distribution over the parameters is *conjugate* to the complete data likelihood, variational learning can be performed using an EM like algorithm which can be considered as a coordinate ascent in the function space of variational distributions. This algorithm is often called Variational Bayes (VB) in the literature.

In the VB algorithm we alternate between maximising $\mathcal{L}(q)$ with respect to the free distributions $q(\boldsymbol{\theta})$ and $q(\mathbf{X}_{\boldsymbol{H}})$ using (3.5.8) and (3.5.9), respectively. Taking the expectation with respect to $q(\boldsymbol{\theta})$ and optimising with respect to $q(\mathbf{X}_H)$ is analogous to the E step of the EM algorithm where the parameters are fixed and we optimise with respect to the latent variables. Taking the expectation with respect to $q(\mathbf{X}_H)$ and optimising with respect to $q(\boldsymbol{\theta})$ is analogous to the M step. In each case the optimal variational posterior has the same functional form as the prior due to our choice of conjugate priors (Beal and Ghahramani, 2004).

We next show how to perform the variational updates in the case that the node is a multinomial variable or a Gaussian variable. We again assume that parent variables are multinomial and can be represented as a single parent. The variables $\mathbf{Z}_n, \mathbf{x}_n$, $\mathbf{z}_n$, $\mathbf{z}_{\pi_n}$ and parameters $\boldsymbol{\rho}_k$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ are defined as in the discussion of EM. In addition we define the *precision* matrix, which is the inverse of the covariance matrix, as $\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$.

**Multinomial.** In the case that $\mathbf{z}_n$ is a multinomial variable the conjugate prior of the parameters $\rho_{kl}$ is the Dirichlet distribution (A.2.11) given by

$$p(\boldsymbol{\rho}_k) = \mathrm{Dir}(\boldsymbol{\rho}_k|\boldsymbol{\alpha}_k^{(0)}) = C(\boldsymbol{\alpha}_k^{(0)}) \prod_{l=1}^{L} \rho_{kl}^{\alpha_{kl}^{(0)}-1} \tag{3.5.10}$$

where we as usual ignore the dependencies of the parameters on the node $v$ in our notation. The factor $C(\cdot)$ is the normalisation constant for the Dirichlet distribution. The parameter $\alpha_{kl}^{(0)}$ can be interpreted as the effective *prior* number of observations associated with each component $l$ of the distribution associated with parent configuration $k$. If the value of $\alpha_{kl}^{(0)}$ is small, then the posterior distribution will be influenced primarily by the data rather than the prior. If the value of $\alpha_{kl}^{(0)}$ is large the posterior distribution will be more influenced by the prior. Thus, $\alpha_{kl}^{(0)}$ can effectively be used to adjust the level of *regularisation* in the learning algorithm.

Assuming that parameters $\boldsymbol{\rho}_k$ are independent we can optimise with respect to each $\boldsymbol{\rho}_k$ independently (Beal and Ghahramani, 2004). Then, from (3.5.6) and (3.5.10) we it follows that

$$\begin{aligned}
\ln q(\boldsymbol{\rho}_k) &= \ln\left\{\prod_{k=1}^{K}\prod_{l=1}^{L}\rho_{kl}^{\alpha_{kl}^{(0)}}\right\} + \sum_{n=1}^{N}\sum_{\mathbf{Z}_n} q(\mathbf{Z}_n)\ln\left\{\prod_{k=1}^{K}\prod_{l=1}^{L}\rho_{kl}^{z_{nl}z_{\pi_{nk}}}\right\} + \text{const} \\
&= \sum_{k=1}^{K}\sum_{l=1}^{L}\alpha_{kl}^{(0)}\ln\rho_{kl} + \sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{n=1}^{N}\sum_{\mathbf{Z}_n} q(\mathbf{Z}_n) z_{nl} z_{\pi_{nk}} \ln\rho_{kl} + \text{const} \\
&= \sum_{k=1}^{K}\sum_{l=1}^{L}\alpha_{kl}^{(0)}\ln\rho_{kl} + \sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{n=1}^{N} r_{nkl}\ln\rho_{kl} + \text{const} \\
&= \sum_{k=1}^{K}\sum_{l=1}^{L}\alpha_{kl}^{(0)}\ln\rho_{kl} + \sum_{k=1}^{K}\sum_{l=1}^{L} N_{kl}\ln\rho_{kl} + \text{const} \qquad (3.5.11)
\end{aligned}$$

where we have defined

$$\begin{aligned}
r_{nkl} &= \sum_{n=1}^{N}\sum_{\mathbf{Z}_n} q(\mathbf{Z}_n) z_{nk} z_{\pi_{nk}} \\
&= \sum_{n=1}^{N}\mathbb{E}[z_{nk}z_{\pi_{nk}}] \qquad (3.5.12) \\
N_{kl} &= \sum_{n=1}^{N} r_{nkl}. \qquad (3.5.13)
\end{aligned}$$

where the expectation is with respect to $q(\mathbf{Z}_n)$. Taking the exponential of both sides in (3.5.11) we get

$$q^*(\boldsymbol{\rho}_k) = \text{Dir}(\boldsymbol{\rho}_k|\boldsymbol{\alpha}_k). \qquad (3.5.14)$$

We see that posterior has the same distribution as the prior. The updated hyperparamter $\boldsymbol{\alpha}_k$ has components

$$\alpha_{kl} = \alpha_{kl}^{(0)} + N_{kl} \qquad (3.5.15)$$

where we see that the $\alpha_{kl}$ parameters are indeed the effective number of observations of component $l$ for parent configuration $k$.

**Gaussian.** Instead of working with the covariance matrix as in the case of EM we shall find that the analysis is simpler if we work with the precision matrix $\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$. The conjugate prior for the joint distribution over $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$ is the Gaussian-Wishart distribution (A.2.24) given by

$$p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = p(\boldsymbol{\mu}_k|\boldsymbol{\Lambda}_k)p(\boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, (\beta_k^{(0)}\boldsymbol{\Lambda}_k^{-1}))\mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k^{(0)}, \nu_k^{(0)}). \qquad (3.5.16)$$

Following a similar line of reasoning as in the case of a multinomial we get the posterior $\ln q^*(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

$$\ln q^*(\boldsymbol{\mu}_k, \boldsymbol{\Lambda})_k = \ln p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) + \sum_{n=1}^{N} r_{nk} \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}) + \text{const} \tag{3.5.17}$$

from which we again see that the posterior has the same form as the prior.

$$q^*(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k) = \mathcal{N}(\boldsymbol{\mu}_k | \mathbf{m}_k, (\beta_k \boldsymbol{\Lambda})^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k | \mathbf{W}_k, \nu_k) \tag{3.5.18}$$

where the updated set of hyperparamter are given by

$$\beta_k = \beta_0 + N_k \tag{3.5.19}$$

$$\mathbf{m}_k = \frac{1}{\beta_k}(\beta_0 \mathbf{m}_0 + N_k \bar{\mathbf{x}}_k) \tag{3.5.20}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + N_k \mathbf{S}_k + \frac{\beta_0 N_k}{\beta_0 + N_k}(\bar{\mathbf{x}}_k - \mathbf{m}_0)(\bar{\mathbf{x}}_k - \mathbf{m}_0)^{\mathrm{T}} \tag{3.5.21}$$

$$\nu_k = \nu_0 + N_k \tag{3.5.22}$$

where

$$N_k = \sum_{n=1}^{N} r_{nk} \tag{3.5.23}$$

$$\bar{\mathbf{x}}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} \mathbf{x}_n \tag{3.5.24}$$

$$\mathbf{S}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} (\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^{\mathrm{T}}. \tag{3.5.25}$$

See (Bishop, 2007) for a detailed derivation of these results. Note that the quantities are analogous to quantities evaluated in the EM algorithm. In the case of diagonal covariance matrices the results are the same with off-diagonal elements set to zero.

### 3.5.1 Inference for variational learning

In order to calculate the responsibilities $r_{nk} = \mathbb{E}[z_{nk} z_{\pi_{nk}}]$ in the latent multinomial case and $r_{nk} = \mathbb{E}[z_{\pi_{nk}}]$ in the observed Gaussian case we need to posterior distributions $q(\mathbf{Z}_n)$. Again, we only require the local posterior probabilities $p(\mathbf{z}_v, \mathbf{z}_{\pi_v})$. Beal and Ghahramani (2004) shows that these probabilities can be obtained from clique potentials using the junction tree algorithm with parameters $\widetilde{\boldsymbol{\theta}}$ implicitly defined as

$$\phi(\widetilde{\theta}) = \mathbb{E}[\boldsymbol{\phi}(\boldsymbol{\theta})] \tag{3.5.26}$$

where the expectation is taken with respect to $q(\boldsymbol{\theta})$, and $\phi(\boldsymbol{\theta})$ is the natural parameters of the exponential family form (A.2.25) of the distribution (Beal and Ghahramani, 2004). Thus, the analogous quantities in VB to the expected sufficient statistics in EM are calculated by running the junction tree algorithm with model parameters given by the expected values of the natural parameters in the exponential family form of the distribution.

In the case of a multinomial variable the natural parameters are $\ln \boldsymbol{\rho}_k$ given by

$$\ln \widetilde{\rho}_{kl} = \mathbb{E}[\ln \rho_{kl}] = \psi(\alpha_{kl}) - \psi(\widehat{\alpha}_k) \tag{3.5.27}$$

where $\psi(\cdot)$ is the digamma function and $\widehat{\alpha} = \sum_{k=1}^{K} \alpha_k$ (see A.2.17). For a Gaussian variable we get that

$$\begin{aligned} \mathbb{E}[(\mathbf{x}_n - \boldsymbol{\mu}^{\mathrm{T}})(\mathbf{x}_n - \boldsymbol{\mu}_k)] &= D\beta_k^{-1} + \nu_k(\mathbf{x}_n - \mathbf{m}_k)^{\mathrm{T}}\mathbf{W}_k(\mathbf{x}_n - \mathbf{m}_k) \\ \mathbb{E}[|\boldsymbol{\Lambda}_k|] &= \sum_{i=1}^{D} \psi\left(\frac{\eta_k + 1 - i}{2}\right) + D\ln 2 + \ln|\boldsymbol{W}_k| \\ &. \end{aligned} \tag{3.5.28}$$

where $D$ is the dimensionality of the observed Gaussian variable. In practice when evaluating the multinomial and Gaussian distributions we use these expectation instead of the quantities $\boldsymbol{\rho}_k$ in (A.2.6) and $|\boldsymbol{\Lambda}_k|$ and $(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}(\mathbf{x}_n - \boldsymbol{\mu}_k)$ in (A.2.1) with $\boldsymbol{\Sigma}_k = \boldsymbol{\Lambda}^{-1}$.

When classifying new data with models learned using VB equations (3.5.13), (3.5.23), (3.5.24) and (3.5.23) may be used instead of (3.3.23), (3.3.34), (3.3.33) and (3.3.35) respectively. Standard Bayesian network classification techniques such as the maximum likelihood probability configurations in Section 3.4.9 may then be applied. This approach was shown to give good results in Valente and Wellekens (2003).

### 3.5.2 Variational lower bound

In the EM algorithm we used the log likelihood as a measure of convergence as this quantity is guaranteed to be strictly increasing at each step of the EM algorithm. The analogous quantity for the VB algorithm is the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_{\mathbf{X}_H} \int q(\mathbf{X}_H, \boldsymbol{\theta}) \ln \left\{ \frac{p(\mathbf{X}_E, \mathbf{X}_H, \boldsymbol{\theta})}{q(\mathbf{X}_H, \boldsymbol{\theta})} \right\} \mathrm{d}\boldsymbol{\theta}. \tag{3.5.29}$$

Bishop (2007, Chapter 10) describes how to evaluate the lower bound for models consisting of latent multinomial variables and observed Gaussian variables.

### 3.5.3 Penalising complex models

An important property of variational learning is automatic model complexity selection. To develop some intuition of the origin of this complexity selection property we consider the lower bound $\mathcal{L}(q)$. We rewrite (3.5.2) as

$$\begin{aligned}\mathcal{L}(q) &= \int \sum_{\mathbf{X}_H} q(\mathbf{X}_H) q(\boldsymbol{\theta}) \ln \frac{p(\mathbf{X}_E, \mathbf{X}_H|\boldsymbol{\theta})}{q(\mathbf{X}_H)} \mathrm{d}\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \ln \frac{p(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} \mathrm{d}\boldsymbol{\theta} \\ &= \mathbb{E}\left[\ln \frac{p(\mathbf{X}_E, \mathbf{X}_H|\boldsymbol{\theta})}{q(\mathbf{X}_H)}\right] - \mathrm{KL}(q(\boldsymbol{\theta})||p(\boldsymbol{\theta})). \qquad (3.5.30)\end{aligned}$$

where the expectation is with respect to both $q(\mathbf{X}_H)$ and $q(\boldsymbol{\theta})$. The first term on the right-hand side is the expected log likelihood. The second term is the KL distance between the prior and posterior over the parameters. As the number of parameters increases, the KL distance increases and thus the lower bound is decreased (Attias, 2000). This is the reason why the VB learning algorithm tends to give preference to less complex models with fewer parameters.

The penalised likelihood becomes transparent in the large sample limit $N \to \infty$ where the parameter posterior is sharply peaked about the most probable value $\boldsymbol{\theta} = \boldsymbol{\theta}_0$. It can be shown that the KL penalty reduces to $(|\boldsymbol{\theta}_0|/2) \ln N$ which is linear in the number of parameters $|\boldsymbol{\theta}_0|$. $\mathcal{L}(q)$ then corresponds precisely to the Bayesian information criterion (BIC) (Bishop, 2007) . Thus, the BIC model selection criterion follows as a limiting case of the VB framework.

## 3.6 Example: Gaussian mixture models

### 3.6.1 Representation

The Gaussian mixture model (GMM) is a popular model for density estimation and is represented by a simple two-node Bayesian network. A GMM is defined as a weighted sum of Gaussian distributions (see Section A.2.1)

$$p(\mathbf{x}) = \sum_{k=1}^{K} \rho_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad (3.6.1)$$

where $\rho_k$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ is the weight, mean and covariance matrix of the $k$-th Gaussian mixture component, respectively, satisfying the condition $\sum_{k=1}^{K} \rho_k = 1$.

In order to represent a Gaussian mixture model as a Bayesian network we introduce a $K$-dimensional binary random variable $\mathbf{z}$. We define the joint distribution $p(\mathbf{x}, \mathbf{z})$ in terms of the marginal distribution $p(\mathbf{z})$ and a conditional distribution $p(\mathbf{x}|\mathbf{z})$. This model corresponds to the Bayesian network in Figure

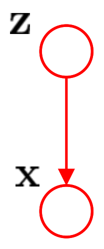

**Figure 3.19:** Bayesian network representation of a Gaussian mixture model.

3.19. The distribution over $\mathbf{z}$ is a multinomial distribution (A.2.6) with the GMM mixing coefficients as distribution parameters

$$p(\mathbf{z}) = \prod_{k=1}^{K} \rho_k^{z_k} \tag{3.6.2}$$

where $0 \leq \rho_k \leq 1$ and $\sum_{k=1}^{K} \rho_k = 1$. The conditional distribution of $\mathbf{x}$ given $\mathbf{z}$ is the Gaussian distribution

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{3.6.3}$$

The joint distribution can thus also be written as

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_{\boldsymbol{k}})^{z_k}, \tag{3.6.4}$$

since at any given time only one $z_k$ can have the value 1 in which case the remaining elements are 0. The joint distribution over $\mathbf{z}$ and $\mathbf{x}$ is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ from which we get that the marginal distribution of $\mathbf{x}$ is obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \rho_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{3.6.5}$$

In the GMM the variable $\mathbf{z}$ is a latent variable and $\mathbf{x}$ is an evidence variable. Therefore, for the GMMs we have that $\mathbf{X}_H = \{\mathbf{z}\}$ and $\mathbf{X}_E = \{\mathbf{x}\}$.

### 3.6.2 Maximum likelihood learning for GMMs

Suppose we have a set of observations $\mathbf{X}_E = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$. We obtain a latent variable $\mathbf{z}_n$ for every observation variable $\mathbf{x}_n$ such that $\mathbf{X}_H = (\mathbf{z}_1, \ldots, \mathbf{z}_N)$. This situation is shown in Figure 3.20 where we have used *plate notation* to indicate that we have $N$ pairs of latent and observed variables. Note that we have also explicitly indicated that $\mathbf{x}_n$ is observed by shading the corresponding node in the graph.

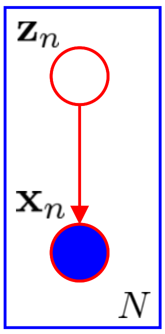

**Figure 3.20:** Gaussian mixture model with $N$ observations represented as a Bayesian network using plate notation.

In order to learn a GMM from data using maximum likelihood we apply the EM algorithm from Section 3.3.1. We have two nodes for which we need to iteratively update expected sufficient statistics and parameters. The latent variable $\mathbf{z}_n$ is a multinomial variable with no parents where we can use the result (3.3.27) to update the parameters. The observed Gaussian variable $\mathbf{x}_n$ has one parent $\mathbf{z}_n$ and the results (3.3.33) and (3.3.35) applies. Figure 3.21 shows a GMM with six components fitted to the *Old Faithful* dataset (Bishop, 2007, Appendix A) using the EM algorithm. The scattered circles represent the observed data. The circles in the center of the ellipses are the means of the GMM components and the ellipses are standard deviation contours scaled for visualisation purposes.

## 3.6.3 Variational learning for GMMs

We next consider variational learning for GMMs. As we are now treating the model parameters as random variables we explicitly represent the variables as hidden nodes in the Bayesian network as shown in Figure 3.22.

From Equation (3.5.4) we get the variational approximation

$$q(\mathbf{Z}, \boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\Lambda}) = q(\mathbf{Z})q(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\Lambda}). \tag{3.6.6}$$

We then iteratively update the variational distributions $q^*(\mathbf{Z})$ and $q^*(\boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\Lambda})$ using (3.5.8) and (3.5.9). In Figure 3.23 - 3.26 we see snapshots from a few iterations of the VB algorithm while fitting a GMM with $K = 6$ to the Old Faithful dataset. The algorithm was initialised using the EM algorithm. We see that the EM algorithm keeps all six components after convergence but using the VB algorithm only two components remains. The remaining components have all converged to their prior distributions indicating that these components take no responsibility for explaining the data. These components correspond to the smallest ellipse in Figure 3.26, which in fact consists of four overlapping distributions. This is an instance of the model complexity selection property discussed in Section 3.5.3.

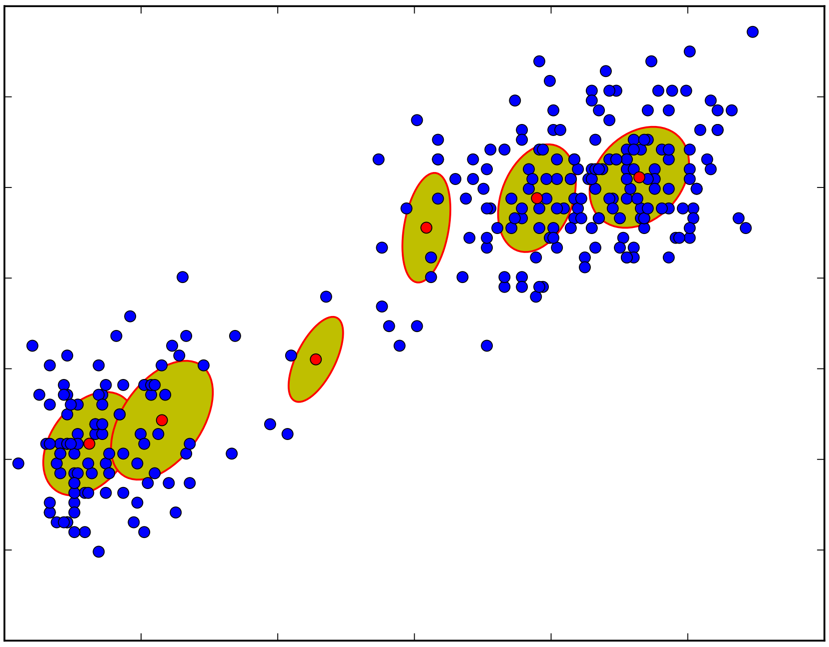

**Figure 3.21:** A GMM with six mixture components fitted to the “Old Faithful” dataset using the EM algorithm.

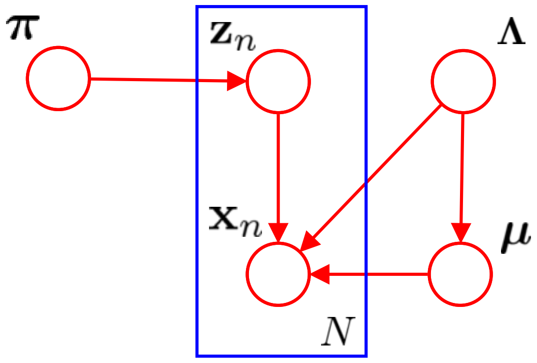

**Figure 3.22:** Bayesian network representation of a GMM where model parameters are represented as random variables in the graph. The plate notation is used to represent $N$ i.i.d. observations.

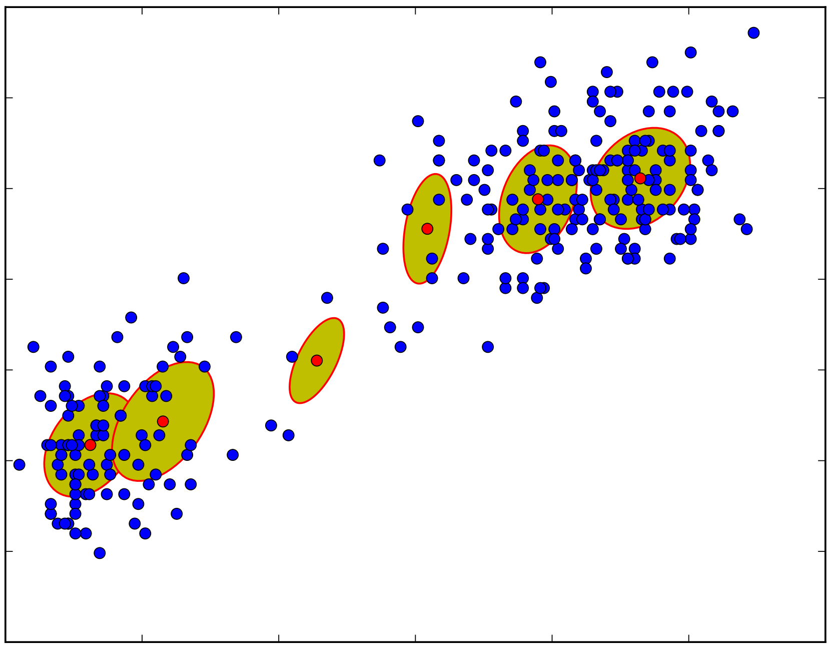

**Figure 3.23:** *Iteration 0*: Example of variational learning in GMMs using the old faithful dataset.

## 3.7 Dynamic Bayesian networks

### 3.7.1 Definition

A dynamic Bayesian network (DBN) is an extension of Bayesian networks that allows for modelling variable-length (and potentially semi-infinite) sequences of hidden and observed random variables and their dependencies. As is the case for static Bayesian networks, a dynamic Bayesian network must constitute a directed acyclic graph (DAG).

A DBN consists of a set of *slices*. When modelling temporal dynamic systems each slice represents a discrete time step. The DBN is defined in terms of *two* Bayesian networks. The first network is the *prior* whose joint distribution is defined as

$$p(\mathbf{x}_V^1) = \prod_{v \in V} p(\mathbf{x}_v^1 | \mathbf{x}_{\pi_v}^1) \tag{3.7.1}$$

where $\mathbf{x}_v^1$ denotes those variables indexed by the node $v \in V$ in the first slice of the Bayesian network. This distribution can be thought of as the initial state of the system.

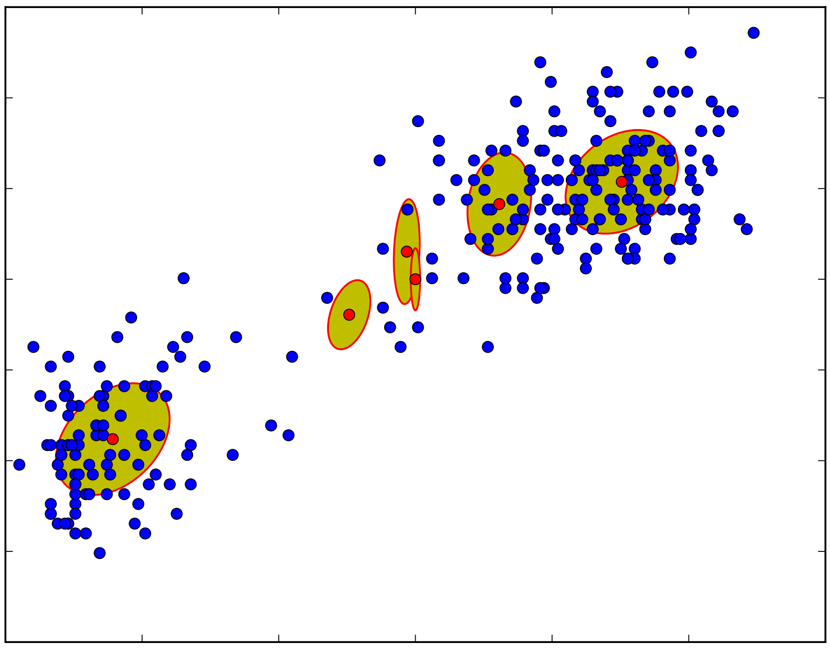

**Figure 3.24:** *Iteration 15*: Example of variational learning in GMMs using the old faithful dataset.

The joint distribution associated with the second Bayesian network is defined by

$$p(\mathbf{x}_V^t|\mathbf{x}_V^{t-1}) = \prod_{v \in V} p(\mathbf{x}_v^t|\mathbf{x}_{\pi_v}^t) \tag{3.7.2}$$

for $t > 1$ where $\mathbf{x}_v^t$ denotes the random variable indexed by node $v$ in time slice $t$ and $\mathbf{x}_{\pi_v}^t$ is the set of parent variables of $\mathbf{x}_v$ in the BN. Equation (3.7.2) can be interpreted as the transition model from one slice to the next. The essential difference between (3.7.1) and (3.7.2) is that in (3.7.2) we allow parents of nodes to lie in the previous time slice in addition to the same slice. Apart from this *inter-slice* dependency the networks are identical. Thus, the network topology is invariant in time. It is the underlying system being modelled that is dynamic. We also assume that the parameters of the model are time-invariant.

The direction of edges between slices are always from left to right, representing the forward flow of time. The arcs within a slice are arbitrary as long as the overall DBN remains a DAG. Intuitively, directed edges within a slice represent instantaneous correlation while edges between slices represent a time-delayed correlation. Figure 3.27 shows an example of DBN with three slices and two nodes in each slice.

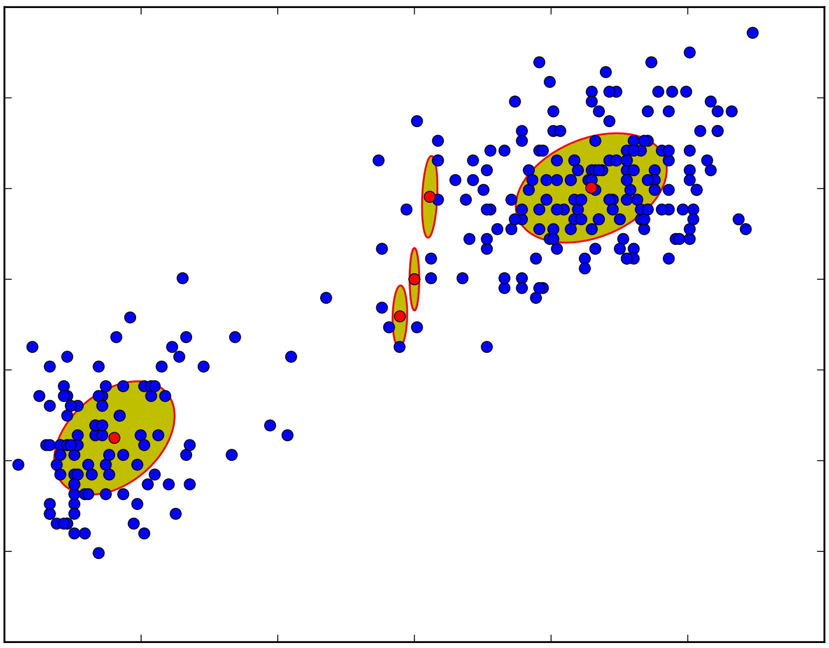

**Figure 3.25:** *Iteration 30*: Example of variational learning in GMMs using the old faithful dataset.

As with static BNs we shall often explicitly represent the dependency of the model on its parameters. Equations (3.7.1) and (3.7.2) then become

$$p(\mathbf{x}_V^1|\boldsymbol{\theta}) = \prod_{v\in V} p(\mathbf{x}_v^1|\mathbf{x}_{\pi_v}^1, \boldsymbol{\pi}_v, \boldsymbol{\theta}_v) \tag{3.7.3}$$

and

$$p(\mathbf{x}_V^t|\mathbf{x}_V^{t-1}, \boldsymbol{\theta}) = \prod_{v\in V} p(\mathbf{x}_v^t|\mathbf{x}_{\pi_v}^t, \boldsymbol{\theta}_v) \tag{3.7.4}$$

where $\boldsymbol{\theta}$ are the transition model parameters and $\boldsymbol{\pi}_v$ are the initial parameters of the nodes in the prior BN that has outgoing arcs to the next time slice in which case $\boldsymbol{\theta}_v = \emptyset$. Nodes in the prior BN that do not have outgoing arcs to the next time slice share parameters with the transition model in which case $\boldsymbol{\pi}_v = \emptyset$. Note in particular that $\boldsymbol{\theta}$ does not depend on $t$ the model parameters are time-invariant.

The semantics of the DBN is obtained by *unrolling* the transition model (3.7.2) for $T$ time-slices. This distribution is given by

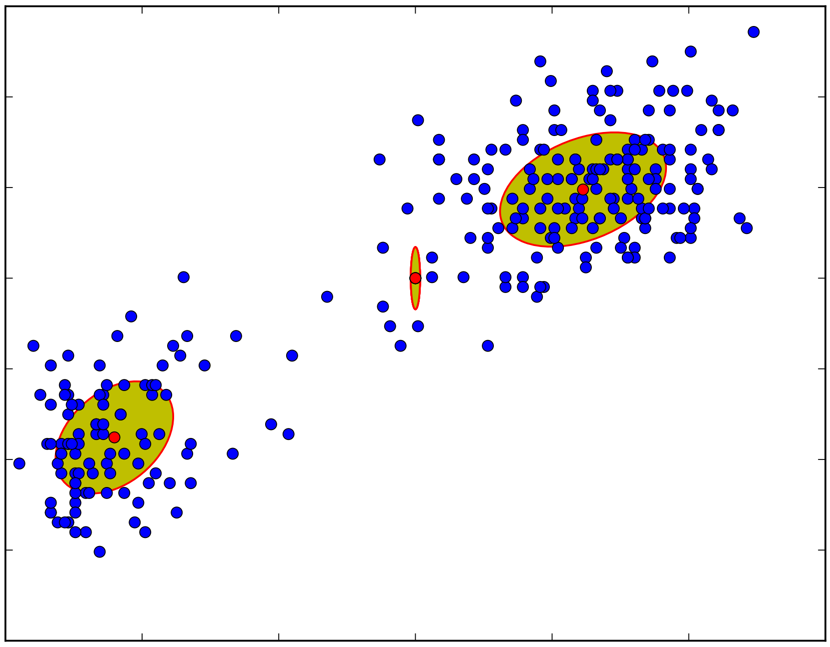

**Figure 3.26:** *Iteration 42*: Example of variational learning in GMMs using the old faithful dataset.

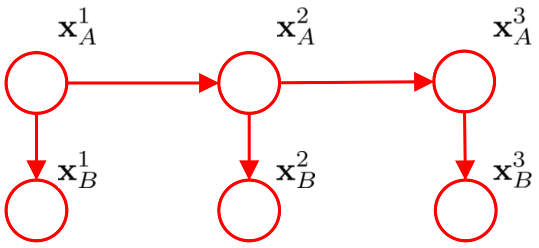

**Figure 3.27:** A DBN consists of multiple slices each containing a BN with additional arcs between slices representing the forward flow of time.

$$p(\mathbf{x}_V) = \prod_{v \in V} \left\{ p(\mathbf{x}_v^1 | \mathbf{x}_{\pi_v}^1, \boldsymbol{\pi}_v, \boldsymbol{\theta}_v) \prod_{t=2}^{T} p(\mathbf{x}_v^t | \mathbf{x}_{\pi_v}^t, \boldsymbol{\theta}_v) \right\} \tag{3.7.5}$$

where again we note that $\boldsymbol{\theta}_v$ does not depend on $t$.

## 3.7.2 Learning and inference in DBNs

Maximum likelihood and variational learning in DBNs are straightforward extensions of the analysis already done for static Bayesian networks. Recall that in DBNs model parameters are tied across time slices. That is, with the exception of the nodes that have outgoing arcs in the initial slice, corresponding nodes share the same parameter set. As a result, when re-estimating the model parameters in DBNs, the expected sufficient statistics for all nodes that share parameters are pooled together as if they were expected sufficient statistics from one node. Thus, the M step remains essentially the same.

It remains to describe how to efficiently obtain the expected sufficient statistics in the case of dynamic Bayesian networks. Recall from Section 3.3 that for learning we need to evaluate the marginals

$$p(\mathbf{x}_v^t, \mathbf{x}_{\pi_v}^t | \mathbf{x}_E, \boldsymbol{\theta}) \tag{3.7.6}$$

which is an inference problem. The most straightforward way to perform exact inference in a DBN is to unroll the DBN for $T$ time slices. The DBN is then no different from a static BN and the junction tree algorithm from Section 3.4 applies without further analysis. However, when learning with sequences that have variable lengths, as is the case in speech recognition applications, it becomes too expensive to repeatedly unroll the DBN and convert it to a junction tree for each observation sequence. Murphy (2002) proposes the *interface junction tree algorithm* as a solution to this problem.

## 3.7.3 The interface junction tree algorithm

The *interface junction tree algorithm* works by "sweeping" a Markov blanket across the length of the DBN, first forward and then backward. Murphy (2002) shows that the set of nodes within a time slice with outgoing edges to the next slice is sufficient to d-separate the past from the future. This set is referred to as the *interface set.*

Let $G = (V, E)$ be the DAG obtained by unrolling the DBN according to (3.7.5) and let $V_t$ denote the set of node indices in time-slice $t$. Let the set of all edges from a time slice $t-1$ to $t$ be denoted by $\vec{E}_t$. This set is defined as all edges $(u, v) \in E$ such that $u \in V_{t-1}$ and $v \in V_t$. The interface set $I_t$ is then defined as all nodes $u \in V_t$ such that $(u, v) \in \vec{E}_{t+1}$ for $v \in V_{t+1}$. That is, the interface set is the set of nodes that have children in the next slice. The set of non-interface nodes is $N_t = V_t \backslash I_t$. We refer to $N_t$ and all earlier nodes as the *past* and $V_{t+1}$ and all later nodes as the *future*.

**Theorem 3.7.1.** *The interface set d-separates the past from the future.*

*Proof.* We need to show that

$$\mathbf{x}_P \perp\!\!\!\perp \mathbf{x}_F \mid \mathbf{x}_{I_t} \tag{3.7.7}$$

where $P = \{V_1, \ldots, V_{t-1}, N_t\}$ is the past and $F = \{V_{t+1}, \ldots, V_T\}$ is the future.

Let $i$ be a node in the interface set that is connected to a node $p$ in the past and a node $f$ in the future. If $p$ is a parent the graph looks like $p \to i \to f$. If $p$ is a child the graph looks like $p \leftarrow i \to f$. In both cases we have that $\mathbf{x}_p \perp\!\!\!\perp \mathbf{x}_f \mid \mathbf{x}_v$ since $v$ is never head-to-head (see discussion of Figure 3.9). The result then follows since all paths between any node in the past and any node in the future are blocked by some node in the interface. □

Consider the subgraph $H^t$ of the unrolled DBN that consists of all nodes in the slice $t$ and $t-1$ minus the non-interface nodes in $t-1$. That is, the set of nodes in $H^t$ are $I_{t-1} \cup V_t$. For the special case $t = 1$ the nodes are $V_1$. We now construct a junction tree from each subgraph $H^t$ as in Section 3.4. Additionally, we require the interface sets $I_{t-1}$ and $I_t$ to each form a clique. This is done by adding edges in the moral graph between all nodes in $I_{t-1}$ and similarly for $I_t$. We finally *glue* all the junction trees together via their interfaces as shown in Figure 3.28.

Figure 3.29 illustrates the construction of interface junction trees in the case of an HMM with Gaussian mixture observation model. Note that we have used $\mathbf{z}$ and $\mathbf{y}$ to denote the latent variables representing HMM states and GMM components, respectively. We can then perform standard junction tree inference in each tree separately and pass messages between the junction trees via the interface nodes. We perform the message passing in a forward-backward procedure along the length of the DBN.

Denoting latent variables in slice $t$ as $\mathbf{x}_{H_t}$ and observed variables as $\mathbf{x}_{E_t}$ the details of the forward-backward passes are then as follows.

**Forward pass** In the forward pass we are given the prior belief state

$$p(\mathbf{x}_{I_{t-1}} | \mathbf{x}_{E_1}, \ldots, \mathbf{x}_{E_{t-1}}) \tag{3.7.8}$$

which is passed from $C_{t-1}$ to $D_t$ where $C_{t-1}$ is the clique in $J_{t-1}$ containing $I_{t-1}$ and $D_t$ is the clique in $J_t$ containing $I_{t-1}$. We then call `CollectEvidence` (Algorithm 1) on $J_t$ with $C_t$ as root node. Finally, we marginalise the distribution over $C_t$ onto $I_t$ to compute $p(\mathbf{x}_{I_t} | \mathbf{x}_{E_1}, \ldots, \mathbf{x}_{E_t})$ and pass this marginal to the next slice. In more detail, the steps are:

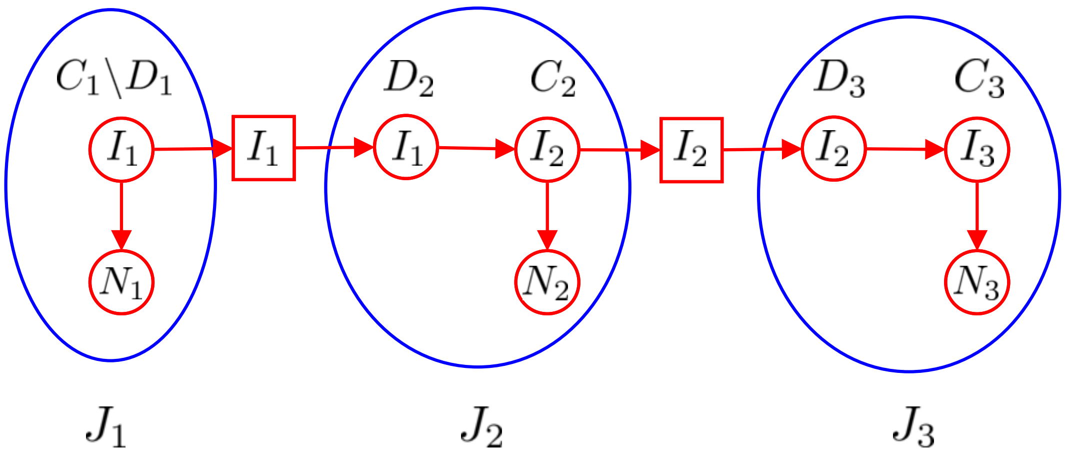

**Figure 3.28:** Illustration of how junction trees are joined in the interface junction tree algorithm. $I_t$ are the interface nodes for slice $t$, $N_t$ are the non-interface nodes. $D_t$ is the clique in $J_t$ containing $I_{t-1}$ and $C_t$ is the clique in $J_t$ containing $I_t$. The square boxes are the separator sets whose domain by definition is $I_t$.

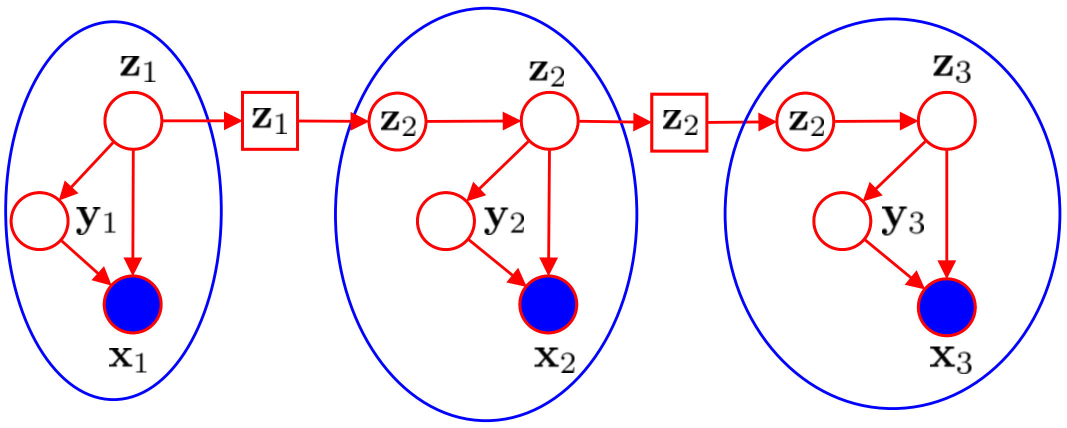

**Figure 3.29:** The DBN for a three-node HMM and mixture of Gaussian observation model with junction trees glued together by the interface nodes $\mathbf{z}_t$. The non-interface nodes are $N_t = \{\mathbf{y}_t, \mathbf{x}_t\}$.

- Construct $J_t$ where all clique and separator potentials are initialised to unity. In the case that all potentials are discrete this means we initialise all elements of the arrays to 1.
- From $J_{t-1}$ extract the potential over $C_{t-1}$ and marginalise onto $I_{t-1}$ to give $p(\mathbf{x}_{I_{t-1}}|\mathbf{x}_{E_1}, \ldots, \mathbf{x}_{E_{t-1}})$. Multiply this marginal onto the potential $D_t$.
- Multiply the CPDs for each node in slice $t$ onto the appropriate potential in $J_t$ taking slices of observed variables $\mathbf{x}_{E_t}$ as necessary.
- Run `CollectEvidence` in $J_t$ using $C_t$ as the root.
- Return all clique and separator potentials in $J_t$.

In the first slice we skip the second step as there is no previous slice to pass a message from. After collecting evidence to $C_t$, not all nodes in $J_t$ will have seen all the evidence $\mathbf{x}_E$. For example, in Figure 3.29 if we collect evidence to $\mathbf{x}_2$, then the posterior distribution over $\mathbf{y}_2$ calculated from the clique potential will be $p(\mathbf{y}_2|\mathbf{x}_2)$ rather then $p(\mathbf{y}_2|\mathbf{x}_1, \ldots, \mathbf{x}_N)$ since $\mathbf{y}_2$ will only have received a message from $\mathbf{x}_2$ below and not $\mathbf{z}_2$ above. Hence, we must also perform `DistributionEvidence` on $C_t$ to compute the correct posterior distribution over all nodes in $J_t$. This operation will be performed in the backward-pass whose details are as follows.

**Backward pass** In the backward pass we run `DistributeEvidence` (Algorithm 2) using $C_t$ as the root and then pass a message from $D_t$ to $C_{t-1}$. The details are as follows.

- The input is the clique and separator potentials over all nodes in $J_t$.
- From $J_t$ extract the potential $D_{t+1}$ and marginalise onto $I_t$ to get $p(\mathbf{x}_{I_t}|\mathbf{x}_E)$.
- Update the potential on $C_t$ in $J_t$ by absorbing from the potential on $D_{t+1}$ in $J_{t+1}$ using
$$\phi_C^* = \phi_C \frac{\sum_{D \setminus C} \phi_D}{\sum_{C \setminus D} \phi_C} \tag{3.7.9}$$
where $C = C_t$ and $D = D_{t+1}$.
- Distribute evidence from the root $C_t$.
- Return all clique and separator potentials.

To start the recursion from the final slice $T$ we distribute evidence from the clique $C_T$. See Murphy (2002) for an analysis of the complexity of the interface junction tree algorithm.

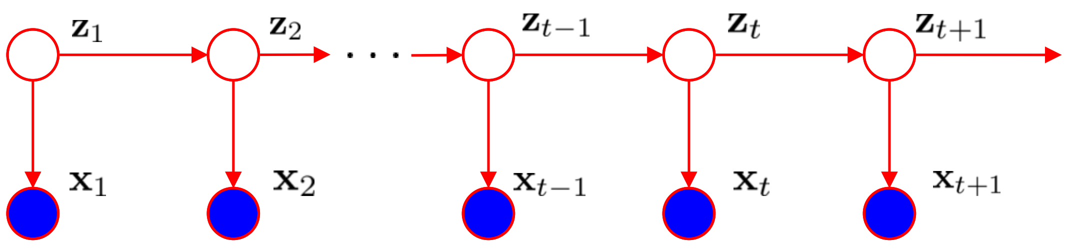

**Figure 3.30:** Hidden Markov Model represented as a dynamic Bayesian network.

## 3.8 Example: Hidden Markov models

A well-known example of a DBN is the hidden Markov Model (HMM). An HMM is a probabilistic model defined by

$$p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = p(\mathbf{x}_1|\boldsymbol{\alpha}) \left[ \prod_{t=2}^{T} p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{A}) \right] \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_t, \boldsymbol{\phi}) \tag{3.8.1}$$

where $\boldsymbol{\alpha}$ is the prior over HMM states, $\mathbf{A}$ is the transition matrix and $\boldsymbol{\phi}$ is the observation model parameters. From (3.7.1), (3.7.2), and (3.8.1) we see that an HMM has the graphical representation shown in Figure 3.30 which we recognise as a DBN.

In the HMM we do not make the i.i.d. assumption for individual observations as in the GMM. In particular, from Figure 3.30 we see that any observation $\mathbf{x}_t$ is conditionally independent of the previous observation $\mathbf{x}_{t-1}$ given state variables $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$. In the case that we have multiple observations sequences we assume that the sequences are i.i.d.

The observation model can in principle have any distribution. In speech recognition applications it is common to model observations as a GMM.

### 3.8.1 Maximum likelihood learning for HMMs

Given observed data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, the likelihood function is obtained from the joint distribution by marginalising over latent variables which gives

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{3.8.2}$$

Recall that the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^*)$ in the M step is given by

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^*) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}^*). \tag{3.8.3}$$

We shall use $\gamma(\mathbf{z}_t)$ to denote the marginal posterior distribution of a latent variable $\mathbf{z}_t$, and $\xi(\mathbf{z}_{t-1}, \mathbf{z}_t)$ to denote the joint distribution of two successive latent variables. We then get

$$\gamma(\mathbf{z}_t) = p(\mathbf{z}_t|\mathbf{X},\boldsymbol{\theta}) \tag{3.8.4}$$
$$\xi(\mathbf{z}_{t-1},\mathbf{z}_t) = p(\mathbf{z}_{t-1},\mathbf{z}_t|\mathbf{X},\boldsymbol{\theta}). \tag{3.8.5}$$

Since $\mathbf{z}_t$ is a discrete variable we can store $\gamma(\mathbf{z}_t)$ using a set of $K$ non-negative numbers that sum to one, and $\xi(\mathbf{z}_{t-1},\mathbf{z}_t)$ as a $K \times K$ matrix of non-negative numbers that again sum to one. We shall use $\gamma(z_{tk})$ to denote the conditional probability of $z_{tk} = 1$, with an analogous notation being used for $\xi(z_{(t-1),j}, z_{tk})$. Note that the expectation of a binary random variable equals the probability that it takes the value 1. This gives us

$$\gamma(z_{tk}) = \mathbb{E}[z_{tk}] = \sum_{\mathbf{z}} \gamma(\mathbf{z}) z_{tk} \tag{3.8.6}$$
$$\xi(z_{(t-1)j}, z_{tk}) = \mathbb{E}[z_{(t-1)j}, z_{tk}] = \sum_{\mathbf{z}} \xi(\mathbf{z}_{(t-1)j}, \mathbf{z}_{tk}). \tag{3.8.7}$$

The quantities $\gamma(z_{tk})$ and $\xi(z_{(t-1)j}, z_{tk})$ are in fact the $r_{ntk}$ quantities from Section 3.3.1 for the $n$-th observation. In the case that we have multiple observation sequences these quantities indeed become $r_{ntk}$ for their respective nodes. In this case the expected sufficient statistics from each sequence is pooled when re-estimating parameters.

Substituting the joint distribution $p(\mathbf{X},\mathbf{Z}|\boldsymbol{\theta})$ given by (3.8.1) into (3.8.3) and using the definitions of $\gamma$ and $\xi$, we obtain

$$\begin{aligned} Q(\boldsymbol{\theta}^*,\boldsymbol{\theta}) &= \sum_{k=1}^{K} \gamma(z_{1k}) \ln \alpha_k + \sum_{t=1}^{T}\sum_{j=1}^{K}\sum_{k=1}^{K} \xi(z_{(t-1)j}, z_{tk}) \ln A_{jk} \\ &\quad + \sum_{t=1}^{T}\sum_{k=1}^{K} \gamma(z_{tk}) \ln p(\mathbf{x}_t|\boldsymbol{\phi}_k). \end{aligned} \tag{3.8.8}$$

Evaluating this expression is the E step of the EM algorithm for HMMs. The marginals $\gamma(\mathbf{z}_t) = p(\mathbf{z}_t|\mathbf{X},\boldsymbol{\theta})$ and $\xi(\mathbf{z}_{t-1},\mathbf{z}_t) = p(\mathbf{z}_{t-1},\mathbf{z}_t|\mathbf{X},\boldsymbol{\theta})$ can be found using the interface junction tree algorithm which, in the special case of HMMs, is equivalent to the classic *forward-backward* algorithm (Rabiner, 1989).

In the M step, we maximise $Q(\boldsymbol{\theta},\boldsymbol{\theta}^*)$ with respect to the parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \mathbf{A}, \boldsymbol{\phi}\}$ while treating $\gamma(\mathbf{z}_t)$ and $\xi(\mathbf{z}_{t-1},\mathbf{z}_t)$ as constant. Using the results from Section 3.3.1 we get

$$\alpha_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{3.8.9}$$
$$A_{jk} = \frac{\sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{tk})}{\sum_{l=1}^{K}\sum_{t=2}^{T} \xi(z_{(t-1)j}, z_{tl})}. \tag{3.8.10}$$

Note that each node in the time-slices $t = 2, \ldots, T$ shares the same parameters, and the expected sufficient statistics are pooled when re-estimating the transition model $\mathbf{A}$ as discussed in Section 3.7.2. This pooling of expected sufficient statistics gives rise to the sum $\sum_{t=2}^{T}$ in the (3.8.10).

Any element of $\mathbf{A}$ that is initially set to zero, remains zero throughout the algorithm. Thus, no particular modification of the EM equations are required for the case of left-to-right models beyond choosing appropriate initial values for the elements of $A_{jk}$.

Assuming that the observation model is a GMM, and noting that only the final term of (3.8.8) depend on $\boldsymbol{\phi}_k$ and that this term has the form of a GMM, we see that to maximise $Q(\boldsymbol{\theta}^*, \boldsymbol{\theta})$ with respect to $\boldsymbol{\phi_k}$ we can use the standard GMM parameter update equations with $\gamma(z_{tk})$ playing the role of responsibilities. This gives us

$$\boldsymbol{\mu}_k = \frac{\sum_{t=1}^{T} \gamma(z_{tk}) \mathbf{x}_t}{\sum_{t=1}^{T} \gamma(z_{tk})} \tag{3.8.11}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{t=1}^{T} \gamma(z_{tk})(\mathbf{x}_t - \boldsymbol{\mu}_k)(\mathbf{x}_t - \boldsymbol{\mu}_k)^{\mathrm{T}}}{\sum_{t=1}^{T} \gamma(z_{tk})} \tag{3.8.12}$$

which should be compared to the results given in Section 3.3.1. Note that in this case the pooling sum $\sum_{t=1}^{T}$ starts at $t = 1$ since the node representing the Gaussian variable $\mathbf{x}_t$ in Figure 3.30 does not have any outgoing arcs and thus the parameters in the initial slice are also shared.

The EM algorithm in general finds a local optimum. It is therefore important to have good initial guesses for the model parameters. In the case of left-to-right HMMs it is common to divide the training data into $K$ equally sized bins, one for each state, and fit a GMM to the data within each bin as if the data is i.i.d. It is common to initialise the GMM means and weights using the $K$-means algorithm and letting the covariance matrices be the identity matrix.

### 3.8.2 Variational learning for HMMs

The variational approximation in the case of the HMM with a GMM observation model is

$$q(\mathbf{Z}, \boldsymbol{\alpha}, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = q(\mathbf{Z}) q(\boldsymbol{\alpha}, \boldsymbol{A}, \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{3.8.13}$$

The variational learning method proceed as usual by updating $q^*(\mathbf{Z})$ and $q^*(\boldsymbol{\alpha}, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ while pooling the expected quantities for nodes that share parameters as in maximum likelihood learning. McGrory and Titterington (2006) shows that variational learning in HMMs have the same model complexity selection properties as the GMM, thus demonstrating that this is a general property of variational learning in mixture models.

## 3.9 Summary

In this chapter we have derived the general framework of Bayesian networks. We presented dynamic Bayesian networks as an extension to the basic framework that allows for modelling variable-length sequences of variables. We presented general learning and inference algorithms for both BNs and DBNs.

We have also showed how variational learning leads to automatic model complexity selection and thus avoids singularities and overfitting problems associated with maximum likelihood learning. The junction tree algorithms for BNs and interface junction tree algorithm for DBNs were derived for the respective models which yield efficient exact inference general in BNs and DBNs.

In the next chapter the purpose of deriving such a general framework shall become apparent as we present several audio-visual DBN models that address the stream weighting and stream integration problems of AVASR. In all the models learning and inference are handled by the general framework without any further analysis necessary.

# Chapter 4

# System Design

## 4.1 Introduction

In this chapter we describe our proposed AVASR system which will be used in the experiments presented in Chapter 5. The system comprises feature extraction, feature stream integration, and classification.

The data corpus we use in the experiments is a small-vocabulary corpus consisting of 36 speakers pronouncing the digits from *zero* to *nine*. Thus, our system is an isolated-digit audio-visual speech recognition system. In this case we can ignore the language model as we are only recognising isolated digits. We can also ignore the pronunciation model as, due to the small vocabulary, we model each digit directly using a separate DBN. Thus, the system will consist of ten audio-visual DBN (AV-DBN) models, one for each digit class $\mathcal{M}_0, \ldots, \mathcal{M}_9$. As the model is conditioned on the class we denote the set of models as $p(\mathbf{x}_\mathcal{V}|\mathcal{M}_0), \ldots, p(\mathbf{x}_\mathcal{V}|\mathcal{M}_9)$.

We first describe how acoustic and visual speech features are extracted from audio and video samples of digit utterances. In particular, we discuss mel-frequency cepstrum coefficients (MFCCs) and active appearance models (AAMs). Next we describe how to integrate the acoustic and visual information contained in the feature streams using audio-visual DBN models. This problem is referred to as the feature stream integration problem. We then describe the observation model used by the system. The observation model is a stream weighted Gaussian mixture model which allows each observation stream (acoustic or visual) to be weighted independently according to some appropriate measure of the reliability of the stream. We then describe how novel observations are classified by the system. Classification is the problem of determining to which class an observation belongs. We discuss implementation issues as appropriate where necessary.

In addition to audio-visual speech recognition, the system may also use only acoustic features or only visual features. We then obtain either an acoustic-only (audio-only) speech recognition system or an automatic lip-reading (visual-only) system.

## 4.2 Audio-visual feature extraction

A fundamental component of any speech recognition system is feature extraction. The raw audio and video data is typically extremely high-dimensional and contains much redundant information. Redundant information is characterised as any information that is not relevant to recognising speech. For instance, a single image of size $720 \times 480$ pixels results in a 345600-dimensional feature vector. If we want to use video sequences to capture dynamic information the situation becomes even worse. In such high-dimensional models the amount of data necessary for training accurate models grows exponentially with the dimensionality of the feature spaces, rendering these models intractable. This is known as the *curse of dimensionality* (see Bishop, 2007, page 33).

Moreover, much of the information contained in audio or video samples is redundant. For instance, in video most of the information relevant to speech is contained in the motion of visible articulators such as the lips, the tongue and jaw. It seems likely that this motion can be expressed using a variable of much lower dimension than for instance the number of pixels in each frame. The process of extracting lower-dimensional and informative features is called *feature extraction.*

We have chosen feature extraction methods that has been shown to yield good results in other studies. In particular, we have chosen MFCCs (Rabiner and Juang, 1993) as acoustic features and AAM coefficients (Matthews *et al.*, 2002) as visual features. In the next section we give an overview of each of these feature extraction methods.

Note that acoustic, and in particular visual, feature extraction is in itself a topic where there is much scope for improvement. However, in this research the main focus has been on modelling, inference and learning. A thorough investigation of feature extraction is beyond the scope of the research and as such we only briefly review each of the feature extraction techniques.

### 4.2.1 Mel-frequency cepstrum coefficients

MFCCs are the standard features used in most modern speech recognition systems. In short, MFCC coefficients are calculated as the cosine transform of the logarithm of the short-term energy spectrum of a signal expressed on the mel-frequency scale. The result is a set of coefficients that approximates the way the human auditory system perceives sound. In Davis and Mermelstein (1980) MFCCs are shown experimentally to give better recognition accuracy than alternative parametric representations for the speech recognition task.

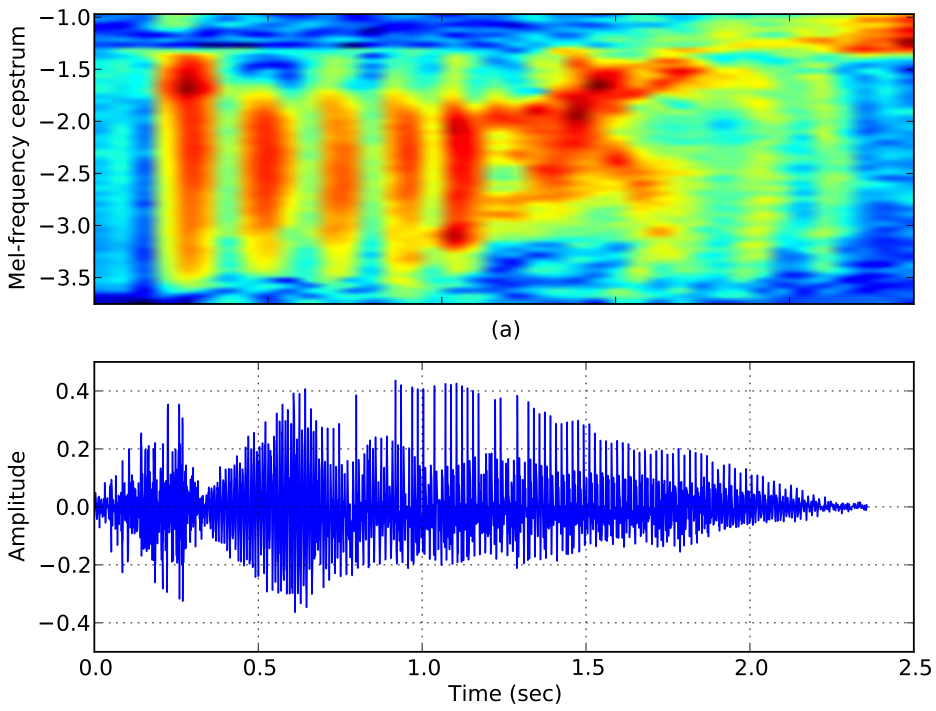

**Figure 4.1:** Acoustic feature extraction from an audio sample of the spoken word “zero”. Mel-cepstrum (top) and original audio sample (bottom).

For the analysis and intuition behind MFCCs and a description of how to calculate the coefficients see Rabiner and Juang (1993).

Figure 4.1 shows an example of an original audio sample of the spoken word “zero” and the corresponding mel-frequency cepstrum. A typical number of MFCCs used in speech recognition systems is 13 coefficients. Note that the first of these coefficients represents the power of the speech signal. We also use the first-order and second-order differences (i.e. velocity and acceleration) of these feature vectors to capture dynamic information. As the feature vectors are typically noisy, calculating these differences directly is numerically unstable. We have used the Savitzky-Golay smoothing filter (Savitzky and Golay, 1964) in which numerically stable first-order and second-order differences are calculated. As a result our feature vectors have a dimension of $13 + 13 + 13 = 39$. Again, the high-dimensional feature vectors are likely to lead to problems due to the curse of dimensionality. Thus, we apply dimensionality reduction through *principal component analysis* (Smith, 2005) to the feature data. The number of principal components to keep was determined by visual inspection of the singular value spectrum. We found that the first 12 singular components contained the majority of the information in the feature data. Thus, the proposed AVASR system uses 12-dimensional acoustic feature vectors which we shall denote as $\bar{\mathbf{x}}_A$, i.e. the observed value of the evidence variable $\mathbf{x}_A$.

## 4.2.2 Active appearance models

While acoustic speech features can be extracted directly through a sequence of transformations applied to the input audio signal, extracting visual speech features is in general more complicated. The visual information relevant to speech is mostly contained in the motion of visible articulators such as lips, tongue and jaw. In order to extract visual information from a video sequence it is advantageous to track the complete motion of the speaker's face and particular facial features.

Active Appearance Model (AAM) fitting is an efficient method for tracking the motion of deformable objects (Edwards *et al.*, 1998; Cootes *et al.*, 2001; Matthews and Baker, 2003). An AAM is a statistical model of variations in shape and texture of an object of interest. Tracking of an object of interest in a video sequence is done by fitting the AAM to each frame of the sequence. To build an AAM it is necessary to provide sample images with the shape of the object annotated. Hence, in contrast to MFCCs, AAMs require prior training before being used for tracking and feature extraction.

The *shape* of an AAM is defined by a mesh given by a set of $(x, y)$ coordinates, represented here in the form of a column vector

$$\mathbf{s} = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n)^{\mathrm{T}}. \tag{4.2.1}$$

Shape variation is restricted to a *base shape* $\mathbf{s}_0$ plus a linear combination of $N$ *shape vectors* $\mathbf{s}_i$

$$\mathbf{s} = \mathbf{s}_0 + \sum_{i=1}^{N} p_i \mathbf{s}_i. \tag{4.2.2}$$

The shape vectors $\mathbf{s}_i$ determine the directions in which the shape is allowed to vary from the base shape. The amount of variation in each direction is given by the shape parameters $p_i$.

The base shape and shape vectors are normally computed by applying PCA to a set of manually annotated training images. Given a set of training shapes $\mathbf{X} = \{\mathbf{x}_i\}$, the base shape $\mathbf{s}_0$ is computed as the mean of the training shapes

$$\mathbf{s}_0 = \bar{\mathbf{x}} \quad \text{where} \quad \bar{\mathbf{x}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i. \tag{4.2.3}$$

The shape vectors are computed in terms of the eigenvectors of the covariance matrix. The covariance matrix is given by

$$\mathbf{\Sigma}_{\mathbf{x}} = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathrm{T}}. \tag{4.2.4}$$

which can also be expressed in terms of its eigenvectors

$$\mathbf{\Sigma_x} = \sum_{k=1}^{D} \lambda_i \mathbf{u}_i \mathbf{u}_i^T. \tag{4.2.5}$$

The shape vectors $\mathbf{s}_i$ are then given by the eigenvectors $\mathbf{u}_i$ sorted according to the corresponding eigenvalues such that $\lambda_i \geq \lambda_{i+1}$. However, not all the eigenvectors are retained. The smallest eigenvalues mostly contain noise and are therefore discarded. The thresholds for which eigenvalues to discard may be chosen manually by visual inspection of the eigenvalue spectrum. In practice, the eigenvectors and eigenvalues are computed using the SVD due to numerical stability issues.

The training shapes typically differ in translation, rotation and scale. This is information that is not relevant to speech and which we would like to remove from the training set before constructing the AAM. This can be achieved by aligning the training shapes prior to performing PCA. A common approach to perform the shape alignment is Procrustes analysis (Cootes *et al.*, 2000). By aligning the shapes we ensure that PCA only capture local, non-rigid deformations, and that translation, rotation, and scale, which do not contain speech relevant information, are ignored. See Cootes *et al.* (1998) for more details on how to perform shape normalisation using Procrustes analysis when training AAMs.

Figure 4.2 shows an example of a shape model with three shape vectors.

#### 4.2.2.1 Appearance

The *appearance* models texture, i.e. the actual pixel values, over the region defined by the base shape $\mathbf{s}_0$. Similar to shape, appearance variation is restricted to a *base appearance* $A_0$ plus a linear combination of $M$ *appearance vectors* $A_i$

$$A(\mathbf{x}) = A_0 + \sum_{i=1}^{M} \lambda_i A_i(\mathbf{x}). \tag{4.2.6}$$

Note that $\mathbf{x}$ is only defined for the pixel coordinates that fall within the base shape $\mathbf{s}_0$. I.e., we shall assume $\mathbf{x} \in \mathit{hull}(\mathbf{s}_0)$.

The mean appearance and appearance vectors are computed from annotated training images. The training images are first shape-normalised by warping each image onto $\mathbf{s}_0$ using a piecewise affine transformation. To this extent, the shape vertices are first triangulated. The collection of corresponding triangles in two shape meshes then defines a piecewise affine transformation between the two shapes. The pixel values within each triangle in the training shape $\mathbf{s}$ are warped onto the corresponding triangle in the base shape $\mathbf{s}_0$ using the affine transformation as defined by the two corresponding triangles.

The appearance model is then computed from the shape-normalised images using PCA. Figure 4.3 shows an example of a base appearance and the first three appearance images.

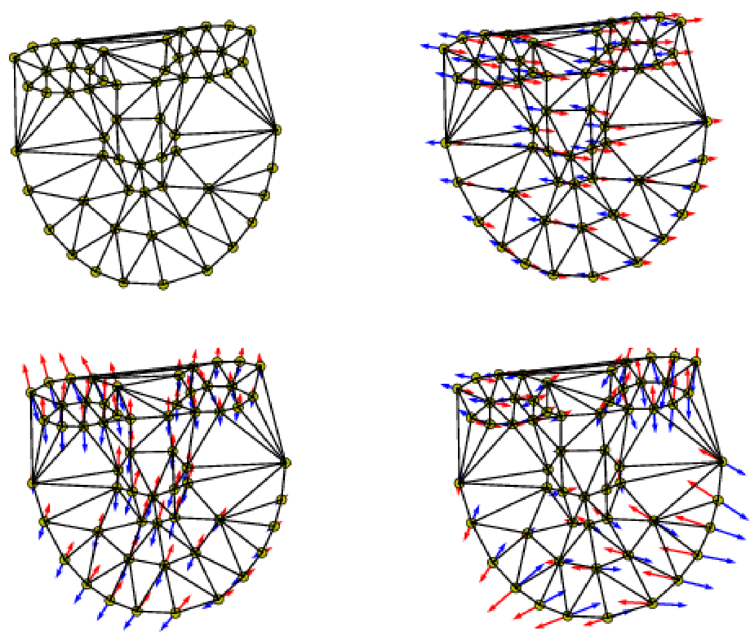

**Figure 4.2:** Triangulated base shape $\mathbf{s}_0$ (top left), and first three shape vectors $\mathbf{p}_1$ (top right), $\mathbf{p}_2$ (bottom left) and $\mathbf{p}_3$ (bottom right) represented by arrows superimposed onto the triangulated base shape.

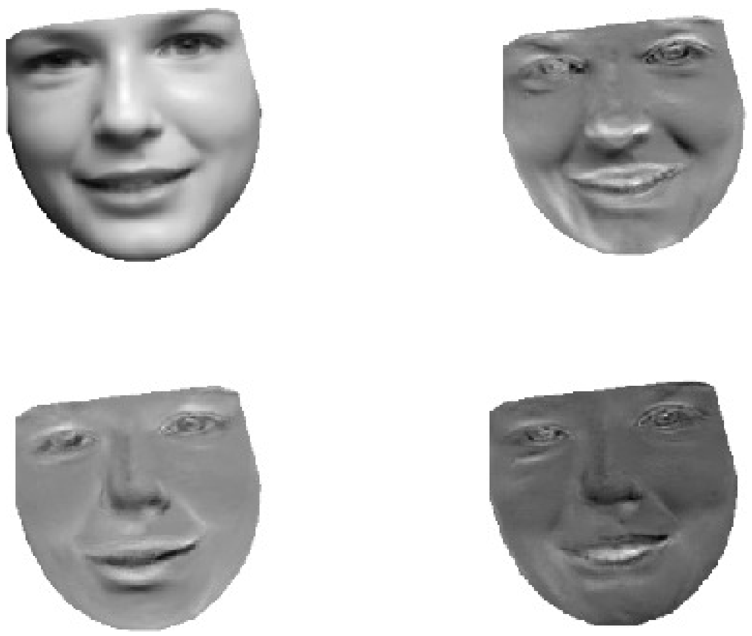

**Figure 4.3:** Mean appearance $A_0$ (top left) and first three appearance images $A_1$ (top right), $A_2$ (bottom left) and $A_3$ (bottom right).

### 4.2.3 Tracking motion using AAM fitting

Tracking of an appearance in a sequence of images is performed by minimising the difference between the base model appearance, and the input image warped onto the coordinate frame of the AAM. For a given image $I$ we determine the optimum parameters.

$$\underset{\boldsymbol{\lambda},\mathbf{p}}{\operatorname{argmin}} \sum_{\mathbf{x}} \left[ A_0(\mathbf{x}) + \sum_{i=1}^{M} \lambda_i A_i(\mathbf{X}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 . \qquad (4.2.7)$$

where $\mathbf{x} \in \mathbf{s}_0$, $\mathbf{p}$ are the shape parameters describing mesh deformation, and$\boldsymbol{\lambda}$ are the appearance parameters describing texture. In the above equation we seek to minimise the distance between the appearance model and an input image $I$ warped onto the frame of the base appearance of the appearance model.

With the purpose of simplifying the presentation, we shall only consider variation in shape and ignore variation in texture. The derivation for the case including texture variation is available in Matthews and Baker (2003). Consequently (4.2.7) reduces to

$$\underset{\mathbf{p}}{\operatorname{argmin}} \sum_{\mathbf{x}} [A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2. \qquad (4.2.8)$$

Solving (4.2.8) for $\mathbf{p}$ is a non-linear optimisation problem.

The quantity that is minimised in (4.2.8) is the same quantity that is minimised in the classic Lucas-Kanade image alignment algorithm (Lucas and Kanade, 1981). In the Lukas-Kanade algorithm the problem is first reformulated as

$$\underset{\Delta\mathbf{p}}{\operatorname{argmin}} \sum_{\mathbf{x}} [A_0(\mathbf{X}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}))]^2. \qquad (4.2.9)$$

This equation differs from (4.2.8) in that we are now optimising with respect to $\Delta\mathbf{p}$ while assuming $\mathbf{p}$ is known. Given an initial estimate of $\mathbf{p}$ we update with the value of $\Delta\mathbf{p}$ that minimises (4.2.9). The new $\mathbf{p}^*$ is given by

$$\mathbf{p}^* = \mathbf{p} + \Delta\mathbf{p}. \qquad (4.2.10)$$

The updated value $\mathbf{p}^*$ will necessarily decrease the value of (4.2.8). Replacing $\mathbf{p}$ with the updated value for $\mathbf{p}^*$, this procedure is iterated until convergence at which point $\mathbf{p}$ yields the (locally) optimal shape parameters for the input image $I$.

To solve (4.2.9) Taylor expansion is used giving

$$\underset{\Delta\mathbf{p}}{\operatorname{argmin}} \sum_{\mathbf{x}} \left[ A_0(\mathbf{W}(\mathbf{x};\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) - \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} \right]^2 \qquad (4.2.11)$$

where $\nabla I$ is the gradient of the input image and $\partial\mathbf{W}/\partial\mathbf{p}$ is the Jacobian of the warp evaluated at $\mathbf{p}$.

The optimal solution to (4.2.11) is found by setting the partial derivative with respect to $\Delta\mathbf{p}$ equal to zero, which gives

$$2\sum_{\mathbf{x}}\left[\nabla\mathbf{I}\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x})) - \nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\Delta\mathbf{p}\right] = 0. \tag{4.2.12}$$

Solving for $\Delta\mathbf{p}$ we get

$$\Delta\mathbf{p} = \mathbf{H}^{-1}\sum_{\mathbf{x}}\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[A_0(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right] \tag{4.2.13}$$

where $\mathbf{H}$ is the Gauss-Newton approximation to the Hessian matrix given by

$$\mathbf{H} = \sum_{\mathbf{x}}\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]^{\mathrm{T}}\left[\nabla I\frac{\partial\mathbf{W}}{\partial\mathbf{p}}\right]. \tag{4.2.14}$$

When tracking motion in a video sequence a proper initialisation of the shape parameters $\mathbf{p}$ is essential for the first frame. For subsequent frames $\mathbf{p}$ may be initialised as the optimal parameters from the previous frame. Figure 4.4 shows an example of an AAM fitted to an input image.

For further details on how to compute the piecewise linear affine warp and the Jacobian as well as an extension of the method that includes appearance variation and a global shape normalising transform see Matthews and Baker (2003).

The AAM fitting method described above is referred to as *forwards-additive* (Baker and Matthews, 2001). In AVASR applications with real-time performance constraints we are often willing to sacrifice accuracy slightly for increased efficiency. In (Matthews and Baker, 2003) several variations of the Lucas-Kanade method is evaluated and it is concluded that the *inverse-compositional* method gives the best trade-off between performance and accuracy. We have used the inverse compositional method for our research and in particular in the experiments presented in Chapter 5. The derivation of the inverse-compositional algorithm is analogous to the forwards-additive one, with some key differences that increase performance. Details can be found in Matthews and Baker (2003).

### 4.2.4 Extracting visual speech features

We would like the visual features to be as independent of the individual speakers in the training data as possible. The standard AAM feature extraction method described so far is insufficient in this regard as the resulting AAM will contain a large amount of information about local variability across different speakers instead of global variations caused by speech. In order to address this

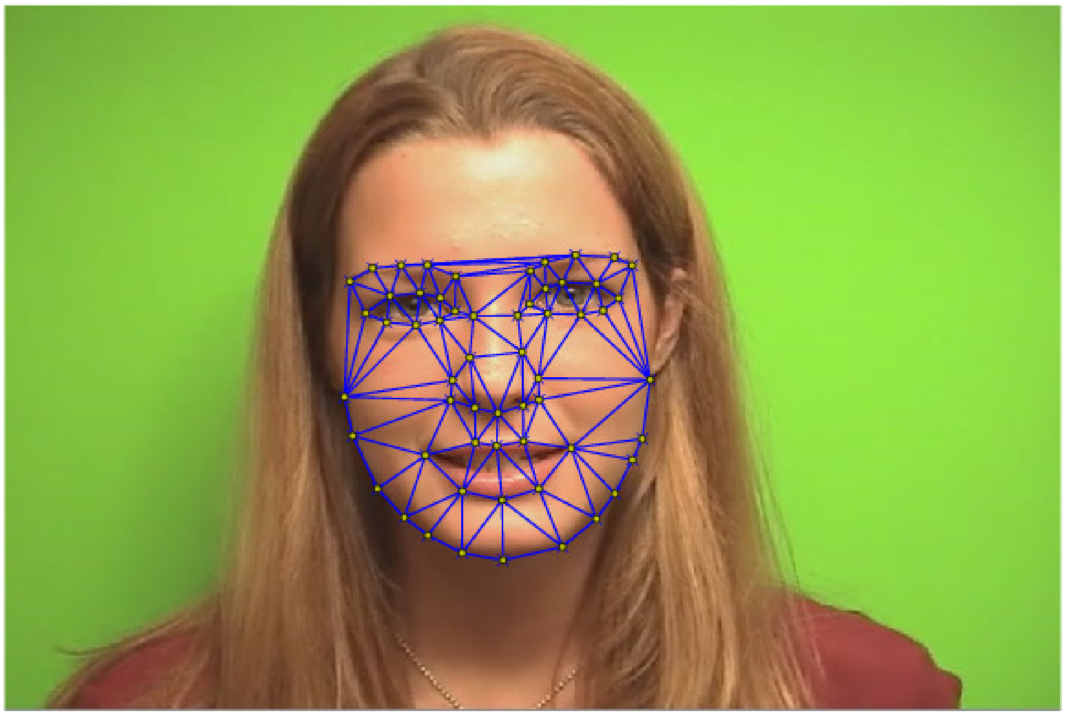

**Figure 4.4:** Example of an AAM fitted to an image. We can think of AAM fitting as moving the vertices of the mesh such that the vertex configuration and image intensity values within each triangle optimally fits the model.

issue we subtract the mean shape and mean appearance of a specific speaker from the feature data from that speaker.

The form of the AAMs discussed so far has been motivated by our need for tracking the motion of a speaker's face and facial features. However, the resulting AAM coefficients may not necessarily be appropriate as features for audio-visual speech recognition. Visual speech information is mostly contained in the motion of visible articulators such as the lips, tongue and jaw. In order to address this issue Papandreou *et al.* (2009) make use of a second *region of interest AAM* (ROI-AAM). The ROI-AAM is confined to the lower face region around the speaker's mouth as shown in Fig 4.5. We observe that the shape vertices of the ROI-AAM form a subset of the vertices in the original AAM. We shall refer to the original AAM as the *facial* AAM. The ROI-AAM is generated from the shapes that result when performing AAM tracking using the facial AAM. The principal components of the ROI-AAM will capture locale information in the region of interest. As a result the principal components contain information that is more relevant to speech than the corresponding principal components of the facial AAM.

By plotting the eigenvalue spectrum we found it sufficient to use the first 10 principal components of the ROI-AAM shape model and the first 16 prin-

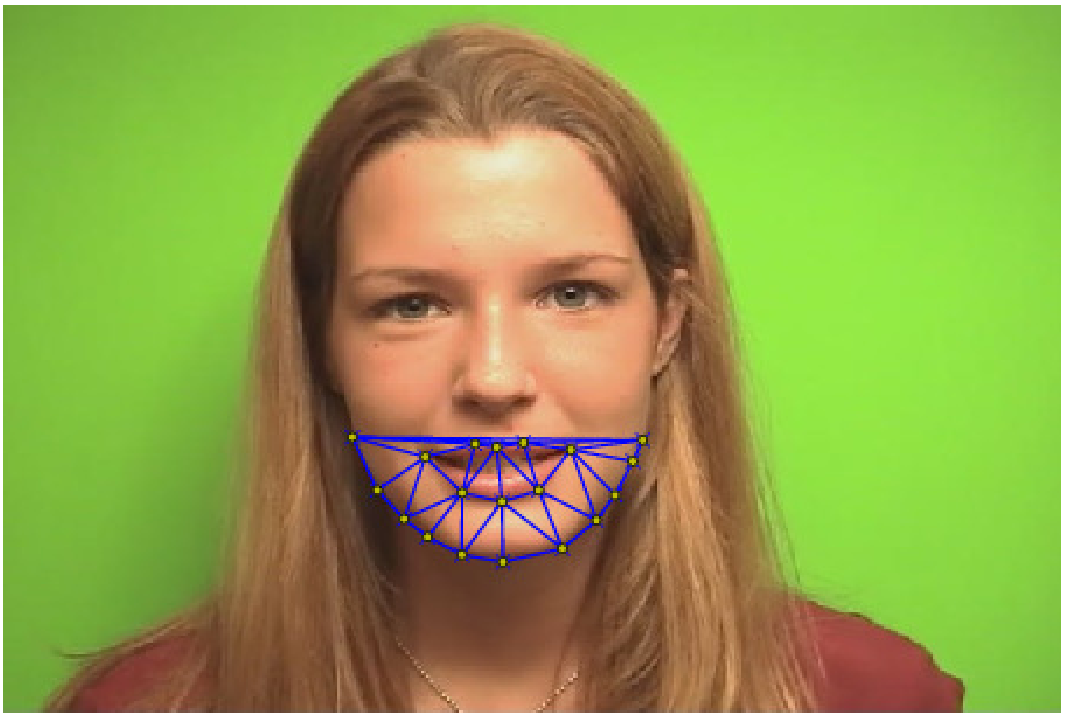

**Figure 4.5:** The ROI-AAM is used to capture the parts of the AAM that contains relevant speech information.

cipal components of the ROI-AAM appearance model as visual feature in the AVASR system. Thus, we obtain a 26-dimensional visual speech feature vector $\bar{\mathbf{x}}_V$ as observed evidence of the visual evidence variable $\mathbf{x}_V$.

## 4.3 Integrating acoustic and visual features

An important problem in AVASR is how to integrate the acoustic and visual observation sequences. Let $\bar{\mathbf{x}}_A$ and $\bar{\mathbf{x}}_V$ denote the acoustic and visual observations associated with the acoustic and visual variables $\mathbf{x}_A$ and $\mathbf{x}_V$, respectively. A simple integration scheme would be to concatenate the two observation vectors into a single vector. Assuming column vectors we get

$$\bar{\mathbf{x}}_E = \begin{pmatrix} \bar{\mathbf{x}}_A \\ \bar{\mathbf{x}}_V \end{pmatrix}. \tag{4.3.1}$$

Using the concatenated feature vectors we can apply standard acoustic-only speech recognition methods such as HMMs. This method is often referred to as *feature fusion* or *early integration*.

Alternatively, we can learn two separate HMMs using the acoustic and visual features, respectively. We will then essentially have two separate speech recognition systems from which an audio-visual speech recognition system can be built by combining the outputs of the two individual systems. This method is called *decision fusion* or *late integration*.

In an HMM the states typically correspond to a notion of "meaning" that represents the cause of the observed data. Feature fusion assumes that the acoustic and visual observation are perfectly *synchronous*. That is, for a given $t$ it assumes that $\mathbf{x}_A^t$ and $\mathbf{x}_V^t$ correspond to the same state at the same time, i.e. the same "meaning". In decision fusion, however, we have two completely independent state spaces for each observation sequence. Thus, decision fusion assumes that the observation $\mathbf{x}_A^t$ is completely independent from $\mathbf{x}_V^t$.

None of these two assumptions are necessarily appropriate for AVASR. Speech production is a complex physical process. In particular, speakers form the facial expression and the shape of the lips prior to a sound being pronounced, resulting in the motion of visible articulators occurring before the actual sound is uttered. Thus, there is a slight delay between the visual and acoustic speech modalities. This delay is referred to as *audio-visual speech asynchrony* and is not constant, but depends on the particular sound that is being uttered as well as the speaker (Benoit, 1992). In part this offset is caused by different channel delays, but is also affected by *forward-articulation* as described in Benoit (1992). In Bregler and Konig (1994) this delay is estimated to be approximately 120 ms on average.

There is an additional problem with the feature fusion method. As the acoustic and visual modalities typically differ substantially in terms of information content and noise, we would like to weight each of the modalities accordingly. This is straightforward in decision fusion as we can weight the outputs of the two independent HMMs as we wish. However, in feature fusion there is no immediate way to perform such a weighting.

Based on the DBN framework we now propose solutions to these problems as extensions to the basic HMM model described in Section 3.7.1. Note that for all models the mathematical formulation is implicit through (3.7.1) and (3.7.2) and that the general inference and learning framework described in Chapter 3 applies with no further refinement necessary. Note that all the AV-DBN models we consider have a left-to-right HMM topology.

**Audio-Visual HMM** The *audio-visual HMM* shown in Figure 4.6 corresponds to early integration where the acoustic and visual features have been concatenated using (4.3.1). It is the equivalent to standard HMM model used in most HMM-based speech recognition systems with acoustic and visual features vectors concatenated into a single $D_A + D_V$ dimensional audio-visual feature vector. The disadvantage of the AV-HMM model is the lack of flexibility in modelling audio-visual asynchrony and weighting the individual feature streams.

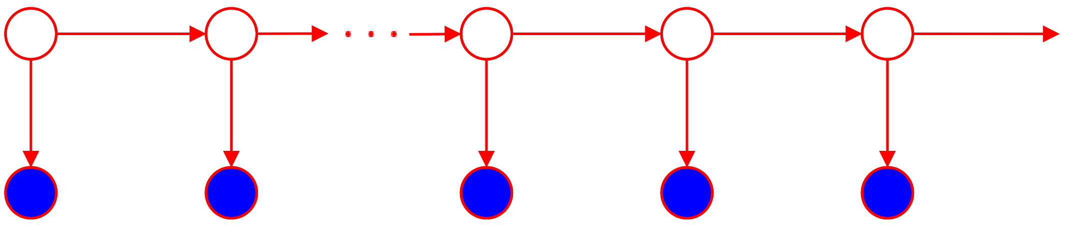

**Figure 4.6:** Audio-visual HMM (AV-HMM)

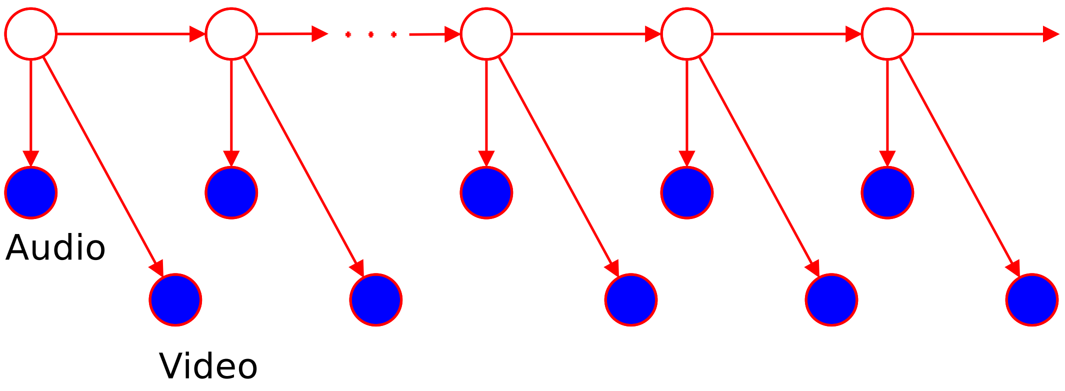

**Figure 4.7:** Audio-visual product HMM (AV-PHMM)

**Audio-Visual Product HMM.** The *audio-visual product HMM* (AV-PHMM) shown in Figure 4.7 is an intermediate integration DBN model. It is a minor modification of the early integration AV-HMM model in which we have two different observation models (corresponding to acoustic and visual observations) represented by separate nodes in the graph. This factored observation model allows us to independently weight each observation stream. In the AV-PHMM the two streams share a single state space and as such the model assumes synchronous acoustic and visual observations. As a result the AV-PHMM might not adequately capture the natural asynchrony between the acoustic and visual observation streams.

**Audio-Visual Independent HMM.** Another possibility is to use two separate HMMs for the acoustic and visual observation sequences independently. Each observations stream will then have a separate state space independent from each other. The resulting DBN is the *audio-visual independent HMM* (AV-IHMM) shown in Figure 4.8. Although this model will allow stream weighting and state asynchrony between the acoustic and visual observation streams, it may fail to capture the natural correlation that exists between the acoustic and visual observation streams.

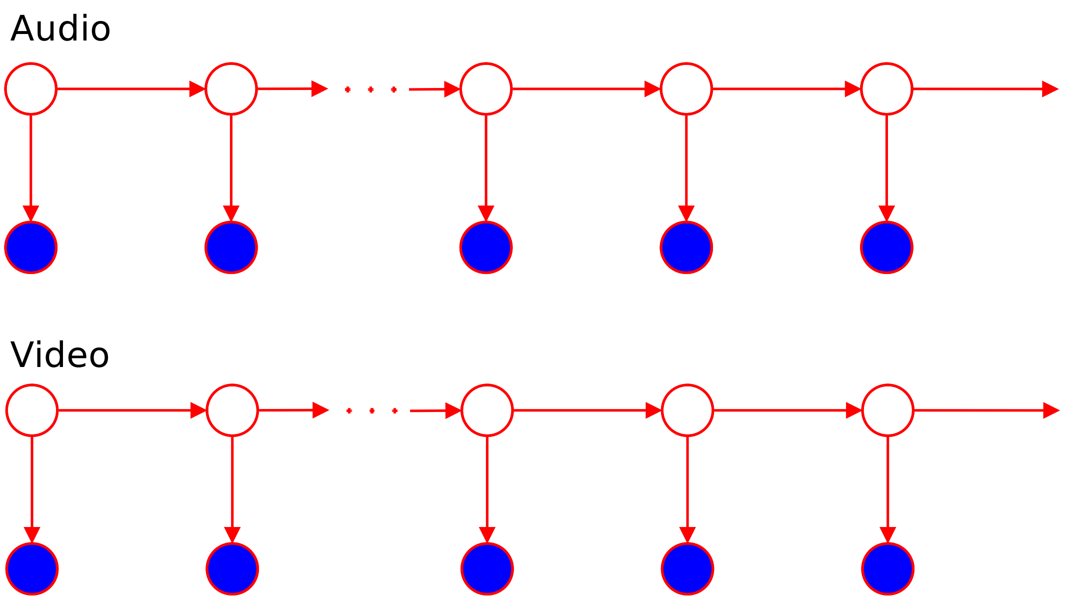

**Figure 4.8:** Audio-visual independent HMM (AV-IHMM)

**Audio-Visual Coupled HMM.** The *audio-visual coupled HMM* (AV-CHMM) shown in Figure 4.9 is a compromise between the AV-PHMM and the AV-IHMM in terms of asynchrony modelling. The AV-CHMM consists of two HMMs similar to the AV-IHMM. However, note that the state spaces of the two models are connected. In particular, each latent state variable in each stream is conditioned on the state variable in the previous time slice of the complementary stream in addition to the state variables in the same stream. Thus, the current state of a stream is dependent on the previous state of both streams. We can constrain the maximum number of states that the two streams are allowed to desynchronise by setting elements of the transition matrices that correspond to "illegal" levels of desynchrony to zero.

Consider the transition model $p(\mathbf{z}_s^t, \mathbf{z}_s^{t-1}, \mathbf{z}_{s'}^{t-1}|\mathbf{A}_s)$ where $s \in \{A, V\}$ indicates the acoustic (A) or visual stream (V) and $s'$ is the complementary stream. Assuming each stream can be in one of $K$ possible states, the transition matrix $\mathbf{A}_s$ is a three-dimensional probability table of size $K^3$ with elements $a^s_{ijk}$ where $i$ is the current state and $j$ is the previous state of the stream $s$, and $k$ is the previous state of the complementary stream. In the proposed AVASR system we use a left-to-right model where the two stream are at most allowed to desynchronise by *one* state. This can be achieved by defining the transition matrix elements as

$$a^s_{ijk} = 0 \qquad \text{if} \begin{cases} i \notin j, j+1 \\ |i-k| \geq 2 \end{cases} . \tag{4.3.2}$$

where the top condition represents the left-to-right structure within a stream, and the bottom the synchronisation between streams.

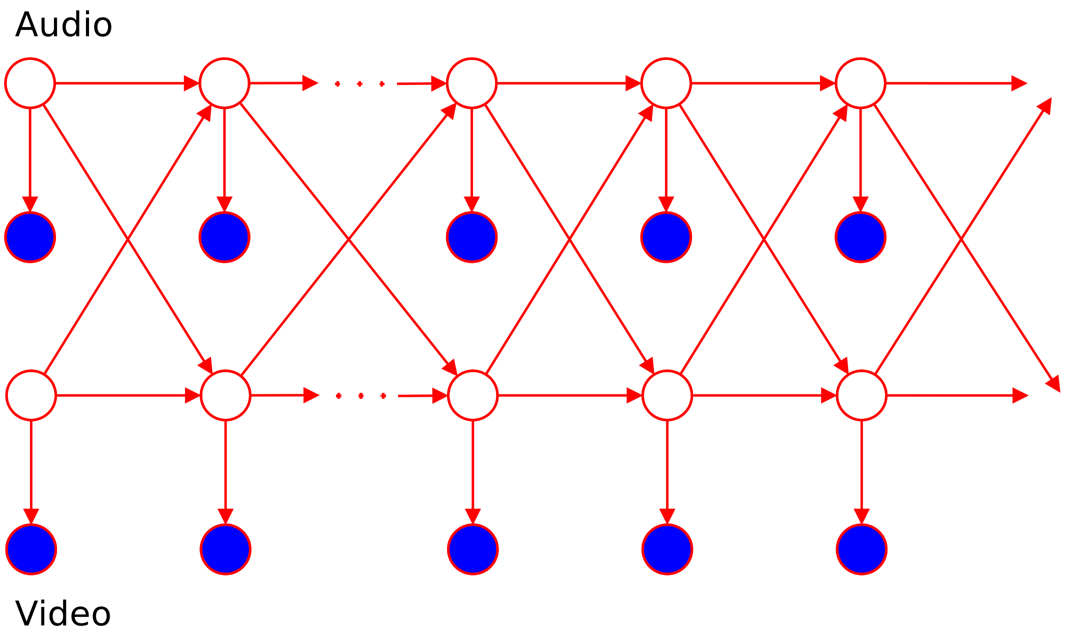

**Figure 4.9:** Audio-visual coupled HMM (AV-CHMM)

We observe that for all the models that we have considered we assume that there is an observation $\mathbf{x}_V^t$ for every $\mathbf{x}_A^t$. However, in most applications the audio sample rate is much higher than the video frame rate. Even after MFCCs have been calculated the number of AAM coefficients are likely to be significantly less than the number of MFCC vectors. We solve this problem by *re-sampling* the visual observation stream to the same number of samples as the acoustic feature stream using the Fourier method (Gasquet and Witomski, 1999).

In the early and intermediate integration models we align the visual and acoustic feature streams by shifting the video stream by 120 ms into the past relative to the audio stream. This is done to compensate for the average delay of acoustic speech relative to visual speech.

## 4.4 Observation model

We use the same state-conditional observation model for all the AV-DBN models. The observation model is the conditional probability distribution associated with the shaded nodes in the graphs. The observation model used in our system is a *stream weighted mixture of Gaussians* (Nefian *et al.*, 2002) defined as

$$b_s(\mathbf{x}_t|z_t^i = 1) = \left[\sum_{j=1}^{M} w_{ij}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{ij}, \boldsymbol{\Sigma}_{ij})\right]^{\lambda_s} \qquad (4.4.1)$$

where $w_{ij}$, $\boldsymbol{\mu}_{ij}$ and $\boldsymbol{\Sigma}_{ij}$ are the weight, mean and covariance matrix of the $j$-th mixture component and $i$-th HMM state, respectively. Note that we omit the dependence of these parameters on the particular stream $s \in \{A, V\}$ in our notation. In the case of the AV-HMM there is only one observation stream and thus there are no stream weights associated with the AV-HMM.

We use *diagonal* covariance matrices as it significantly reduces the number of parameters in the high-dimensional model without degrading accuracy performance. In fact, in most cases where the amount of available training data is limited, diagonal covariance matrices improve accuracy as it reduces the number of parameters thus reducing the effects of the curse of dimensionality.

The stream exponent $\lambda_s$ is used to weight the acoustic and visual observation streams independently according to some measure of reliability and the noise in the respective streams. The stream weights are subject to the conditions $\lambda_s \geq 0$ and $\sum_s \lambda_s = 2$. During the training phase both stream weights and fixed at $\lambda_s = 1$ in which case we have a standard Gaussian mixture model.

## 4.5 Classification

In speech recognition applications we need to classify novel observations (samples) as belonging to a particular model class such as a word or a phoneme. In our application the classes are the digits from zero to nine. In general the likelihood of an observation belonging to a model class is context-dependent. For instance, for words the context is given by the language model and for phonemes by the pronunciation model. In fact, it is possible to model all levels of speech from the top-level language model down to the observation model using a single hierarchical DBN (Terry and Katsaggelos, 2008). However, as we only perform isolated digit recognition there is no language model. Also, due to the small vocabulary we can ignore the pronunciation model and learn separate AV-DBN models for each digit directly.

Suppose we have a set of trained DBN models for each digit $p(\mathbf{x}_\mathcal{V}|\mathcal{M}_i)$ where we have conditioned the model on the particular model class $\mathcal{M}_i \in \{\mathcal{M}_0, \ldots, \mathcal{M}_9\}$ representing the digits from zero to nine. Also suppose we have observed evidence in the form of corresponding audio and video observation sequences $\bar{\mathbf{x}}_E = (\bar{\mathbf{x}}_A, \bar{\mathbf{x}}_V)$. We classify this observation as belonging to the model with largest posterior probability given the observations,

$$\begin{aligned} \underset{\mathcal{M}_0,\ldots,\mathcal{M}_9}{\operatorname{argmax}} \left\{ \max_{\mathbf{x}_H} p(\mathbf{x}_H, \mathcal{M}_i | \bar{\mathbf{x}}_E) \right\} &= \underset{\mathcal{M}_0,\ldots,\mathcal{M}_9}{\operatorname{argmax}} \left\{ \max_{\mathbf{x}_H} \frac{p(\mathbf{x}_H, \bar{\mathbf{x}}_E, \mathcal{M}_i)}{p(\bar{\mathbf{x}}_E)} \right\} \\ &= \underset{\mathcal{M}_0,\ldots,\mathcal{M}_9}{\operatorname{argmax}} \left\{ \frac{p(\mathcal{M}_i)}{p(\bar{\mathbf{x}_E})} \max_{\mathbf{x}_H} p(\mathbf{x}_H, \bar{\mathbf{x}}_E | \mathcal{M}_i) \right\}, \end{aligned} \tag{4.5.1}$$

where $p(\mathcal{M}_i)$ is the prior probability distribution over model class $i$ for $i = 0, \ldots, 9$. Recall that $\mathbf{x}_H$ is the set of latent variables and $\bar{\mathbf{x}_E}$ is the observed variables fixed at their observed values. Typically $p(\mathcal{M}_i)$ is the language model. However, as each digit is equally likely to occur in the data corpus we can drop this factor and perform classification using

$$\underset{\mathcal{M}_0,\ldots,\mathcal{M}_9}{\operatorname{argmax}} \left\{ \max_{\mathbf{x}_H} p(\mathbf{x}_H, \bar{\mathbf{x}}_E | \mathcal{M}_i) \right\}. \tag{4.5.2}$$

Note that we have used the maximum probability interpretation of the marginal when evaluating the marginal of $p(\mathbf{x}_H|\mathbf{x}_E, \mathcal{M}_i)$. The intuition is as follows. We classify the observation to the model class that, given the observed data, has the maximum posterior probability, That is, the model that is *most likely* to have generated the observation sequences. The maximum probability can be evaluated efficiently for any of the proposed AV-DBN models using the interface junction tree algorithm in "max mode" (see Section 3.4.9).

When classifying a novel observation sequence the stream weights $\lambda_s$ are set according to some measure of reliability and noise in the acoustic and visual observations. In particular, in our experiments we use synthetic noise added to the observations and hence the noise level is assumed known. This is not an unrealistic model as several methods exist for measuring the reliability of the acoustic and visual streams (Papandreou *et al.*, 2009). However, investigating such methods is beyond the scope of this research.

## 4.6 Learning

In general, we consider learning as a separate system whose output are the models used by the AVASR system. However, many of the components described in this chapter are also required for the learning system. In particular, feature extraction and feature stream integration remain precisely the same. Features that are extracted for learning form part of the *training data*. The training data is entered as evidence into any of the learning algorithms described in Chapter 3 from which model parameters are estimated.

In Chapter 5 we shall compare the performance of models learned using maximum likelihood and variational learning. Variational learning is in general expected to perform better than maximum likelihood as it reduces the effects of overfitting.

The learning algorithms are determined to have converged when the change during one iteration of the algorithm is less than a predefined threshold. In maximum likelihood learning the log likelihood in Equation (3.3.5) is used to measure change with each iteration, and for variational learning we use the lower bound in Equation (3.5.2). We consider a learning algorithm to have converged when the change is less than 0.2% relative error rate during an iteration.

## 4.7 Summary

In summary, our proposed AVASR system is a small-vocabulary isolated digit audio-visual speech recognition system. The system comprises acoustic and visual feature extraction, early, intermediate, and late feature stream integration, and classification. Figure 4.10 shows a diagrammatic overview of the proposed AVASR system.

Early integration corresponds to the use of the AV-HMM model and intermediate integration to the use of AV-PHMM or AV-CHMM. In late integration the AV-IHMM model is used. Given a raw audio sample and a corresponding video sample the system will extract acoustic and visual features from the respective samples. The acoustic features are MFCCs and visual features are AAM coefficients. In early and intermediate integration the two streams are aligned by shifting the video observation stream 120 ms into the past relative to the audio stream. This is done to compensate for the average delay of acoustic speech relative to visual speech. In late integration this step is not necessary as the two streams are independent. The resulting features are introduced as observed evidence to the system. It will then evaluate the likelihood that the observation was generated by any of the previously trained digit models. The observation is classified as the model (digit) that is most likely to have generated the observation.

An AVASR system as described in this chapter was implemented as a part of the research and used in the experiments presented in the next chapter. See Appendix C for more details on the software.

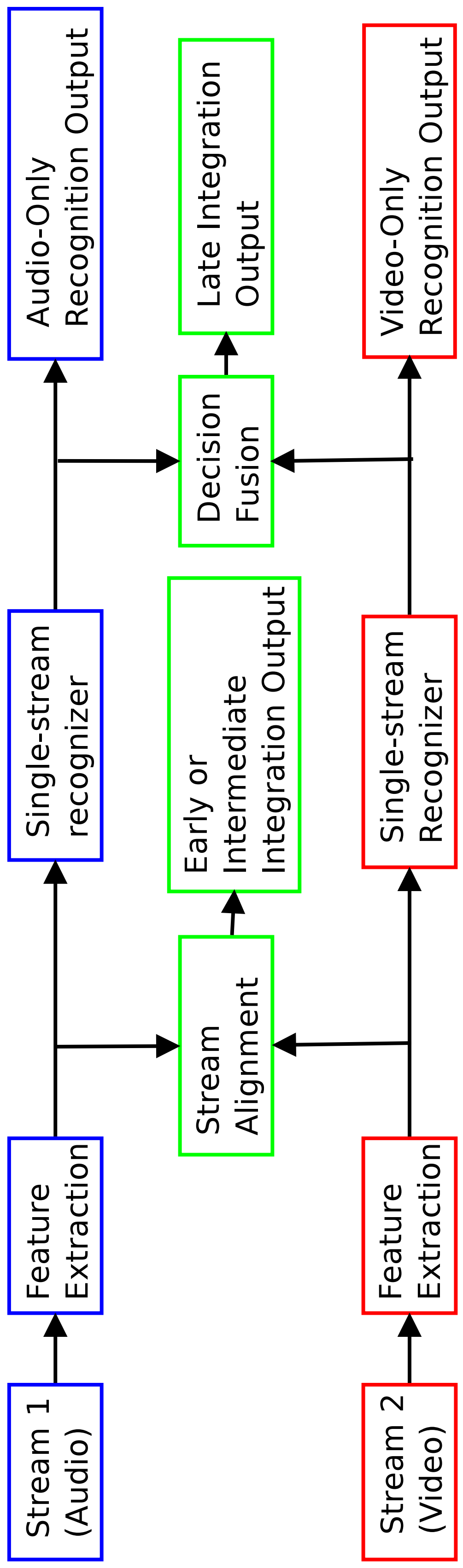

**Figure 4.10:** Schematic overview of the design of an AVASR system

# Chapter 5

# Experiments

## 5.1 Introduction

In this chapter we evaluate the performance of the DBN models and learning algorithms that we have proposed for AVASR. Performance is evaluated based on classification accuracy. In the case of digit recognition this means that we classify multiple examples of each digit and report the performance as the number of digit samples classified incorrectly divided by the total number of digit samples classified. This metric is called the *misclassification rate.*

AVASR is in particular expected to perform better than audio-only speech recognition in noisy acoustic environments as the visual modality is not affected by acoustic noise. Thus, we are particularly interested in comparing the performance of the AVASR models and learning algorithms in various levels of acoustic noise. For each experiment we have added white Gaussian noise (AWGN) ranging in signal-to-noise ratio (SNR) from $-6$ to 18 dB in steps of 4 dB to the acoustic feature stream. We then evaluate the performance of each model or algorithm at each SNR level. The optimal stream weights are discriminatively determined for each SNR level. We expect to weight the visual stream more as the amount of acoustic noise increases (SNR level decreases).

Learning the models consists of two phases. First we estimate model parameters using maximum likelihood or variational learning. Next we discriminatively estimate optimal stream weights for each SNR level. We then use the model parameters and stream weights during classification. Note that by doing so we are assuming that the level of acoustic noise, and hence which pair of stream weights to use, is known during classification. In practice this is not the case. Several methods exist for measuring the reliability on both the audio and visual modality (Papandreou *et al.*, 2009), however, but we shall not consider such methods here.

The following AVASR experiments are considered:

- **Experiment 1.** In the first experiment we investigate the effect of stream weighting in AVASR. In order to evaluate the effects of stream weighting we compare the performance of the AV-HMM model, which does not feature stream weighting, and the AV-PHMM that does.

- **Experiment 2.** In the second experiment we evaluate the performance of the various feature stream integration strategies (early, intermediate, late). In particular, we compare the performance of AV-PHMM, AV-IHMM and AV-CHMM audio-visual DBN models.
- **Experiment 3.** In the third experiment we evaluate the performance of AVASR compared with audio-only and visual-only speech recognition. The comparison is performed by evaluating the misclassification rate of the audio-visual AV-CHMM model to HMM models where we have only used acoustic feature or visual features respectively.
- **Experiment 4.** In the final experiment we evaluate the performance of audio-visual DBN models learned using maximum likelihood estimation (EM) versus variational learning (VB). We compare the performance of the AV-CHMM model learned using each of these algorithms. We refer to the model learned using VB as AV-VBCHMM.

We use a left-to-right transition matrix with no skips for all models, meaning that only the diagonal and super-diagonal elements of the transition matrix are non-zero. This transition matrix means that we can only move from one state to its direct successor, for instance from state three to four. In the case of the AV-CHMM the transition model as defined by (4.3.2) is used. For all models we use eight HMM states and eight mixtures in the GMM observation model. The dimensionality of the acoustic feature vector is 12 and of the visual feature vector is 16. We use diagonal covariance matrices for the Gaussian distributions.

In order to initialise the models we divide each observation sequence in the training set into eight consecutive equal-sized bins and use the data in each bin to initialise the state conditional GMMs for the respective state. The reason for this approach is that in a left-to-right no-skip model we expect the data in bin $k$ to roughly belong to the HMM state $k$. The transition matrices are initialised by setting diagonal elements equal to 0.9 and super-diagonal elements equal to 0.1. The remaining elements in the matrix are 0 as a consequence of the left-to-right no-skip topology. Thus, the probability of remaining in the same state is initially set to 0.9 and the probability of transitioning is initially 0.1.

The means and weights of the GMMs are initialised using the $K$-means algorithm (Bishop, 2007, Chapter, 9). The means of the eight GMM components of the corresponding HMM state $k$ are initialised, using the $K$-means (with $K = 8$) on the data from bin $k$. The weights of each GMM component is initialised as the relative proportion of training data points in bin $k$ that are closer to the corresponding mean value than any other mean values for any other component corresponding to state $k$. The covariance matrices are initialised as the identity matrix.

After the models have been initialised learning is performed using the EM algorithm as described for general Bayesian networks in Section 3.3.1. In Experiment 4 when we do variational learning the models are first initialised using both the above procedure involving $K$-means and EM before running the VB algorithm.

As is pointed out by Alpaydin (2004) we should always keep in mind that whatever conclusion we draw from the analysis is conditioned on the dataset given. Thus, we are not comparing models and learning algorithms in a domain independent manner. Any result we present is only valid for the particular application of AVASR and for the dataset used. As stated in the *No Free Lunch Theorem* (Wolpert and Macready, 1997) there is no such thing as the "best" learning algorithm in general. For any learning algorithm, there will be a dataset where it is very accurate and another were it is very poor. Thus, our results are only valid for the particular application of AVASR and in particular for the digit data corpus we have chosen. This data corpus is discussed next.

## 5.2 Data

Unfortunately, there is no large publicly available annotated data corpus such as TIMIT (Garofolo *et al.*, 1993) available for audio-visual speech recognition. The scarcity of large audio-visual speech corpora for use in AVASR is a known problem in AVASR research (Lucey *et al.*, 2004). The only data corpus in existence suitable for large-vocabulary continuous audio-visual speech recognition (LV-AVCSR) is the IBM ViaVoice$^{\text{TM}}$ audio-visual database (Neti *et al.*, 2000). Unfortunately, this database is not publicly available. We did contact IBM during the course of the research regarding acquisition of the dataset. The request was unfortunately denied. The amount of resources necessary for collecting and transcribing such a dataset is substantial. Therefore, we have based our experiments on one of the small-vocabulary data corpora that are publicly available.

We have chosen the Clemson University audio-visual experiments data corpus (CUAVE) (E.K. Patterson *et al.*, 2002). CUAVE is a small-vocabulary multi-speaker data corpus consisting of 36 speakers, 19 male and 17 female, uttering isolated and connected digits (0-9) under various conditions. Video of the speakers is recorded in frontal, profile and while moving. In our experiments we only use the portion of the corpus where the speakers are stationary and facing the camera while uttering isolated digits. This part is referred to as the "Normal" section in the CUAVE documentation. A sample frame from each individual in the CUAVE data corpus is shown in Figure 5.1.

The audio sample rate in CUAVE is 16000 samples per second. From these samples we calculate 13 MFCC coefficients using overlapping windows

**Figure 5.1:** Sample frames from the CUAVE database

of 256 samples. Delta and delta-delta coefficients are calculated and PCA is performed on the resulting high-dimensional feature vector as described in Section 4.2.1. In order to compensate for the average delay of the acoustic feature stream, the stream is shifted 120 ms into the past relative to the video stream as explained in Section 4.3.

The video frame rate in CUAVE is 30 frames per second. From the video samples we train a separate AAM for each speaker in the data corpus. This AAM is only used for extracting visual features from that particular speaker. The AAMs are trained through a procedure whereby the shape of the object of interest is manually annotated in the training data. The annotation itself is performed using a bootstrapping procedure where initially only a few frames are annotated. The AAM is trained from these initial annotated frames and subsequently fitted to the remaining set of frames. As the initial AAM is trained using only a few frames, the fit to the remaining frames are poor in general. Thus, we manually correct the fit for a subset of the remaining frames (preferably frames where the fit is worst) and, using the augmented set of manually annotated frames, we re-train the AAMs. This procedure is repeated until the fit to new frames appeared to have converged in the sense that no correction is necessary. We found it sufficient to annotate approximately every 50th frame of each video. Finally, the AAM is fitted to all the frames in the video sequence. Visual speech features are extracted using the techniques of ROI-AAMs and speaker-mean subtraction described in Section 4.2.4.

## 5.3 Experimental protocol

Training the audio-visual speech recognition system consists of two phases; learning audio-visual DBNs for each digit and estimating optimal stream weights for each SNR level. We shall refer to the first phase as the training phase and the second as the validation phase. In addition to these two phases we have the actual classification phase where the system is tested on data that ideally has not been used during training and validation. We refer to this phase as the test phase. Thus, each experiment will consist of three phases; training, validation and testing.

Ideally, we would use separate datasets for all three phases. However, due to the limited availability of data we will have to reuse portions of our dataset across phases. As the dataset consists of five utterances of each digit by each speaker, we divide the data into five subsets $\mathcal{D}_1, \ldots, \mathcal{D}_5$ where each subset has one sample from each speaker of each digit. As we have 36 speakers the result is five subsets each containing 36 audio and video samples of each word. In the experiments we choose one of the subsets for testing. The remaining four subsets are used for training and validation. From the four subsets we use three for training the DBN models and the remaining subset for validation. The performance of the model or algorithm subject of the experiment is measured as the misclassification rate on the test dataset.

In order to increase the accuracy of the experimental results, we iteratively train, validate and test on different subsets. That is, we run five different experiments where we sequentially use $\mathcal{D}_1$ then $\mathcal{D}_2$ until $\mathcal{D}_5$ as the test dataset and in each case the remaining subsets for training and validation. Each iteration will give a misclassification rate. We report the average of the five misclassification rates as the performance of the model or algorithm subject to the experiment. In this way we will eventually have utilised all the data for testing. Thus, in total we have 36 speakers $\times$ 10 digits $\times$ 5 samples $= 1800$ classifications to evaluate the performance of the model or algorithm subject. This technique is known as (five-fold) cross-validation.

The optimal stream exponents are determined discriminatively from the misclassification rates when classifying samples in the validation set using a brute-force grid search. For each SNR level we calculate the misclassification rate while varying $\lambda_A$ from 0.0 to 2.0 in steps of 0.2 resulting in 11 different possible values for $\lambda_A$. Since $\lambda_V = 2 - \lambda_A$ we immediately get the corresponding weights for the visual feature stream for each $\lambda_A$. The optimal stream weight pair for a given SNR level is chosen as the weight parameters $\lambda_A$ and $\lambda_V$ that results in the smallest misclassification rate for that SNR level. Finally, we use the estimated model parameters and optimal stream weights to evaluate the performance of the model or algorithm subject to experiment as the misclassification rate on the as of yet unseen test dataset.

Note that the speakers in the training and validation sets are the same as in the test set. Thus, the experiments will not tell us how well the models and algorithms generalise to speakers not in the training and validation sets. As a consequence, the results reported here are applicable to speech recognition systems with *closed-speaker sets*. We have chosen the closed-speaker set approach as there was not sufficient data available to train a speaker-independent system.

In all experiments learning is performed using the EM algorithm with notable exception of Experiment 4 where the VB algorithm is used for one of the models with the purpose of comparing the performance of maximum likelihood and variational learning on audio-visual DBN models.

In each experiment AWGN ranging from -6dB to 18dB in steps of 4 dB SNR is added to the acoustic feature stream in the validation data when estimating stream weights and the test data when evaluating system performance. Note that noise is never added to the training data. During the training phase the stream weights are set to $\lambda_A = 1$ and $\lambda_V = 1$ in which case the observation models become standard GMMs in which the parameter update results from Section 3.3 and 3.5 applies.

## 5.4 McNemar's test

We wish to make sure that differences in the experimental results that we report are not caused by random experimental noise, but are indeed true findings by showing statistical significance (Alpaydin, 2004). In a statistical significance test we show that the likelihood of the result having occurred by chance is for instance less than 5% likely.

McNemar's test is a non-parametric method used to determine if there is enough evidence to claim that two experimental subjects, for instance two different models or algorithms, differ by a certain significance level, that is, that any observed differences are less than for instance 5% likely to have occurred by chance. McNemar's test is applied to $2 \times 2$ contingency tables. In the case of classification the first subject $C_1$ is represented by the rows and the second subject $C_2$ by the columns. Table 5.1 shows an example of a $2 \times 2$ contingency table where we have defined

$$\begin{aligned}
N_{00} &= \text{Number of utterances which } C_1 \text{ classifies correctly} \\
&\quad\ \text{and } C_2 \text{ classifies correctly} \\
N_{01} &= \text{Number of utterances which } C_1 \text{ classifies} \\
&\quad\ \text{correctly and } C_2 \text{ classifies incorrectly} \\
N_{10} &= \text{Number of utterances which } C_1 \text{ classifies incorrectly} \\
&\quad\ \text{and } C_2 \text{ classifies correctly} \\
N_{11} &= \text{Number of utterances which } C_1 \text{ classifies incorrectly} \\
&\quad\ \text{and } C_2 \text{ classifies incorrectly.}
\end{aligned}$$

| | | $C_2$ | |
|---|---|---|---|
| | | Correct | Incorrect |
| $C_1$ | Correct | $N_{00}$ | $N_{01}$ |
| | Incorrect | $N_{10}$ | $N_{11}$ |

**Table 5.1:** Contingency table of errors made by $C_1$ and $C_2$.

Note that $N = N_{00} + N_{01} + N_{10} + N_{11}$ is the total number of test samples being classified. For each $N_{ij}$ we also define a probability $q_{ij}$, such that for a given utterance $u_i$ we have for instance

$$q_{01} = p(C_1 \text{ classifies } u_i \text{ correctly, } C_2 \text{ classifies } u_i \text{ incorrectly}) \tag{5.4.1}$$

and $q_{00}$, $q_{10}$ and $q_{11}$ are defined similarly. Note that we have $\mathbb{E}[N_{ij}] = Nq_{ij}$.

The null hypothesis $\mathbf{H}_0$ in McNemar's test is that

$$q_{00} + q_{01} = q_{00} + q_{10} \tag{5.4.2}$$

and

$$q_{10} + q_{11} = q_{01} + q_{11}. \tag{5.4.3}$$

That is, the marginal probabilities for each outcome are the same. From (5.4.2) and (5.4.3) we get the null hypothesis

$$\mathbf{H}_0 : q_{01} = q_{10}. \tag{5.4.4}$$

Defining

$$q = \frac{q_{10}}{q_{01} + q_{10}} \tag{5.4.5}$$

we get the equivalent null hypothesis

$$\mathbf{H}_0 : q = \frac{1}{2} \tag{5.4.6}$$

which follows from

$$2q = \frac{q_{10}}{q_{01} + q_1} + \frac{q_{01}}{q_{01} + q_{10}} = 1 \tag{5.4.7}$$

since the null hypothesis is that $q_{01} = q_{10}$.

The parameter $q$ represents the conditional probability that $C_1$ will make an error on an utterance given that only one of the two algorithms makes an error. The null hypothesis $q = 1/2$ represents the assertion that, given that only one of the algorithms makes an error, it is equally likely to be either one.

In order to test $\mathbf{H}_0$ it is only necessary to examine the utterances on which only one of the algorithms made an error. No information about the relative performance of $C_1$ and $C_2$ is available from utterances where $C_1$ and $C_2$ were both right or both wrong. Intuitively, this makes sense, if we want to say

| | | $C_2$ | |
|---|---|---|---|
| | | Correct | Incorrect |
| $C_1$ | Correct | 1325 | 3 |
| | Incorrect | 13 | 59 |

**Table 5.2:** Example of contingency table of errors made by $C_1$ and $C_2$

whether two models or algorithms are different we should look at where they disagree, not where they agree.

If we condition on the number of utterances $K = N_{10} + N_{01}$ on which only one algorithm made an error, then for the observed $K = k$ $N_{10}$ has a binomial distribution $\mathrm{Bin}(k, q)$ (A.2.4). Furthermore, under $\mathbf{H}_0$ we have that $N_{10}$ has a binomial distribution $\mathrm{Bin}(k, 1/2)$. The null hypothesis is tested by applying a two tailed test to the observation of a random variable $M$ drawn from a $\mathrm{Bin}(k, 1/2)$ distribution

$$p = \begin{cases} 2p(n_{10} \leq M \leq k) & \text{when} \quad n_{10} > k/2 \\ 2p(0 \leq M \leq n_{10}) & \text{when} \quad n_{10} < k/2 \\ 1.0 & \text{when} \quad n_{10} = k/2. \end{cases} \tag{5.4.8}$$

$\mathbf{H}_0$ is rejected when $p$ is less than some significance level $\alpha$. If $k$ is large enough (k $>$ 50) and $n_{10}$ is not too close to $k$ or 0 a normal approximation to the exact binomial probability (A.2.4) may be used. Under $\mathbf{H}_0$ and conditional on $K = k$ we have that $\mathbb{E}[n_{10}] = k/2$ and $\mathrm{var}[N_{10}] = k/4$. Then, the random variable

$$k = \frac{|N_{10} - k/2| - 1/2}{\sqrt{k/4}} \tag{5.4.9}$$

is approximately $\mathcal{N}(0, 1)$ under $\mathbf{H}_0$. We can then compute the $p$-value

$$p = 2p(Z \geq w) \tag{5.4.10}$$

where $Z$ is a random variable with distribution $\mathcal{N}(0, 1)$ and $w$ is the realised value of $W$. We reject $\mathbf{H}_0$ if $p < \alpha$ where $\alpha$ is the chosen significance level. We shall set this level to 5%. The $-1/2$ in the numerator of (5.4.9) is the Yates' correction factor for continuity (Yates, 1934).

As an example consider the contingency table shown in Table 5.2. Here $k = 16$, $N_{10} = 13$ and $n_{01} = 3$. Calculating the $p$-value using the normal approximation (5.4.9) we get that $p = 0.0244$. This means that the observed difference would arise by chance on about 2% of occasions and hence we can conclude that in this case there is evidence of a genuine difference given a 5% significance level.

## 5.5 Results

We here present the experimental results from the four experiments discussed in Section 5.1. We only show graphical plots of misclassification rates for different models and algorithms at different SNR levels. The actual numerical values are available in Appendix B. In Appendix B we also report the optimal stream weights used in each experiment. The stream weights may vary slightly when using different validation sets. Thus, we have reported the average of the stream weights found from the five different validations sets used for each experiment. We perform McNemar's tests for all of the results which are also presented in Appendix B. We shall refer to the appendix where appropriate.

**Experiment 1.** In Figure 5.2 we show the performance results of the AV-HMM and AV-PHMM audio-visual DBN models. Recall that the AV-PHMM allows for weighting individual streams while the AV-HMM does not. We observe that for small SNR levels from -2 dB to 2 dB the AV-PHMM performs significantly better than AV-HMM. This is because the AV-PHMM allows for weighting the visual stream more in this region, thereby diminishing the effect of the acoustic noise. This is not possible with the AV-HMM and results in the acoustic noise having a particularly negative effect on classification performance. In the region from 6 dB to 10 dB the two models appear to have similar performance. For the SNR levels 14 dB and 18dB the advantage of the weighting scheme is apparent again as the AV-PHMM model appears to be performing better the AV-HMM model.

The significance tests for Experiment 1 are shown in Table B.2. We see that for SNR levels -6 dB, -2 dB, 2 dB, and 18 dB there is sufficient evidence to reject the null hypothesis that the performance of the two models is the same and accept the alternative hypothesis that there is indeed a genuine difference at a 5% significance level. For SNR levels 6 dB, 10 dB and 14 dB there is not sufficient evidence to reject the null hypothesis that the two models perform the same.

**Experiment 2.** Figure 5.3 shows the misclassification rate at different SNR levels for the AV-PHMM, AV-IHMM and AV-CHMM models. We observe that, in severe acoustic noise, the AV-IHMM and AV-PHMM models perform better than the AV-CHMM model. The reason for this behaviour is likely that, due to the dependency between the two streams in the AV-CHMM model, when one stream is contaminated by noise the other stream is also negatively affected. In this case it would appear that the presence of stream exponents does not sufficiently compensate for the noise in the acoustic feature stream.

As the amount of acoustic noise decreases, however, the AV-CHMM appears to be performing better than the AV-PHMM and AV-IHMM. This is likely due to the asynchrony model of the AV-CHMM model which, as opposed to the AV-PHMM, allows for audio-visual asynchrony and, as opposed to the

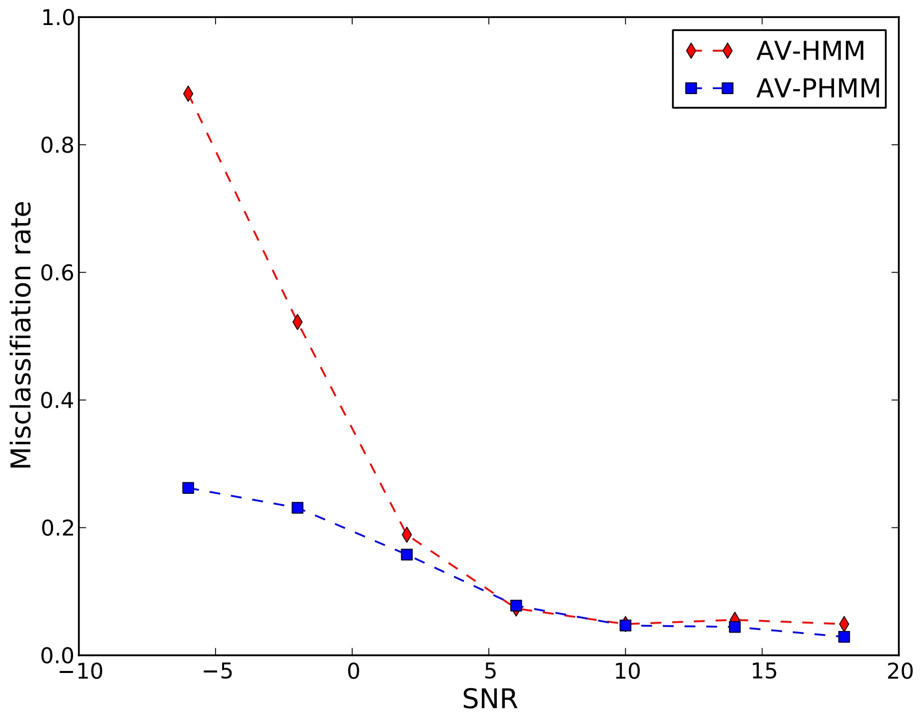

**Figure 5.2:** Performance comparison between AV-HMM and AV-PHMM. The use of stream weights in the AV-PHMM yields superior results to the AV-HMM where both streams are considered equally reliable.

AV-IHMM, restricts the level of asynchrony to one state thereby maintaining the natural correlation between the acoustic and visual speech modalities. The AV-PHMM model appears to be most robust to noise while having the worst performance at low noise levels.

Table B.4 shows the McNemar's test results for AV-CHMM versus AV-PHMM. We observe that there is sufficient evidence of a genuine difference everywhere except for the 2 dB SNR level. The McNemar's test for AV-CHMM versus AV-IHMM in Table B.5 shows the same results.

**Experiment 3.** Figure 5.4 shows the performance results of the audio-only, visual-only, and audio-visual classifiers. We have chosen the AV-CHMM as the audio-visual classifier as it has the best performance in the moderate to high SNR levels which is typically of most interest. The audio-only and visual-only classifiers use standard HMMs.

We see that the performance of the visual-only classifier is fixed at 25.6% misclassification rate for all noise levels. This is because the visual channel is not affected by acoustic noise. At the lowest SNR level (-6 dB) we observe that

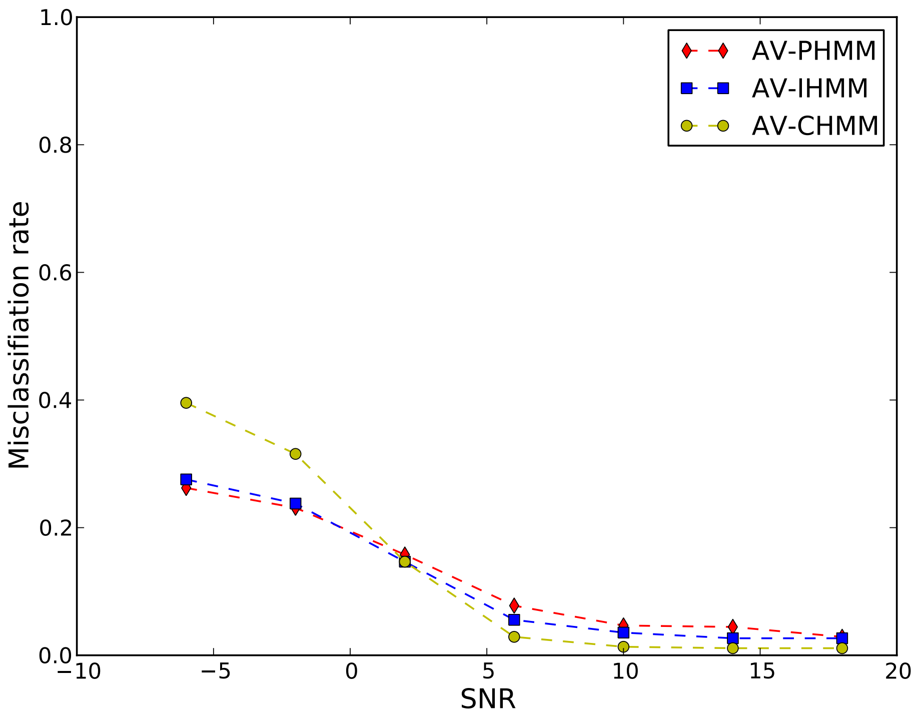

**Figure 5.3:** Performance comparison between AV-PHMM, AV-IHMM and AV-CHMM classifiers. The sophisticated feature stream integration stream in the AV-CHMM yields better results.

the audio-only classifier performs very poorly at approximately 84.2% misclassification rate. Recall that for a digit recognition experiment the expected misclassification rate from random assignment or "guessing" is 90%. Thus, the audio-only classifier only performs marginally better than this in the most severe noise case. As expected, as the noise level decreases the accuracy of the audio-only classifier increases to near 0% misclassification rate (actual 2.7%) in the least noisy environment.

We observe that the audio-visual classifier performs significantly better than audio-only in severe noise conditions. The corresponding performance increase under less noisy acoustic conditions is less substantial, but as shown in Table B.7, significant with 1.1% misclassification rate in the least noisy case for audio-visual classifier and 2.7% for the audio-only classifier. Thus, we achieve more than a halving of the misclassification rate when using the audio-visual classifier. The $p$-value is $p = 0.000181$ indicating the performance difference is indeed significant at significance level $\alpha = 0.05$.

We also observe that under noisy conditions the visual-only classifier performs better than the audio-visual classifier. We can again attribute this ob-

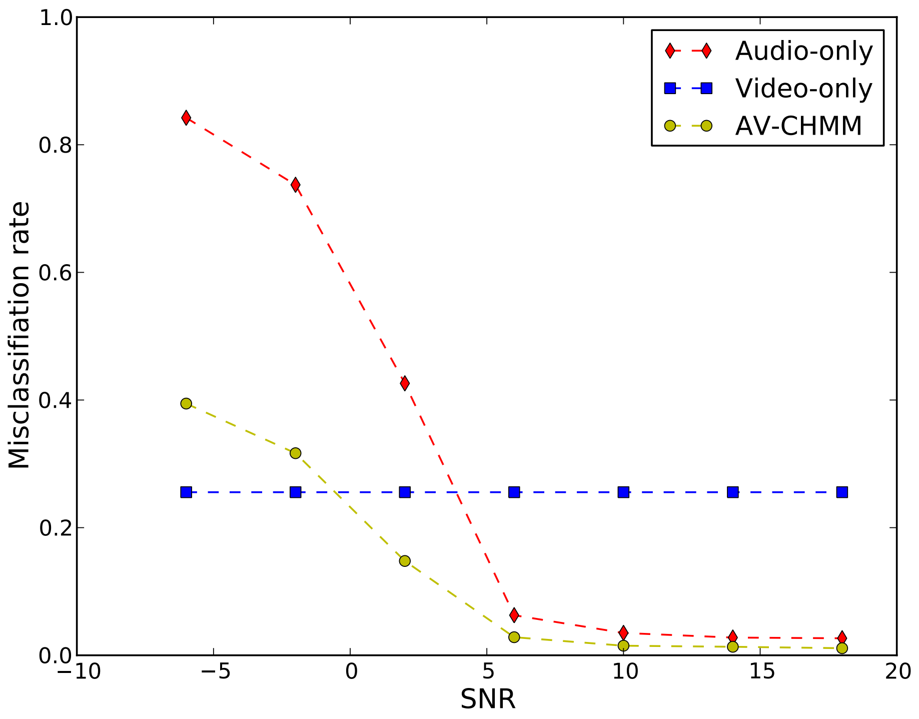

**Figure 5.4:** Performance comparison of audio-visual, audio-only, and visual-only speech recognition classifiers. Note that the video-only classifier is not affected by acoustic noise. We see that even for large SNR levels the audio-visual classifier performs better than audio-only.

servation to the coupling in the AV-CHMM model causing noise in the acoustic stream to affect the visual stream negatively.

The McNemar's test results for Experiment 3 is shown in Table B.7 and B.8. Table B.7 shows audio-visual versus audio-only classifiers. We see that for all SNR levels there is enough evidence to reject the null hypothesis and claim that the audio-visual classifier performs better than audio-only.

**Experiment 4.** In Experiment 4 we compare performance between an audio-visual DBN model learned using variational learning and the same model learned using maximum likelihood estimation. We have chosen the AV-CHMM audio-visual DBN as the subject of the experiment as this model has the largest number of parameters that needs to be estimated and as such should be most prone to overfitting. In the case of eight HMM states and eight GMM components the observation model of the AV-CHMM will have $8 \times 8 \times 8 = 512$ weight parameters, means and covariance matrices; one for each combination of audio and video HMM states and GMM component. We might expect the relatively limited $36 \times 3 = 108$ observation sequences in the training data for each digit to result in overfitting problems for this model in which case the performance of variational learning should be superior due to its automatic model complexity selection property.

Recall that variational learning requires us to choose parameter values for the parameter priors. We use the value $\alpha_0 = 10^{-3}$ for the initial parameters of the Dirichlet priors over both HMM state variables and mixing coefficients. For the Gaussian distributions of the observation model we have used the parameters $\beta_0 = 1$, $\boldsymbol{\mu}_0 = \vec{\mathbf{0}}$, $\upsilon_0 = 20$, $\mathbf{W}_0 = \mathbf{I}$ where $\vec{\mathbf{0}}$ is the zero vector and $\mathbf{I}$ the identity matrix.

Figure 5.5 shows the performance of AV-CHMM models that have been learned using maximum likelihood (EM) and variational learning (VB). We shall refer to the AV-CHMM model learned using VB as AV-VBCHMM and the AV-CHMM model learned using EM as simply AV-CHMM.

In the experiment we observed similar model complexity selection properties for the AV-VBCHMM as in the VB-GMM case in Section 3.6. We found that the number of GMM components remaining in the observation model after convergence of the VB algorithm (components with parameters that have not converged to their prior distributions) differs depending on the digit class and to a lesser extent on which portion of the data was used for learning. Intuitively, this is caused by the different complexities of the distribution of observations from the different digit classes. We found that the number of GMM components remaining in the observation model ranges from five to eight out of the initial eight. For the transition model we found that all eight HMM components remain. The latter is likely due to the relatively smaller number of parameters in the left-to-right no-skip one-state-only-asynchronous transition model used in the AV-VBCHMM which only has $3+6\times2\times3+3 = 42$ non-zero entries in the $8 \times 8 \times 8$ transition matrix. This is a much smaller number of parameters than the 512 weights, means and covariance matrices of the observation model and thus explains why the model complexity selection property is only observed for the observation model.

From Figure 5.5 we see that AV-VBCHMM appears to be more robust to noise than AV-CHMM. This is likely due to its more compact form. However, as the amount of acoustic noise decreases, the AV-CHMM appears to be performing better than AV-VBCHMM. It is likely that the AV-VBCHMM becomes subject to over-smoothing in this SNR region. That is, due to the model complexity selection property, the resulting model may not be sufficiently complex in this region resulting in decreased performance. However, as the noise level approaches clean audio we observe that the two models appear to converge to the same accuracy which seems to contradict that over-smoothing is an issue. We are unsure of the reasons for this apparent contradiction.

The results of McNemar's test for Example 4 is shown in Table B.10. The results confirm that there is indeed a significant difference between the two algorithms with AV-VBCHMM being superior to AV-CHMM for SNR levels

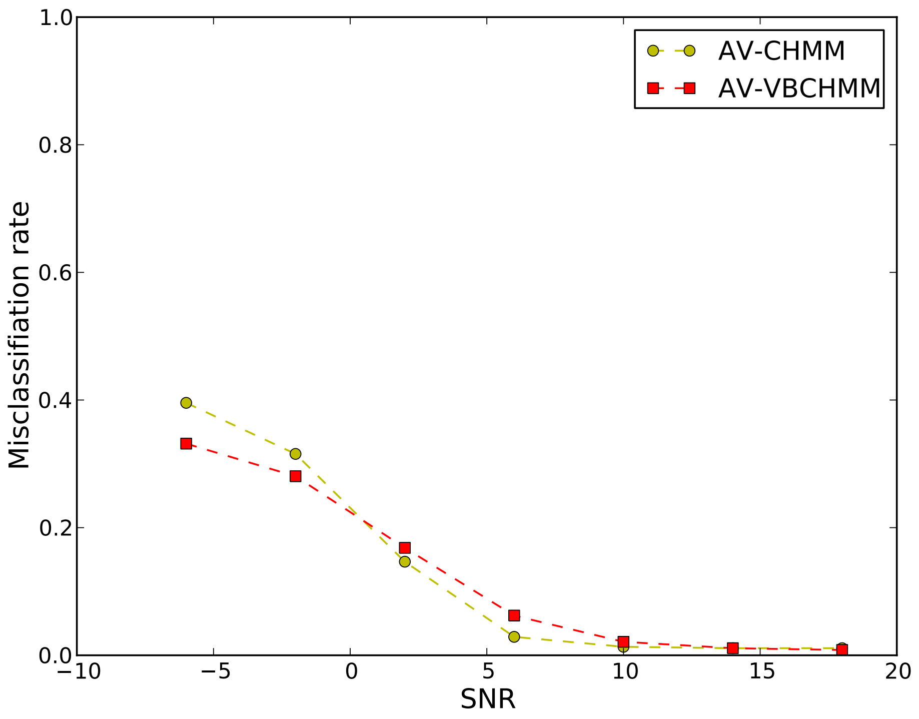

**Figure 5.5:** Comparison of AV-CHMM models learned using EM and VB learning. There is not evidence to support a significant difference between the AV-VBCHMM and AV-CHMM for large values of SNR.

−6 dB and −2 dB and AV-CHMM being superior to AV-CHMM in the region from 2 dB to 10 dB. We note that for SNR levels 14 dB and 18 dB the misclassification rate of AV-VBCHMM is slightly lower than that of AV-CHMM, but that the data is not sufficient to claim within a 5% significance level that AV-VBCHMM performs better than AV-CHMM for high SNR levels.

## 5.6 Discussion

The results show that the visual speech modality indeed contains information relevant to speech recognition. The difference between audio-visual and audio-only speech recognition even at large SNR levels is significant. At 18 dB SNR the performance of the audio-visual classifier is 1.1% versus 2.7% for the audio-only classifier which is more than a halving of the error rate when using the audio-visual classifier.

We found that the misclassification rate of our visual-only classifier is 25.6%. Papandreou *et al.* (2009) reported misclassification rates below 20% for visual-only speech recognition using the same data corpus and experimental protocol. The audio-visual speech recognition results presented in Papandreou *et al.* (2009) are also in general better than ours. However, their system uses a sophisticated adaptive uncertainty compensation instead of our relatively simple stream weighting scheme. In light of this observation, our results seem reasonable.

At the lowest and highest SNR levels, the results also appear to conform to results presented in Valente and Wellekens (2003) and Somervuo (2002), which show that speech GMM models learned using VB perform better than the same models learned using EM. However, in the case of high SNR we did not have enough experimental data to significantly support the claim that variational learning is better than maximum likelihood for AVASR. For SNR levels between 6 dB and 10 dB the models learned using variational learning performs worse than the model learned using maximum likelihood estimation. The reason for the decrease in performance in this region is unknown, but may be related to over-smoothing problems. Thus, the question whether VB is the preferred learning algorithm for audio-visual DBN models remains unanswered and is an interesting problem for future studies. Further investigation is also necessary on how to optimally choose hyper-parameters for the parameter priors such as to avoid over-smoothing while maintaining the desirable automatic model complexity selection property of variational learning.

We found that the AV-PHMM model performs better than the AV-HMM from which we conclude that the stream weighting scheme is advantageous to AVASR. Finally, we observed that the AV-CHMM model performed better than the AV-PHMM and AV-IHMM from which conclude that the stream integration scheme of the AV-CHMM model is superior to the synchronous AV-PHMM model and the independent AV-IHMM model integration schemes.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In this thesis we have presented audio-visual automatic speech recognition (AVASR) and showed how the framework of dynamic Bayesian networks (DBNs) can be used to solve the feature stream integration, learning and classification problems in AVASR. In AVASR, video of a speaker is used as a complementary source of information to audio for speech recognition purposes. The use of the visual modality has been shown to enhance speech recognition performance, especially in noisy acoustic environments as the visual modality is not affected by acoustic noise.

We have proposed and implemented a full-featured AVASR system. We have in particular focused on the problems of feature stream weighting, asynchrony modelling, learning and classification. We have shown how each of the problems can be addressed in the framework of dynamic Bayesian networks. The additional requirement of an AVASR system is acoustic and visual feature extraction. The proposed AVASR system uses MFCCs as acoustic feature and visemic AAM coefficients as visual features which we briefly reviewed in Chapter 4.

The theory of Bayesian networks and dynamic Bayesian networks was presented in Chapter 3. We showed that inference and learning can be performed in general in such networks and we gave the details for networks that consist of latent multinomial variables and observed Gaussian variables. We presented maximum likelihood learning and variational learning resulting in the EM and VB algorithms. Variational learning has the advantage of leading to automatic model complexity selection thereby reducing the problem of overfitting for models with a large number of parameters for which there is not sufficient training data to accurately estimate parameters.

We proposed four DBN models that provide different approaches to solving problems in AVASR. The audio-visual HMM (AV-HMM) is an early integration model where the acoustic and visual features are concatenated before being entered as evidence into a standard left-to-right HMM. The AV-HMM assumes that the acoustic and visual feature streams are synchronous and does not allow weighting individual feature streams. The audio-visual product HMM (AV-PHMM) is similar to the AV-HMM, but with the observation model factored over the acoustic and visual observation variables allowing us to weight each stream independently.

The AV-PHMM still assumes that the acoustic and visual features are synchronous. This is in general not the case for audio-visual speech as, when speaking, the motion of visible articulator such as lips, tongue and jaw comes prior the actual sound being uttered. Hence, there is a slight delay between the acoustic and visual feature streams. This delay is not constant, but depends on the particular sound that is being uttered as well as the speaker. The audio-visual independent HMM (AV-IHMM) uses two separate HMM models with independent state spaces and as such allows desynchrony between the two streams. However, the AV-IHMM may fail to capture the natural correlation between the acoustic and visual feature stream due to the independence assumption. In order to solve this problem, the audio-visual coupled HMM (AV-CHMM) uses two separate HMM models for each stream, but couples the streams at the state level. This coupling allows us to control the level of asynchrony without assuming that the streams are synchronous. In our experiments we have allowed the two streams to desynchronise by at most one HMM state.

The experiments were performed using the CUAVE data corpus. CUAVE consists of 36 different speakers uttering the digits from zero to nine, thereby rendering the experiments as multi-speaker digit recognition experiments. The results show that visual speech indeed contains information valuable to speech recognition. The performance of visual-only speech recognition is in general, as we might have expected, lower than audio-only speech recognition. However, by integrating acoustic and visual speech information in audio-visual DBNs we are able to increase speech recognition performance above that of audio-only speech recognition. This is because the combination of acoustic and visual speech information is superior to any of the two modalities alone. The performance increased is most pronounced in noisy acoustic environments as the visual modality is not affected by acoustic noise.

We also found that the AV-PHMM model performs better than AV-HMM, indicating that stream weighting is beneficial to AVASR. Further we found that from the AV-PHMM, AV-IHMM and AV-CHMM models, the AV-CHMM performs best on the audio-visual digit recognition task from which we conclude that allowing the acoustic and visual feature streams to desynchronise while restricting the level of asynchrony most accurately models the dynamics of audio-visual speech.

Finally, we found that learning audio-visual DBNs and in particular the AV-CHMM using variational learning results in a model that appear to be more robust to noise than the same model learned using maximum likelihood estimation. This is likely due to the more compact form of the model resulting from variational learning. However, at small levels of acoustic noise we were not able to shown any significant difference between models learned using the two different algorithms. At intermediate levels of noise maximum likelihood learning gave better results than variational learning illustrating the risk of over-smoothing using variational learning. Further investigation into variational learning is necessary to determine how to appropriately choose parameter priors in order to avoid such issues while taking advantage of the automatic model complexity selection property of variational learning.

## 6.2 Future work

There are interesting problems yet to be solved in AVASR with many aspects where there is room for significant improvement to existing systems.

Further investigation into visual feature extraction would be a valuable contribution. As mentioned in Section 4.2.2 the standard AAM coefficients are not necessarily the most appropriate for use in AVASR, as they will contain much information about the variability across speakers when trained on a multi-speaker dataset, while we are more interested in speech information only. In our research we have partly solved this problem by building individual AAMs for each speaker. However, general AVASR system must be able to handle speakers that are not featured in the training set. Although the visemic AAM proposed by Papandreou *et al.* (2009) is an improvement to the standard HMM, we believe that it is possible to develop methods that separates speech even better from non-speech related information.

The AAM-based method used in this research (Section 4.2.2) use PCA as a technique for dimensionality reduction. PCA reduces dimensionality by finding the smaller dimensional subspace that captures most of the information contained in the data. This subspace is found by considering similarities within a particular class. However, in speech recognition applications we are in general more interested in the subspace that optimises the difference *between* the classes. A dimensionality reduction technique that optimises the difference between classes is *linear discriminant analysis* (LDA). It would be interesting to evaluate the performance of an AVASR system using LDA as a dimensionality reduction technique for acoustic and visual feature extraction.

When learning model parameters using maximum likelihood or variational learning we are also only optimising for a specific class such as a digit. That is, we are finding the model parameters that best models features that the samples from a particular class has in common. However, again we are more interested in the parameters that model features that separates the particular class from other classes. Estimating model parameters by maximising the difference between classes is referred to as *discriminative learning* (Jebara, 2002). Discriminative learning has been successful in audio-only speech recognition systems (Jiang, 2010). It is therefore likely that this will also be the case in AVASR. Thus, the application of discriminative learning to AVASR will be an interesting research topic. The objective function to be maximised in discriminative learning is typically significantly more complicated and sensitive to initial conditions than in the case of maximum likelihood and variational learning.

We have proposed several audio-visual DBNs that solve problems associated with AVASR. However, the DBN framework is a comprehensive one. There may be models that we have not considered and that model different aspects of AVASR that we have not considered. These models may perform even better than the models proposed here. Using the framework of Bayesian networks that we presented in Chapter 3 and the software described in Appendix C it is trivial to propose new AVASR models and evaluate their performance. It should be emphasised that a thorough understanding of the theory presented in Chapter 3 is essential when modelling Bayesian networks. However, the software may be treated as a "black box" as the interpretation of the output is immediate from the theory.

Further study is needed regarding the performance of variational learning for speech recognition applications. In particular, it is necessary to investigate further how to optimally choose hyperparameters for the parameter priors so as to avoid over-smoothing while maintaining the automatic model complexity selection property of variational learning. As variational learning is in particular expected to address the problem of overfitting, which becomes a problem in models with a large number of parameters and not enough training data, it would be interesting to investigate the effect of variational learning while varying the amounts of training data, and the number of parameters that needs to be estimated in the model.

We have on several occasions mentioned the lack of a publicly available large-vocabulary multi-speaker AVASR data corpus. The collection of such a data corpus would be of immense value to the research field. A collaborative initiative by multiple universities and researchers is a cost-effective way to solve the problem. See Voxforge (2006) for an example of how such a collaborative data acquisition project may be carried out. Alternatively, video sharing sites such as YouTube contain enormous amounts of suitable data. However, the ownership and right-of-use of such content is often problematic.

Speech will undoubtedly be one of the primary forms of communicating with information systems in the future. Increasingly more computing is happening away from the traditional desktop setting. Video cameras and microphones in handheld devices form a second pair of "eyes and ears" and the Internet allows us to share our experiences with the results of the world. There is no doubt that there is great advantage in having computer systems that can analyse and interpret this data. Such technologies will improve human-computer interaction and assist when searching and organising content, while at the same time giving rise to interesting scientific and technological problems to be solved.

# Appendices

# Appendix A

# Probability theory and densities

In this appendix we list a selection of results from probability theory and probability distributions that we have made use of in the thesis.

## A.1 Probability theory

### A.1.1 Rules of probability

The fundamental results from probability theory are the sum rule and the product rule. Let $\mathbf{x}_A$ and $\mathbf{x}_B$ be two random variables. The sum rule and product rule is then as follows

**The sum rule**

$$p(\mathbf{x}_A) = \sum_{\mathbf{x}_B} p(\mathbf{x}_A, \mathbf{x}_B). \tag{A.1.1}$$

**The product rule**

$$p(\mathbf{x}_A, \mathbf{x}_B) = p(\mathbf{x}_B|\mathbf{x}_A)p(\mathbf{x}_A). \tag{A.1.2}$$

These rules also generalise to the case where $A$ and $B$ are arbitrary subsets of variables. Using the product rule twice it is possible to prove the following result

**Bayes' theorem**

$$p(\mathbf{x}_B|\mathbf{x}_A) = \frac{p(\mathbf{x}_A|\mathbf{x}_B)p(\mathbf{x}_B)}{p(\mathbf{x}_A)}. \tag{A.1.3}$$

In Bayes' theorem the denominator $p(\mathbf{x}_A)$ can be calculated using the sum rule

$$p(\mathbf{x}_A) = \sum_{\mathbf{x}_B} p(\mathbf{x}_A|\mathbf{x}_B)p(\mathbf{x}_B). \tag{A.1.4}$$

## A.2 Probability densities

### A.2.1 Gaussian

The Gaussian distribution, also known as the *normal* distribution, is the most commonly used distribution for a continuous random variable. In this thesis we are mostly interested in the *multivariate* Gaussian over a $D$-dimensional real random vector variable $\mathbf{x}$. The multivariate Gaussian has as parameters a $D$-dimensional mean vector $\boldsymbol{\mu}$ and a $D \times D$ covariance matrix $\Sigma$. Note that we require the covariance matrix to be symmetric and positive-definite. The multivariate Gaussian is defined by

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}) \right\}. \tag{A.2.1}$$

From the above we can show that the following results hold

$$\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu} \tag{A.2.2}$$
$$\mathrm{cov}[\mathbf{x}] = \boldsymbol{\Sigma}. \tag{A.2.3}$$

The inverse of the covariance matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is referred to as the precision matrix, which is also symmetric and positive definite. The conjugate prior for the mean vector $\boldsymbol{\mu}$ is again a Gaussian. The conjugate prior for the precision matrix $\boldsymbol{\Lambda}$ is the Wishart distribution (A.2.19), and the conjugate prior for the parameter pair $(\boldsymbol{\mu}, \boldsymbol{\Lambda})$ is the Gaussian-Wishart distribution (A.2.24).

Note that in the case $D = 1$ the multivariate Gaussian reduces to the standard univariate Gaussian distribution with mean $\mu$ and variance $\sigma^2$.

### A.2.2 Binomial

The binomial distribution is a distribution over a binary variable x defined as

$$\mathrm{Bin}(m|N, \mu) = \binom{N}{m} \mu^m (1-\mu)^{N-m} \tag{A.2.4}$$

where $\mu \in [0, 1]$ is the probability that $x = 1$. The binomial distribution gives the probability of observing $m$ occurrences of $x = 1$ in a set of $N$ samples. For the binomial distribution we have that

$$\begin{aligned} \mathbb{E}[m] &= N\mu \\ \mathrm{var}[m] &= N\mu(1-\mu). \end{aligned} \tag{A.2.5}$$

### A.2.3 Multinomial

The multinomial distribution is a discrete distribution over a $K$-dimensional binary variable $\mathbf{z}$ with components $z_k \in \{0, 1\}$ such that $\sum_k z_k = 1$. It is defined by

$$p(\mathbf{z}) = \prod_{k=1}^{K} \mu_k^{z_k}. \tag{A.2.6}$$

From the above definition we can show that

$$\begin{aligned}
\mathbb{E}[z_k] &= \mu_k & \text{(A.2.7)} \\
\text{var}[z_k] &= \mu_k(1-\mu_k) & \text{(A.2.8)} \\
\text{cov}[z_j z_k] &= I_{jk}\mu_k & \text{(A.2.9)}
\end{aligned}$$

where $I_{jk}$ is the $j,k$ element of the identity matrix. The value of $\mu_k$ gives the probability of $z_k = 1$ and so these parameters are subject to the constraints $0 \leq \mu_k \leq 1$ and $\sum_k \mu_k = 1$. The conjugate prior for the parameter vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)^{\mathrm{T}}$ is the Dirichlet distribution (A.2.11).

### A.2.4 Dirichlet

The Dirichlet distribution is a multivariate distribution over $K$ random variables $\mu_k$ for $k = 1, \ldots, K$ subject to

$$0 \leq \mu_k \leq 1, \qquad \sum_{k=1}^{K} \mu_k = 1. \tag{A.2.10}$$

Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)^{\mathrm{T}}$ denote the vector of random variables and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)^{\mathrm{T}}$ the vector of distribution parameters. The Dirichlet distribution is then defined by

$$\text{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = C(\boldsymbol{\alpha}) \prod_{k=1}^{K} \mu_k^{\alpha_k - 1} \tag{A.2.11}$$

where

$$C(\boldsymbol{\alpha}) = \frac{\Gamma(\widehat{\alpha})}{\Gamma(\alpha_1), \ldots, \Gamma(\alpha_K)} \tag{A.2.12}$$

and

$$\widehat{\alpha} = \sum_{k=1}^{K} \alpha_k. \tag{A.2.13}$$

The following results can be shown from the definition of the Dirichlet distribution

$$\mathbb{E}[\mu_k] = \frac{\alpha_k}{\widehat{\alpha}} \tag{A.2.14}$$

$$\text{var}[\mu_k] = \frac{\alpha_k(\widehat{\alpha} - \alpha_k)}{\widehat{\alpha}^2(\widehat{\alpha}+1)} \tag{A.2.15}$$

$$\text{cov}[\mu_j \mu_k] = -\frac{\alpha_j \alpha_k}{\widehat{\alpha}^2(\widehat{\alpha}+1)} \tag{A.2.16}$$

$$\mathbb{E}[\ln \mu_k] = \psi(\alpha_k) - \psi(\widehat{\alpha}) \tag{A.2.17}$$

where $\psi(\cdot)$ is the *digamma* function defined by

$$\psi(a) = \frac{d}{da} \ln \Gamma(a). \tag{A.2.18}$$

The parameters $\alpha_k$ are subject to the constraint $\alpha_k > 0$ in order to ensure that the distribution can be normalised.

The Dirichlet distribution is the conjugate prior to the multinomial distribution (A.2.3). In this case, the parameters $\alpha_k$ can be interpreted as the effective number of observations of the corresponding values of a $K$-dimensional binary vector $\mathbf{z}$. The Dirichlet distribution has finite density everywhere provided $\alpha_k \geq 1$ for all $k$.

## A.2.5 Wishart

The Wishart distribution is the conjugate prior for the precision matrix of the multivariate Gaussian. It is defined by

$$\mathcal{W}(\mathbf{\Lambda}|\mathbf{W}, \nu) = B(\mathbf{W}, \nu)|\mathbf{\Lambda}|^{(\nu-D-1)/2} \exp\left(-\frac{1}{2}\text{trace}(\mathbf{W}^{-1}\mathbf{\Lambda})\right) \tag{A.2.19}$$

where $\mathbf{W}$ is a $D \times D$ symmetric, positive definite matrix and

$$B(\mathbf{W}, \nu) = |\mathbf{W}|^{-\nu/2} \left(2^{\nu D/2} \pi^{D(D-1)/4)} \prod_{i=1}^{D} \Gamma\left(\frac{\nu+1-i}{2}\right)\right) \tag{A.2.20}$$

where $\Gamma(\cdot)$ is gamma function. The parameter $\nu$ is called the *number of degrees of freedom* of the distribution constrained by

$$\nu > D - 1 \tag{A.2.21}$$

to ensure that the Gamma function in the normalisation factor is well-defined.

The following results hold for the Wishart distribution:

$$\mathbb{E}[\mathbf{\Lambda}] = \nu \mathbf{W} \tag{A.2.22}$$

$$\mathbb{E}[\ln |\Lambda|] = \sum_{i=1}^{D} \psi\left(\frac{\nu+1-i}{2}\right) + D \ln 2 + \ln |\mathbf{W}|. \tag{A.2.23}$$

where $\psi(\cdot)$ is the digamma function defined by (A.2.18).

### A.2.6 Gaussian-Wishart

The Gaussian-Wishart distribution is the conjugate distribution for both the mean and precision matrix of the multivariate Gaussian. It is the product of a Gaussian distribution for $\boldsymbol{\mu}$, whose precision is proportional to $\boldsymbol{\Lambda}$, and a Wishart distribution over $\boldsymbol{\Lambda}$. The Gaussian-Wishart distribution is defined by

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\mathbf{m}, \beta, \boldsymbol{W}, \nu) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}, (\beta\boldsymbol{\Lambda})^{-1})\mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}, \nu). \tag{A.2.24}$$

### A.2.7 The Exponential Family

All the models that have been presented in this appendix are part of the *exponential family* of probability distributions (Wainwright and Jordan, 2008). The exponential family of distributions over $\mathbf{x}$, given parameters $\boldsymbol{\eta}$, is defined as

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta})\exp\{\boldsymbol{\eta}^T\mathbf{u}(\mathbf{x})\} \tag{A.2.25}$$

where $\mathbf{x}$ may be a scalar or vector, and may be discrete of continuous. Here $\boldsymbol{\eta}$ are called the *natural parameters* and $\mathbf{u}(\mathbf{x})$ is some function of $\mathbf{x}$. The function $g(\boldsymbol{\eta})$ can be interpreted as the coefficient that ensures that the distribution is normalised and therefore satisfies

$$g(\boldsymbol{\eta})\int h(\mathbf{x})\exp\{\boldsymbol{\eta}^T\mathbf{u}(\mathbf{x})\}\mathrm{d}\mathbf{x} = 1. \tag{A.2.26}$$

# Appendix B

# Experimental results

## B.1 Experiment 1

| SNR | -6 | -2 | 2 | 6 | 10 | 14 | 18 |
|---|---|---|---|---|---|---|---|
| **AV-HMM** | | | | | | | |
| $\lambda_A$ | n/a | | | | | | |
| $\lambda_V$ | n/a | | | | | | |
| WER | 0.880556 | 0.521667 | 0.187778 | 0.075556 | 0.047222 | 0.054444 | 0.048889 |
| **AV-PHMM** | | | | | | | |
| $\lambda_A$ | 0.0 | 0.24 | 1.04 | 1.6 | 1.64 | 1.8 | 2.0 |
| $\lambda_V$ | 2.0 | 1.76 | 0.96 | 0.4 | 0.36 | 0.2 | 0.0 |
| WER | 0.263333 | 0.232222 | 0.158333 | 0.077222 | 0.046667 | 0.044444 | 0.029444 |

**Table B.1:** Average stream weights and word error rates for Experiment 1.

| SNR | -6 | -2 | 2 | 6 | 10 | 14 | 18 |
|---|---|---|---|---|---|---|---|
| | **AV-PHMM** | | | | | | |
| **AV-HMM** | $\begin{matrix}180 & 35\\1146 & 439\end{matrix}$ | $\begin{matrix}765 & 96\\617 & 322\end{matrix}$ | $\begin{matrix}1384 & 78\\131 & 207\end{matrix}$ | $\begin{matrix}1622 & 42\\39 & 97\end{matrix}$ | $\begin{matrix}1690 & 25\\26 & 59\end{matrix}$ | $\begin{matrix}1668 & 34\\52 & 46\end{matrix}$ | $\begin{matrix}1667 & 45\\80 & 8\end{matrix}$ |
| $p$ | 0.000000 | 0.000000 | 0.000322 | 0.824141 | 1.000000 | 0.066779 | 0.002358 |

**Table B.2:** McNemar's tests for Experiment 1. (AV-PHMM vs. AV-HMM)

## B.2 Experiment 2

| SNR | -6 | -2 | 2 | 6 | 10 | 14 | 18 |
|---|---|---|---|---|---|---|---|
| **AV-PHMM** | | | | | | | |
| $\lambda_A$ | 0.0 | 0.24 | 1.04 | 1.6 | 1.64 | 1.8 | 2.0 |
| $\lambda_V$ | 2.0 | 1.76 | 0.96 | 0.4 | 0.36 | 0.2 | 0.0 |
| WER | 0.261667 | 0.230000 | 0.160000 | 0.076111 | 0.046667 | 0.045000 | 0.028889 |
| **AV-IHMM** | | | | | | | |
| $\lambda_A$ | 0.0 | 0.32 | 1.76 | 1.8 | 1.8 | 1.96 | 2.0 |
| $\lambda_V$ | 2.0 | 1.68 | 0.24 | 0.2 | 0.2 | 0.04 | 0.0 |
| WER | 0.276111 | 0.237778 | 0.147778 | 0.055556 | 0.035556 | 0.029444 | 0.026667 |
| **AV-CHMM** | | | | | | | |
| $\lambda_A$ | 0.16 | 0.4 | 1.24 | 1.6 | 1.76 | 1.84 | 1.88 |
| $\lambda_V$ | 1.84 | 1.6 | 0.76 | 0.4 | 0.24 | 0.16 | 0.12 |
| WER | 0.395556 | 0.313889 | 0.146667 | 0.028889 | 0.013333 | 0.011667 | 0.011111 |

**Table B.3:** Average stream weights and word error rates for Experiment 2.

| SNR | -6 | | -2 | | 2 | | 6 | | 10 | | 14 | | 18 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **AV-PHMM** | | | | | | | | | | | | | |
| **AV-CHMM** | 921 | 167 | 1062 | 173 | 1370 | 166 | 1639 | 109 | 1707 | 69 | 1713 | 66 | 1732 | 48 |
| | 408 | 304 | 324 | 241 | 142 | 122 | 24 | 28 | 9 | 15 | 6 | 15 | 16 | 4 |
| $p$ | 0.000000 | | 0.000000 | | 0.190011 | | 0.000000 | | 0.000000 | | 0.000000 | | 0.000107 | |

**Table B.4:** McNemar's test for experiment 2-1 (AV-CHMM vs. AV-PHMM)

| SNR | -6 | | -2 | | 2 | | 6 | | 10 | | 14 | | 18 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **AV-IHMM** | | | | | | | | | | | | | |
| **AV-CHMM** | 956 | 132 | 1122 | 113 | 1412 | 126 | 1672 | 76 | 1727 | 49 | 1730 | 48 | 1736 | 44 |
| | 347 | 365 | 250 | 315 | 124 | 140 | 28 | 24 | 9 | 15 | 17 | 5 | 16 | 4 |
| $p$ | 0.000000 | | 0.000000 | | 0.949571 | | 0.000004 | | 0.000000 | | 0.000198 | | 0.000491 | |

**Table B.5:** McNemar's test for experiment 2-2 (AV-CHMM vs. AV-IHMM)

## B.3 Experiment 3

| SNR | -6 | -2 | 2 | 6 | 10 | 14 | 18 |
|---|---|---|---|---|---|---|---|
| | **Audio-Only** | | | | | | |
| $\lambda_A$ | n/a | | | | | | |
| $\lambda_A$ | n/a | | | | | | |
| WER | 0.842222 | 0.737222 | 0.426111 | 0.062778 | 0.035000 | 0.027778 | 0.026667 |
| | **Video-Only** | | | | | | |
| $\lambda_A$ | n/a | | | | | | |
| $\lambda_A$ | n/a | | | | | | |
| WER | 0.255556 | 0.255556 | 0.255556 | 0.255556 | 0.255556 | 0.255556 | 0.255556 |
| | **AV-CHMM** | | | | | | |
| $\lambda_A$ | 0.16 | 0.4 | 1.24 | 1.6 | 1.76 | 1.84 | 1.88 |
| $\lambda_V$ | 1.84 | 1.6 | 0.76 | 0.4 | 0.24 | 0.16 | 0.12 |
| WER | 0.394444 | 0.316667 | 0.147778 | 0.028333 | 0.015000 | 0.013333 | 0.011111 |

**Table B.6:** Average stream weights and word error rates for Experiment 2.

| SNR | -6 | | -2 | | 2 | | 6 | | 10 | | 14 | | 18 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Audio-only** | | | | | | | | | | | | | |
| **AV-CHMM** | 208 | 882 | 367 | 863 | 931 | 603 | 1653 | 96 | 1724 | 49 | 1735 | 41 | 1740 | 40 |
| | 76 | 634 | 106 | 464 | 102 | 164 | 34 | 17 | 13 | 14 | 15 | 9 | 12 | 8 |
| $p$ | 0.000000 | | 0.000000 | | 0.000000 | | 0.000025 | | 0.000009 | | 0.000835 | | p = 0.000181 | |

**Table B.7:** McNemar's test for experiment 3-1 (AV-CHMM vs. Audio-Only)

| SNR | -6 | | -2 | | 2 | | 6 | | 10 | | 14 | | 18 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Video-only** | | | | | | | | | | | | | |
| **AV-CHMM** | 913 | 177 | 1035 | 195 | 1220 | 314 | 1324 | 425 | 1335 | 438 | 1334 | 442 | 1336 | 444 |
| | 427 | 283 | 305 | 265 | 120 | 146 | 16 | 35 | 5 | 22 | 6 | 18 | 4 | 16 |
| $p$ | 0.000000 | | 0.000001 | | 0.000000 | | 0.000000 | | 0.000000 | | 0.000000 | | 0.00000 | |

**Table B.8:** McNemar's test for experiment 3-2 (AV-CHMM vs. Video-Only)

## B.4 Experiment 4

| SNR | -6 | -2 | 2 | 6 | 10 | 14 | 18 |
|---|---|---|---|---|---|---|---|
| **AV-CHMM** | | | | | | | |
| $\lambda_A$ | 0.16 | 0.4 | 1.24 | 1.6 | 1.76 | 1.84 | 1.88 |
| $\lambda_V$ | 1.84 | 1.6 | 0.76 | 0.4 | 0.24 | 0.16 | 0.12 |
| WER | 0.395556 | 0.315556 | 0.146667 | 0.028889 | 0.012222 | 0.011667 | 0.011111 |
| **AV-VBCHMM** | | | | | | | |
| $\lambda_A$ | 0.0 | 0.16 | 0.8 | 1.56 | 1.72 | 1.8 | 1.84 |
| $\lambda_V$ | 2.0 | 1.84 | 1.2 | 0.44 | 0.28 | 0.2 | 0.16 |
| WER | 0.331666 | 0.280555 | 0.168333 | 0.062222 | 0 0.021111 | 0.011111 | 0.008333 |

**Table B.9:** Average stream weights and word error rates for Experiment 2.

| SNR | -6 | | -2 | | 2 | | 6 | | 10 | | 14 | | 18 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **AV-VBCHMM** | | | | | | | | | | | | | |
| **AV-CHMM** | 841 | 247 | 1046 | 186 | 1359 | 176 | 1666 | 82 | 1754 | 24 | 1770 | 9 | 1777 | 3 |
| | 362 | 350 | 249 | 319 | 138 | 127 | 22 | 30 | 8 | 14 | 10 | 11 | 8 | 12 |
| $p$ | 0.000004 | | 0.002952 | | 0.036795 | | 0.000000 | | 0.008010 | | 1.0 | | 0.227800 | |

**Table B.10:** McNemar's test for experiment 4 (AV-CHMM vs. AV-VBCHMM)

# Appendix C

# Software

As part of the research, an AVASR system as described in Chapter 4 was implemented in Python (Reikeras *et al.*, 2010*a*) and using the Matlab Bayesian Network Toolbox (BNT) (Murphy, 2001). We used `mlabwrap` (Schmolck and Rathod, 2010) to interface with the Matlab toolbox from Python. The future plan is to port the Matlab part of the system to Python and as such make it a pure Python project. Such a port is already in progress (Gouws, 2010). However, it does not yet support Gaussian distributions and the inference junction tree algorithm needed for DBNs.

In summary, the following software was used in the research:

- `BNT`. The Bayesian networks toolbox (Murphy, 2001) is a Matlab toolbox for inference and learning of Bayesian networks and dynamic Bayesian networks. It supports a wide range of conditional probability distributions including multinomial, Gaussian, multi-layer percepton and softmax densities. BNT is available at `http://bnt.googlecode.com`.

- `PyAAM`. The Python AAM toolkit (Reikeras *et al.*, 2010*a*) is a Python toolkit for building, evaluating, and tracking AAMs. PyAAM is available at `https://bitbucket.org/helger/pyaam`.

- `Talkbox`. Talkbox (Cournapeau, 2008) is a set of python functions for audio signal processing. It includes the calculation of MFCCs. Talkbox is available at `http://scikits.appspot.com/talkbox`.

In addition to the above software we modified BNT to perform variational learning by using results from Section 3.5. Recall that the only part of the EM algorithm that is necessary to modify is how the model parameters are updated from the expected sufficient statistics during the M step. The interface junction tree algorithm remains unchanged.

# List of References

Alpaydin, E. (2004). *Introduction to machine learning.* MIT Press.

Attias, H. (2000). A variational bayesian framework for graphical models. In: *In Advances in Neural Information Processing Systems 12*, pp. 209–215. MIT Press.

Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In: *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090–1097.

Beal, M.J. and Ghahramani, Z. (2004). Variational Bayesian Learning of Directed Graphical Models with Hidden Variables. *Bayesian Analysis*, pp. 1–44.

Benoit, C. (1992). The Intrinsic Bimodality of Speech Communication and the Synthesis of Talking Faces. *Journal of the Hungarian Telecommunication Association*, , no. 43, pp. 32–40.

Bishop, C.M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics).* 1st edn. Springer.

Bregler, C. and Konig, Y. (1994). "Eigenlips" for robust speech recognition.

Chen, T. (2001). Audiovisual speech processing: Lip reading and lip synchronization. *EEE Signal Processing Mag.*, vol. 18, no. 1, pp. 9–21.

Chen, T. and Rao, R. (1998). Audio-visual integration in multimodal communication. *Proceedings of the IEEE*, vol. 86, no. 5, pp. 837–852.

Chu, S.M. and Huang, T.S. (2007). Audio-Visual Speech Fusion Using Coupled Hidden Markov Models. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 1–2.

Cootes, T., Taylor, C. and Pt, M.M. (2000). Statistical models of appearance for computer vision.

Cootes, T.F., Edwards, G.J. and Taylor, C. (1998). Active appearance models. In: *European Conference on Computer Vision*, vol. 2, pp. 484–498.

Cootes, T.F., Edwards, G.J. and Taylor, C.J. (2001). Active appearance models. *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 681–685.

Cournapeau, D. (2008). Talkbox.
Available at: `http://scikits.appspot.com/talkbox`

Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech, and Signal Processing [see also IEEE Transactions on Signal Processing], IEEE Transactions on*, vol. 28, no. 4, pp. 357–366.

Edwards, G.J., Taylor, C.J. and Cootes, T.F. (1998). Interpreting face images using active appearance models. In: *FG '98: Proceedings of the 3rd. International Conference on Face & Gesture Recognition*, pp. 300–305. IEEE Computer Society, Washington, DC, USA.

E.K. Patterson, S. Gurbuz, Z. Tufekci and J.N. Gowdy (2002). Cuave: A new audio-visual database for multimodal human-computer interface research. In: *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pp. 2017–2020.

Garofolo, J.S., Lamel, L.F., Fisher, W.M., Fiscus, J.G., Pallett, D.S. and Dahlgren, N.L. (1993). Darpa timit acoustic phonetic continuous speech corpus cdrom.

Gasquet, C. and Witomski, P. (1999). *Fourier analysis and applications: filtering, numerical computation, wavelets.* Springer-Verlag New York, Inc., New York, NY, USA.

Glotin, H., Vergyri, D., Neti, C., Potamianos, G. and Luettin, J. (2001). Weighting schemes for audio-visual fusion in speech recognition. Decision fusion.

Goldschen, A.J. (1993). *Continuous automatic speech recognition by lipreading.* Ph.D. thesis, Washington, DC, USA.

Gouws, A. (2010). GrMPy: Graphical models in Python.
Available at: `http://dip.sun.ac.za/vision/trac-git/agouws-GrMPy.bzr`

Gowdy, J., Subramanya, A. and Bartels, C. (2004). DBN based multi-stream models for audio-visual speech recognition. In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 1, pp. 993–996.

Gurban, M., Thiran, J.-P., Drugman, T. and Dutoit, T. (2008). Dynamic modality weighting for multi-stream HMMs in audio-visual speech recognition. In: *IMCI '08: Proceedings of the 10th international conference on Multimodal interfaces*, pp. 237–240. ACM, New York, NY, USA.

Hershey, J., Attias, H., Jojic, N. and Kristjansson, T. (2004). Audio-visual graphical models for speech processing. In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 5.

Jebara, T. (2002). *Discriminative, Generative and Imitative Learning.* Ph.D. thesis, Massachusetts Institute of Technolog.

Jiang, H. (2010). Discriminative training of hmms for automatic speech recognition: A survey. *Comput. Speech Lang.*, vol. 24, pp. 589–608.

Jordan, M. (2003). An introduction to graphical models.

Kass, M., Witkin, A. and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331.

Kruskal, J.B. (1956). On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50.

Liew, A.W.-C. and Wang, S. (2009). *Visual speech recognition; lip segmentation and mapping.*

Liu, X., Zhao, Y., Pi, X., Liang, L. and Nefian, A.V. (2002). Audio-visual continuous speech recognition using a coupled hidden markov model.

Lucas, B.D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In: *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*, pp. 674–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Lucey, P., Martin, T. and Sridharan, S. (2004). Confusability of phonemes grouped according to their viseme classes in noisy environments.

Lv, G., Jiang, D., Zhao, R. and Hou, Y. (2007). Multi-stream asynchrony modeling for audio-visual speech recognition. In: *ISM '07: Proceedings of the Ninth IEEE International Symposium on Multimedia*, pp. 37–44. IEEE Computer Society, Washington, DC, USA.

Marcheret, E. and Libal, V.P. (2007). Dynamic Stream Weight Modeling for Audio-Visual Speech Recognition.

Mase, K. and Pentland, A. (1991). Automatic lip-reading by optical flow analysis.

Matthews, I. and Baker, S. (2003). Active appearance models revisited. *International Journal of Computer Vision*, vol. 60, pp. 135–164.

Matthews, I., Cootes, T.F., Bangham, J.A., Cox, S. and Harvey, R. (2002 February). Extraction of visual features for lipreading. *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, pp. 198–213.

McGrory, C. and Titterington, D. (2006). Variational Bayesian Analysis for Hidden Markov Models. *Australian & New Zealand Journal of Statistics*, vol. 51, no. 2, pp. 227–244.

McGurk, H. and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature*, vol. 264, no. 5588, pp. 746–748.

Murphy, K.P. (2001). The Bayes net toolbox for Matlab.
Available at: `http://bnt.googlecode.com/`

Murphy, K.P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, UC Berkeley, Computer Science Division.

Nefian, A.V., Liang, L., Pi, X., Liu, X. and Murphy, K. (2002). Dynamic bayesian networks for audio-visual speech recognition. *EURASIP J. Appl. Signal Process.*, vol. 2002, no. 1, pp. 1274–1288.

Neti, C., Potamianos, G., Luettin, J., Matthews, I., Glotin, H., Vergyri, D., Sison, J., Mashari, A. and Zhou, J. (2000). Audio-visual speech recognition. Tech. Rep. WS00AVSR, Johns Hopkins University, CLSP.

Papandreou, G., Katsamanis, A., Pitsikalis, V. and Maragos, P. (2009). Adaptive multimodal fusion by uncertainty compensation with application to audiovisual speech recognition. *Trans. Audio, Speech and Lang. Proc.*, vol. 17, no. 3, pp. 423–435.

Parisi, G. (1988). Statistical Field Theory. vol. 66.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems : networks of plausible inference*. Morgan Kaufmann.

Petajan, E., Bischoff, B., Bodoff, D. and Brooke, N.M. (1988). An improved automatic lipreading system to enhance speech recognition. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '88, pp. 19–25. ACM, New York, NY, USA.

Petajan, E.D. (1984). *Automatic lipreading to enhance speech recognition (speech reading)*. Ph.D. thesis, Champaign, IL, USA.

Potamianos, G., Neti, C., Gravier, G., Garg, A. and Senior, A.W. (2003). Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, vol. 91, no. 9, pp. 1306–1326.

Potamianos, G., Neti, C., Luettin, J. and Matthews, I. (2004). Audio-Visual Automatic Speech Recognition: An Overview. *Issues in Visual and Audio-Visual Speech Processing.*

Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of speech recognition.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Rabiner, L.R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. In: *Proceedings of the IEEE*, pp. 257–286.

Reikeras, H., Herbst, B., du Preez, H. and Engelbrech, H. (2010*a*). Audio-Visual Automatic Speech Recognition in Python. *Proceedings of the 9th Python in Science conference (SciPy 2010).*

Reikeras, H., Herbst, B., du Preez, H. and Engelbrech, H. (2010*b*). Audio-Visual Automatic Speech Recognition using Dynamic Bayesian Networks. *Proceedings of the 21st Annual Symposium of the Pattern Recognition Association of South Africa.*

Saenko, K. and Livescu, K. (2006). An asynchronous DBN for audio-visual speech recognition. *Spoken Language Technology Workshop, 2006. IEEE*, pp. 154–157.

Savitzky, A. and Golay, M.J.E. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639.

Schmolck, A. and Rathod, V. (2010). Mlabwrap.
Available at: `http://mlabwrap.sourceforge.net/`

Smith, L.I. (2005). A Tutorial on Principal Component Analysis. Web.

Sober, E. (1996). Parsimony and predictive equivalence. *Erkenntnis*, vol. 44, no. 2, pp. 167–197.

Somervuo, P. (2002). Speech modeling using variational bayesian mixture of gaussians. In: *Proc. ICSLP 2002*, pp. 1245–1248.

Taylor, D. (1987). *Hearing by Eye: The Psychology of Lip-Reading.*

Terry, L. and Katsaggelos, A. (2008). A phone-viseme dynamic bayesian network for audio-visual automatic speech recognition. pp. 1–4.

Valente, F. and Wellekens, C. (2003). Variational Bayesian GMM for Speech Recognition. In: *Eurospeech 2003, 8th european conference on speech communication and technology - September 1-4, 2003, Geneva, Switzerland.*

Voxforge (2006).
Available at: `http://www.voxforge.org/`

Wainwright, M.J. and Jordan, M.I. (2008). *Graphical Models, Exponential Families, and Variational Inference*, vol. 1 of *Foundations and Trends in Machine Learning*.

Watanabe, S., Minami, Y., Nakamura, A. and Ueda, N. (2003). Application of Variational Bayesian Approach to Speech Recognition. In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 1, pp. 568–571. MIT Press.

Wolpert, D.H. and Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82.

Yates, F. (1934). Contingency table involving small numbers and the Xi-squared test. *Statistical methods for rates and proportions (2nd ed.)*, vol. 1, no. 2, pp. 217–235.

Yuhas, B., Goldstein, M.H., J. and Sejnowski, T. (1989). Integration of acoustic and visual speech signals using neural networks. *Communications Magazine, IEEE*, vol. 27, no. 11, pp. 65 –71.