Modelling Speech Dynamics with Trajectory-HMMs

Le Zhang

Doctor of Philosophy

The Centre for Speech Technology Research
Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh

2009

Abstract

The conditional independence assumption imposed by the hidden Markov models (HMMs) makes it difficult to model temporal correlation patterns in human speech. Traditionally, this limitation is circumvented by appending the first and second-order regression coefficients to the observation feature vectors. Although this leads to improved performance in recognition tasks, we argue that a straightforward use of dynamic features in HMMs will result in an inferior model, due to the incorrect handling of dynamic constraints. In this thesis I will show that an HMM can be transformed into a Trajectory-HMM capable of generating smoothed output mean trajectories, by performing a per-utterance normalisation. The resulting model can be trained by either maximising model log-likelihood or minimising mean generation errors on the training data. To combat the exponential growth of paths in searching, the idea of delayed path merging is proposed and a new time-synchronous decoding algorithm built on the concept of token-passing is designed for use in the recognition task. The Trajectory-HMM brings a new way of sharing knowledge between speech recognition and synthesis components, by tackling both problems in a coherent statistical framework. I evaluated the Trajectory-HMM on two different speech tasks using the speaker-dependent MOCHA-TIMIT database. First as a generative model to recover articulatory features from speech signal, where the Trajectory-HMM was used in a complementary way to the conventional HMM modelling techniques, within a joint Acoustic-Articulatory framework. Experiments indicate that the jointly trained acoustic-articulatory models are more accurate (having a lower Root Mean Square error) than the separately trained ones, and that Trajectory-HMM training results in greater accuracy compared with conventional Baum-Welch parameter updating. In addition, the Root Mean Square (RMS) training objective proves to be consistently better than the Maximum Likelihood objective. However, experiment of the phone recognition task shows that the MLE trained Trajectory-HMM, while retaining attractive properties of being a proper generative model, tends to favour over-smoothed trajectories among competing hypothesises, and does not perform better than a conventional HMM. We use this to build an argument that models giving a better fit on training data may suffer a reduction of discrimination by being too faithful to the training data. Finally, experiments on using triphone models show that increasing modelling detail is an effective way to leverage modelling performance with little added complexity in training.

Acknowledgement

“So what kind of research you’d like me to do in CSTR?”, I asked uneasily four years ago at our first meeting. The reply from Steve was “So, what kind of research you want to do in CSTR?”. It has been a privilege working with Steve Renals, whose skillfull supervision undoubtedly helps me all the way through. I am indebted to the various wonderful and brilliant people I worked with at CSTR. The discussion with Simon King has always been stimulating. I thank Joe Frankel for detail discussions on speech recognition, Korin Richmond for help in using the MOCHA-TIMIT data and articulatory modelling, and Hiroshi Shimodaira for general research discussion. Also thank Miles Osborne for being my second supervisor. I enjoyed his technical discussion as much as his humour.

Discussion with John Bridle initially raised my interest in the Trajectory-HMM. My work would not be possible without the help from Keiichi Tokuda and Heiga Zen of Nagoya Institute of Technology. I would like to thank Junichi Yamagishi for explaining HMM-based synthesis, Chris Williams for answering my questions on probabilistic modelling and Qin Chao for discussion on the details of the articulatory inversion experiment.

I am especially grateful for Heriberto Cuayáhuitl, Sabrina Pei-Yun Hsueh and Ivan Meza-Ruiz, with whom we ran a monthly progress meeting in the last three years. Thank you for always being there. In addition, I want to thank Tim Mills, Fiona Couper Kenney, Yoshinori Shiga, Gabriel Murray, Peter Bell, Songfang Huang and Leonardo Badino for sharing an office with me. I enjoy the random chat, as well as the more technical ones, with you.

The informatics computer support term does an excellent job and all I can say is that I am impressed! The supporting work by Avril Heron really made my study life a lot easier.

While not directly related to my work reported here, I am equally grateful for friends from local churches, for their hospitality, love and encouragement in the highs and lows of my PhD. I am particularly grateful to Elizabeth Macke, Bob Akroyd and Catriona Macdonald from Buccleuch Free Church; Maurie Sween, Xylia, Cai Qiong, Lin Yiling, Yuan Yuan and Samuel from the fellowship of Mayfield Chinese Church; Dave Hill, Dave Thresher, John Matthews and Ritsia Thomas from Kings Church Edinburgh.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Le Zhang*)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The book “Computer Speech Processing” (Fallside and Woods, 1985) published some 23 years ago begins with the following: “Research in speech recognition and synthesis is one of the more fascinating areas of research today, both as an investigation into human abilities and as an emerging engineering discipline”. Indeed, looking back at the last 20 years history of speech processing research, one will be surprised by how much progress has been made. Modern automatic speech recognition (ASR) systems based on hidden Markov models (HMMs), coupled with advanced acoustic modelling and speaker adaptation techniques, have achieved word error rates as low as 10% on speaker independent broadcast news recognition tasks (Liu et al., 2005). In speech synthesis, sophisticated concatenated and model-based systems are able to produce intelligible, natural-sounding speech that can be easily adapted to different voices (Black and Taylor, 1997; Yamagishi and Kobayashi, 2007).

Despite these advances, many of the basic research principles employed in speech research have remained unchanged. In particular, the two core problems of speech processing, namely speech recognition and synthesis, have largely been tackled differently and each has its own dedicated framework designed and developed over the years. In speech recognition, classes of sub-word speech units are modelled by statistical models such as HMMs with model parameters estimated from speech data. In speech synthesis, optimisation models are now used to produce a realisation of speech by minimising some cost functions over the concatenated sub-word modelling units. The similarity between the techniques used in today’s speech recognition and synthesis suggests that it might be possible to integrate the two problems into a unified statistical framework. The benefit of such an integration would include a deeper knowledge sharing between the recognition and synthesis components, and a computational model

closer to the human speech skill acquisition process that involves the “simultaneous” learning of speech perception and of speech production (Fallside, 1992).

The idea of integrating speech analysis and synthesis is not new. This principle was embraced by the developer of the very first vocoder back in the 1930s (Dudley, 1939). The theory behind the vocoder is to analyse the recorded speech to produce a series of parameters that models the changes of voice over time, such as the fundamental frequency. In so doing, the speech was compressed into a form that requires much less storage space. In the synthesis stage, the process was reversed and the fundamental frequency was created in an oscillator to reconstruct an approximation to the original waveform. Later attempts to bridge the two problems together include speech recognition using synthesis by rule (Bridle and Ralls, 1985) and acquisition of speech by machines (ASM) (Fallside, 1992). These efforts, while not tremendously successful in their day, indicate a way of integrating speech knowledge for both recogniser and synthesiser.

A technical difficulty in designing a unified framework for both speech recognition and synthesis is the lack of a coherent statistical model that can be used for both problems. HMMs that have been successfully used in ASR as a recognition component, were found to be inadequate for use as the generative output component in a model-based synthesis system. The mean output vectors from an HMM have a stepwise trajectory due to the piecewise stationarity and conditional independence assumptions of the model. The consequence is that the output distribution will be the same over time as long as the underlying HMM state remains unchanged.

Recent advances in HMM-based speech synthesis bring promise in using HMMs as a unified modelling tool for both recognition and synthesis. Tokuda et. al. (1995) discovered a way of producing a smoothed, continuous mean trajectory from a given HMM state sequence using Gaussian outputs. The idea is to use dynamic features, which are usually the first and second order regression coefficients computed from the static observation vectors, to impose constraints on the output mean trajectory. This discovery ultimately led to the development of the Trajectory-HMM (Tokuda et al., 2004), which can be seen as an HMM with an augmented input, normalised over the original static observation space. As a proper generative model, the Trajectory-HMM overcomes a major weakness of using HMMs in text-to-speech (TTS) synthesis, namely the conditional independence assumption between state outputs. Using HMMs in model-based speech synthesis has some advantages compared to other approaches, such as the Unit Selection method (Clark et al., 2007). The whole system is

parametrised rather than consisting of stored pre-recorded waveforms, and therefore is *trainable*. Moreover, HMMs provide a much more compact representation compared to the concatenated approach, and are often an order of magnitude smaller. Finally, the speech produced from an HMM-TTS system has a overall lower variance than Unit Selection, and the style of the voice can be controlled parametrically (Yamagishi and Kobayashi, 2007).

The Trajectory-HMM is still in an early stage of development and researchers have not fully understood its many properties. For example, training a Trajectory-HMM is computationally much more expensive than Baum-Welch training of HMMs. Although researchers have tried fitting a Trajectory-HMM using Maximum Likelihood criterion (Tokuda et al., 2004), it may be beneficial to explore other parameter estimation criterion. Another current difficulty in applying Trajectory-HMM to wider speech applications is the lack of an decoding algorithm to do the state inference task. Some have resorted to N-best list rescoring (Tokuda et al., 2004). However, it remains unclear what the advantage is of using a full scale Trajectory-HMM decoder. Beside application in speech synthesis, and some early experiments in ASR using "clean" data designed for TTS, so far there have been no attempts on other areas of speech processing using more realistic, and potentially "noisier", data. The premise for this research is to explore the possibility of using the Trajectory-HMM in a wider area of speech processing other than speech synthesis, and to have a better understanding of the relationship between the Trajectory-HMM and conventionally trained HMMs.

1.1 Contributions of the Thesis

In this thesis I try to answer some of the research questions regarding the use of the Trajectory-HMM in speech processing that have not been addressed before. Starting with the training, I analysed the effect of Maximum Likelihood Estimation (MLE) and proposed a discriminative training criterion based on the mean RMS generation error that delivers better performance in an articulatory inversion task. Following that, enhancement of the modelling details with sub-word units was studied and the use of triphone units was found to be an effective way of increasing the modelling power without resorting to expensive trajectory parameter update, although triphone models were also found to be more likely to suffer overfitting. Furthermore, the use of the Trajectory-HMM as a recognition model was also explored and a novel Trajectory Decoding algorithm was proposed to create a full scale trajectory decoder that is capable

of performing phone recognition with grammar constraints, a task previously impossible with N-best list rescoring. On the application side, a joint acoustic-articulatory HMM modelling method was created to estimate human articulator movements from acoustic signal. Each of the main contributions is discussed in detail below:

1. **Analysis of Trajectory-HMM Training Methods**.
   Although Baum-Welch trained parameters have been found to be effective in trajectory parameter generation algorithm (Tokuda et al., 1995), to take the full advantage of the Trajectory-HMM, the model parameters need to be trajectory updated accordingly (Tokuda et al., 2004). In this thesis I investigate the improvement of trajectory trained models compared to Baum-Welch trained ones. Furthermore, there exist different ways of updating model parameters, albeit with varied levels of computational complexity. One can choose to update mean parameters only, variance parameters only, or both. Through experiment I expect to reveal the strength and weakness of different approaches. In addition, an alternative training criterion that seeks to minimise Root Mean Square (RMS) mean generation error on training data was proposed and compared to the Maximum Likelihood criterion.

2. **New Trajectory Decoding Algorithm**.
   In this thesis I introduce a new decoding algorithm tailored for decoding Trajectory-HMMs. The algorithm extends the "token-passing" concept for decoding HMMs and uses a delayed decision to do the decoding, as suggested in (Zen et al., 2004). This makes it possible to do full scale phone recognition using Trajectory-HMM. The decoding algorithm is benchmarked on a phone recognition task.

3. **Increase Modelling Details of Trajectory-HMM**s.
   All the reported experiments on Trajectory-HMMs are based on monophone models. Although in large scale HMM-based TTS system the basic modelling unit has been extended using contextual features (Zen et al., 2006), no attempt so far has tried trajectory training a Trajectory-HMM using sub-phone units, which potentially can bring better modelling power. In this thesis I will study the effect of trajectory training triphone HMMs compared to monophone ones on an acoustic-articulatory inversion task.

4. **Acoustic-Articulatory Inversion with Trajectory-HMMs.**
   Estimating the human articulator movements from acoustic signal has been an

active research topic for decades (Richmond, 2002). Unlike most attempts on this task where a direct mapping between the acoustic and articulatory signals was learnt, the proposed method centres around the idea of jointly optimising a single HMM model for both acoustic and articulatory channels. The result is a compact acoustic-articulatory HMM inversion model that delivers comparable performance with systems with much more free parameters.

1.2 Structure of Thesis

The structure of this thesis is as follows. Chapter 2 starts with the general speech modelling problem, it then proceeds to review representative statistical models for capturing speech dynamics. The chapter concludes by drawing a relationship between Trajectory-HMM and conventional HMM.

Chapter 3 presents the Trajectory-HMM framework used in this thesis in detail. It begins with the problem HMMs face when dynamic features are used. Following that, a mathematical treatment of the Trajectory-HMM is given, including the transformation of a normal HMM into a Trajectory model via a per-utterance normalisation. Training methods are then discussed, including both Maximum Likelihood training and the newly proposed Minimum RMS training. A decoding algorithm tailored for the Trajectory-HMM is presented in detail. Followed by the discussion on the issue of sampling from the model. Finally, the chapter ends with a demonstration on synthetic data using a one-state delta-chain system.

Chapter 4 puts the theory of Trajectory-HMM into practice, proposing an Acoustic-Articulatory modelling method for recovering the articulatory movement from speech signal. The task also serves as a testbed for analysing trajectory parameter update methods. The inversion problem is first reviewed. Then the methodology of the Acoustic-Articulatory modelling framework was presented. The experimental results using acoustic-articulatory parallel data are given together with the analysis of the effect of MLE training, the comparison between MLE and RMS trained models, and the difference between three parameter update methods.

Chapter 5 looks at the possibility of increasing the modelling power of Trajectory-HMM through the use of more detailed modelling units than monophones. To achieve this, a tree-based triphone clustering method was employed. Triphone systems with varying numbers of tied-states were created and evaluated on the articulatory inversion task. Experiments indicated that the use of triphone unit is an effective way of

improving Baum-Welch trained HMMs without resorting to expensive trajectory training. However, a trajectory updated triphone system was not found to perform much better than trajectory updated monophone model, due to problems related to overfitting.

Chapter 6 moves on to address the issue of using the Trajectory-HMM as a recognition component. It starts with a discussion of the difficulty of using Trajectory-HMM in speech recognition. Then it discusses the trade-off made in designing the trajectory decoding algorithm and some of the practical limitations of the trajectory decoder. Phone recognition experiments were conducted using two speakers from the corpus used. The general conclusion is that, while being a good generative model, in certain conditions Trajectory-HMM can become "too faithful" to the data, weakening the discrimination between competing hypothesis. As such, in our experiment a higher phone error rate was reported compared to a conventionally trained HMMs.

As a conclusion, the final chapter summarises the theoretical and experimental results of the thesis, and put the Trajectory-HMM into the broader perspective of HMM modelling: we use to think recognition and synthesis are radically different. As shown in this thesis, the two sides can be reconciled within the same HMM framework. And it suggests some new directions for further research.

Chapter 2

Background

This chapter reviews typical speech modelling methods for capturing the dynamic aspects of speech. It begins by placing the problem within the context of stochastic modelling. Then it proceeds to exam typical statistical speech models that represent the state-of-the-art. Over the years, a significant amount of efforts have been put to capture dynamic aspects of speech, such as coarticulation, which are difficult for a plain HMM to model.

2.1 Speech Modelling

Modelling speech statistically means to stochastically capture the relationship between an utterance represented as a sequence of words, $\mathbf{W}$, and its acoustic representation $\mathbf{A}$, subject to the available training data. The criterion for the target relationship varies from application to application. For Automatic Speech Recognition (ASR), we want to build a speech recogniser that will return the most probable word sequence $\widehat{\mathbf{W}}$ corresponding to $\mathbf{A}$:

$$\widehat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{A}) \tag{2.1}$$

For the Text to Speech (TTS) task, the goal is to create a speech synthesiser that will generate the most likely acoustic realisation $\widehat{\mathbf{A}}$, given the word sequence $\mathbf{W}$[1]:

$$\widehat{\mathbf{A}} = \arg\max_{\mathbf{A}} P(\mathbf{A} \mid \mathbf{W}) \tag{2.2}$$

[1] An actual TTS system could include other resources such as contextual and prosodic features in addition to the word sequence $\mathbf{W}$.

A typical speech application is usually divided into a number of probabilistic components. For example, using Bayes' rule, the right hand of (2.1) can be written as:

$$P(\mathbf{W} \mid \mathbf{A}) = \frac{p(\mathbf{A} \mid \mathbf{W}) \cdot P(\mathbf{W})}{p(\mathbf{A})} \tag{2.3}$$

here the denominator $p(\mathbf{A})$ in (2.3), the likelihood of the observation sequence, is a constant in the maximisation and can be dropped safely. The term $p(\mathbf{A} \mid \mathbf{W})$ represents the likelihood that the acoustic evidence $\mathbf{A}$ can be generated from the word sequence $\mathbf{W}$, and is usually referred to as the *acoustic model*. $P(\mathbf{W})$ is the probability of the word sequence $\mathbf{W}$ according to some linguistically meanfully criteria, and is often called a *language model*. In addition, as a stochastic model, all the model parameters, be they for $p(\mathbf{A} \mid \mathbf{W})$ or $P(\mathbf{W})$, can be statistically derived from the available training data. As a trainable system, the parameters can be obtained in such a way that the task's objective functions are optimised on the training data. As a result, the system can be automatically ported across domains if additional training data is available. The rest of this review will focus on the acoustic modelling aspect of statistical speech modelling.

2.2 Acoustic Models

2.2.1 Hidden Markov Models

For decades, the mainstream acoustic models in ASR have been dominated by the hidden Markov model (HMM) and its variants (Bahl et al., 1983; Poritz, 1988; Rabiner, 1989). As its name suggests, the component parts from an HMM is a Markov chain consisting of a set of hidden discrete states and their transition probabilities. Some of the states are associated with a probability density function (PDF) and are collectively called the emitting states. Others do not have an output function and serve as connecting points between HMM units, such as the non-emitting entry and exit states. The discrete state sequence of an HMM is hidden and the output is modelled by the corresponding PDFs associated with each emitting state. In a typical HMM-based ASR system, the speech signal is parametrised into sequence of frames, each of which is represented as a low-dimensional vector.

The dimensionality of observation vector of frame t, $\mathbf{o}_t$, is often 39 in speech recognition, which typically consists of the 12^{th} order Mel frequency cepstral coefficients (MFCCs) or Perceptual linear prediction (PLP) features plus log energy and their first

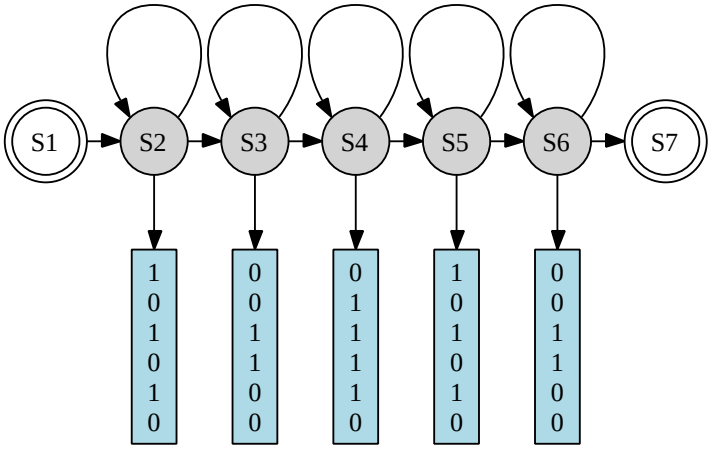

Figure 2.1: A left-to-right HMM with 7 states, the first and last of which are non-emitting.

and second-order temporal derivatives. For an HMM with N states, the parameters can be defined as:

- set of transition probabilities $a_{ij} = P(q_t = j \mid q_{t-1} = i)$, the probability of moving from state i to state j.
- the parameters of the output probability density function $b_j(\mathbf{o}_t) = p(\mathbf{o}_t \mid q_t = j)$, associated with the emitting state j.

The transition probabilities must satisfy $\Sigma_j^N a_{ij} = 1, \forall i$. The probability density function for state j at time t, $b_j(\mathbf{o}_t)$, is assumed to be conditionally independent of all other observations given the discrete state $q_t = j$. Figure 2.1 illustrates such an HMM where only emitting states can generate output.

A common choice for the state output function is the multivariate Gaussian distribution whose density function is given by:

$$b_j(\mathbf{o}_t) = \mathcal{N}(\mathbf{o}_t \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \tag{2.4}$$

$$= \frac{1}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}_j|}} \exp\left\{ -\frac{1}{2} (\mathbf{o}_t - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j (\mathbf{o}_t - \boldsymbol{\mu}_j) \right\} \tag{2.5}$$

where $\boldsymbol{\Sigma}_j$ and $\boldsymbol{\mu}_j$ are the covariance matrix and mean vector for the Gaussian distribution of state j, and p is the dimensionality of the observation vector.

A Gaussian distribution $\mathcal{N}(\mathbf{o}_t \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ has only one mode at the mean $\boldsymbol{\mu}_j$ and is insufficient to model the acoustic variability in real world speech. Gaussian mixture models (GMMs) are often used instead of single Gaussian densities to provide a richer modelling capacity. The probability density function of a GMM is defined as:

$$b_j(\mathbf{o}_t) = \sum_{m=1}^{M} C_{jm} \mathcal{N}(\mathbf{o}_t \mid \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \tag{2.6}$$

where M is the number of Gaussian components. C_{jn} are the mixture weights and $\boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}$ are the mean and covariance matrix for the m^{th} mixture component. To make $b_j(\mathbf{o}_t)$ a valid probability density function, the mixture weights must satisfy $\sum_{m=1}^{M} C_{jm} = 1$. The covariance matrix can take various forms, for example diagonal, block-diagonal or full covariance matrix (Gales, 1999). For a p-dimensional Gaussian distribution, a diagonal covariance matrix has p parameters, and a full covariance matrix has $p(p+1)/2$ parameters. The limited amount of training data makes robust estimation of a full covariance matrix in large HMM system a hard task. Although some work has been done to reduce the number of parameters of the Gaussian covariance matrix (Axelrod et al., 2002; Olsen and Gopinath, 2004; Bell and King, 2007), it is more common to use diagonal covariance matrices in a GMM setting.

The parameters of an HMM can be estimated from training data via Maximum Likelihood Estimation as:

$$\hat{\lambda} = \arg\max_{\lambda} p(\mathbf{o} \mid \lambda) = \arg\max_{\lambda} \log p(\mathbf{o} \mid \lambda) \tag{2.7}$$

Here the logarithm of the likelihood function is used for mathematical convenience. The estimation would be straightforward if the speech frame/state alignment of the training data were known. However, as the state sequence is hidden, there is no closed form solution. A well-established technique for estimating HMM parameters from an observation sequence with an unknown state sequence is the Baum-Welch algorithm (Baum and Petrie, 1966), which is an instance of the expectation maximisation (EM) algorithm that fits the model parameters to the data iteratively until a local maximum is reached (Dempster et al., 1977). The algorithm uses the so-called *forward-backward* procedure to find the discrete state posterior probabilities and the state conditional densities given the observation sequence and the current set of parameters $\lambda^{(t)}$. In so doing, the forward variable, $\alpha_j(t)$, is defined and calculated as the joint likelihood of the observation sequence up to frame t and being in state j at frame t; and the

backward variable, $\beta_i(t)$, is defined and calculated as the posterior likelihood of the partial observation sequence from frame $t+1$ to T given being in state j at frame t. Once the alpha and beta variables have been collected, a set of new parameters $\lambda^{(t+1)}$ can be estimated from $\lambda^{(t)}$. The process is iterated a few times until the likelihood of the model on the training data converges to a local maxima.

During recognition, input speech is first parametrised into a sequence of feature vectors by the signal processing module, which are then mapped into possible sequences of HMM states using the Viterbi search, a dynamic programming algorithm for finding the most likely state sequence, and returns the joint likelihood of the observation sequence given the model. A language model can be employed in the decoding process to help constrain the selection to a linguistically meaningful state sequence.

In an HMM the basic modelling units (usually phonemes or context-dependent phonemes) are modelled by a sequence of hidden states (typically 2-5 states per unit), and the observed acoustic feature vector at each frame $\mathbf{o}_t$ is assumed to be independently generated by the hidden state at that frame q_t. In addition, different levels of HMMs, like word level HMMs and phone level HMMs, can be connected via null transitions to form a composite HMM. Then the same training and decoding algorithms can be applied to this composite HMM with little modification.

There exist many effective techniques to improve the performance of an HMM-based ASR system including: the use of dynamic (delta) features (Furui, 1986) and context-dependent phoneme units to model short-term dependencies, applying decision tree-based parameter tying scheme to overcome data sparseness problem (Young and Woodland, 1994), explicit duration modelling (Hochberg, 1992), training HMMs discriminatively based on Maximum Mutual Information Estimation (MMIE) or Minimum Classification Error (MCE) criteria (Bahl et al., 1986; Valtchev et al., 1997), and Maximum Likelihood Linear Regression (MLLR) speaker adaptation (Woodland and Leggetter, 1994).

After decades of experiment and theoretical development, the performance of HMM-based ASR system has been greatly improved. Yet, a current state-of-the-art HMM-based speech recogniser still has a word error rate as high as 20.7% on an English telephone speech transcription task (Evermann et al., 2004). This suggests that some inherent deficiencies of the HMM modelling paradigm may prevent us from going further. More specifically, the following three assumptions made by basic HMMs may be inappropriate for modelling speech patterns (Holmes and Russell, 1999):

1. *piecewise stationarity.* Speech is split into small segments associated with dis-

crete states, with instantaneous transitions between those states.

2. *conditional independence*. The acoustic observation of each frame is modelled independently given the discrete state of that frame. Beyond that, there is no way to model correlation between adjacent frames in an HMM.

3. *state duration distribution*. The state duration distribution of an HMM is a simple geometric distribution, as a consequence of the "self-loop" transition probability. Sophistated pronunciation models are required for more complex duration modelling.

Although none of these assumptions is true for real speech, the independence assumption is regarded by many to be the major drawback of the use of HMM in speech recognition (Ostendorf et al., 1996). It is known that in human speech the same phoneme can be pronounced differently according to its surrounding phoneme context to ensure a smooth transition between syllables. This phenomenon, also called coarticulation in phonology, leads to strong correlations between adjacent speech segments and is difficult to model with an HMM.

Although developed as a generative model, the HMM is not suitable for generating a smooth, time-varying trajectory as required in certain applications such as speech synthesis. Without special treatment (as will be discussed in detail in chapter 3), the mean output trajectory from an HMM will be constant within a state and will jump instantly between state boundaries, which is clearly a very poor approximation of observed speech trajectories. This is illustrated in figure 2.2.

Partly motivated by these insights, many attempts have been made to develop alternative acoustic models that can better handle the dynamic character of speech. A significant motivation of some of those models is to relax the independence assumption, which is believed to be the main weakness of HMMs.

2.2.2 Hybrid Connectionist/Markov Model Approach

One successful alternative approach to HMMs is to build acoustic models around Artificial Neural Networks (ANNs), which is usually referred to as *connectionist speech recognition*. The idea behind the so-called hybrid HMM/ANN systems is to learn directly a non-linear mapping from sequences of parametrised speech into sequences of phone/word labels associated with those frames using a neural network, while main-

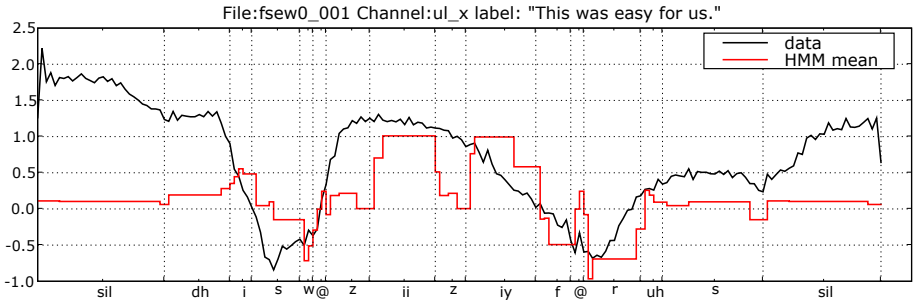

Figure 2.2: Example of HMM stepwise mean output trajectory of the 1st MFCC feature for one recorded utterance. With in each phone, the mean of the Gaussian state output distribution (red line) remains the same as long as the underlying HMM state is unchanged, which is clearly wrong compared to the real data (black line).

taining the underlying Markov network structure (Bourlard and Morgan, 1993; Robinson, 1994).

A typical hybrid ANN system trains multiple-layer perceptrons (MLPs) over a sequence of acoustic observations in order to estimate a phone class posterior probability $p(q_k \mid \mathbf{o}_{t-L}^{t+L})$, where q_k is a given state corresponding to a phone class, and $\mathbf{o}_{t-L}^{t+L}$ denotes parametrised speech over a window of $(2L+1)$ frames. The class posterior probability $p(q_k \mid \mathbf{o}_{t-L}^{t+L})$ can be divided by $p(q_k)$ to get a scaled likelihood which can be used in a standard viterbi style decoder in place of the likelihood score of Gaussian mixtures. Figure 2.3 illustrates the hybrid ANN framework.

Unlike a GMM-based HMM system that does not make efficient use of inter-frame contextual information, phone posteriors estimated by MLPs can incorporate correlations that span several frames. Indeed, if a recurrent neural network (RNN) is used, the contextual information from the first frame to several frames ahead of the current frame can be incorporated [2]. Moreover, it can be shown that any linear transformation may be built into the first layer of an MLP by modifying the weights before the non-linearity.

Hybrid ANN systems have shown significantly better performance than similar HMM systems on phone recognition tasks. The recurrent net based system that uses

[2] In a typical system using a nine-frame context window e.g. (Robinson et al., 2002), the actual phone *posterior* is computed as $p(q_k \mid \mathbf{o}_1^{t+4})$, where $\mathbf{o}_1^{t+4}$ is the observation vector sequence up to frame $t+4$.

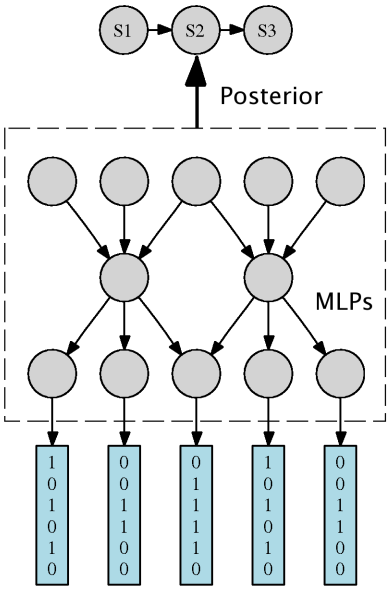

Figure 2.3: Hybrid ANN system: phone class posterior is estimated by MLPs

only monophone models as reported by (Robinson, 1994) has much lower phone classification error rate on TIMIT database compared to the standard HMM (Lamel et al., 1986). One might attribute the success of using ANNs in ASR to the discriminative nature of the system, as the learnt posterior distribution $p(q_k \mid \mathbf{o}_{t-L}^{t+L})$ may carry important discriminative information that is useful for ASR but is not usually available in a plain HMM system. This has been verified in the Tandem approach where appending the phone posterior feature to the usual acoustic features clearly improve the performance of the baseline HMM system (Hermansky et al., 2000). That said, ANN modelling approach has not been very successful using context-dependent units. Another drawback of using ANN in speech recognition is the difficulty of incorporating adaption technique into the ANN framework, which makes it less attractive than HMMs.

2.2.3 Dynamic Bayesian Networks

Many probabilistic models, such as HMMs and Kalman smoothers, may be regarded as special instances of a class of general graphical models called Dynamic Bayesian Networks (DBNs) (Murphy, 2002). A DBN is basically a Bayesian Network unwrapped in time, where the network nodes represent random variables and the edges between nodes indicate conditional dependence assumptions.

A distinguishing feature of DBNs is that there exist general inference algorithms

independent from the underlying network structure. This means that a more flexible state space compared to an HMM can be employed by the modeller without worrying about the inference algorithms. Exact inference of DBNs can be carried out by passing messages in a junction-tree transformed from the original network (Pearl, 1988). For problems that are intractable, approximate inference is also possible using variational or stochastic sampling methods. Moreover, general-purpose software such as the Graphical Models Toolkit (GMTK) [3] can be used to build a complete DBN-based ASR system.

Compared to an HMM-based system where many constraints such as tied parameters and skip-silence training scheme are implemented implicitly in the software, every constraint in a DBN must be modelled explicitly using graphical constraints (Bilmes et al., 2001). Therefore it is difficult to build a DBN-based system that is as efficient as an equivalent HMM system. However, the flexibility of DBNs makes it possible to explicitly model long-term articulatory and acoustic context as the conditional probability distributions between factored states in a network, which is difficult or impossible to model using an HMM.

DBN-based acoustic models are currently actively pursued as promising alternatives to conventional HMMs (Zweig and Russell, 1998; Livescu et al., 2003; Frankel et al., 2007).

2.2.4 Segment Models

Segment Modelling (Russell and Moore, 1985; Russell, 1993; Levinson, 1986; Wellekens, 1987; Kenny et al., l990; Gales and Young, 1993) is a general framework to address some of the limitations of HMMs. Unlike HMMs where each state can only generate one observation per frame, in a segment model a variable-length sequence of observations can be generated from one segment (corresponding to one HMM state). In a segment model the joint likelihood of a sequence of l observations $\mathbf{o} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_l\}$ and a unit a may be written as (Ostendorf et al., 1996):

$$p(\mathbf{o}, l \mid a) = p(\mathbf{o} \mid l, a)p(l \mid a) \tag{2.8}$$

where $p(l \mid a)$ is a duration model that gives the likelihood of length l given a, $p(\mathbf{o} \mid l, a)$ represents a family of output densities that describe the observation sequence of dif-

[3] The Graphical Models Toolkit (GMTK), available from http://ssli.ee.washington.edu/~bilmes/gmtk/.

ferent lengths, and does not necessary have the conditional independence assumption between observation sequences from the same state. This can be shown in fig 2.4.

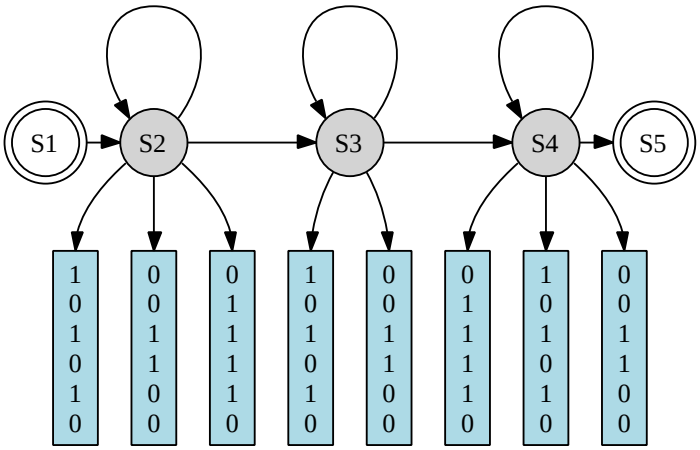

Figure 2.4: Illustration of Segment Model.

Since the basic modelling unit in a segment model is a variable-length observation sequence, there are many opportunities to model short-term temporal correlations within frames belonging to the same segment, by creating a collection of distribution mappings that maps each frame $\mathbf{o}_i$ in the segment to a particular output distribution. Depending on the distribution assumption chosen by the modeller, there are several different ways to model feature dynamics within a segment. Typical approaches include Constrainted Mean Trajectory (Ostendorf et al., 1996), Conditional Gaussian Models (Wellekens, 1987; Kenny et al., l990), and Linear Dynamic Models (Digalakis et al., 1991). Training and decoding schemes for general segment models is similar to that used for HMM's and can be found in (Ostendorf et al., 1996).

Duration modelling is addressed in segment models by the duration distribution $p(l \mid a)$, which can be Poisson distribution (Russell and Moore, 1985), Gamma distribution (Levinson, 1986), context-dependent clustered Gamma models, or simply smoothed relative frequencies (Ostendorf et al., 1992). It has been found that sophisticated duration model can indeed bring a small improvement to ASR (Hochberg, 1992).

In practice, the added complexity and computational cost for parameter estimation and decoding of the segment model are not strongly justified by its performance over conventional HMMs (Holmes and Russell, 1999). Moreover, as a generative model designed for ASR task, few attempts have been made in other speech applications like model-based TTS, although legally it is certainly capable.

2.2.5 Hidden Dynamic/Trajectory Model

The Hidden Dynamic Model (HDM) is an attempt to model human speech dynamics from a speech production point of view (Richards and Bridle, 1999; Bridle et al., 1998). HDMs assume that the hidden dynamics behind coarticulation and the transition between neighbouring phones can be learnt through a non-linear mapping from the space in which the dynamic occurs to the surface representation of speech (acoustic observations).

Speech patterns can be generated from an HDM in a way similar to a classic speech-synthesis-by-rule system (Holmes et al., 1964). First, the input phone label sequences are mapped to some target values (the dimension is usually between 4 to 8), which are then passed through a low-pass filter to get a hidden representation and finally mapped to the surface acoustic form using a non-linear mapping (usually multiple-layer perceptrons (MLPs)). The overall speech pattern generation process is illustrated in figure 2.5.

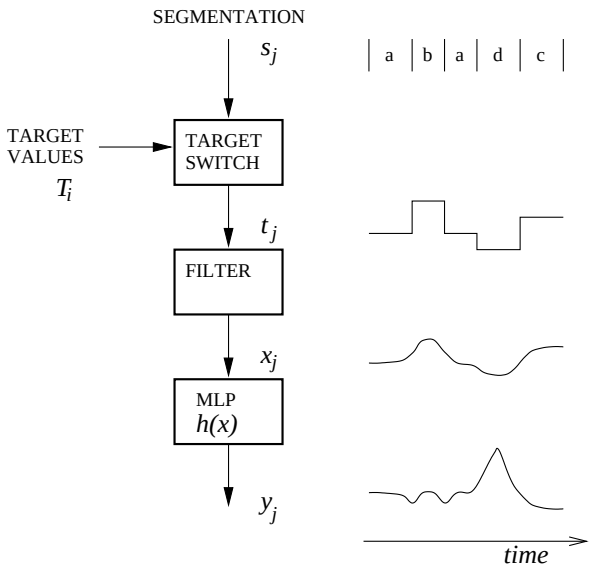

Figure 2.5: The HDM as a speech pattern generator. (reproduced with permission from (Bridle et al., 1998, page3))

Two variants of HDM were studied during the 1998 workshop at Johns Hopkins University [4], the Deterministic Hidden Dynamic Model (DHDM) and the Vocal Tract Resonances (VTR) model (Bridle et al., 1998). The main difference between the

[4]Dynamic segmental models of speech coarticulation: http://www.clsp.jhu.edu/ws98/projects/dynamic/.

two is how the hidden space is represented. DHDM uses a deterministic phone-to-target mapping to map phones to (unconstrained) hidden states, while VTR chooses formant-like resonances as its internal model. The non-linear mapping in both systems are MLPs whose parameters can be trained using gradient descent method with back-propagating.

A trained HDM can be used to rescore N-best list outputted from a conventional HMM system. The scoring criterion can be the distance between the synthesised speech pattern produced by the HDM for each phone sequence and the real acoustic observation, as shown in figure 2.6:

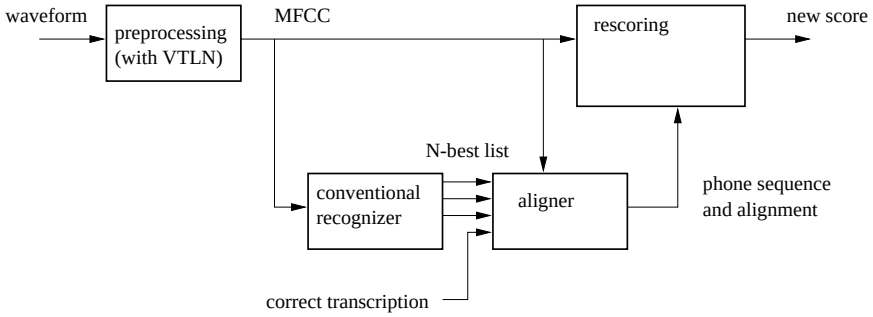

Figure 2.6: Rescoring the output from a conventional recogniser using HDM (reproduced with permission from (Bridle et al., 1998, page 21))

Picone et al. (1999) reports experimental results of HDM applied to the switchboard corpus. The HDM was found to perform well when the reference transcription is included in the rescoring N-best list, but was slightly inferior to a conventional HMM system when not exposed to the reference transcription. In general, the performance of HDM on this data set is unconvincing, especially considering the computational complexity of such model.

The Vocal Tract Resonance variant of HDM has received further development over the years and grows into the Hidden Trajectory Model (HTM) (Deng et al., 2006). In an HTM the speech dynamics (coarticulation and phonetic reduction) is represented implicitly by temporal filtering of Vocal Tract Resonane (VTR) targets, the hidden dynamics of which is capable of capturing long-span context-dependent properties in acoustic features. This suggests that the HTM can model not only cross-frame correlation, but also cross-unit ones.

As a generative model, HTM connects its HTM phone sequence, and the observed

cepstral trajectory via a layer of stochastic VTR targets, which are modelled by distributions from multivariate Gaussian family. The likelihood score of an utterance is computed by integrating out the hidden VTR target variables. Because the likelihood of each observation frame depends on a set of neighbouring frames, and because the segment likelihood can be computed only when all phone boundaries are available, decoding an HTM model is much more difficult than decoding an HMM. Yu et al. (2006) present an A^* time-asynchronous lattice-constrained decoding algorithm. When processing the lattice output from an HMM system, a significant improvement is observed on the TIMIT phone recognition task. Analysis shows the improvements are most significant in the sonorant class (vowels, semivowels, nasals), while there is no improvement in the fricative-consonant class (Deng et al., 2006), which suggests that the target-filtering component of the HTM has been better designed than the acoustic residual component where a single Gaussian per-state is used for the modelling.

2.2.6 Linear Dynamic Model

The Linear Dynamic Model (LDM), also called a Kalman smoother, or a continuous state linear Gaussian model, is yet another approach to model hidden dynamics in speech. As a Kalman model, the LDM is described by a pair of equations (Frankel, 2003; Rosti, 2004):

$$\mathbf{x}_t = \mathbf{F}_t \mathbf{x}_{t-1} + \mathbf{w}_t, \qquad \mathbf{w}_t \sim \mathcal{N}(\mu_w, \sigma_w^2) \tag{2.9}$$

$$\mathbf{y}_t = \mathbf{H_t} \mathbf{x}_t + \mathbf{v}_t, \qquad \mathbf{v}_t \sim \mathcal{N}(\mu_v, \sigma_v^2) \tag{2.10}$$

where $\mathbf{x}_t$ denotes the hidden, continuous state variable of the system at time t, which evolves from one frame to the next via the state transition matrix $\mathbf{F}_t$ and the addition of some Gaussian noise $\mathbf{w}_t$. The output of the system, $\mathbf{y}_t$, is obtained by a linear transformation made through the matrix $\mathbf{H}_t$ and the addition of the Gaussian noise $\mathbf{v}_t$.

The LDM can be seen as the continuous version of discrete-state HMMs where the discrete hidden states are replaced with Gaussian continuous variables (Minka, 1998). The dimensionality of the hidden state variable controls the motion complexity an LDM can model. Moreover, the parameters of an LDM can be learnt using Expectation Maximisation (EM) algorithm in a way similar to the iterative training of HMMs (Ghahramani and Hinton, 1996).

Linear Dynamic Models are suitable to model smoothly varying (but noisy) trajectories, such as the motion of articulators, although attempts have been made to model

features from the acoustic domain directly. Frankel et al. (2000) report a phone recognition system based on the combination of Linear Dynamic Models and Neural Networks. The system is unique in the way it uses articulatory features. First, articulatory features are extracted from the acoustic domain using the non-linear mapping provided by a recurrent neural network. Then an LDM is applied to the articulatory traces to estimate phone posterior probability, which is then rescored by a bigram language model to produce final recognition result.

The recognition result of using LDM on phone recognition is comparable but not significantly superior to a standard HMM system, although as a generative model it can recover spectrum data from a speech signal well.

2.2.7 Trajectory-HMM

Enhancing ASR performance through the use of dynamic features has been a standard practice since 1980s. However, training a conventional HMM directly on the augmented feature vectors will result in an inconsistent model, as the use of dynamic features effectively introduces inter-frame dependencies that an HMM can not handle. This is especially problematic for applications that demand the generation of smoothed output trajectory, as is the case for speech synthesis. As an effort to use dynamic features in HMM-based speech synthesis, a parameter generation algorithm that can derive smoothed mean trajectory from an HMM subject to the maximal likelihood criterion was proposed by Tokuda et al. (2000). Later developments lead to the discovery of a new form of HMM that is a consistant generative model for both static and dynamic features by performing a per-utterance normalisation, which effectively integrating out the dynamic part of the output variables (Tokuda et al., 2004) (see section 3.2.2).

The resulting model is referred to as the Trajectory-HMM and has been successfully applied to model-based speech synthesis (Tokuda et al., 2000). In this case, during training both spectrum and F0 parameters are trained as a conventional context-independent (CI) HMM system. Then the CI models are expanded into context-dependent ones with linguistic context features. Then tree-based clustering is performed and state duration models are derived from the training data. In the synthesis stage, the word sequence is first expanded into context-dependent label sequences, which is then mapped into a concatenated HMM. A parameter generation algorithm (case 2 in (Tokuda et al., 2000)) that is equivalent to generating a smoothed mean trajectory is applied to ob-

tain the speech parameter sequence subject to a maximum likelihood criterion. Finally an MLSA (Mel Log Spectral Approximation) filter is applied to generate synthesised speech.

The main advantages of HMM-based TTS over the popular unit-selection technique are the compact model size, a trainable formula based around the HMM architecture, and a consistant quality of the synthesised speech. The footprint of a high-quality HMM-based TTS system is usually around a few MBs (Zen and Toda, 2005). Moreover, as all the parameters of the system can be derived from training data automatically, the system can be quickly ported to new voice and language domains. Being an extension to HMMs also means that the vast majority of an HMM code base can be reused and techniques like MLLR speaker adaptation can be applied to change the spectral and prosodic characteristics of the synthesised speech (Yamagishi et al., 2006).

The parameters of the Trajectory-HMM as used in today's HMM-based TTS system are all updated using the conventional Baum-Welch algorithm, largely due to the computational cost of the trajectory parameter update. Thus it will be interesting to ask how does a properly trained Trajectory-HMM compare with a Baum-Welch trained one, when used as a generative model for ASR. Moreover, the lack of a trajectory decoding algorithm, because of the complexity in searching, limits the use of Trajectory-HMM in speech recognition task. Although an initial attempt to use the Trajectory-HMM to rescore the N-best recognition result has been reported in (Zen et al., 2004), we have yet explored the various options like using a full-scale decoder in recognition. These issues will be examined in detail in chapter 3 and 6 of this thesis.

2.3 Discussion

There is no doubt that HMMs are still the most popular choice as acoustic models for speech recognition today. The HMM performs surprisingly well considering almost every assumption it makes conflicts with real speech observations, namely piecewise stationary within states, the framewise independence assumption on state output given current state, and simple geometric duration distribution (Holmes and Russell, 1999).

There are generally two different views in seeking an alternative model to HMMs (Bridle, 2004). The first view regards the piecewise stationary statistics an inherent deficiency of HMMs, and proposes to replace HMM with a new model that can explicitly model the inter-frame dependencies and hidden dynamics in speech. Models like

segmental HMMs, Hidden Dynamic Models and Hidden Trajectory Models have been pursued under this view. The second view, while acknowledging that the conventional HMM is incapable of dealing with inter-frame dependence, tries to fix this problem by explicitly modeling dynamic features in an HMM through a proper normalisation. This leads to the Trajectory-HMM studied in this work, which provides a principled way to overcome the limitation of HMMs through the use of dynamic features (Tokuda et al., 2004).

In this research I propose to model the dynamic patterns of speech using a Trajectory-HMM (Tokuda et al., 2004), which is a properly normalised version of HMM that models feature derivatives explicitly without imposing any conditional independence assumption. The use of Trajectory-HMMs in speech recognition is still in its early stages (Tokuda et al., 2004) although this model has been successfully applied to speech synthesis (Tokuda et al., 2000).

In the rest of this thesis, I will first establish the mathematical framework of Trajectory-HMM in chapter 3, covering the model structure, the parameter training algorithm, and the decoding strategy. The effectiveness of a properly trained Trajectory-HMM as a proper generative model will first be evaluated on an articulatory feature inversion task in chapter 4, using a speaker-dependent acoustic-articulatory parallel data. In chapter 5, I will explore the possibility of increasing modelling details using context-dependent triphone modelling units. After that, in chapter 6 I will look into the issue of using Trajectory-HMM as the acoustic model for speech recognition. A full-scale decoder (see section 3.4) will be employed to derive the decoding result.

Chapter 3

The Trajectory-HMM Framework

This chapter presents the theoretical framework for speech modelling with Trajectory-HMMs. First, I will discuss the problem of using dynamic features directly in the context of a conventional HMM and how this is addressed in the Trajectory-HMM. Based on the work of Tokuda et al. (2004), I then give the parametric form of the Trajectory-HMM in section 3.2, and discuss different training options in section 3.3. As an extension to Zen et al. (2004), a novel decoding algorithm based on the concept of Token-passing is presented in section 3.4. Sampling from the Trajectory-HMM is covered in section 3.5. Finally in section 3.6 the usefulness of the Trajectory-HMM will be demonstrated on a synthetic task.

3.1 Limitation of HMMs?

As mentioned in chapter 2, a conventional HMM assumes strong conditional independence over the state observations, which introduces two kinds of problems when dealing with real speech patterns:

1. Stepwise mean trajectory
   Because the output distribution of each discrete HMM state is modelled independently, the mean trajectory of an HMM will be the same as long as the underlying state does not change, and will have an instantaneous transition between state boundaries. This is clearly inappropriate for real speech data.

2. Inconsistency with dynamic features
   A simple method to enhance HMM's modelling power is to augment the original acoustic feature vector with dynamic features computed as a linear trans-

formation of several adjacent acoustic feature vectors (Furui, 1986). The use of dynamic features effectively introduces the intra-frame dependence into the augmented observation vectors. Unfortunately the conventional HMM can not take full advantage of the dynamic features. Training an HMM directly on the augmented feature vectors will result in an inconsistent model: on the one hand, the use of dynamic feature imposes certain constraints on the acoustic observations; on the other hand, the independence assumption of HMM instructs the model to ignore the correlation between static observations and model them independently. Empirically, systems trained this way perform much better than systems using only the original acoustic features for ASR and speaker recognition tasks. However, this improvement in performance is by no means the best we can hope for, since we are modelling the augmented observations using an inferior model.

These two problems may have some impacts on the performance of HMM when it comes to speech modelling. For example, in speech recognition the stepwise mean output trajectory from a conventional HMM can be a bad approximation to the real feature trajectory, and is likely a major cause of higher Word Error Rate (WER) in ASR task. Moreover, in HMM-based speech synthesis, direct use of the HMM mean trajectory will cause the generated speech to sound discountinous (Masuko et al., 1996). The conventional HMM can never be used as a proper generative model as long as these issues remain unfixed.

3.1.1 Handling Dynamic Features

Recently a theoretical framework called Trajectory-HMM has been set up to give an interpretation of the dynamic feature in HMM (Tokuda et al., 2004; Williams, 2005). The first step in deriving a Trajectory-HMM is to model the likelihood function that is used for model comparison as a function of $\mathbf{c}$, the static feature vectors, rather than as a function of $\mathbf{o}$, the augmented feature vectors. Since $\mathbf{o}$ is just a linear transformation of $\mathbf{c}$, it is fully determined once $\mathbf{c}$ is known. The next, more crucial step in fixing the inconsistency problem of the conventional HMMs is to abandon the independence assumption. Without imposing any conditional independence assumption we can derive the likelihood function $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ by performing per state-sequence normalisation of the original likelihood function $p(\mathbf{o} \mid \mathbf{q}, \lambda)$:

$$p(\mathbf{c} \mid \mathbf{q}, \lambda) = \frac{1}{Z_{\mathbf{q}}} p(\mathbf{o} \mid \mathbf{q}, \lambda) \tag{3.1}$$

where $Z_{\mathbf{q}}$ is a normalisation term that depends on the whole state sequence $\mathbf{q}$.

In later sections we will see that the probabilistic density function $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ is in fact a Gaussian distribution[1], whose covariance matrix is no longer block-diagonal, as is the case of conventional HMMs. The almost-full covariance matrix of a Trajectory-HMM provides a means of capturing the inter-frame dependencies introduced with the dynamic features.

Intuitively, given a correctly trained Trajectory-HMM, and a static acoustic feature vector sequence, an HMM state sequence corresponding to the constraints in the observation will have a higher "trajectory likelihood" than those that do not respect the dynamic constraints present in the observation vectors. Therefore the trajectory likelihood $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ can be used to rescore the N-best hypothesis produced by a conventional HMM-based ASR recogniser. A more principled way of using a Trajectory-HMM is to apply a "trajectory decoder" directly on the acoustic input. Section 3.4 describes a decoding algorithm specifically tailored for the Trajectory-HMM. Experiments on a phone recognition task can be found in Chapter 6.

As a proper generative model taking account of the dynamic features, the Trajectory-HMM excels in speech synthesis (Tokuda et al., 2000), where the problem can be phrased as: given a state sequence corresponding to an utterance (concatenated from sub-word HMM models), and a properly trained Trajectory-HMM, generate speech parameters that obey dynamic constraints. Thus doing speech synthesis is equivalent to drawing samples from the underlying probabilistic model corresponding to an utterance.

The parameters of a Trajectory-HMM are exactly the same as those of a conventional HMM with the same topology. This is because the Trajectory-HMM only provides an alternative way of modelling dynamic features in HMMs, by introducing a proper per-sequence normalisation term without requiring additional parameters. This suggests that the bulk of the existing HMM codebase can be reused for a Trajectory-HMM. In contrast, other methods such as Hidden Dynamic Models or Segment Models require more substantial changes to the basic infrastructure.

A drawback of the Trajectory-HMM is the level of the complexity involved in

[1] Here we assume the Gaussian component corresponding to each state in $\mathbf{q}$ has been chosen.

both training and decoding. Since fewer conditional independence assumptions are imposed, more bookkeeping is needed in training and decoding (the complexity is sometimes an order of magnitude greater than algorithms for HMMs). We will address these problems in later sections.

3.2 The Trajectory-HMM Framework

3.2.1 Preliminaries

Let $\mathbf{c} = [\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_T^T]^T$ denote a sequence of T acoustic feature vectors used in an HMM-based ASR system. $\mathbf{c}_t$ is the feature vector at time t, which is an M-dimensional real vector [2]:

$$\mathbf{c}_t = [c_t^1, c_t^2, \ldots, c_t^M]^T \tag{3.2}$$

here c_t^i is the i^{th} real-value feature of $\mathbf{c}_t$, and superscript ^{T} denotes matrix transpose.

Let $\mathbf{q} = \{q_1, q_2, \ldots, q_T\}$ be the corresponding HMM state sequence over T frames. We can write the probability of observing $\mathbf{c}$, given the state sequence $\mathbf{q}$ and a model λ as:

$$p(\mathbf{c} \mid \mathbf{q}, \lambda) = \prod_{t=1}^{T} p(\mathbf{c}_t \mid q_t, \lambda) \tag{3.3}$$

The right hand side of (3.3) factorises into the products of the individual state output probability $p(\mathbf{c}_t \mid q_t, \lambda)$, reflecting the fact that each state output $\mathbf{c}_t$ is independent of the outputs at all other frames once the current state is known:

$$\mathbf{c}_t \perp \mathbf{c}_{k \neq t} \mid q_t \tag{3.4}$$

here $\perp$ denotes the independence relationship.

We can obtain this conditional independence relation using the graphical view of an HMM (figure 3.1) where each node denotes a random variable, and the edges between nodes indicate conditional dependencies.

Because of the conditional independence assumption (3.4), algorithms exist for efficiently manipulating statistics in an HMM: the parameters can be trained with Baum-Welch algorithm, and the most probable state sequence corresponding to an observation $\mathbf{c}$ can be found using the Viterbi algorithm in a time-recursive manner. Sampling

[2] For typical ASR task that uses Mel-Cepstral features, M is usually 13.

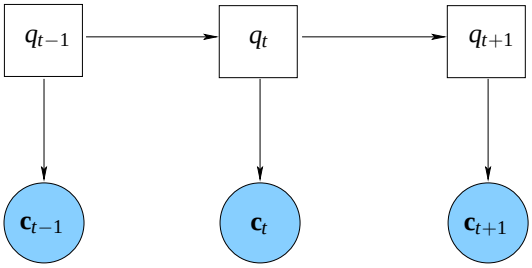

Figure 3.1: Graphical view of a standard HMM.

from an HMM is a simple matter of repeatedly generating samples from individual state output distributions once the underlying state sequence is known.

A common practice to circumvent the conditional independence limitation of an HMM is to augment the static feature vectors $\mathbf{c}_t$ with difference statistics obtained from adjacent frames. For the i^{th} static feature at time t, c_t^i , the so-called delta and delta-delta features can be calculated as weighted sum over a window of size $2L+1$ frames:

$$\Delta c_t^i = \sum_{\gamma=-L}^{L} w^{(1)}(\gamma) c_{t+\gamma}^i \tag{3.5}$$

$$\Delta^2 c_t^i = \sum_{\gamma=-L}^{L} w^{(2)}(\gamma) c_{t+\gamma}^i \tag{3.6}$$

where L is the half width of the window, γ is frame index, $w^{(1)}(\gamma)$ and $w^{(2)}(\gamma)$ are the weights associated with frame γ for the delta and delta-delta features respectively. As suggested in (Furui, 1986), the theoretical interpretation of $w^{(1)}(\gamma)$ and $w^{(2)}(\gamma)$ is that they are the first order orthogonal polynomial coefficients that represent the slope of the time function of each parameter in each segment.

An augmented feature vector $\mathbf{o}_t$ with a dimension of $3M$ by 1 is then created by appending the dynamic features to the static feature vector $\mathbf{c}_t$:

$$\mathbf{o}_t = [\mathbf{c}_t^T, \Delta\mathbf{c}_t^T, \Delta^2\mathbf{c}_t^T]^T \tag{3.7}$$

$\mathbf{o} = [\mathbf{o}_1^T, \mathbf{o}_2^T, \ldots, \mathbf{o}_T^T]^T$ is then the augmented feature vector sequence over T frames.

The use of dynamic features effectively makes each augmented vector $\mathbf{o}_t$ depend on its neighbouring frames, and is capable of capturing short term temporal dynamics over $2L+1$ frames. Unfortunately, in standard HMMs the augmented feature vectors

$\mathbf{o}_t$ are usually treated as independent observations, since $p(\mathbf{o} \mid \mathbf{q}, \lambda)$ is given by:

$$p(\mathbf{o} \mid \mathbf{q}, \lambda) = \prod_{t=1}^{T} p(\mathbf{o}_t \mid q_t, \lambda) \tag{3.8}$$

Although improvement in ASR and speaker recognition by using (3.8) over (3.3) has been verified empirically, and (3.8) has been used as a standard technique in HMM-based ASR systems, it is not clear that (3.8) will deliver the optimal performance that the dynamic features can provide. First, the conditional independence imposed by (3.8) cannot take into account of the fact that $\mathbf{o}_t$ are correlated. Consequently, applying the Viterbi algorithm on a conventionally trained HMM is unlikely to recover the true state sequence respecting the temporal constraints. Moreover, the dynamic constraints over HMM outputs are completely lost when a standard HMM is used as a generative model. The samples generated from $p(\mathbf{o} \mid \mathbf{q}, \lambda)$ do not obey the temporal constraints extant in the training data, and are in fact from the underlying HMM mean sequences.

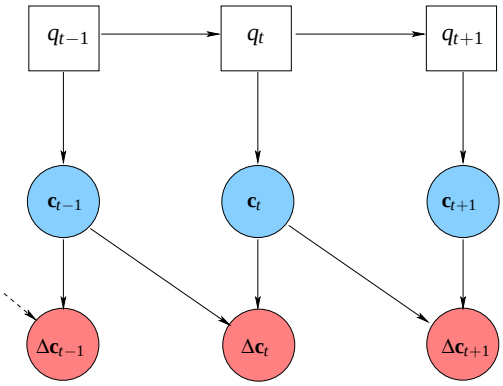

Figure 3.2: Graphical view of a standard HMM with delta features computed from two frames.

The introduction of dynamic features makes the static observation $\mathbf{c}_t$ no longer independent of each other because of the "explaining away" phenomena (Pearl, 1988), as shown in figure 3.2. The dynamic features then serve as a temporal constraint on $\mathbf{c}_t$. Therefore a correct probabilistic model should be defined as a function of $\mathbf{c}$, the static feature vectors, rather than the augmented feature vectors $\mathbf{o}$. In the next section we will derive a model that correctly respects the temporal constraints between nearby frames.

3.2.2 Deriving the Trajectory-HMM

To simplify discussion, we assume the static output distribution of an HMM is modelled as single Gaussians. i.e.:

$$p(\mathbf{o}_t \mid q_t, \lambda) = \mathcal{N}(\mathbf{o}_t \mid \boldsymbol{\mu}_{q_t}, \boldsymbol{\Sigma}_{q_t}, \lambda) \tag{3.9}$$

where $\boldsymbol{\mu}_{q_t}$ and $\boldsymbol{\Sigma}_{q_t}$ are $3M$ by 1 mean vector and $3M$ by $3M$ covariance matrix of the Gaussian corresponding to state q_t.

Since each output $\mathbf{o}_t$ is modelled "independently" in an HMM, the probabilistic function $p(\mathbf{o} \mid \mathbf{q}, \lambda)$ over T frames can be written as a big Gaussian:

$$p(\mathbf{o} \mid \mathbf{q}, \lambda) = \prod_{t=1}^{T} p(\mathbf{o}_t \mid q_t, \lambda) \tag{3.10}$$

$$= \prod_{t=1}^{T} \mathcal{N}(\mathbf{o}_t \mid \boldsymbol{\mu}_{q_t}, \boldsymbol{\Sigma}_{q_t}, \lambda) \tag{3.11}$$

$$= \mathcal{N}(\mathbf{o} \mid \boldsymbol{\mu}_{\mathbf{q}}, \boldsymbol{\Sigma}_{\mathbf{q}}, \lambda) \tag{3.12}$$

where $\boldsymbol{\mu}_{\mathbf{q}}$ and $\boldsymbol{\Sigma}_{\mathbf{q}}$ are the $3TM$ by 1 mean vector and $3TM$ by $3TM$ block-diagonal covariance matrix of the big Gaussian:

$$\boldsymbol{\mu}_{\mathbf{q}} = [\boldsymbol{\mu}_{q_1}^T, \boldsymbol{\mu}_{q_2}^T, \ldots, \boldsymbol{\mu}_{q_T}^T]^T \tag{3.13}$$

$$\boldsymbol{\Sigma}_{\mathbf{q}} = \mathrm{diag}[\boldsymbol{\Sigma}_{q_1}, \boldsymbol{\Sigma}_{q_2}, \ldots, \boldsymbol{\Sigma}_{q_T}] \tag{3.14}$$

We further note that each augmented feature vector $\mathbf{o}_t$ can be obtained from $\mathbf{c}$, the whole static feature vector sequence, by applying a linear transformation:

$$\mathbf{o}_t = \mathbf{w}_t \mathbf{c} \tag{3.15}$$

where $\mathbf{w}_t$ is a $3M$ by TM matrix given by:

$$\mathbf{w}_t = \begin{bmatrix} \underbrace{0, \ \ldots, \ 0,}_{t-L-1} & \underbrace{w^{(1)}(-L), \ \ldots, \ w^{(1)}(0), \ \ldots, \ w^{(1)}(L),}_{2L+1} & \underbrace{0, \ \ldots, \ 0}_{T-(t+L)} \\ 0, \ \ldots, \ 0, & w^{(2)}(-L), \ \ldots, \ w^{(2)}(0), \ \ldots, \ w^{(1)}(L), & 0, \ \ldots, \ 0 \\ 0, \ \ldots, \ 0, & 0, \ \ldots, \ 1, \ \ldots, \ 0, & 0, \ \ldots, \ 0 \end{bmatrix} \otimes \mathbf{I}_{M \times M} \tag{3.16}$$

The elements in the first M rows of $\mathbf{w}_t$ are mostly zeros except those in the t^{th} block of $M \times M$ elements, which is an M by M identity matrix. The second and the third M rows of $\mathbf{w}_t$ represent the weights for computing the delta and delta-delta features, and $w^{(n)}(\gamma), (n = 1,2; \gamma = -L, \ldots, L)$ are defined according to (3.5) and (3.6).

The augmented feature vectors over T frames is then given by a set of equations:

$$\begin{bmatrix} \mathbf{o}_1 \\ \mathbf{o}_2 \\ \vdots \\ \mathbf{o}_T \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_T \end{bmatrix} \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_T \end{bmatrix} \tag{3.17}$$

Let $\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_T \end{bmatrix}$, which is a $3TM$ by TM matrix, we have:

$$\mathbf{o} = \mathbf{W}\mathbf{c} \tag{3.18}$$

We have discussed in section 3.2.1 that $p(\mathbf{o} \mid \mathbf{q}, \lambda)$ is improper when used with dynamic features. The correct model should be defined as a function of $\mathbf{c}$, and can be obtained by normalising $p(\mathbf{o} \mid \mathbf{q}, \lambda)$:

$$p(\mathbf{c} \mid \mathbf{q}, \lambda) = \frac{1}{Z_\mathbf{q}} p(\mathbf{o} \mid \mathbf{q}, \lambda) \tag{3.19}$$

$$= \frac{1}{Z_\mathbf{q}} p(\mathbf{W}\mathbf{c} \mid \mathbf{q}, \lambda) \tag{3.20}$$

$$= \frac{1}{Z_\mathbf{q}} \mathcal{N}(\mathbf{W}\mathbf{c} \mid \boldsymbol{\mu}_\mathbf{q}, \boldsymbol{\Sigma}_\mathbf{q}, \lambda) \tag{3.21}$$

where $Z_\mathbf{q}$ is a partial integration of $\mathcal{N}(\mathbf{W}\mathbf{c} \mid \boldsymbol{\mu}_\mathbf{q}, \boldsymbol{\Sigma}_\mathbf{q}, \lambda)$, a $3MT$-dimension Gaussian, over the TM by 1 variable $\mathbf{c}$:

$$Z_\mathbf{q} = \int p(\mathbf{o} \mid \mathbf{q}, \lambda) d\mathbf{c} \tag{3.22}$$

$$= \int \mathcal{N}(\mathbf{W}\mathbf{c} \mid \boldsymbol{\mu}_\mathbf{q}, \boldsymbol{\Sigma}_\mathbf{q}, \lambda) d\mathbf{c} \tag{3.23}$$

It can be shown that the normalised probability density function (p.d.f.) $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ is a TM-dimension Gaussian distribution:

$$\boxed{\begin{aligned} &P(\mathbf{c} \mid \mathbf{q}, \lambda) = \mathcal{N}(\mathbf{c} \mid \overline{\mathbf{c}}_{\mathbf{q}}, \mathbf{P}_{\mathbf{q}}, \lambda) \\ &\text{satisfying:} \\ &\qquad \mathbf{R}_{\mathbf{q}} = \mathbf{W}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1} \mathbf{W} \\ &\qquad \mathbf{r}_{\mathbf{q}} = \mathbf{W}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1} \boldsymbol{\mu}_{\mathbf{q}} \\ &\qquad \mathbf{P}_{\mathbf{q}} = \mathbf{R}_{\mathbf{q}}^{-1} \\ &\qquad \mathbf{R}_{\mathbf{q}} \overline{\mathbf{c}}_{\mathbf{q}} = \mathbf{r}_{\mathbf{q}} \end{aligned}} \tag{3.24}$$

where $\overline{\mathbf{c}}_{\mathbf{q}}$ and $\mathbf{P}_{\mathbf{q}}$ are the TM by 1 mean vector and TM by TM covariance matrix and the normalisation term $Z_{\mathbf{q}}$ is given by:

$$Z_{\mathbf{q}} = \frac{\sqrt{(2\pi)^{TM} |\mathbf{P}_{\mathbf{q}}|}}{\sqrt{(2\pi)^{3TM} |\boldsymbol{\Sigma}_{\mathbf{q}}|}} \exp\left\{ -\frac{1}{2} \left(\boldsymbol{\mu}_{\mathbf{q}}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1} \boldsymbol{\mu}_{\mathbf{q}} - \mathbf{r}_{\mathbf{q}}^T \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}} \right) \right\} \tag{3.25}$$

One can verify that (3.25) is the correct expression by substituting (3.25) into (3.19) and see that the two sides are equal. The detail derivation can be found in appendix A.2.

Let us have a close look at (3.24) before moving on to the next section. Note $\boldsymbol{\Sigma}_{\mathbf{q}}^{-1}$ is the $3TM$ by $3TM$ conventional HMM covariance matrix of the state sequence $\mathbf{q}$, and is a banded matrix. And the $3TM$ by TM sparse matrix $\mathbf{W}$ encodes the linear coefficients (dynamic coefficients) that transform a static feature vector $\mathbf{c}$ into the augmented feature vector $\mathbf{o}$. Therefore multiplying $\mathbf{W}^T$ and $\mathbf{W}$ to $\boldsymbol{\Sigma}_{\mathbf{q}}^{-1}$ transforms the covariance matrix into a TM by TM matrix $\mathbf{R}_{\mathbf{q}}$, which turns out to be the precision matrix $\mathbf{R}_{\mathbf{q}}$ of the trajectory model and is an $M(4L+1)$ band-diagonal symmetric matrix. The inverse of $\mathbf{R}_{\mathbf{q}}$ is the covariance matrix of the Gaussian likelihood $p(\mathbf{c} \mid \mathbf{q}, \lambda)$, $\mathbf{P}_{\mathbf{q}}$, and is a TM by TM full matrix. In addition, the mean vector of the new model, $\overline{\mathbf{c}}_{\mathbf{q}}$, is the solution of the linear equation $\mathbf{R}_{\mathbf{q}} \overline{\mathbf{c}}_{\mathbf{q}} = \mathbf{r}_{\mathbf{q}}$, the right hand side of which is a linear transformation of the conventional HMM variance and mean parameters $\boldsymbol{\Sigma}_{\mathbf{q}}^{-1}$ and $\boldsymbol{\mu}_{\mathbf{q}}$. Consequently the mean parameter of the new model depends on both the mean and variance parameters of the original HMM it is based on. As $\mathbf{P}_{\mathbf{q}}$ is a full matrix, the calculation of the likelihood $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ can not be decomposed into frame-level statistics as in a conventional HMM.

We call the new model (3.24) the *Trajectory-HMM*, and refer to $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ as the *trajectory likelihood*. The parameters of the Trajectory-HMM, $\overline{\mathbf{c}}_{\mathbf{q}}$ and $\mathbf{P}_{\mathbf{q}}$, are computed from the corresponding HMM parameters $\boldsymbol{\mu}_{\mathbf{q}}$ and $\boldsymbol{\Sigma}_{\mathbf{q}}$, so the new model has the same number of free parameters as the conventional HMM with the same structure. However, the fact that $\mathbf{P}_{\mathbf{q}}$ is not block-diagonal means we can not use the standard

algorithms to do learning and inference based on frame-level statistics. This makes decoding such a model particularly difficult, and will be discussed in more detail in section 3.4.

3.3 Trajectory Training

3.3.1 Maximum Likelihood Training

Assume the state sequence $\mathbf{q}$ is known, the Trajectory-HMM is specified by the final Gaussian $\mathcal{N}(\mathbf{c} \mid \bar{\mathbf{c}}_\mathbf{q}, \mathbf{P}_\mathbf{q}, \lambda)$, which can be computed from the parameters of the corresponding HMM $\{\boldsymbol{\mu}_\mathbf{q}, \boldsymbol{\Sigma}_\mathbf{q}\}$. Although the new model has the same number of free parameters as that of an HMM with the same topology, the meanings of the parameters are different. In conventional HMMs the parameters $\boldsymbol{\mu}_\mathbf{q}$ and $\boldsymbol{\Sigma}_\mathbf{q}$ are trained to maximise the likelihood function $p(\mathbf{o} \mid \mathbf{q}, \lambda)$, and they have the interpretation of the mean vector and covariance matrix of the HMM state-output distribution. In contrast, the parameters $\boldsymbol{\mu}_\mathbf{q}$ and $\boldsymbol{\Sigma}_\mathbf{q}$ in a Trajectory-HMM are trained to maximise the trajectory likelihood $p(\mathbf{c} \mid \mathbf{q}, \lambda)$, and can not be interpreted as the parameters of the output distribution, which is modelled by the Gaussian $\mathcal{N}(\mathbf{c} \mid \bar{\mathbf{c}}_\mathbf{q}, \mathbf{P}_\mathbf{q}, \lambda)$.

The parameters in a Trajectory-HMM can be represented as:

$$\mathbf{m} = [\boldsymbol{\mu}_1^T, \boldsymbol{\mu}_2^T, \ldots, \boldsymbol{\mu}_N^T]^T \tag{3.26}$$

$$\boldsymbol{\phi} = [\boldsymbol{\Sigma}_1^{-1}, \boldsymbol{\Sigma}_2^{-1}, \ldots, \boldsymbol{\Sigma}_N^{-1}]^T \tag{3.27}$$

where N is the total number of Gaussians in the model (N is equal to the number of states if we use single Gaussians for state output), $\mathbf{m}$ and $\boldsymbol{\phi}$ are $3MN$ by 1 vector of the means and diagonal covariances of the Gaussians, respectively.

We can relate the new parameters $\mathbf{m}$ and $\boldsymbol{\phi}$ with $\boldsymbol{\mu}_\mathbf{q}$ and $\boldsymbol{\Sigma}_\mathbf{q}^{-1}$ by introducing a transformation matrix $\mathbf{S}_\mathbf{q}$:

$$\boldsymbol{\mu}_\mathbf{q} = \mathbf{S}_\mathbf{q}\mathbf{m} \tag{3.28}$$

$$\boldsymbol{\Sigma}_\mathbf{q}^{-1} = \mathrm{diag}(\mathbf{S}_\mathbf{q}\boldsymbol{\phi}) \tag{3.29}$$

$$\boldsymbol{\Phi} = \mathrm{diag}(\boldsymbol{\phi}) \tag{3.30}$$

where $\mathbf{S}_\mathbf{q}$ is a $3TM$ by $3MN$ matrix whose elements are zero or one determined according to the state sequence $\mathbf{q}$. $\boldsymbol{\Phi}$ is a $3MN$ by $3MN$ diagonal matrix.

Given a static feature vector sequence $\mathbf{c}$, we want to train a model λ that maximises the trajectory log-likelihood:

$$L(\lambda) = \log p(\mathbf{c} \mid \lambda) \tag{3.31}$$

$$= \log \sum_{\text{all } \mathbf{q}} p(\mathbf{c}, \mathbf{q} \mid \lambda) \tag{3.32}$$

Introducing an auxiliary distribution $q(\mathbf{q})$ over $\mathbf{q}$ and applying Jensen's inequality, we can obtain a lower bound on the log-likelihood objective function:

$$L(\lambda) = \log \sum_{\text{all } \mathbf{q}} q(\mathbf{q}) \frac{p(\mathbf{c}, \mathbf{q} \mid \lambda)}{q(\mathbf{q})} \tag{3.33}$$

$$\geq \sum_{\text{all } \mathbf{q}} q(\mathbf{q}) \log \frac{p(\mathbf{c}, \mathbf{q} \mid \lambda)}{q(\mathbf{q})} \tag{3.34}$$

$$= \sum_{\text{all } \mathbf{q}} q(\mathbf{q}) \log p(\mathbf{c}, \mathbf{q} \mid \lambda) - \sum_{\text{all } \mathbf{q}} q(\mathbf{q}) \log q(\mathbf{q}) \tag{3.35}$$

Applying the EM algorithm to maximise the lower bound (3.35) gives the following setup:

$$\textbf{E-step:} \quad q(\mathbf{q})^{(t+1)} = p(\mathbf{q} \mid \mathbf{c}, \lambda^{(t)}) \tag{3.36}$$

$$\textbf{M-step:} \quad \lambda^{(t+1)} = \arg\max_{\lambda} \sum_{\text{all } \mathbf{q}} q(\mathbf{q})^{(t+1)} \log p(\mathbf{c}, \mathbf{q} \mid \lambda) \tag{3.37}$$

$$= \arg\max_{\lambda} \sum_{\text{all } \mathbf{q}} p(\mathbf{q} \mid \mathbf{c}, \lambda^{(t)}) \log p(\mathbf{c}, \mathbf{q} \mid \lambda) \tag{3.38}$$

Repeatedly performing the E-step and the M-step will guarantee that our model converges to a local maximum (Dempster et al., 1977).

Because in the Trajectory-HMM the observations at different times are correlated with each other (the covariance matrix $\mathbf{P}_{\mathbf{q}}$ is generally full), we can not use the conventional forward-backward algorithm to enumerate all possible state sequences. For the same reason the usual Viterbi search can not be used to recover the most probable state sequence. However, this is an issue we must address if we want to go beyond the conditional independence assumption. The solution here is to approximate the lower bound (3.35) using only one sub-optimal state sequence obtained from either a standard HMM decoder or the trajectory decoding algorithm to be described in section 3.4.

With the Viterbi approximation, the M-step is equivalent to:

$$\lambda^{(t+1)} = \arg\max_{\lambda} \log p(\mathbf{c}, \mathbf{q}^* \mid \lambda) \tag{3.39}$$

where $\mathbf{q}^*$ is a sub-optimal state sequence obtained with model $\lambda^{(t)}$. Maximising (3.39) is equivalent to maximising the trajectory log-likelihood $\log p(\mathbf{c} \mid \mathbf{q}, \lambda)$ with respect to λ:

$$\log p(\mathbf{c} \mid \mathbf{q}, \lambda) = -\frac{1}{2}\left\{TM\log(2\pi) - \log|\mathbf{R_q}| + \mathbf{c}^T\mathbf{R_q}\mathbf{c} + \mathbf{r_q}^T\mathbf{P_q}\mathbf{r_q} - 2\mathbf{r_q}^T\mathbf{c}\right\} \quad (3.40)$$

Differentiating (3.40) with respect to the parameters $\mathbf{m}$ and $\boldsymbol{\phi}$, and setting the derivatives to zero, we can get new parameters.

We can now express the term $\mathbf{r_q}$ in terms of $\mathbf{m}$ and $\boldsymbol{\phi}$:

$$\mathbf{r_q} = \mathbf{W}^T\boldsymbol{\Sigma}_\mathbf{q}^{-1}\mathbf{S_q}\mathbf{m} \quad (3.41)$$

$$= \mathbf{W}^T\mathbf{S_q}\boldsymbol{\Phi}\mathbf{m} \quad (3.42)$$

In (3.42) we make use of the equation $\boldsymbol{\Sigma}_\mathbf{q}^{-1}\mathbf{S_q} = \mathbf{S_q}\boldsymbol{\Phi}$. Also note $\mathbf{S_q}$ is not a square matrix and, like $\mathbf{W}$, is not invertable.

Setting the partial derivative of (3.40) with respect to $\mathbf{m}$ to zero corresponds to solving the following set of linear equations:

$$\mathbf{S_q}^T\mathbf{W}\mathbf{P_q}\mathbf{W}^T\mathbf{S_q}\boldsymbol{\Phi}\mathbf{m} = \mathbf{S_q}^T\mathbf{W}\mathbf{c} \quad (3.43)$$

The dimensionality of (3.43) is $3MN$ and can be solved to obtain $\mathbf{m}$.

Maximising $\log p(\mathbf{c} \mid \mathbf{q}, \lambda)$ with respect to $\boldsymbol{\phi}$ leads to a non-linear optimisation problem, which can be solved using gradient descent methods utilising the first order gradient vector:

$$\frac{\partial \log p(\mathbf{c} \mid \mathbf{q})}{\partial \boldsymbol{\phi}} = \frac{1}{2}\mathbf{S_q}^T\mathrm{diag}^{-1}\Big(\mathbf{W}\mathbf{P_q}\mathbf{W}^T - \mathbf{W}\mathbf{c}\mathbf{c}^T\mathbf{W}^T + \mathbf{W}\overline{\mathbf{c}}_\mathbf{q}\overline{\mathbf{c}}_\mathbf{q}^T\mathbf{W}^T - 2\boldsymbol{\mu}_\mathbf{q}\overline{\mathbf{c}}_\mathbf{q}^T\mathbf{W}^T + 2\boldsymbol{\mu}_\mathbf{q}\mathbf{c}^T\mathbf{W}^T\Big) \quad (3.44)$$

In (3.44) the diag^{-1} operator takes the diagonal elements of the arguments, which are then summed according to the Gaussian component used via the transformation matrix $\mathbf{S_q}^T$, resulting in a $3MN$ by 1 gradient vector. Detailed derivation of those formulae can be found in appendix A.3.

Extending the above training scheme to multiple observation sequences is trivial. Assume that the training data consists of K observation sequence: $\mathbf{C} = \{\mathbf{c}^{(1)}, \mathbf{c}^{(2)}, \ldots, \mathbf{c}^{(K)}\}$, with corresponding state sequences $\mathbf{Q} = \{\mathbf{q}^{(1)}, \mathbf{q}^{(2)}, \ldots, \mathbf{q}^{(K)}\}$, then the log-likelihood objective function becomes:

$$L(\lambda) = \sum_{i=1}^{K} \log p(\mathbf{c}^{(i)} \mid \mathbf{q}^{(i)}, \lambda) \tag{3.45}$$

And the derivatives can be computed in a similar way.

Finally, it is also possible to train the model parameters using an objective function other then maximum-likelihood. In section 3.3.2, we will see trajectory model parameters being optimised to directly minimise the root mean square (RMS) error on the training data in an articulatory inversion task.

3.3.2 Minimum RMS Error Training

As discussed above, we see that just as the conventional HMM, the Trajectory-HMM can be trained using MLE objective function. However, the MLE criterion only measures the model's fitness to the data in the likelihood sense, it does not reflect the distance between the generated trajectories and the training vectors which is directly linked to the articulatory feature inversion performance to be discussed in chapter 4. Development in speech recognition has suggested that a training criterion that directly minimises the error measure on the training set, such as the Maximum Mutual Information Estimation (MMIE) or Minimum Classification Error (MCE) criteria (Bahl et al., 1986; Valtchev et al., 1997), tends to produce a better model for recognition. Theoretically, the MLE objective function should be preferred only when the model is correct, and in real-world applications where the model correctness is unknown the discriminative objective function is preferable for its robustness (Nádas et al., 1988). In this section I give the formula to minimise the RMS error directly on the training data. The detailed derivation can be found in Appendix A.4.

For mathematical convenience, we seek to minimise the sum of square error on the training data:

$$E = (\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})^T (\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}}) \tag{3.46}$$

where $\mathbf{c}$ is the static training vector, and $\overline{\mathbf{c}}_{\mathbf{q}}$ is the mean vector corresponding to the state sequence $\mathbf{q}$.

It can be shown that the gradient vector with respect to (w.r.t) $\mathbf{m}$ is:

$$\frac{\partial E}{\partial \mathbf{m}} = -2\mathbf{\Phi}\mathbf{S}_\mathbf{q}^T\mathbf{W}\mathbf{P}_\mathbf{q}(\mathbf{c} - \mathbf{P}_\mathbf{q}\mathbf{W}^T\mathbf{S}_\mathbf{q}\mathbf{\Phi}\mathbf{m}) \tag{3.47}$$

This leads to the set of linear equations to get the analytic solution of updated $\mathbf{m}$:

$$\mathbf{\Phi}\mathbf{S}_\mathbf{q}^T\mathbf{W}\mathbf{P}_\mathbf{q}\mathbf{P}_\mathbf{q}\mathbf{W}^T\mathbf{S}_\mathbf{q}\mathbf{\Phi}\mathbf{m} = \mathbf{\Phi}\mathbf{S}_\mathbf{q}^T\mathbf{W}\mathbf{P}_\mathbf{q}\mathbf{c} \tag{3.48}$$

The gradient vector w.r.t $\boldsymbol{\phi}$ is then:

$$\frac{\partial E}{\partial \phi} = 2\mathbf{S}_\mathbf{q}^T\text{diag}^{-1}\left(\mathbf{W}\mathbf{P}_\mathbf{q}\mathbf{r}_\mathbf{q}(\mathbf{c} - \bar{\mathbf{c}}_\mathbf{q})^T\mathbf{P}_\mathbf{q}\mathbf{W}^T + \boldsymbol{\mu}_\mathbf{q}(\mathbf{c} - \bar{\mathbf{c}}_\mathbf{q})^T\mathbf{P}_\mathbf{q}\mathbf{W}^T\right) \tag{3.49}$$

3.4 Trajectory Decoding

The goal of the decoding is to find a state sequence $\mathbf{q}$ that has the highest trajectory likelihood given observation sequence $\mathbf{c}$: $p(\mathbf{c} \mid \mathbf{q}, \lambda)$. In a time-synchronous decoder, this computation is usually accumulated as frame-based likelihood $p(\mathbf{c}_t \mid \mathbf{q}_t, \lambda)$ at time t. Unlike conventional HMMs where efficient algorithms exist for calculating $p(\mathbf{c}_t \mid \mathbf{q}_t, \lambda)$, decoding a Trajectory-HMM is much more complex due to the following issues:

- First, the state output at time t, $\mathbf{c}_t$, depends on the whole state sequence $\mathbf{q}$, making the usual Viterbi algorithm inapplicable. This means the likelihood of the whole utterance will not be available until the full state sequence is known, which is impractical in a time-synchronous decoder.

- Second, because there is no piecewise stationarity assumption over the state output, the number of valid paths grows exponentially to the length of the utterance being decoded. We must resort to some approximations to make the search tractable.

- Third, the formula of trajectory likelihood (3.24) involves high dimensional matrix manipulation, and is more expensive to evaluate than conventional HMM likelihood. Tradeoffs between accuracy and decoding time need to be investigated to make the computation acceptable.

To facilitate the time-synchronous decoding process, a way of recursively evaluating the trajectory likelihood based on (Zen et al., 2004) is first discussed in section 3.4.1. The idea is to factorise the likelihood term $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ into frame-level statistic

$p(\mathbf{c}_t \mid \mathbf{q}_{t+L}, \lambda)$, by looking at the history of the state sequence up to $t+L$ frames (L frames look ahead). To combat the exponential growth of paths, the idea of delayed path merging is proposed and will be discussed in section 3.4.2. Finally we conclude by describing a new decoding algorithm based on an extension of the Token-passing algorithm (Young et al., 1989).

3.4.1 Time-recursive Likelihood Computation

Since the searching is carried out in a left-to-right, time-recursive manner, it is necessary to find a way to compute the "trajectory likelihood", $p(\mathbf{c} \mid \mathbf{q}, \lambda)$, time recursively.

Recall $p(\mathbf{c} \mid \mathbf{q}, \lambda) = Z_{\mathbf{q}}^{-1} p(\mathbf{o} \mid \mathbf{q}, \lambda)$. $p(\mathbf{o} \mid \mathbf{q}, \lambda)$ is just the conventional HMM likelihood, and can be computed time-recursively, as can the $|\mathbf{\Sigma_q}|$ and $\boldsymbol{\mu}_{\mathbf{q}}^T \mathbf{\Sigma}_{\mathbf{q}}^{-1} \boldsymbol{\mu}_{\mathbf{q}}$ terms in $Z_{\mathbf{q}}$ (3.25). We need to compute the remaining terms of $Z_{\mathbf{q}}$, namely $|\mathbf{P_q}|$ and $\mathbf{r}_{\mathbf{q}}^T \mathbf{P_q} \mathbf{r_q}$, time-recursively.

Because the precision matrix $\mathbf{R_q}$ is an $M(4L+1)$ band-diagonal symmetric positive definite matrix, $\mathbf{R_q}$ can be decomposed by Cholesky-decomposition:

$$\mathbf{R_q} = \mathbf{U_q}^T \mathbf{U_q} \tag{3.50}$$

where $\mathbf{U_q}$ is an upper $M(2L+1)$ band triangular matrix. Then we can compute $|\mathbf{P_q}|$ time-recursively as the products of the diagonal elements of $\mathbf{U_q}$:

$$\begin{aligned} |\mathbf{P_q}| &= |\mathbf{R_q}|^{-1} && (3.51)\\ &= |\mathbf{U_q}^T \mathbf{U_q}|^{-1} && (3.52)\\ &= |\mathbf{U_q}|^{-2} && (3.53)\\ &= \prod_{t=1}^{T} |\mathbf{U}_{\mathbf{q}_{t+L}}^{(t,t)}|^{-2} && (3.54) \end{aligned}$$

where $\mathbf{q}_{t+L} = \{q_1, \ldots, q_{t+L}\}$ denotes the state sequence up to time $t+L$, $\mathbf{U}_{\mathbf{q}_{t+L}}^{(t,t)}$ is the diagonal element of $\mathbf{U}_{\mathbf{q}_{t+L}}$ at time t, and only depends on state sequence up to time $t+L$.

Using similar techniques, we can rewrite $\mathbf{r}_{\mathbf{q}}^T \mathbf{P_q} \mathbf{r_q}$ as:

$$\begin{aligned} \mathbf{r}_{\mathbf{q}}^T \mathbf{P_q} \mathbf{r_q} &= \mathbf{r}_{\mathbf{q}}^T \mathbf{P}_{\mathbf{q}}^T \mathbf{P_q}^{-1} \mathbf{P_q} \mathbf{r_q} && (3.55)\\ &= \bar{\mathbf{c}}_{\mathbf{q}}^T \mathbf{U_q}^T \mathbf{U_q} \bar{\mathbf{c}}_{\mathbf{q}} && (3.56)\\ &= \mathbf{g_q}^T \mathbf{g_q} && (3.57) \end{aligned}$$

where $\mathbf{g_q}$ is a TM by 1 vector obtained from:

$$\mathbf{g_q} = \mathbf{U_q}\bar{\mathbf{c}}_\mathbf{q} \tag{3.58}$$

$$= \mathbf{U_q}^{T-1}\mathbf{r_q} \tag{3.59}$$

The inner product $\mathbf{g_q}^T\mathbf{g_q}$ can be expressed in the sum form:

$$\mathbf{g_q}^T\mathbf{g_q} = \sum_{t=1}^{T} [\mathbf{g}^{(t)}_{\mathbf{q}_{t+L}}]^T \mathbf{g}^{(t)}_{\mathbf{q}_{t+L}} \tag{3.60}$$

Therefore,

$$\exp\left\{\mathbf{r_q^T P_q r_q}\right\} = \exp\left\{\sum_{t=1}^{T}\left[\mathbf{g}^{(t)}_{\mathbf{q}_{t+L}}\right]^T \mathbf{g}^{(t)}_{\mathbf{q}_{t+L}}\right\} \tag{3.61}$$

$$= \prod_{t=1}^{T} \exp\left[\mathbf{g}^{(t)}_{\mathbf{q}_{t+L}}\right]^T \mathbf{g}^{(t)}_{\mathbf{q}_{t+L}} \tag{3.62}$$

Putting all these together we can write $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ in the product form that can be computed time-recursively:

$$p(\mathbf{c} \mid \mathbf{q}, \lambda) = \prod_{t=1}^{T} \frac{1}{\mathbf{Z}^{(t)}_{\mathbf{q}_{t+L}}} p(\mathbf{o}_t \mid q_t, \lambda) \tag{3.63}$$

where each $\mathbf{Z}^{(t)}_{\mathbf{q}_{t+L}}$ can be calculated with some look ahead up to frame $t+L$.

$$Z^{(t)}_{\mathbf{q}_{t+L}} = \frac{\sqrt{(2\pi)^M}|\mathbf{U}^{(t,t)}_{\mathbf{q}_{t+L}}|^{-1}}{\sqrt{(2\pi)^{3M}|\boldsymbol{\Sigma}_{q_t}|}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\mu}^T_{q_t}\Sigma^{-1}_{q_t}\boldsymbol{\mu}_{q_t} - \left[\mathbf{g}^{(t)}_{\mathbf{q}_{t+L}}\right]^T \mathbf{g}^{(t)}_{\mathbf{q}_{t+L}}\right)\right\} \tag{3.64}$$

3.4.2 Delayed Path Merging

The decoding problem in ASR can be phrased naturally as a tree search, as illustrated in figure 3.3, where nodes at the t^{th} level of the search tree represents valid HMM states at time t. Starting from an empty root node at time 0, we gradually extend the leaf nodes of the tree using nodes subject to language model constraints. At time T the path with the highest score will be picked up as the decoding result.

It is easy to see that the tree search procedure outlined above is infeasible to implement as the number of paths grows exponentially as t increases. For a conventional HMM, this exponential growth in path numbers can be avoided by assuming that nodes

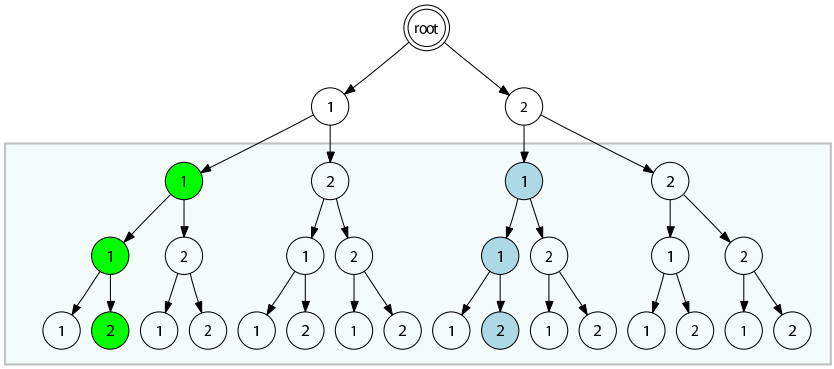

Figure 3.3: Illustration of path merging for a search tree of a two-state HMM. Two paths $1 \rightarrow 1 \rightarrow 1 \rightarrow 2$ and $2 \rightarrow 1 \rightarrow 1 \rightarrow 2$ will be merged as they have the same partial path within the shaded window spanning 3 frames ($D = 1, L = 1$). Only the one with a higher partial likelihood score will be retained.

at the t^{th} level of the tree only depend on nodes at the $(t-1)^{th}$ level (1st order Markov assumption). Therefore two paths are equivalent if their last two nodes end with the same state. We can generalise this idea further for Trajectory-HMM by assuming two paths are equivalent if they share the same fixed-length history. More specifically, for given integers $D \geq 1$, $L \geq 1$. two paths $\mathbf{q}_1^{t+L}$ and $\mathbf{p}_1^{t+L}$ are considered to be equal if $\mathbf{q}_{t-D}^{t+L} = \{q_{t-D}, \ldots, q_{t+L}\} = \{p_{t-D}, \ldots, p_{t+L}\} = \mathbf{p}_{t-D}^{t+L}$. Here D and L are the number of frames we need to look back/ahead in order to compute the likelihood score at time t. In other words, whenever we see two equivalent paths we only need to retain the one with the higher score and discard the other. We refer the process of retaining the highest-score path out of its equivalent class $\Psi(\mathbf{q}_1^{t+L}) = \mathbf{q}_{t-D}^{t+L}$ to as path merging. After extending all the nodes at frame $t-1$ to t, we perform path merging for all paths within a window $[t-D, t+L]$, which effectively discards half of the hypothesis at frame t. The usual beam pruning can then be applied to further reduce the hypotheses set. This process is illustrated in figure 3.3.

3.4.3 Extended Token-Passing Algorithm

The Token-passing algorithm was proposed by (Young et al., 1989) as a way of abstracting the time-synchronous search problem as tokens passing around a network. In that algorithm, a static network is first built from the task grammar where network nodes represent HMM states and the links between nodes denote the valid paths con-

strained by the grammar. State transition probabilities, word insertion penalties, and pronunciation scaling factors can all be represented as weights attached to a node link. The decoding process starts with an empty token at the ENTER node. Then for each frame, each node first passes a copy of its token through all its connected nodes, and increments the token's score by its frame likelihood. After that, each node compares the incoming tokens and only retain the one with the highest score. This is equivalent to applying the Viterbi criterion in a time-synchronous decoder. While doing this, each token also maintains a history of nodes travelled through, so that the most probable path can be backtraced from the EXIT node at the end of the decoding process. The token-passing algorithm is a concise abstraction of Viterbi decoding with a few benefits:

- It is possible to use a hand-crafted grammar, which is important for certain tasks like spoken dialogue system.
- Unigram and back-off Bigram statistics can be incorporated easily as weights of between-word links.
- Once built, the decoding network remains unchanged in the decoding process and the footprint of the decoder is predictable; although the memory requirement is proportional to the size of the expanded network.

For a Trajectory-HMM, a few extensions need to be introduced in order to use the token-passing framework. The first extension is the concept of *partial path*. Instead of storing the full state history of a path, which is impractical, for each path we store the partial state history $\mathbf{q}_{t-D}^{t+L}$ in the frame window $[t-D, t+L]$, where $\mathbf{q}_{t-D}^{t}$ is the history, and $\mathbf{q}_{t}^{t+L}$ is the look ahead part that looks into possible future state sequences. We also need to keep track the word or state history so that at the end of decoding we can backtrace the full word or state sequence.

The next extension is the concept of token. Since at each HMM node, rather than looking at only one state, in a Trajectory-HMM we need to store the information of *all the partial paths* within window $[t-D, t+L]$. Thus an extended token at node j consists of *all partial paths whose* t^{th} *state is state* j, plus their likelihood information. This is illustrated in figure 3.4.

As with the original token-passing algorithm, at each time frame each node will pass a copy of its token to its connected nodes, where all incoming tokens will be merged. When passing tokens around, all partial paths in the token will be expanded

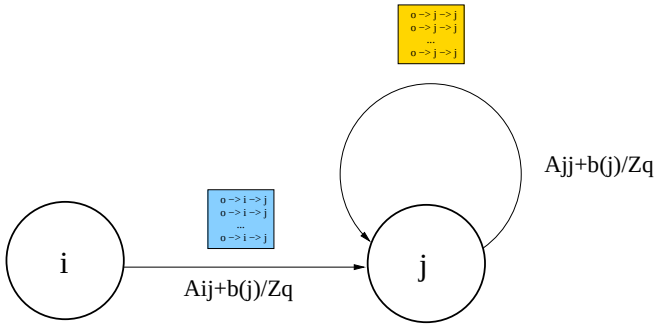

Figure 3.4: Illustration of passing a token with a set of partial paths.

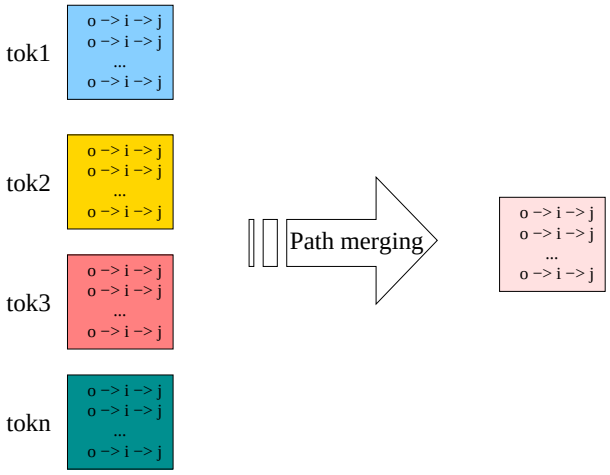

Figure 3.5: Illustration of path merging, where half of the paths are pruned away after merging.

one state further. Because we only retrain paths in the window of $[t-D, t+L]$. This means after a path expansion, each path will find another path sharing the same partial state history $\mathbf{q}_{t-D}^{t+L}$, and we only need to keep the one with a higher score. This effectively execute the delayed path merging concept of section 3.4.2, and can be illustrated in figure 3.5

The main extended token-passing algorithm is outlined in Algorithm 1.

3.4.4 Discussion

The algorithm presented above is a sub-optimal alternative to the full search by enumerating all possible state sequence to solve $\mathbf{q}^* = \arg\max_{\mathbf{q}} p(\mathbf{q} \mid \mathbf{c}, \lambda)$, which is too expensive. Compared to the conventional Token-passing/Viterbi algorithm, the main change is to base our prediction at time t on a short context window including the

Algorithm 1 Extended Token-Passing Algorithm

1: Put an NULL token into the ENTER node
2: **for** $t = 0$ to T **do**
3: Propagate all nodes' tokens to their connected nodes
4: Perform path merging for all paths within a window $[t-D, t+L]$
5: **end for**
6: Return the highest-score path at time T

previous D frames and the next L frames. This is also called D-frame delayed Viterbi decoding (Zen et al., 2004).

If our model has N different HMM states, the extended Token-passing algorithm needs to maintain the statistics of $N^{(D+L)}$ different partial-paths within each token, significantly more bookeeping compared with the conventional Viterbi searching for HMMs, where only statistics for one path need to be recorded at each token. When passing each token around at each frame, the statistics of each of those $N^{(D+L)}$ paths also need to be updated based on the contextual information available.

3.5 Sampling from the Model

Certain applications, such as HMM-based speech synthesis, can benefit from being able to generate a smoothed trajectory from such models. Since samples from a Trajectory-HMM will respect the dynamic constraints in the model, which is more appropriate for real speech data.

Sampling from a Trajectory-HMM is more difficult than sampling from a standard HMM, because the covariance matrix $\mathbf{P_q}$ of the final Gaussian $\mathcal{N}(\mathbf{c} \mid \bar{\mathbf{c}}_\mathbf{q}, \mathbf{P_q}, \lambda)$ is generally full. In this section we describe two approaches to sample from such model, both assume the state sequence $\mathbf{q}$ is known.

1. Sampling with Cholesky-decomposition
   Assuming the parameters are all single Gaussians, we can sample from the final Gaussian $\mathcal{N}(\mathbf{c} \mid \bar{\mathbf{c}}_\mathbf{q}, \mathbf{P_q}, \lambda)$ directly using the Cholesky-decomposition, a standard technique to sample from multivariate Gaussian distribution. The Cholesky-decomposition of the covariance matrix is:

$$\mathbf{P_q} = \mathbf{L}\mathbf{L}^T \tag{3.65}$$

where $\mathbf{L}$ is a unique TM by TM lower triangular matrix.
The following steps draw a sample from $\mathcal{N}(\mathbf{c} \mid \overline{\mathbf{c}}_\mathbf{q}, \mathbf{P}_\mathbf{q}, \lambda)$ [3]:

(a) Draw a TM-dimensional real vector $\mathbf{U}$ from the standard Gaussian $\mathcal{N}(0,1)$:

$$\mathbf{U} = [u_1, u_2, \ldots u_{TM}]^T \qquad u_i \sim \mathcal{N}(0,1) \tag{3.66}$$

(b) Obtain the final sample:

$$\mathbf{Y} = \mathbf{L}\mathbf{U} + \overline{\mathbf{c}}_\mathbf{q} \tag{3.67}$$

Although this method works well for single Gaussian case, applying it to mixtures of Gaussian (GMMs) will be difficult. Since in the later case the final distribution of the Trajectory-HMM is a big GMM whose number of components grows exponentially with the length of the sequence T. It is infeasible to compute those components and their weights. In such cases, stochastic sampling method such as Gibbs Sampling may be used.

2. Sampling with Gibbs Sampling
Gibbs Sampling (Geman and Geman, 1984) is a method to sample from complex multivariate distribution without knowing the full joint p.d.f. $p(\mathbf{c})$, as long as we can compute the full conditional p.d.f. $p(\mathbf{c}_t \mid \mathbf{c}_{k \neq t})$. In the case of Gaussian mixtures, the final distribution as a big GMM is unknown. However, since each output $\mathbf{c}_t$ only depends on its neighbour output $\{\mathbf{c}_{t-L}^{t+L}\}$, we can compute the conditional probability of $\mathbf{c}_t$ given that the values of all other observations are fixed:

$$p(\mathbf{c}_t \mid \mathbf{c}_1, \ldots, \mathbf{c}_{t-1}, \mathbf{c}_{t+1}, \ldots, \mathbf{c}_T) = p(\mathbf{c}_t \mid \mathbf{c}_{t-L}, \ldots, \mathbf{c}_{t-1}, \mathbf{c}_{t+1}, \ldots, \mathbf{c}_{t+L}) \tag{3.68}$$

If each state output is modelled as a GMM, the conditional p.d.f. (3.68) is also a GMM, whose parameters can be computed from its neighbour's statistics using standard formula of multivariate Gaussian distribution (Anderson, 2003). We can construct a Gibbs Sampler to sample from $p(\mathbf{c} \mid \mathbf{q}, \lambda)$.

[3] Here we assume the mean vector $\overline{\mathbf{c}}_\mathbf{q}$ is given. Otherwise it can be obtained by solving $\mathbf{U}_\mathbf{q}\overline{\mathbf{c}}_\mathbf{q} = \mathbf{g}_\mathbf{q}$.

3.6 Experiment on Synthetic Data

3.6.1 Motivation

Having introduced the basic framework for the Trajectory-HMM, this section will describe some experiments conducted on synthetic data, using a sampling-based approach. Doing experiment on synthetic data has several advantages. First, the sampling-based approach used here makes it possible to systemically compare learnt models with the original model used to generate the data. Whilst the real world data is always generated by some unknown process, the result obtained on synthetic data can be generalised to real data if we assume a Trajectory-HMM can be used as a proper generative model. Viewing the samples generated graphically also provides us with a visual clue to the modelling capability of such model. Moreover, conducting experiments in a controlled environment can prevent the system from being affected by irrelevant factors that exist in real speech data such as variability in gender and speaking style. Last, the current training procedure of Trajectory-HMM is computationally expensive, so a small dataset makes a lot of sense for rapid system prototyping.

3.6.2 Learning a Delta Chain

3.6.2.1 The Delta Chain Model

One simple yet interesting Trajectory-HMM is the delta chain model studied in (Bridle, 2004). The delta chain used here is a one-state system parametrised by two single Gaussians: a static distribution $\mathcal{N}_s(\mu_s, \sigma_s^2)$ characterising the state-output distribution, and a dynamic (delta) distribution $\mathcal{N}_d(\mu_d, \sigma_d^2)$ modelling the dynamic constraints computed as the difference between two adjacent state outputs:

$$\Delta c_t = c_t - c_{t-1} \tag{3.69}$$

The corresponding dynamic window template used to compute the augmented observations is:

$$w_t = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 1 & 0 \end{bmatrix} \tag{3.70}$$

The delta chain presented above is a one-state Trajectory-HMM that generates 1D

observations. In this experiment the following chain configuration is used:

$$\mathcal{N}_s \sim \mathcal{N}(0, 30) \tag{3.71}$$

$$\mathcal{N}_d \sim \mathcal{N}(10, 1) \tag{3.72}$$

Also the delta effects at both ends of the chain are ignored, which gives the same setting as used in the simplest case studied in (Bridle, 2004).

3.6.2.2 Sampling from a Delta Chain

A general method to sample from such a chain is to construct a Gibbs Sampler (Geman and Geman, 1984) using the full conditional p.d.f. computed from Gaussian statistics with a window of size 3. Gibbs Sampling tends to be computationally slow, and the convergence rate of the sampler is sensitive to the choice of the initial state of the chain. Fortunately, for the simple chain studied here where all parameters are single Gaussians, there exists faster way to sample. We can first work out the mean and covariance matrix of the joint Gaussian of the whole chain directly using (3.24), then sample from the joint distribution using standard technique for sampling from multivariate Gaussian distribution (e.g. using Cholesky-decomposition method).

The left part of figure 3.6 shows a sample of 30 frames generated from such a chain. The X-Axis shows the frame index, from 1 to 30, and Y-Axis indicates the output value. The circle points represent the state output at each frame, while the square points correspond to the mean sequence of the underlying "trajectory Gaussian". The temporal covariance matrix of the chain is plotted in the right part of figure 3.6.

In this delta chain each output c_t is affected by three distributions namely the static distribution $\mathcal{N}_s \sim \mathcal{N}(0, 30)$ and two delta distributions $\mathcal{N}_d \sim \mathcal{N}(10, 1)$ from its neighbour points (c_{t-1} and c_{t+1}). The static distribution acts like a gravity force to constrain point c_t around μ_s (mostly within $\pm\sigma_s$), while the delta distribution $\mathcal{N}_d$ can be interpreted as the repulsion forces from adjacent points c_{t-1} and c_{t+1} which tend to drive c_t away from them. Those three kinds of forces interact with each other and propagate through the whole chain until some balance state is reached. As a result, the actual amount of "delta force" imposed on c_t only depends on the points to the left and right sides of c_t. Consequently points near the two edges will receive near maximal "delta force" while points in the central frame will have "delta force" on both sides cancelled almost exactly, resulting the "S" shape in Figure 3.6. In fact we can check that the ver-

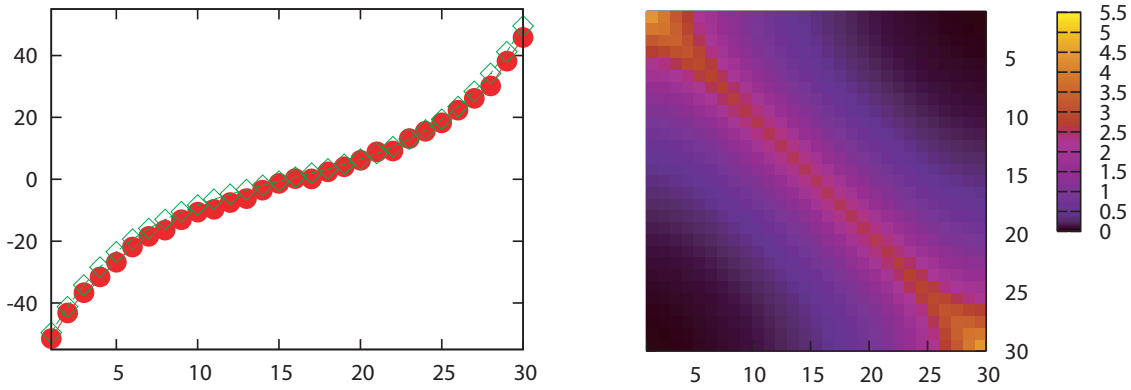

Figure 3.6: A sample of 30 frames from a one-state Delta Chain (left), and its full covariance matrix (right), showing the correlation between frames.

tical difference between two points at the ends of the chain is roughly 10 units (8.33 for the left end, and 7.57 for the right end).

Although the delta chain described here has only one state, the actual output of the system at different times, as a result of the interaction between static and dynamic constraints, can influence each other. We can verify that the mean sequence of the delta chain (the square line in figure 3.6) indeed changes over time. This behaviour is completely different from a conventional HMM, where each state-output is considered to be independent. In addition, the temporal covariance matrix of the Trajectory-HMM is in general a full matrix compared to the diagonal covariance matrix of an HMM. Also in figure 3.6 we can see there are much stronger correlation around the diagonal area than other part of the covariance matrix. This is mainly because the delta feature used is computed from a short window of 3 frames. We conclude that under such a configuration, short term temporal correlation is better modelled than long term temporal correlation.

3.6.2.3 Learning a Delta Chain

If the real acoustic data can be generated by a Trajectory-HMM with dynamic constraints, it will be interesting to see what kind of models can be learnt from such data. Our assumption here is that a statistical model that matches the underlying generative model should not only give a higher likelihood on testing data than a model learnt with incorrect assumption, but also converge to the paramater setting of the original model.

First, I generated 1000 samples of 30 frames using the above delta chain. 900 samples are used for training and 100 samples are for testing. I then trained a Trajectory-HMM and a conventional HMM on the same training data, both have only one state, and compute the per-frame "trajectory likelihood" scores on testing data using the learnt models. The Gaussian parameters were initialised from the global mean and variance computed from the training data. The result is shown in table 3.1.

Model	Static Param.	Delta Param.	loglik-train	loglik-test
original	$\mathcal{N}(0,30)$	$\mathcal{N}(10,1)$	-1.372	-1.362
HMM	$\mathcal{N}(-0.007,519.287)$	$\mathcal{N}(3.302,5.807)$	-2.370	-2.365
trajectory	$\mathcal{N}(-0.007,30.681)$	$\mathcal{N}(9.840,0.999)$	-1.372	-1.362

Table 3.1: Performances of different delta-chain models.

First note that the HMM parameters shown in table 3.1 is nowhere near the true parameters of the delta chain. The variance of the static distribution learnt from the data is notably higher than the static variance of the original model. Although the static distribution $\mathcal{N}_s$ of the original model has a rather small variance of 30, the samples generated show a larger variance of 519.29 the constraints imposed by delta distribution $\mathcal{N}_d$. This suggests a model based on different assumption can not learn the delta chain correctly. As a result, the HMM has a lower trajectory per-frame log-likelihood on both training and testing data.

The last row of table 3.1 shows the parameters of a Trajectory-HMM after 15 iterations of training. The parameters show the model learnt through the MLE trajectory training algorithm (see section 3.3.1) indeed converges to the original model. The per-frame log-likelihood on both training and testing data are only slightly lower than those of the original model.

3.7 Trajectory-HMM in Practice

Using constraints in the dynamic features to generate smoothed output trajectories from an HMM is not a new idea. HMM-based speech generation using this technique has been around for more than a decade (Tokuda et al., 1995). However, not until the discovery of trajectory MLE training algorithm, along with the formation of Trajectory-HMM as a properly normalised HMM, did the Trajectory-HMM start

to generate interest outside of the model-based speech synthesis community (Bridle, 2004; Minami et al., 2004; Williams, 2005). Maximum likelihood training was found to improve a phone recognition task by rescoring the N-best list returned by a conventional HMM decoder. However, the lack of a full scale trajectory decoding algorithm makes it difficult to evaluate the usefulness in the wider context of speech recognition, where rescoring N-best list may not be an option. Furthermore, the recognition experiment conducted in (Tokuda et al., 2004; Zen et al., 2004) used a "clean" speech corpus specifically designed for the purpose of speech synthesis. It is unknown how the Trajectory-HMM performs under typical recognition environment where the recorded speech is expected to be much noisier. Still, the use of Trajectory-HMM as a recognition model is still in its early stage. As for speech synthesis, a number of improvements have been made recently. For instance, linguistically derived contextual units replaced the monophone models to provide a richer representation of contextual speech variation. Regulation method considering global variance has been introduced to combat the over-smoothness output from the original parameter generation algorithm (Toda and Tokuda, 2005). Sophisticated speaker adaption algorithms have been proposed to change the spectral and prosodic characteristics of the synthesised speech (Yamagishi et al., 2006). The resulting HMM-based TTS can be quite compact (under 2MB footprint) yet still provide competitive synthesis performance (Zen and Toda, 2005; Zen et al., 2006). However, the parameters of current HMM-based TTS system are still updated using the Baum-Welch algorithm designed for conventional HMMs. It is therefore of great research interest to investigate the usefullness of trajectory updated parameters in real-world speech processing tasks. In addition, there has not yet been a systematic study of the effect of different parameter update method and the possibility of using alternative training criteria other than Maximum Likelihood Estimation. These questions will be addressed in later chapters of this thesis.

3.8 Summary

We have explored the Trajectory-HMM framework in detail. The learning algorithm has been demonstrated on the synthetic data. In the following chapters we will submit the model to real-world tests. First in chapter 4 we use Trajectory-HMM as a proper generative model to recover articulatory features from speech signal. Then in Chapter 6 the proposed decoding algorithm is employed to perform a speaker-dependent phone recognition task.

Chapter 4

Acoustic-Articulatory Modelling with Trajectory-HMM

The previous chapter shows that the mean output of a Trajectory-HMM is a continuous trajectory that closely resembles the static part of the training vector. In this chapter, I will explore how this property can be used in a real-world task to recover the movement of human articulators from speech signal. In particular, I propose an HMM-based acoustic-to-articulatory inversion system where the Trajectory-HMM is used in a complementary way to the conventional HMM acoustic modelling techniques, utilising joint acoustic-articulatory optimisation. In the system the link between the articulatory domain and the acoustic domain is represented by the HMM state sequence. To achieve this, the mapping from the acoustics to the HMM state sequence is posed as a recognition problem, where speech recognition techniques are employed. Following that, the parameter generation algorithm is used to recover the articulatory mean trajectory from the reconstructed state sequence. This two-stage inversion is in contrast to most approaches where a direct mapping from acoustics to the articulatory domain is learnt. The benefit is that each inversion stage can be optimised using established techniques from speech recognition and synthesis, within a unified HMM framework. In addition, the intermediate use of an HMM state sequence gives the system the ability to synthesise articulatory movements from sources other than acoustic signal. This can be achieved by doing articulatory inversion from a textual representation of speech using a duration model.

As training Trajectory-HMMs is a subject of both theoretical and practical interest, different aspects of training will be examined in a controlled environment. This includes the effectiveness of the MLE and RMS training objective functions, the be-

haviour of individual parameter update method, and the possible optimisations that can be applied to the task.

Experiments on a speaker-dependent acoustic-articulatory parallel corpus indicate that the jointly trained acoustic-articulatory models are more accurate (having a lower RMS error) than the separately trained ones, and that Trajectory-HMM training results in greater accuracy compared with conventional Baum-Welch parameter updating. The proposed RMS training objective has proved to be consistently better than MLE objective on this task.

In what follows, I will first review the articulatory inversion problem in section 4.1, covering data acquisition, the difficulties of inversion, and reviewing of previous inversion attempts within the statistical paradigm. Then in section 4.2 I will discuss the motivation and methodology of using Trajectory-HMMs on this task. Following that, experiment set-up and result analysis will be given in section 4.3 and 4.4, respectively. Finally this chapter finishes with a summary in section 4.5.

4.1 Articulatory Inversion Problem

4.1.1 Articulatory Data Acquisition

Physically, human speech production is the joint effect of the changes in vocal tract shapes and the movement of speech articulators, such as the tongue, jaw and lips. Knowing the relationship between the movement of speech articulators and the resulting speech will provide valuable information to speech science. Compared to the acoustic features commonly used in speech recognition, articulatory movement can provide a simple and straightforward explanation for speech characteristics. For example, a simple movement of the F2 formant from high to low could be easily explained by the tongue moving from the front of the mouth to the back, using the articulatory features, but is more complicated in the domain of standard acoustic parameters, such as mel-cepstra or line spectral frequencies (LSF). Moreover, articulatory features, recorded directly from articulator positions, are more robust to environmental conditions, such as the frequency response of acoustic recorders, the distance between the speaker's mouth and microphone, or the background noise (Ling et al., 2008). On a practical side, applications like speech training, speech coding and speech visualisation can all benefit from recovering articulatory movement from speech signal (Richmond, 2002, page 5-7). Furthermore, a conventional speech recogniser can potentially take

advantage of the added information in the articulatory domain, which provides useful constraints from speech production that is otherwise unavailable in the acoustic domain. Zlokarnik (1995) reported an 18-25% relative error rate reduction in a speaker-dependent isolated word recognition task using an HMM recogniser. In that experiment the articulatory features were appended to conventional acoustic features in the training stage and the articulatory movements during the testing phase were estimated using a multilayer perceptron that performed the acoustic-to-articulatory mapping. It is also found that the information from articulatory data can improve the accuracy of model-based speech synthesis: Ling et al. (2008) integrated articulatory features into an HMM-based parametric speech synthesis system and discovered that the accuracy of acoustic parameter prediction can be improved significantly by modelling acoustic and articulatory features together in a shared-clustering and asynchronous-state model structure.

While speech signals can be recorded acoustically via microphone, the movement of human speech articulators is difficult to track physically. Early articulatory data was obtained by means of X-Ray filming, also called cineradiography (Houde, 1967; Munhall et al., 1998). This method is effective and will not interfer the subject's speaking process. However, the obvious drawback of using X-Ray filming is the radiation to the subject's head. Although computer-controlled X-Ray can alleviate this problem by reducing the time of subject's exposure under X-Ray, this method is largely replaced by safer methods such as MRI (Magnetic Resonance Imaging) and EMA (**E**lectro**m**agnetic **A**rticulography).

In Magnetic Resonance Imaging (MRI), subjects' voices can be recorded simultaneously with axial, coronal, or midsagittal MR images of their vocal tracts while the subjects are speaking. Area functions describing the individual tract shapes can be obtained by measurements performed on the captured MR images. In (Baer et al., 1991), the subject laid in the supine position on a horizontal patient couch with his head inside a saddle-shaped radio frequency coil for receiving the resonance signal from a whole body MRI system that generated a field strength of $0.15T$. And it took a long time (about 3 to 4 hours) to collect all the images needed to specify a given vocal tract configuration, which is the major drawback of using MRI in speech studies. Recent development in real-time MR scanning (RT-MRI) has improved the acquisition speed considerably, and it is even possible to do a real-time continuous monitoring of the vocal tract with an actual time resolution of 5-50 images per second, although the frame rates cannot be further increased by accelerating the readout scans for safety reasons

(Bresch et al., 2008). Compared to the invasive EMA approach described below, MRI has the advantage of producing a complete view of the entire vocal tract including the pharyngeal structures in a non-invasive manner.

In Electromagnetic Articulography (EMA), a number of miniature sensor coils are attached to the subject's articulators such as the lips, the tongue body or the tongue tip. The subject's head is then surrounded by three magnets outside of the head. The movement of the coils can be inferred as sets of x, y coordinates by comparing the relative strength of the signals from each of the three magnets. Compared to MRI system, EMA machine costs much less and is preferred for examining speech and articulatory movements at a high frequency. A practical issue of EMA recording is that it is difficult to keep the sensor coils fixed during long recording sessions. In the recording mentioned in (Richmond et al., 2003), for example, tongue-blade and velum coils have been re-attached during recording and the EMA data need to be processed to remove discontinuity in the position of re-attachment. In addition, before each recording session the subjects also need some time to get used to the presence of sensor coils in their mouths.

Another recent development in capturing articulatory movement is the ultrasound imaging of tongue technique, such as the EdgeTrak System (Li et al., 2005), which tracks the tongue surfaces through a sequence of 2D ultrasound images. Image analysis can be carried out to recover the articulatory movement.

Having mentioned all the data acquisition methods, the experiment reported in this thesis will use EMA data recorded in MOCHA database, which will be described in detail in section 4.3.

4.1.2 The Challenge of Inversion

Estimating human articulator movements from non-articulatory data has been studied for decades. One of the first studies was done by (Schroeder, 1967), who showed that vocal-tract-shape information can be obtained by purely acoustic measurements. Schroeder considered two kinds of acoustic methods that can determine an approximation to the cross sectional area of the vocal tract as a function of distance along its axis. Researchers soon realised that although it is possible to estimate vocal-tract shape from speech signals, the inversion problem presents its unique difficulties.

The first difficulty is the non-uniqueness of the mapping. Empirically, it has been observed that the mapping from speech to articulatory domain is one-to-many. Atal

et al. (1978) studied the relationship between the shape of the vocal tract and its acoustic output. They observed that large changes in the shape of the vocal tract can be made without changing the formant frequencies. In particular, Atal et al demonstrated that the same English vowel /i/ can correspond to different mouth openings of their model while retaining its formant characteristics, and different vocal tract shapes for several vowels can produce acoustic signals with almost identical values for the first three formant frequencies. Lindblom et al. (1977) conducted the so-called bite-block experiment using formant frequency data of Swedish vowels produced both with fixed and unconstrained mandible. They found that subjects were able to produce formant patterns within the ranges of the normal vowels, in spite of physiologically unnatural jaw openings. The observations were explained by hypothesising that the "instantaneous" learning of highly unfamiliar tasks, such as compensatory articulation of fixed-mandible vowels, is possible because normal speech motor programming is indeed "compensatory" rather than due to either speakers drawing upon past similar experience or invoking special motor mechanisms distinct from those of natural speech. This is in line with the hypothesis of compensatory articulation that different people can produce the same sound with dissimilar vocal tract shapes. It also helps explain the art of ventriloquism that many sounds can be produced with a wide range of mouth openings. Theoretically, the non-uniqueness of the mapping can also be demonstrated by inspecting Webster's horn equation, a second-order linear differential equation used to derive transfer function of a tube under appropriate boundary conditions. It has been shown that for lossless vocal tracts with fixed boundary conditions, the area functions $A(x)$ and $1/A(L-x)$ (where L is the length of the vocal tract) produce the same acoustics (Qin and Carreira-Perpiñán, 2007).

While it is universally accepted that the mapping from acoustics to arcitulatory features is one to many, how frequently does non-uniqueness occur in normal human speech, and how does this affect the inversion algorithms that can only do one-to-one? An empirical investigation conducted by (Qin and Carreira-Perpiñán, 2007) on a small acoustic-articulatory dataset showed that only 5% of the acoustic vectors yield a multi-modal cloud in articulatory space. This suggests that, while nonuniqueness does happen, by and large, it is an infrequent situation, and most times a unique vocal tract shape is used for human speech production.

Another challenge is the jaggedness of acoustic features compared to the articulatory data which is of a smooth, slow-varying nature. The inversion algorithm needs to be robust so that the input jaggedness does not transform to the output articulatory

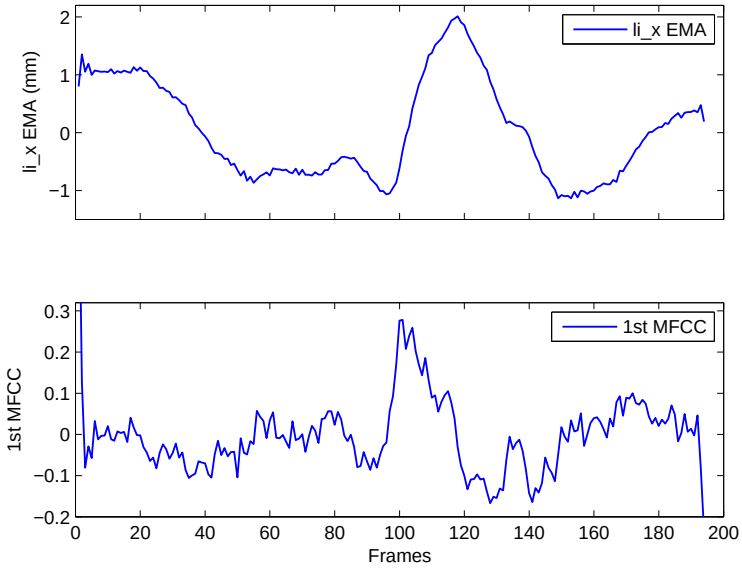

Figure 4.1: Comparing EMA channel li_x (up) and the 1st order MFCC feature (bottom) from the first utterance in the articulatory-acoustic parallel data. The MFCC has much jaggedness compared to EMA.

targets to cause large inversion error. Some researchers therefore opt to smooth the inversion output trajectory in the post-procession stage, such as applying a low-pass filter (Richmond, 2002). Figure 4.1 plots acoustic and articulatory features from one utterance in the articulatory-acoustic parallel data where the jaggedness of acoustic MFCC feature is easy to see.

Nevertheless, the articulatory inversion task is an ill-posed problem and one can not guarantee unique inversion result from acoustic signal without imposing additional constraints. When designing a computational inversion model, we should bear in mind that not all configurations generated by an inversion model are physiologically possible in human speech production as a result of the lack of physiological accuracy (Richmond, 2002, page 48).

4.1.3 Previous Attempts to Acoustic-to-Articulatory Inversion

Many of the early attempts are so called analytical approaches, where mathematical analysis of an acoustic signal was performed to yield the area function of a tube model

that might have generated it (Richmond, 2002, page 23). With the advance of modern computing, the research community shifted to statistical methods. The rest of this section will focus mainly on reviewing representative inversion methods in the statistical department.

1. Codebook Approach

   Also referred to as Articulatory Codebooks approach (Schroeter and Sondhi, 1994), the coodbook method builds look-up table consists of pairs of segmental acoustic and articulatory parameters from parallel recorded articulatory-acoustic data. For example, in the experiment of (Hogden et al., 1996) the acoustic vectors were categorised into a table with 256 codes using vector quantisation (VQ) method by finding the shortest Euclidean distance between the acoustic vectors and each of a small set of numbered reference vectors. A VQ codebook was used to map from acoustic segments to VQ codes, and a look-up table was then used to map from the VQ code to an estimated articulatory configuration. Root-mean-squared errors around 2 mm were reported for coils on the tongue by (Hogden et al., 1996). Being a discrete method, the VQ approach does not give the same level of approximation to the target distribution without significantly increasing the size of look-up table, compared to methods employing continuous variables. Today this method has largely been replaced by more sophisticated models.

2. Neural-Network-based Method

   As a “universal function approximator”, Neural-Network-based methods are popular choices for the acoustic-articulatory inversion task. (Papcun et al., 1992) used a neural network trained on paired data to do inversion for "gesture recognition", which is to recognise speech gestures from inferred trajectories of the three articulators. Rahim et al. (1991) built an assembly of multi-layer perceptrons for inversion and Kobayashi et al. (1991) trained a four-layer-feed-forward network for this task. Variants of Neural-networks such as Mixture Density Networks (MDN) has also been reported recently (Richmond, 2007). A Neural-network based inversion system is easier to implement than more dedicatedly built models with comparative performance. The choice of network structure (number of hidden layers, and node per hidden layer etc.) is still largely a trial and error approach. In addition, it is not straightforward to incorporate phonetic constraints into the system.

3. Kalman Filtering Method

(Dusan and Deng, 2000) tried an extended Kalman filter trained on paired acoustic-articulatory data. After the acoustic features were acquired, they repetitively applied extended Kalman filter on each segment of speech using the parameters of different phonological models. Then a likelihood measure was computed for each filtering and based on it, carried out segmentation and recognition of models. Finally, the articulatory trajectories were estimated by applying the extended Kalman smoother using the parameters of the recognised models.

4. Gaussian Mixture Models

   In (Toda et al., 2004), an inversion model based on Gaussian Mixture Models was described. The mapping function from an acoustic feature vector $\mathbf{x}_t$ to an articulatory feature vector $\mathbf{y}_t$ in frame t is defined as $\widehat{\mathbf{y}}_t = \sum_{i=1}^{M} p(m_i \mid \mathbf{x}_i) p(\mathbf{y}_t \mid \mathbf{x}_t, m_i)$, where M is the total number of mixture components, $p(m_i \mid \mathbf{x}_i)$ is the component weight conditioned on $\mathbf{x}_i$, and $p(\mathbf{y}_t \mid \mathbf{x}_t, m_i)$ is a conditional Gaussian distribution with full covariance matrices. The set of GMMs were MLE trained on joint probability $p(\mathbf{x}, \mathbf{y})$ using parallel acoustic-articulatory data. The training can be augmented by considering articulatory dynamic features which, when used with a low-pass filter in post-processing, helped to reduce the RMS error further. In order to get a good inversion accuracy, a large number of Gaussian mixture components from 1 to 128 were used in their experiments. GMM models with 1 mixture component was reported to give an RMS error of 2.11mm on the test data. This was reduced to 1.68mm when a mixture of 32 components was used.

5. HMM-based Inversion

   (Hiroya, 2006) proposed an HMM-based production model for acoustic-articulatory inversion. His system consists of two parts. First HMMs of articulatory parameters were built for each phoneme or diphone, and the mapping from articulatory to acoustic domain was approximated in a piece-wise linear form, whose parameters can be trained via maximum a posteriori (MAP) criterion from the parallel acoustic-articulatory data. At the inversion stage, an HMM state sequence was derived from speech signal via Viterbi decoding, then articulatory feature values were estimated via an MAP mapping and a smoothed output trajectory was generated via the trajectory parameter generation algorithm (see 3.24 on page 31). Phonetic constraints can be incorporated to facilitate the derivation of more

accurate HMM state alignment for inversion.

Among the reviewed approaches, (Hiroya, 2006)'s method is the closest to the modelling approach discussed in the present chapter: both methods use HMM state as inversion unit, and both used the trajectory parameter generation algorithm. However, I would like to highlight the main differences here:
First, in our model the parameter generation algorithm is an integral part of the Trajectory-HMM framework to derive smoothed output mean trajectories. Moreover the training criterion can be tailored to minimise the RMS error between the generated mean vector and the data. In contrast, the parameter generation algorithm was used as a smoother in the post-processing stage of (Hiroya, 2006). There is no guarantee the constraints between the static and dynamic features can be honoured. Nevertheless, the parameter generation algorithm has proved to be an effective smoothing algorithm in similar context. Richmond used the algorithm to smooth the static output from a Mixture Density Networks with the help of dynamic features and showed improvement over the low-pass filter smoothing (Richmond, 2007).

Second, in (Hiroya, 2006) the mapping between acoustic and articulatory domain was approximated as a linear function, which is a naive approximation considering the fact that the relation between arcitulatory and acoustic domain is non-linear. There is no such restriction in the proposed framework.

4.2 Trajectory-HMM for Articulatory Feature Recovering

4.2.1 Motivation

Hidden Markov models (HMMs) are the standard approach to speech recognition, where the underlying task is to maximise the discrimination between similar phones or words (Valtchev et al., 1997). Speech synthesis models, on the other hand, use different techniques such as unit selection (Clark et al., 2007) to make the synthesised speech sound as natural as possible. This suggests that different modelling approaches may be required for recognition and synthesis; however Tokuda et al have shown that the Trajectory-HMM formulation may be successfully applied to speech synthesis as well (Tokuda et al., 2000; Zen and Toda, 2005).

The inversion of articulatory data involves both synthesis and recognition: we start

with the acoustic signal and pose the recovery of the missing articulatory information as a synthesis problem. Conversely, the recovered articulatory information can have a complementary role in the modelling of pronunciation and acoustic variability in speech recognition.

As previously reviewed, most attempts to recover articulatory movement from the speech signal involve building a mapping from the acoustic domain to the articulatory domain, either manually or constructed automatically from parallel data. Often the inversion system is built separately from the recognition framework, particularly because the slowly varying nature of articulation may be best modelled in a different way to speech acoustics which change more rapidly, and are noisier.

Our system, based on the Trajectory-HMM, differs from others in the sense that both recognition (acoustic) and synthesis (articulatory) models are constructed in the same framework, and are jointly modelled as a two-stream HMM. The use of publicly available articulatory-acoustic parallel data enables us to compare our findings with the published results from other researchers, and to use the task as a testbed for trying out Trajectory-HMM algorithms. The Trajectory-HMM extends the conventional HMM framework, and many established HMM building techniques can be re-used. Additionally, in the inversion stage only the HMM state sequence is needed, so it is possible to synthesise articulatory movement from a textual representation without the speech signal.

4.2.2 Trajectory-HMM

Temporal derivative features, or delta features, are well known to improve the accuracy of HMM-based speech recognition systems (Furui, 1986). However, as discussed in chapter 3, the simple incorporation of delta features in an HMM leads to an inconsistent generative model (Bridle, 2004). These inconsistencies may be resolved by performing a per-utterance normalisation, leading to the Trajectory-HMM (Tokuda et al., 2004).

Let $\mathbf{c}$ denote the static observation vector sequence, and let $\mathbf{o}$ denote the sequence of observation vectors augmented with delta features. Then the likelihood of observing the static observation vector sequence given the HMM state sequence $\mathbf{q}$ and the model parameters λ is obtained by normalising the likelihood of obtaining the augmented observation vector sequence:

$$p(\mathbf{c} \mid \mathbf{q}, \lambda) = \frac{1}{Z_{\mathbf{q}}} p(\mathbf{o} \mid \mathbf{q}, \lambda) \tag{4.1}$$

where $Z_{\mathbf{q}}$ is a normalisation term that depends on the state sequence (also see 3.25 on page 31):

$$Z_{\mathbf{q}} = \int p(\mathbf{o} \mid \mathbf{q}, \lambda) d\mathbf{c}. \tag{4.2}$$

A smoothed output mean trajectory $\overline{\mathbf{c}}_{\mathbf{q}}$ can be derived from an HMM state sequence:

$$\overline{\mathbf{c}}_{\mathbf{q}} = \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}} \tag{4.3}$$

The model parameters include the Gaussian mean and variance components and can be updated using gradient-based methods.

Unlike the stepwise mean output of a conventional HMM, the mean output from $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ is a smoothed trajectory, and can be used as a proper generative model, as in parametric speech synthesis. Equation (4.3) serves as the core generative component in the articulatory inversion system. It is possible to train the Trajectory-HMM to maximise the generative model likelihood $p(\mathbf{c} \mid \mathbf{q}, \lambda)$. This has considerably higher complexity than conventional maximum likelihood training for HMMs, and is rarely done for HMM-based speech synthesis systems (Zen and Toda, 2005).

4.2.3 Joint Acoustic-Articulatory HMM Modelling

Instead of seeking a direct mapping between the acoustic and articulatory signals, our methodology centres around the idea of jointly optimising a single model for acoustic and articulatory information. The model has two components, both are modelled as multi-state phone-level HMMs: an articulatory synthesis model (articulatory HMMs) which (given an HMM state sequence) generates a smoothed mean trajectory (4.3); and an alignment model (acoustic HMMs) which derives the state sequence for synthesis from the acoustic representation of an utterance.

Both the articulatory HMMs and the acoustic HMMs have the same model topology: i.e. they have exactly the same set of HMM states, phoneme definitions, so that the HMM state alignment from one can be mapped to the other directly. This structure enables the HMM state alignment, as a highly abstracted speech form, to bridge the articulatory and acoustic domains.

For inversion we first derive a representative HMM state alignment from the acoustic HMMs. Then the parameter generation algorithm (3.24) is executed to produce the smoothed mean trajectory (4.3) in the articulatory domain. One feature of the system is the flexibility in obtaining the HMM state alignment at the inversion stage. Depending on the available resources, it can be the state sequence returned by an HMM

decoder, the forced alignment derived from phone labels, or the synthesised state sequence from a textual representation, using a suitable duration model. An overview of the acoustic-articulatory model is illustrated in figure 4.2.

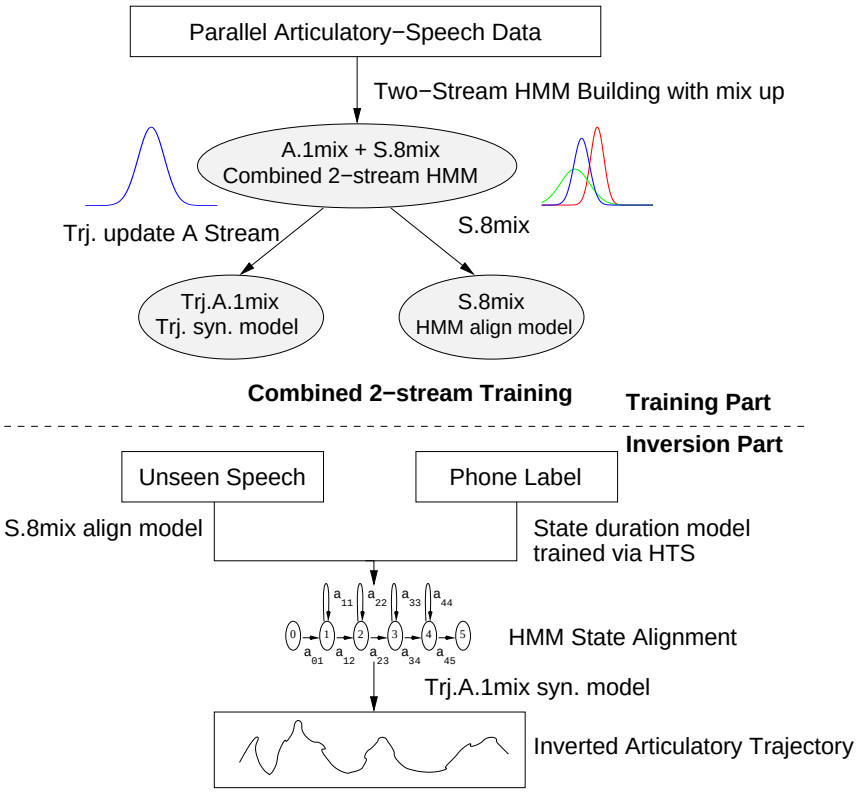

Figure 4.2: Overview of the articulatory-acoustic modelling system. Using two-stream combined training will result in greater accuracy compared with the separately trained ones. The trajectory stream is modelled by trajectory updated HMMs with single Gaussian state output (Trj.A.1mix), and the acoustic stream is modelled by Baum-Welch trained HMMs with 8-mixture GMM state output (S.8mix).

The use of two conceptually different HMMs makes training more complex. First the two HMMs operate on domains with different nature: the articulatory feature is slowly varying while the acoustic feature is fast changing and more noisy. Second, the output distribution of the articulatory HMMs can only be modelled using single Gaussian densities, which is a limitation of the present Trajectory-HMM framework. Nevertheless, the acoustic HMMs can use Gaussian mixtures as in a conventional HMM

system. Third, the quality of the intermediate HMM state alignment, either for trajectory training the articulatory HMMs, or for the articulatory feature inversion, plays a vital role in the overall inversion performance.

I considered two ways of training the models. The first is referred to as separate training, where the articulatory HMMs are trained on the articulatory data only, and the acoustic HMMs are built from the acoustic data alone using the standard HMM training procedure. The HMM state alignment required for trajectory training of articulatory HMM parameters is derived from the articulatory data using a conventionally trained baseline HMM model. The idea behind the separate training is that training the two types of HMMs individually is likely to bring out the best performance from each channel.

The second scheme, on the other hand, tries to find a way to optimise the parameters of both types of HMMs simultaneously in the hope that the performance of the two as a whole can be further improved. This requires a way to tie the two HMMs at state level during training. And HTK's multi-stream functionality makes this type of training possible, by combining the two set of HMMs into a single, two-stream HMM model. Starting from single Gaussian output distribution, the number of Gaussian mixtures in the acoustic stream is increased during training while the number of Gaussian mixtures in the articulatory stream remains unchanged. Furthermore, the alignment for the trajectory parameter update can be obtained by aligning the articulatory-acoustic parallel training data using the combined HMM set. After that, either the Trajectory MLE estimation or the RMS training criteria can be carried out on the articulatory stream only.

The dynamic features, which play a central role in Trajectory-HMM, are usually obtained from the regression coefficients that represent the temporal slope of each feature (Furui, 1986). In the HTK system[1], delta coefficients are computed from the previous and next two frames. The delta-delta coefficients are computed in the same way using the previous and next two deltas, meaning that the whole window covers nine frames. In the HTS HMM-based speech synthesis system[2] a simpler three frame window is employed, using a quadratic regression for delta-deltas. I experimented with both kinds of windows, and found that choice of coefficient type has an impact on the articulatory inversion task. In what follows, I will refer to the three-frame dynamic window (as used in HTS) as *dw3*, and the nine-frame window (as used in HTK) one as

[1] `http://htk.eng.cam.ac.uk/`.
[2] `http://hts.sp.nitech.ac.jp/`.

dw9. The differences will be discussed in section 4.4.7.

4.3 Experimental Set-up

4.3.1 The EMA Data from MOCHA-TIMIT

The **M**ulti**ch**annel **A**rticulatory (MOCHA) corpus[3] is a speaker-dependent recording of TIMIT sentences with EMA articulatory information captured along with the the acoustic signal. Currently it includes one male speaker (msak0) and one female speaker (fsew0), each uttering 460 "phonetically-diverse" TIMIT sentences. Electromagnetic receiver coils are attached to 7 articulators[4] in both x and y-coordinates during recording, providing a total 14 channels of articulatory information sampled at 500 Hz. The female data (fsew0) was used in this work.

4.3.2 Data Pre-processing

In preparing for the experiment, we down-sampled the EMA data to 100 Hz to match the 10 ms frame-shift of the acoustic features, which are the standard 12-order MFCCs with log-energy plus their delta and delta-deltas. All delta features were computed using the three-frame *dw3* window unless mentioned otherwise. A mean-filtering normalisation was performed to compensate some EMA measure errors introduced in the recording stage (Richmond, 2002). I set aside the utterances whose recording number ends with 2 for validation (46 utterances), those ending with 6 for testing (46 utterances) and the remaining 368 utterances for training. This is the same split as used in (Richmond, 2002). The validation set will be used to early stop the training process when gradient-based method was used, to prevent from overfitting the data. The monophone set consists of 45 phones including silence. The inversion performance will be reported as the average RMS error compared with the recorded articulatory data:

$$E_{RMS} = \frac{1}{14}\sum_{m=1}^{14}\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{c}_i^m - \mathbf{t}_i^m)^2} \tag{4.4}$$

where N is the total number of frames in the data, $\mathbf{c}_i^m$ and $\mathbf{t}_i^m$ are the i^{th} frame for the

[3] http://www.cstr.ed.ac.uk/research/projects/artic/mocha.html.

[4] These 7 articulators are: upper and lower incisor, upper and lower lip, tongue tip, tongue blade and tongue dorsum.

generated mean vector and the test vector of articulatory channel m.

The silence portion at the beginning and the end of each utterance was excluded when computing the RMS error. As during these unvoiced regions the articulators can be in any position. This practice is in line with other published results on the MOCHA data. We did not do this in an early experiment, and slightly higher RMS errors were reported in (Zhang and Renals, 2008).

4.3.3 The L-BFGS Optimiser

While the trajectory mean parameter update can be obtained in closed form (3.43, 3.48), it is difficult to obtain the update of variance parameter analytically due to the fact that the gradient vector (3.44, 3.49) is not a quadratic function of the variance variable. We therefore resort to a general gradient-descent method to do the optimisation. Conventional gradient-descent optimisers that do not utilise the second-order Hessian matrix can be slow to converge, especially on tasks involving a large number of variables. I chose to use an off-the-shelf gradient-based optimiser L-BFGS-B developed in (Zhu et al., 1997). L-BFGS stands for " Limited memory BFGS method", which is a limited-memory version of the quasi-Newton algorithm that, instead of calculating a full Hessian at each iteration, stores a low rank Hessian approximation by analysing successive gradient vectors. A bounded version of the algorithm that can solve large non-linear optimisation problem subject to simple bounds on the variables will be used [5]. The variance variables are constrained to be within the range of $[\log 1^{-4}, \log 1^{4}]$, to avoid extreme values that will cause numerical problem. The bound is given in the log domain as the variance parameters are transformed into log domain to extend the search space of each variance parameter to $\mathcal{R}$.

4.3.4 Inversion Configurations

Similar to building an HMM-based speech recognition system, we refine our inversion models incrementally. Models for both articulatory and acoustic streams start from a single Gaussian, three-state, left-to-right monophone HMM trained using HTK.

The inversion model refers to the HMM that was first initialised on articulatory data using HTK. Then its mean and variance parameters were updated via the trajectory training algorithm. There are different ways to update the model parameters. The nature of the mean parameter update (closed form analytical solution) and the variance

[5] The fortran L-BFGS-B code can be obtained from http://www.ece.northwestern.edu/~nocedal/lbfgsb.html.

update (gradient-descent training) calls for a different treatment. Zen et al. (2004) used a procedure that first updated the mean parameter analytically, then employed a gradient-descent method to optimise the variances, repeating the process until the model log-likelihood on the training data stabilised. A possibly better way of training is to update both mean and variance parameters simultaneously using gradient-decent method, thus removing the need for iterative training as long as the state alignment is fixed. We can also choose to update mean or variance only. Since the Trajectory-HMM is still relatively new with no standard training protocol, I explored different training approaches.

Depending on the parameter update method used, three groups of articulatory inversion models were built:

- Group A: baseline HMM model trained on the articulatory data using Baum-Welch algorithm as implemented in the `HERest` tool from HTK. This group has only one model: A.hmm.

- Group B: Articulatory HMMs trained separately from acoustic HMMs, with the Gaussian mean and variance components updated using the forced alignment provided by the articulatory HMMs in the baseline model A.hmm. Six models can be built using two different training objective functions (MLE and RMS) and three parameter update schemes:

  - m: update Gaussian mean parameters only by solving the set of linear equations (3.43, 3.48).
  - v: update Gaussian variance parameters only using gradient-based method.
  - mv: update both mean and variance parameters using gradient-based method.

  The trained models will be named with suffix so that B.mle-mv is the Trajectory-HMM with both mean and variance parameters updated using MLE objective function via gradient-descent method on the articulatory data.

- Group C: Similar to Group B but both the articulatory stream and the acoustic stream were jointly trained using HTK's multiple-stream facility, with a default stream weight of 1.0 being applied to both streams. Then the Gaussian mean and variance parameters of the articulatory stream were updated using the same procedure as in Group B, yielding six different articulatory inversion models named as C.mle-m, C.mle-mv etc.

The configurations of different inversion models are summarised in table 4.1.

A	A.hmm	baseline HMM trained with HTK
B	B.mle-m, B.mle-v, B.mle-mv	separate training, MLE objective
	B.rms-m, B.rms-v, B.rms-mv	separate training, RMS objective
C	C.mle-m, C.mle-v, C.mle-mv	joint training, MLE objective
	C.rms-m, C.rms-v, C.rms-mv	joint training, RMS objective

Table 4.1: List of inversion model configurations.

Since a Trajectory-HMM equivalent of the Baum-Welch algorithm has not been discovered, it is prohibitively expensive to train Trajectory-HMMs whose output distribution is Gaussian mixture with more than one component. Thus our articulatory synthesis component is limited to single component Gaussian densities (1mix). In deriving the alignment for training, however, there is no such restriction. We can therefore obtain a more accurate alignment by using more mixture components in the acoustic HMMs. In the experiments, Gaussian mixture densities up to eight (8mix) components were used to derive the HMM state alignment.

As the MOCHA-TIMIT data we used is similar to the original TIMIT data, it is of interest to consider the accuracy obtained by the state-of-the-art techniques on the original TIMIT data, which is a standard benchmark for phone recognition. For standard context-dependent triphone HMMs, a phone error rate (PER) of 27.1% has been reported in (Lamel and Gauvain, 1993). Using a recurrent neural network gives 26.1% (Robinson, 1994), and the lowest PER so far 24.4%, was obtained by a landmark segment-based approach (Glass, 2003).

Finally, table 4.2 gives the phone error rate on validation and testing sets using a phoneloop grammar. These figures, while not directly comparable to the state-of-the-art performance on the original TIMIT data, do give us an idea of the performance of monophone Baum-Welch trained HMMs.

4.3.5 Questions to be Answered

Before moving to the result analysis section, let us step back and look at some questions of both theoretical and practical value, that will be answered by the experiments in this chapter.

	Articulatory Feature		Acoustic Feature	
	val	test	val	test
1mix	55.93	53.13	57.49	55.71
4mix	48.94	46.87	52.54	51.78
8mix	45.13	43.25	51.43	47.97

Table 4.2: Phone error rates on fsew0 data using Baum-Welch trained HMMs. The first column gives the number of Gaussian mixture components used in the HMM state output distribution.

1. How does the Trajectory-HMM compare to MLE trained HMM?
   Conventional MLE trained HMMs have already been used in the context of speech synthesis for more than a decade with good results (Tokuda et al., 1995). Due to the complexity of trajectory training , even the latest HMM-based speech synthesis system creates models using the classic Baum-Welch algorithm, albeit with sophisticated enhancements such as the use of contextual features, applying EM-based mixture component selection in the parameter generation stage, and parameter generation considering global variance (Zen et al., 2007). However, the fact that a conventionally trained HMM does not hornour the relationship between the statics and deltas, as described in chapter 3, is unlikely to bring out the full potential of the model. Through experiment, I hope to discover empirically the level of improvement we are likely to gain in a real-world application by honouring the dynamic constraints in training.

2. We want to see if jointly trained two-stream models (group C) have a performance advantage over the separately optimised two-stream models (group B).
   The two streams in the inversion system have different properties. The acoustic features are fast changing while the articulatory features have a slow varying nature. It is tempting to hypothesise that training the two stream separately can achieve optimal modelling on both streams. However, the fact that the two stream are tied by the same HMM state, which is vital for both acoustic alignment and articulatory inversion, introduces dependencies between the two. In other applications such as audio-visual recognition, it has been confirmed that joint-stream optimisation is helpful for modelling the asynchrony between audio

and visual state sequences (Brand et al., 1997; Gravier et al., 2002). This kind of optimisation will be explored in section 4.4.5.

3. The effect of different parameter update schemes.
   As discussed earlier, different training schemes can have an impact on the system performance. It will be useful to find out the tradeoff between training complexity and model fitness resulting from varying individual parameter update method. For instance, in terms of computational cost, the analytic solution of mean update has a clear advantage over the gradient-based methods. Through experiment I would like to find out how will the low cost update methods compete with the more expensive ones when it comes to system performance.

4. The effect of RMS training objective function.
   So far all the work on training the Trajectory-HMM is based on the Maximum-likelihood principle. While MLE-training has a sound theoretical foundation, as described in section 3.3.2 it is possible to directly minimise the RMS error on the training data. The experiment in section 4.4.3 is designed to see how the RMS training objective function stacks against the MLE one.

5. The quality of state alignment.
   A vital component in the inversion system is to use acoustic HMMs to derive an HMM state alignment for articulatory inversion. Although the articulatory stream is restricted to single Gaussian, there is no such restriction for acoustic HMM. In our experiment we choose to increase the number of Gaussian components up to 8 mixture components in the acoustic stream. We will investigate how such a practice performs in a real-world task.

6. The effect of dynamic coefficients.
   An often ignored area in previous research is the choice of dynamic coefficient. Different types of dynamic coefficient differ not only in the width of the dynamic window, the formula for deriving the delta and delta-deltas from the static features, but also in the generated mean trajectories, and ultimately the inversion performance. These differences will be examined in section 4.4.7.

4.4 Experiments

4.4.1 Effectiveness of MLE Training

The first problem to be addressed is the effectiveness of MLE training. Zen et al. (2004) observed that MLE training helped to reduce the N-best rescoring error in a phone recognition context. In this experiment, I investigated the effectiveness of MLE training and explored the relationship between the increase in log-likelihood and the reduction in RMS error. All training that involved variance update was done via the L-BFGS method and the validation data was used for early-stopping the training process. Table 4.3 gives the per-frame log-likelihood and RMS error of the trained model on training, validation and testing data, using alignments derived from aligning articulatory data. 8-mixture component articulatory HMMs were employed to obtain the state alignment.

	Log-likelihood			RMS		
	train	val	test	train	val	test
A.hmm	1.928	2.080	1.492	1.663	1.688	1.704
B.mle-m	5.508	5.332	4.930	1.351	1.403	1.373
B.mle-v	4.986	3.806	4.209	1.580	1.606	1.605
B.mle-mv	10.536	10.445	10.134	1.416	1.470	1.434

Table 4.3: Per-frame trajectory log-likelihood and averaged RMS errors of different MLE trained models.

A few observations can be made from table 4.3:

1. All trained models achieve a higher log-likelihood than baseline A.hmm, regardless of data set, confirmed by paired t-test at $p < 0.005$ (n is the number of utterances involved, which is 368 for training set, and 46 for validation and testing set, respectively). This is in line with the expection of MLE training that the trained model will give a higher trajectory log-likelihood compared to untrained ones.

2. Variance update (B.mle-v) gives a higher log-likelihood on all data sets than baseline A.hmm, but the gain compared to mean update alone (B.mle-m) is smaller. Consequently, there is less RMS reduction of B.mle-v trained model

compared to B.mle-m trained model. This shows updating mean parameters is more effective than updating variance parameters.

3. In the columns of RMS errors it can be seen that all MLE updated models have a lower RMS error compared to A.hmm, confirmed by paired t-test at $p < 0.005$. However, the reduction of RMS error is not a simple linear function of the change in log-likelihood. B.mle-mv update, which generates the greatest improvement in log-likelihood, actually has less RMS reduction than B.mle-m trained models.

4. Surprisingly, the simple mean update (B.mle-m) that requires the least computational power, is most effective in terms of RMS reduction, with lower errors than those obtained by updating both mean and variance together (B.mle-mv). This is an interesting observation as updating both mean and variance is expected to be more effective for reducing the RMS errors, judging from the steady increase in log-likelihood. A possible explanation is that gradient-based update of the mean and variance is likely to find local minima while the analytic update of the mean vector is always a global solution.

4.4.2 Effectiveness of RMS Training

Following the MLE training experiment, in this section we examine the effectiveness of training using RMS objective function. The experiment condition was the same as in previous section, with the training criterion changed to RMS criterion (see 3.46 on page 35). The result is displayed in table 4.4. I report the RMS error as well as the per-frame log-likelihood on training, validation and testing data, using an aligned state sequence derived from 8-mixture articulatory HMMs.

	Log-likelihood			RMS		
	train	val	test	train	val	test
A.hmm	1.928	2.080	1.492	1.663	1.688	1.704
B.rms-m	2.097	1.791	1.343	1.321	1.378	1.372
B.rms-v	-3013.511	-3368.367	-3385.122	1.410	1.469	1.474
B.rms-mv	-20.641	-22.879	-21.947	1.255	1.341	1.324

Table 4.4: Per-frame trajectory log-likelihood and averaged RMS errors of different RMS-trained models.

Table 4.4 shows that, after RMS training, the trajectory log-likelihood of most models is lowered compared to the baseline HMM. This is not surprising as the RMS training objective function does not guarantee the increase of log-likelihood. And it is likely a model that has a lower RMS generation error on the training data has a lower "fit" in the sense of log-likelihood. Interestingly, the log-likelihood "reduction" in table 4.4 shows a different pattern to that of the MLE training in table 4.3 . Here the mean update (B.rms-m) has the least log-likelihood decrease from baseline HMM, followed by updating both mean and variance (B.rms-mv). However, updating variance alone (B.rms-v) drives the log-likelihood much lower than the other models, suggesting a radically different set of model parameters was chosen towards the end of training.

Different parameter updating methods again play differently when it comes to RMS reduction. B.rms-mv has the lowest RMS on validation and testing data, followed by B.rms-m (the difference is statistically significant at $p < 0.005$, $n = 46$). The variance-updated model B.rms-v, though achieves a lower RMS error than A.hmm, is not as effective as either B.rms-m or B.rms-mv. This is in line with the observation we made on MLE trained models and suggests that variance update of Trajectory-HMMs is not as effective as other parameter update methods for the articulatory inversion task.

4.4.3 MLE vs RMS Objective Function

Having analysed MLE and RMS training criteria individually, it is time to look at the difference between the two types of training objective functions. The RMS errors of MLE and RMS trained models are compared in figure 4.3. We observed that all RMS-trained models consistently achieve greater RMS error reduction than MLE trained ones. The differences are all significant at $p < 0.005$ level using one-tail paired test ($n = 368$ for training, $n = 46$ for validation and testing). This agrees with our expection that RMS training is better suited for this task. This also confirms the hypothesis that training will be most effective when the training objective and the error measure of the task match. Given the limited amount of data, RMS training criterion is better suited for the inversion task than the MLE one.

4.4.4 Effect of Parameter Update Method

Before moving on to further analysis, I would like to summarise the most effective parameter update method using what has been observed so far. Therefore in the later analysis I will focus on the result from this parameter update scheme to make the

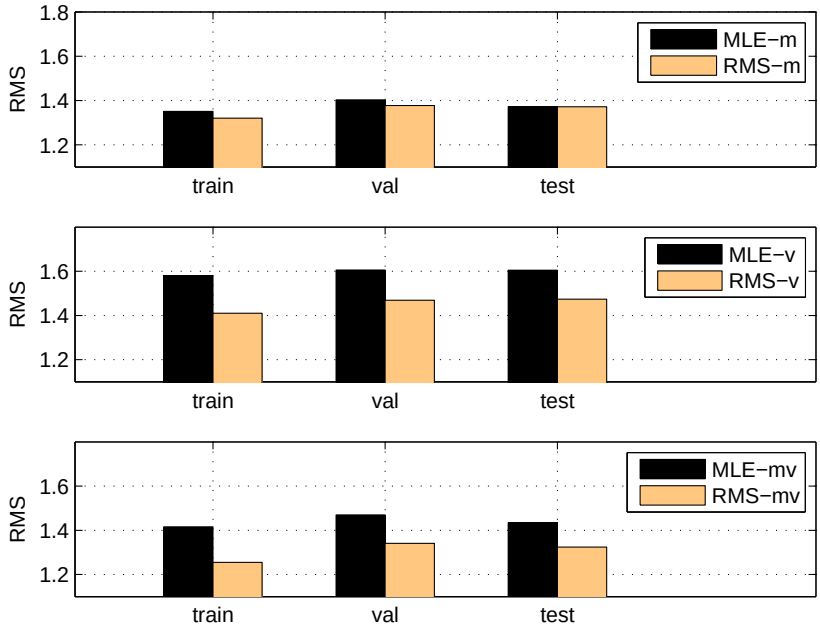

Figure 4.3: Comparison of RMS error reduction using MLE RMS training criteria.

presentation clearer.

From the previous section we observe that the RMS objective function generally produces better articulatory inversion model than MLE trained ones, judging from the RMS error on the forced-aligned alignment from articulatory data. Furthermore, looking at table 4.4, we find that updating both mean and variance simultaneously (B.rms-mv) gives best RMS reduction on the testing set, which is followed by mean update only (B.rms-m), with variance update being the least effective method. The difference is easier to see as illustrated in figure 4.4. There is no doubt rms-mv is consistently better than the other parameter update methods. I therefore decide to present further analysis using rms-mv trained models.

4.4.5 Articulatory Stream Updating: Jointly or Separately

The two main components of our HMM-based inversion system are realised as two set of HMMs: one for modelling the acoustic feature to be used for deriving an valid HMM state alignment from the unseen acoustic representation of an utterance, and one for modelling the articulatory data which will be trajectory trained to estimate smoothed articulatory trajectories based on the HMM state alignment provided by the

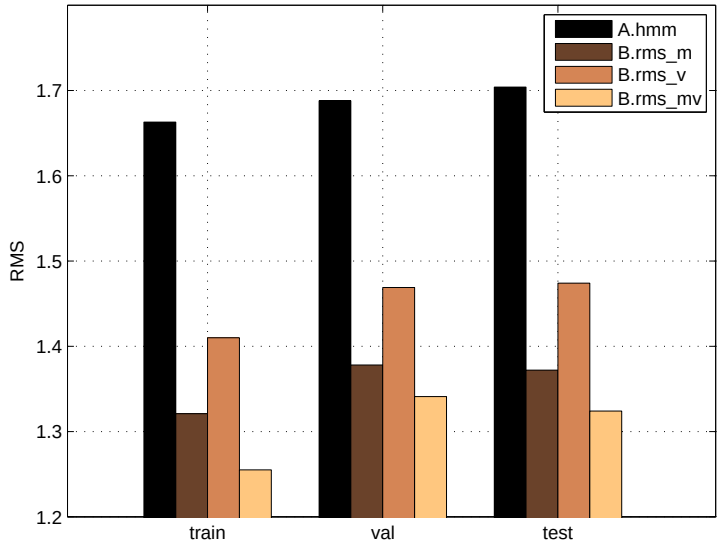

Figure 4.4: The RMS errors from different parameter update methods using RMS objective function. Group A is the Baum-Welched trained HMM, and group B are the trajectory updated HMMs.

acoustic HMMs.

The two set of HMMs are not independent from each other. They have the same HMM topology, and may be considered as an HMM with multiple observation stream. For instance, the HMM state corresponding to English vowel */ax/* is capable of generating both the acoustic realisation of the phone */ax/* and its articulatory realisation, given a suitable HMM state sequence.

As we have parallel acoustic-articulatory data, a choice arises as to how we should train the two HMM streams. Given the different nature of the acoustic and articulatory features, one may argue that training the two streams separately will yield a model that fits the data better. However, as the inversion quality largely depends on the quality of the HMM state alignment, which is derived from the acoustic HMM, I hypothesise that training the two streams together will help the inversion performance although the error measure (log-likelihood or RMS error) on the individual stream will be worse.

Both separate training (group B) and joint training (group C) were conducted. The RMS errors and per-frame trajectory log-likelihood of rms-mv trained models are given in table 4.5.

		RMS		Log-likelihood	
		B.rms-mv	C.rms-mv	B.rms-mv	C.rms-mv
A-Align	train	1.255	1.286†	-20.641	-8.009
	val	1.341	1.352	-22.879	-9.885
	test	1.324	1.316	-21.947	-8.316
S-Align	val	1.525	1.458	-44.702	-17.613
	test	1.482	1.403	-46.536	-15.876
S-Decode	val	1.712	1.663	-59.409	-26.421
	test	1.755	1.697	-64.738	-25.953

Table 4.5: Per-frame log-likelihood and RMS errors of jointly and separately trained models using rms-mv update method. The first column indicates the sources of HMM state alignment for generating output mean trajectory. The Alignment A-Align is obtained by force-aligning the articulatory data, S-Align alignment is obtained by force-aligning the acoustic data, and S-Decode gets the alignment by directly decoding the speech signal using an HMM decoder. The alignments were obtained using 8-mixture HMMs.

In table 4.5 A-Align refers to the alignment derived from aligning articulatory data using the HMM Viterbi algorithm. Since the articulatory data is not available in a real-world scenario, where the alignment is usually derived by an acoustic HMM either from directly decoding the speech signal (S-Decode) or doing an alignment (S-Align), it is more appropriate to interpret the result from A-Align as the upper-bound one can obtain from inversion system with a single Gaussian articulatory model. Looking at the A-Align result in table 4.5, the first thing we notice is that the RMS errors on the training and validation data are slightly higher for C.rms-mv, compared to B.rms-mv. This is not suprising as the jointly-trained model is supposed not to perform as well as separately trained model on the training data. However, the difference is only significant on the training set (marked with †, $p<0.005$), and is not statistically significant on the validation and testing sets, at $p=0.05$. The results from alignments derived from speech, S-Align and S-Decode, however, are encouraging. In all these cases the joint training gives a lower RMS error compared with separately trained models, and the differences are all significant at the $p < 0.005$ level of a paired one-tail t-test. This

confirms our hypothesis on the advantage of joint training.

The log-likelihood figures are unexpected. Although not directly optimised for log-likelihood, the figures in table 4.5 reveal that jointly-trained models all have higher log-likelihood than separately trained ones. Combined with the reduction in RMS error, I hypothesise that the effect of joint-optimisation acts as a regulariser, which penalises models with lower log-likelihood, thus, improves the "model correctness".

4.4.6 Quality of State Alignment

In this section let us look at how the quality of HMM state alignment can have an impact on the articulatory inversion performance. As described earlier in section 4.2.3, in the inversion stage, an HMM state alignment is needed by the parameter generation algorithm to generate the smoothed mean trajectories as the estimate of the articulatory movement for the unseen speech. In the proposed inversion system the HMM state alignment is derived from acoustic HMMs. This can be achieved by either force-aligning the phone label sequence, or decoding the unseen speech directly using the trained acoustic HMMs. The two types of alignment, represented by S-Align and S-Decode, have different level of accuracy. The forced alignment is likely to perform better.

A second factor that affect the accuracy of derived HMM alignments is the complexity of the acoustic HMMs. In HMM based speech recognition it is common practice to use Gaussian mixture as the state emission distribution for increased performance. Although the articulatory HMM is restricted to single Gaussian, there is no such restriction for the acoustic HMM that will be used to derive HMM state alignment from unseen speech. Therefore it is possible to increase the number of Gaussian component in the acoustic HMM prior to conducting the trajectory training. In our experiment, up to 8 Gaussian mixture components were employed to obtain a better HMM alignment.

The effect of using different number of Gaussian mixture components in the alignment deriving stage can be analysed by comparing the average RMS errors based on different alignments, which are reported in table 4.6. The inversion configuration is C.rms-mv, i.e. joint HMM training using RMS objective function to update both mean and variance parameters. Validation data was used to early stop the training process.

To study the effect of alignment derived from HMM with different number of mixture components, we present the average RMS error using C.rms-mv in table 4.6:

		1mix	4mix	8mix
train	A-Align	1.316	1.291	1.286
val	A-Align	1.392	1.367	1.352
	S-Align	1.536	1.467	1.458
	S-Decode	1.760	1.652†	1.663
test	A-Align	1.352	1.328	1.316
	S-Align	1.457	1.425	1.403
	S-Decode	1.891	1.794	1.697

Table 4.6: The average RMS errors computed from different alignments using C.rms-mv trained model.

In each row of table 4.6 we observe that the more mixture components used for alignment, the more RMS reduction is observed. For the same type of HMM alignment, RMS errors computed from alignments derived from 4-mixture acoustic HMMs are consistently lower than those from 1-mixture acoustic HMM. Similarly, RMS error computed from alignment derived from 8-mixture acoustic HMMs are lower than those from 4 mixtures. All the differences were confirmed to be significant by paired t-test at $p < 0.005$. The only exception is S-Decode on the validation set, where 8mix result (1.663) is higher than 4mix one (1.652, marked by †). Overall, this is in agreement with our assumption that even when the articulatory HMM uses single Gaussian output, the inversion system can still benefit from using HMM state alignment derived from HMMs with more expressive output distribution.

Moreover, looking at the figures column-wise, the use of phone label information (S-Align) consistently produce better alignment for inversion than relying solely on the information in the speech signal (S-Decode), confirmed at $p < 0.05$, which is as expected. The differences are further plotted in figure 4.5, using results from alignment of 8-mixture component HMMs. The difference between S-Align and S-Decode in figure 4.5 is much larger than that between S-Align and A-Align. This suggests that there is still much room for improving the quality of HMM state alignment derived from S-Decode. In a practical inversion system, it would be possible to first produce a phone label sequence from a sophisticated ASR component known to have good phone accuracy, such as a Neural-network recogniser (Robinson, 1994), then to force-align the phone label sequence to get a more accurate HMM state alignment for the purpose

of inversion.

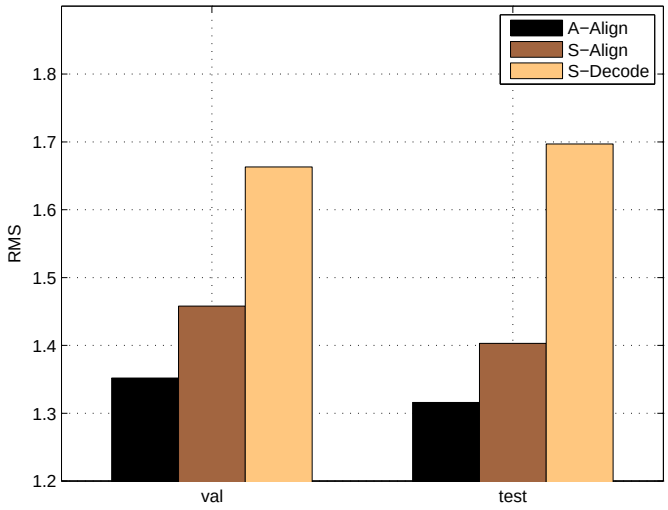

Figure 4.5: RMS errors on validation and test data using different types of alignment suggest there is still room for improving the quality of alignment derived from S-Decode.

4.4.7 Effect of Dynamic Coefficients

One area that has not received much attention in the literature on HMM modelling is the choice of formula for calculating the dynamic coefficients. In speech recognition, the most common way of computing delta and delta-delta coefficients are given as:

$$\Delta \mathbf{c}_t = \sum_{\tau=1}^{n} \frac{\tau(\mathbf{c}_{t+\tau} - \mathbf{c}_{t-\tau})}{2\sum_{k=1}^{n} k^2} \tag{4.5}$$

$$\Delta^2 \mathbf{c}_t = \sum_{\tau=1}^{n} \frac{\tau(\Delta\mathbf{c}_{t+\tau} - \Delta\mathbf{c}_{t-\tau})}{2\sum_{k=1}^{n} k^2} \tag{4.6}$$

where $\mathbf{c}_t$ is the static feature at frame t, n is the half size of the window used to compute the dynamic feature at frame t. $\Delta\mathbf{c}_t$ and $\Delta^2\mathbf{c}_t$ are the calculated delta and delta-delta feature for frame t respectively. In ASR a typical choice for n is 2, and this yields a 5-frame window for calculating delta features for frame t, and a 9-frame

window for calculating delta-deltas:

$$\begin{aligned}\Delta\mathbf{c}_t =& -0.2\mathbf{c}_{t-2} - 0.1\mathbf{c}_{t-1} + 0.0\mathbf{c}_t + 0.1\mathbf{c}_{t+1} + 0.2\mathbf{c}_{t+2} \\ \Delta^2\mathbf{c}_t =& 0.04\mathbf{c}_{t-4} + 0.04\mathbf{c}_{t-3} + 0.01\mathbf{c}_{t-2} - 0.04\mathbf{c}_{t-1} - 0.1\mathbf{c}_t \\ & - 0.04\mathbf{c}_{t+1} + 0.01\mathbf{c}_{t+2} + 0.04\mathbf{c}_{t+3} + 0.04\mathbf{c}_{t+4}\end{aligned}$$

We will refer this type of window, which was implemented in HTK toolkit, to as *dw9* type window.

Another type of dynamic coefficient, derived from quadratic function as implemented in the HTS toolkit, is given as:

$$\Delta\mathbf{c}_t = \frac{\sum_{\tau=-n}^{n}(\mathbf{c}_{t+\tau} - \mathbf{c}_t)}{\sum_{\tau=-n}^{n}\tau^2} \tag{4.7}$$

$$\Delta^2\mathbf{c}_t = 2\frac{\sum_{\tau=-n}^{n}\tau^2\mathbf{c}_{t+\tau} - \frac{1}{N}(\sum_{\tau=-n}^{n}\tau^2)(\sum_{\tau=-n}^{n}\mathbf{c}_{t+\tau})}{\sum_{\tau=-n}^{n}\tau^4 - \frac{1}{N}\sum_{\tau=-n}^{n}\tau^2} \tag{4.8}$$

here $N = 2n+1$ is the width of the window used to compute the dynamic feature at frame t. Setting n to 1 will produce a 3-frame window for calculating both delta and delta-deltas for frame t:

$$\begin{aligned}\Delta\mathbf{c}_t =& -0.5\mathbf{c}_{t-1} + 0.0\mathbf{c}_t + 0.5\mathbf{c}_{t+1} \\ \Delta^2\mathbf{c}_t =& 1.0\mathbf{c}_{t-1} - 2.0\mathbf{c}_t + 1.0\mathbf{c}_{t+1}\end{aligned}$$

We will refer this type of window to as *dw3* type window.

The 9-frame window obviously requires more computation than the 3-frame window. For conventional HMMs, the dynamic features are computed once and then fixed in both training and decoding, so the difference in computational cost can be ignored. The dynamic coefficient window in Trajectory-HMM formula actively participates in every step of trajectory training in the form of matrix $\mathbf{W}$ (3.16), and the dw9 type window will make the training time longer than that of dw3 type window. Furthermore, in the Trajectory decoding algorithm (section 3.4 on page 36), the decoder will look ahead a number of frames equal to half window size, and the dw9 type window will pull in many more paths for looking ahead than the dw3 type window, which will slow down the decoding process considerably.

Traditionally, the dw9 type of coefficient was used in speech recognition, and the dw3 type of coefficient found its use in HMM-based speech synthesis, but not vice versa.

		1mix		4mix		8mix	
		dw3	dw9	dw3	dw9	dw3	dw9
val	A	55.93	54.52	48.94	46.89	45.13	41.95
	S	57.49	54.17	52.54	45.55	51.13	41.81
	A+S	43.29	39.41	40.61	35.95	40.32	34.75
test	A	53.13	51.26	46.87	43.90	43.25	41.58
	S	55.71	51.13	51.78	43.00	47.97	42.03
	A+S	39.25	36.67	37.44	32.92	36.22	31.05

Table 4.7: Phone error rates from the acoustic HMMs using dw3 and dw9 type dynamic coefficient. Three feaure sets are used: articulatory feature (A), acoustic feature (S) and the combined articulatory-acoustic feature (A+S).

To see the difference between the two on the articulatory inversion task, two inversion system using C.rms-mv configurations were built using the data from the female speaker fsew0, one with dw3 type of dynamic coefficient and one with dw9 type of coefficient. The phone error rates on the training, validation and testing sets using conventionally trained HMMs from both systems are given in table 4.7:

The phone error rates in table 4.7 suggest that the dw9 type of dynamic coefficient, when it comes to recognition task, is consistently better than the dw3 type coefficient. The average reduction in phone error rate using dw9 based system compared to dw3 based system is 2.19% for articulatory feature (A), 6.49% for acoustic feature (S) and 4.40% for the combined articulatory-acoustic feature (A+S). However, the articulatory inversion results, as presented in table 4.8, show a slightly different trend. Here the dw3 type coefficient has a performance advantage over the dw9 type coefficient using A-Align and S-Align alignments. But the dw9 coefficient produces a lower RMS error when HMM alignment was derived directly from speech (S-Decode). Combined with the finding in the phone error rate, we conclude that the dw9 type of coefficient gives better discrimination for speech and, consequently, yields more accurate alignment in S-Decode configuration.

Although both the dw3 and dw9 types of coefficient give similar RMS figures for S-Decode alignment, we observe a notable difference in the recovered trajectories:

		dw3	dw9
train	A-Align	1.286	1.377
val	A-Align	1.352	1.440
	S-Align	1.458	1.516
	S-Decode	1.663	1.567
test	A-Align	1.316	1.407
	S-Align	1.403	1.487
	S-Decode	1.697	1.671

Table 4.8: The average RMS errors computed from dw3 and dw9 dynamic window types, using C.rms-mv trained model with 8-mixture acoustic HMMs.

inspection of the recovered articulatory trajectories shows that the dw9 system gives a noisy, jigsaw-like result which, when compared to the smoother output from the dw3 system, is inferior (see figure 4.6). This is contrary to our assumption that a longer dynamic window is capable of capturing long range correlations between frames, and should produce smoother mean output trajectories.

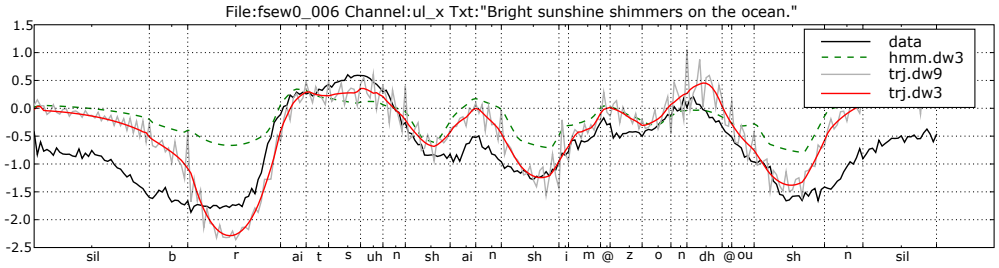

Figure 4.6: Recovered trajectory for the movement of upper lip in x co-ordinate ul_x of a test utterance fsew0_006. The trained Trajectory-HMM (hmm.dw3) shows a closer fit to the data than the baseline HMM (trj.dw3), with state alignment derived from a 8-mixture jointly MLE-m trained 2-stream HMM. The light gray trajectory of (trj.dw9) shows the noisy effect of using 9-frame dynamic window.

To investigate the possible cause of the smoothness result produced from the dw9 type dynamic window, the covariance matrices for a segment of the utterance corresponding to silence are plotted in figure 4.7, using both dw3 and dw9 dynamic win-

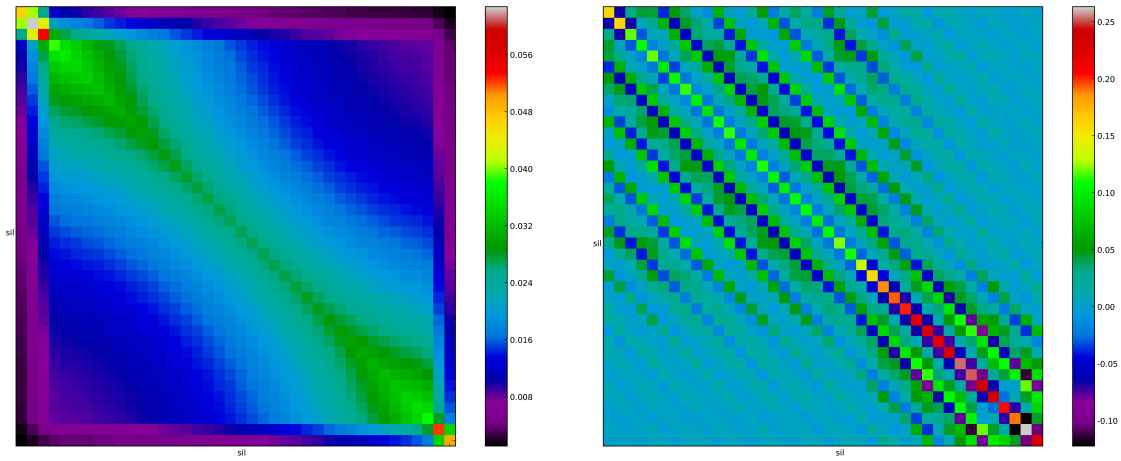

Figure 4.7: Covariance matrices for a segment corresponding to silence in the utterance fsew0_006. The left is from a dw3 type dynamic window and the right is from a dw9 type dynamic window. The dw9 covariance matrix shows many negative elements in the off-diagonal position.

dows.

The main difference between the two matrices is the distribution of off-diagonal elements. The covariance matrix produced from a dw3 type dynamic window shows a stronger diagonal structure, which decreases steadily towards the off-diagonal part of the matrix. The covariance matrix from a dw9 type dynamic window shows a different patten: while most off-diagonal elements are close to zero as expected, there are many near diagonal elements that are negative. The existence of negative elements in a covariance matrix generally suggests some degree of anticorrelation. Inspecting samples generated from the dw9 system reveals a similar pattern of jaggedness as observed in the mean output trajectory. Referring to (3.24) we can see the covariance matrix $\mathbf{P_q}$ is the inverse of $\mathbf{R_q}$, which is a linear transformation of the original HMM precision matrix $\mathbf{\Sigma_q}^{-1}$. However, it is still an open question why the linear transform $\mathbf{W}$ used in a dw9 window produces an inverse covariance matrix with negative off-diagonal elements, and how this is related to the structure of the dynamic coefficients.

So far the experimental results suggest that both types of dynamic coefficients have their own strengths and weaknesses. The dw9 coefficients show increased discriminative power in phone decoding and give improved accuracy for HMM alignment derived from speech alone, compared to the dw3 system. The drawback of the dw9 coefficient, however, is the jaggedness of the recovered articulatory trajectories, which renders the dw9 type useless in real-world inversion scenario without some kind of post-smoothing

(such as applying a low-pass filter to the estimated trajectory as in (Richmond, 2002)). The cause behind this difference, however, is unclear.

The proposed inversion system utilises two streams, articulatory and acoustic, each modelled by a set of HMMs. The articulatory stream is used to build the Trajectory-HMM inversion model to generate the smoothed mean trajectories from given HMM state sequences. The acoustic stream is used to train a set of conventional acoustic HMMs to derive an accurate HMM state alignment from the acoustic representation of the speech.

Given the functional difference between the two streams, it may be profitable to use the dw3 type of coefficient for the articulatory stream, and to use the dw9 type of coefficient for the acoustic stream. The hypothesis is that the use of dw9 coefficient in the acoustic stream will help produce more accurate HMM state alignment for inversion. The inversion result from this arrangement compared to either dw3 based or dw9 based systems is presented in table 4.9, where the combined use of dynamic coefficient is denoted as dw3-9.

		dw3	dw9	dw3-9
train	A-Align	1.286	1.377	1.269
val	A-Align	1.352	1.440	1.354
	S-Align	1.458	1.516	1.472
	S-Decode	1.663	1.567	1.575
test	A-Align	1.316	1.407	1.324
	S-Align	1.403	1.487	1.411
	S-Decode	1.697	1.671	1.678

Table 4.9: The average RMS errors computed from dw3, dw9 and dw3-9 dynamic window configurations,using C.rms-mv trained model with 8-mixture acoustic HMMs.

The result in table 4.9 confirms this hypothesis, giving a lower RMS inversion error than the dw3 configuration when alignment was derived from speech only, with smooth trajectories compared with the dw9 configuration (which has slightly lower RMS errors for the S-Decode alignment).

4.4.8 Overfitting and Regularisation

Like many probabilistic models the Trajectory-HMM can suffer from overfitting on training data. As a result, the trained model will have less generalisation power when applied to unseen data. All the training in this thesis used a separate validation set to monitor the training process and to stop the training if overfitting is observed. One possible solution is to add a so-called *weight decay* term to the error function, which is analogous to ridge regression used for linear models:

$$E = (\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})^T (\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}}) + \lambda (\sum_i \sigma_i^2) \tag{4.9}$$

where $\lambda >= 0$ is a tuning parameter that is estimated using cross-validation, and σ_i is the i^{th} variance parameter of the model. Larger values of λ will tend to shrink the weights toward zero, therefore the weight decay penalty term causes the weights to converge to smaller absolute values than they otherwise would (Hastie et al., 2001).

Another possible regularisation method is to consider the global variance in addition to the error term. Toda and Tokuda (2005) showed that an improvement to the naturalness of synthetic speech was observed in a perceptual evaluation of a TTS task by considering a global variance constraint in training.

4.4.9 Final Result

We have discussed various factors that can influence the performance of the inversion system: the effectiveness of training objective function, the method of parameter update, joint or separate optimisation, the quality of HMM state alignment, and the estimation of dynamic coefficients. The conclusion is that, for the particular articulatory inversion task, the RMS training criterion is superior to the MLE training objective function, due to the match of the error measure across training and testing condition. Updating both mean and variance parameters has an advantage over updating either mean or variance parameter alone, albeit adding computational expense. Furthermore, joint parameter optimisation of the articulatory-acoustic two stream HMMs was found to have an edge over the separate parameter estimation. As for the quality of HMM state alignment, 8-mixture component acoustic HMMs were found to produce a more accurate alignment than 1 or 4-mixture component HMMs. In addition, the dw3 type of dynamic window was observed to give a smoother output trajectory compared to the dw9 type, and further improvement can be obtained by using the two types of dynamic

coefficients separately.

Combining all these findings, the best inversion configuration was C.rms-mv, which jointly optimises both mean and variance parameters using RMS training criterion, with all the alignments being derived from 8-mixture acoustic HMMs.

At present there are two speakers in the publicly available MOCHA TIMIT data: the female speaker `fsew0` as used in this chapter and a male speaker `msak0`. The final articulatory inversion system was set up to run on data from both speakers using C.rms-mv trained model and 8-mixture acoustic HMMs, and the inversion results are presented in table 4.10.

dynamic window		dw3		dw3-9	
speaker		fsew0	msak0	fsew0	msak0
train	A-Align	1.286	1.236	1.269	1.236
val	A-Align	1.352	1.350	1.354	1.323
	S-Align	1.458	1.458	1.472	1.421
	S-Decode	1.663	1.684	1.575	1.625
test	A-Align	1.316	1.357	1.324	1.364
	S-Align	1.403	1.410	1.411	1.414
	S-Decode	1.697	1.733	1.678	1.712

Table 4.10: The average RMS errors computed from different speakers using C.rms-mv trained model.

The lowest inversion error from the the speech signal alone is 1.678 mm on testing data for speaker fsew0, which compares well with an error of 1.62 mm obtained when using an MLP for direct acoustic-articulatory mapping (Richmond et al., 2003), especially since in this approach the articulatory trajectory is generated using single Gaussian densities. Adding the phone label information to produce the S-Align alignment, the average RMS error can be as low as 1.403 mm, using dw3 type dynamic coefficient. More recently, the TMDN approach with many more free parameters has resulted in a decreased RMS error of 1.40 mm on this data set (Richmond, 2007).

Finally, the per-channel RMS errors from the best inversion system configurations are given in table 4.11.

	S-Align		S-Decode	
articulator	fsew0	msak0	fsew0	msak0
ll_y	2.065	1.491	2.579	1.814
ll_x	1.059	1.177	1.170	1.264
tb_y	1.872	2.082	2.314	2.375
ul_x	0.864	0.696	0.976	0.761
ul_y	0.895	0.782	1.116	0.989
tt_y	2.248	2.344	2.459	2.918
v_x	0.402	0.583	0.555	0.674
tb_x	2.085	2.001	2.405	2.652
li_x	0.775	0.549	0.832	2.575
li_y	1.124	0.861	1.291	0.946
v_y	0.386	1.088	0.467	1.136
tt_x	2.242	2.203	2.598	3.068
td_y	1.718	1.952	2.467	2.466
td_x	1.907	1.931	2.258	2.332
average RMS (mm)	1.403	1.410	1.678	1.712

Table 4.11: The final per-channel RMS errors (mm) for two speakers in MOCHA-TIMIT data. The dynamic window type was dw3 for S-Align alignment and dw3-9 for S-Decode alignment. The training configuration was C.rms-mv.

4.5 Summary

Recent interest in the use of HMM-based systems for speech synthesis, and the development of the the Trajectory-HMM, has resulted in a resurgence of interest in the development of unified models for speech recognition and synthesis with a principled statistical basis. In this chapter we demonstrate that a common generative model for acoustic-articulatory data can be used for both recognition and synthesis of acoustic and articulatory signals with appropriate marginalisation. In particular, HMM state sequences are used as the hidden representation of both acoustic and articulatory channels. The benefit of introducing an intermediate layer into existing framework can also be found in speech recognition (Russell and Jackson, 2005), where the relationship between symbolic (phonetic) and surface (acoustic) representation of speech is regulated by an 'articulatory' representation under a multi-level segmental HMM framework.

The task developed in this chapter served as a testbed for trying out new algorithms for Trajectory-HMMs. It gave us an opportunity to systematically study various aspects of trajectory training. The experiments in this chapter showed that the newly proposed RMS training objective function was found to be consistently better than the MLE training criterion for the inversion task. The result also revealed that training such a model jointly (Group C) results in more accurate generation of articulatory trajectories, compared with separately trained models (Group B). The final inversion result compares favourably to models with much more free parameters.

Despite its theoretical attractions, the Trajectory-HMM has a major limitation at the present time. In the absence of a "Trajectory-HMM Baum-Welch" algorithm, training models with multiple mixture components is prohibitively expensive. Thus, in this work, the articulatory inversion model was limited to Trajectory-HMMs with single Gaussian densities. In the HTS speech synthesis system, this limitation is implicitly addressed through the use of detailed context. In this work we have used monophone models, and it is clear that the use of context-dependent models is worth investigating. The use of more detailed modelling units will be explored in chapter 5.

Although there are significant technical challenges related to Trajectory-HMM training, there are several advantages to pursuing the Trajectory-HMM as a unified model for synthesis and recognition. The fact that existing software frameworks for HMMs may be reused provides a platform for experimentation, and a principled, efficient way to initialise models (using conventional HMM parameter estimation). In the case of articulatory-acoustic modelling, the use of duration modelling approaches

developed in HMM-based speech synthesis enables articulatory movement to be generated without the need for acoustics, and it is also possible to apply speaker adaptation approaches used successfully in recognition and synthesis.

Chapter 5

Trajectory Triphone Modelling

The basic modelling units of the Trajectory-HMMs explored so far are monophones where each phoneme is modelled by a multi-state HMM with single Gaussian output function. While this keeps the model relatively compact and easy to train, the limited number of model parameters may be inadequate to model the more detailed structure of speech. In the articulatory inversion experiment of Chapter 4 it was observed that the final RMS inversion error, though much lower than that produced by conventionally trained HMMs, is still inferior to figures obtained by more sophisticated models employing a richer structure with more parameters, such as the Mixture Density Networks (Richmond, 2007). This suggests there is still room for improvement, considering the RMS reduction obtained by using a rather small number of free parameters in a monophone system. If that is the case, i.e., the model turns out to be too simple for the task, one solution could be to increase the number of model parameters without substantial infrastructure changes. Moreover, a model is of little practical use if it can not scale to large scale tasks. Therefore, it is natural to wonder if further performance gains can be obtained by increasing the modelling details of current Trajectory-HMM framework. In this chapter, I will explore the use of triphone models as a way to increase the modelling capacity of the Trajectory-HMM. Besides using sub-word models, it is also possible to employ more expressive state output distributions, which will be discussed in Section 5.1. Section 5.2 introduces the tree-based triphone clustering algorithm as implemented in HTK. Trajectory triphone training is then covered in Section 5.3. Experimental results are given in Section 5.4, and a conclusion is drawn in Section 5.5.

5.1 Increase Modelling Details in HMMs

Generally speaking, there are two ways of increasing the modelling power of an HMM system: enriching the state output distribution or using more detailed modelling units. In the former approach, the single Gaussian output distribution with diagonal covariance can be replaced by Gaussian mixture models, full covariance Gaussians or variants of neural networks, all of which have proved to deliver better performance in speech recognition. Among these, GMMs are the most popular choice for output distribution as it only requires modest modification to the training and decoding algorithms of HMMs. Due to the effectiveness and ease of use of the GMM, it has become the standard choice for output distribution in today's HMM-based ASR systems.

For the Trajectory-HMM, using Gaussian mixture models as the state output distribution brings challenges in both training and decoding. The parameter update algorithm presented in Section 3.3 of Chapter 3 assumes that the Gaussian component at each state has been fixed, which is similar to Viterbi training of HMMs. For mixture models, the contribution of each mixture component at each state needs to be considered in training and the likelihood function is the sum of all possible mixture component combinations which, without the conditional independence assumption of HMMs, will grow exponentially with the length of the training utterance. To properly update the parameters in a mixture model, a trajectory equivalent of Baum-Welch, probably with approximation, needs to be discovered. The lack of such algorithm makes trajectory training of Gaussian mixture components a formidable task at the time of writing.

Similarly, the size of the search trellis will experience an exponential growth when switched to mixture modelling. Considering the complexity of the monophone single Gaussian decoder developed in Chapter 3, it will be impractical to perform mixture model decoding without efficient approximation method.

A second way to use detailed modelling structure is to replace the simple modelling units (monophone) with more detailed ones, such as biphone or triphone models. A triphone is basically a monophone enriched with left and right phoneme context. For example, the monophone sequence:

```
sil dh i s w @ z sil
```

can be expanded into the folowing triphone sequence in HTK:

```
sil sil-dh+i dh-i+s i-s+w s-w+@ w-@+z @-z+sil sil
```

where a triphone unit `dh-i+s` shows an `/i/` phone enriched with left context `/dh/`

and right context /s/. Based on the surrounding phonetic context, a monophone model can be expanded into a number of triphones with different left and right contexts. The enhanced context provides a finer level of modelling constraints over monophone models, especially when the monophone output distribution is not sophisticated enough to cope with the variation in the observation space caused by contextual changes.

Moving from monophone to triphone models, the number of phone models grows significantly, which causes some problems. First, most triphone models in the training data only appear a handful of times, so there is not enough data to make a reliable estimate of the triphone parameters (the under-training problem). Second, not all triphone models derived from monophones can be found in the training data. Thus triphone models unseen in the training phrase will cause problems in recognition, or when force-aligning new data. Tree-based clustering and parameter sharing technologies have been developed to solve the problem of training unseen triphones and to improve the trainability of the model (Young et al., 1994). Before training begins, a list of contextual questions, usually linguistically derived, is presented to the clustering algorithm which will build a decision tree according to the questions. During training, a sophisticated parameter sharing (also called parameter tying) method ties the parameter of acoustically similar phones together so that the output observations for those tied models can be used to estimate the tied parameter. In the recognition stage, the parameter of unseen triphone models can be identified to the closest tried parameter by tracing through the tree. Nevertheless, decoding a triphone system is much more difficult as triphone context need to be examined at word boundaries. The reader is referred to (Odell, 1995) for more details on that subject.

5.2 Triphone Clustering and Parameter Sharing

The triphone building method used in this chapter is the tree-based clustering method as proposed in (Young and Woodland, 1994), and implemented in the HTK toolkit. In HTK's decision tree clustering algorithm there are two commands working together to control the clustering process: RO and TB, where two thresholds f_{RO} and f_{TB} are manually chosen to control the clustering process. First a file with statistics of occupation of each state must be generated by running the HERest tool. Then along with a list of manually provided questions the RO command is used with a threshold f_{RO} to remove outlier states during clustering: this effectively controls which questions are allowed to be used to split each node. Running the TB command will perform a

top-down clustering of the states or models specified in its argument. The clustering starts by placing all items in a single root node and then choosing a question from the current set to split the node in such a way as to maximise the likelihood of a single diagonal covariance Gaussian at each of the child nodes generating the training data. This splitting continues until the increase in likelihood falls below threshold f_{TB} or no questions are available which do not pass the outlier threshold test (Young et al., 2006). Obviously the choice of thresholds f_{RO} and f_{TB} will affect the number of parameters in the final tied decision tree model. If f_{RO} is too small, states that does not occur many times in the training will be included in the clustering process and will produce a large tree. Conversely, if the f_{TB} is set to be too large, very few states will be used in the clustering and, as a result, a smaller tree with less parameters will be produced. It is worth mentioning that it is not healthy for the final tree to have either too many parameters (larger tree) or too few parameters (smaller tree). To many parameters, or tree branches, will produce a tree that fits well on the training data but has otherwise bad generalisation ability. Moreover, a tree with too many parameters is easier to suffer under-training with limited amount of training data. Tree with too few parameters, on the other hand, will be too generalised to perform well on unseen data. Thus a decision tree model that balances between model complexity and generalisation ability is likely to perform well. Unfortunately, this also means some tedious trial-and-error experiments are needed to find the optimal combination of f_{RO} and f_{TB}.

5.3 Triphone Parameter Update

Similar to building a monophone HMM that starts with a Baum-Welch trained model, the triphone Trajectory-HMM is derived from a Baum-Welch trained triphone HMM. A monophone single Gaussian output HMM was first trained, then the monophone models were expanded into triphones and decision-tree clustering algorithm was executed to pool acoustically similar triphones together and a compact triphone model was then generated and submitted to a few Baum-Welch updates, resulting in a triphone baseline model.

To perform the trajectory parameter update, the number of Gaussian mixture components in baseline triphone was increased to 8 and was used to derive a triphone state alignment for training.

In a tied-state triphone system, the state output distribution from a few different triphone models can be tied together to share the same physical emitting state. This dra-

matically reduces the number of free parameters in the model. During trajectory training those tied structures are honoured and the shared Gaussian parameters are updated accordingly. Theoretically speaking, the training complexity of triphone Trajectory-HMM only depends on the number of physical emitting states, not the total number of (logical) triphone states. In practice, however, the size of the triphone Trajectory-HMM is restricted by the physical memory available, due to the fact that the matrix storage for computing the gradient vectors (3.43) grows quadratically with the number of Gaussian components.

5.4 Experimental Set-up and Result Analysis

Trajectory training using tied triphone parameters was implemented, so the parameter tying structure from HTK trained model can be honoured and updated accordingly. I choose to only test it on the articulatory inversion task. Although it is possible to modify the decoder to do phone recognition using triphones, this is not done due to the time constraint of the project.

The experiment condition was the same as that in Chapter 4. First a monophone, single Gaussian output HMM system was trained using HTK. Then tree-based state clustering was conducted using the `HHEd` tool from HTK. Then triphone HMM alignment was derived from HTK trained triphone HMMs, whose Gaussian output mixture components have been increased to 8-mixture. Following that, trajectory training was executed with both mean and variance updated. And finally, articulatory trajectory were estimated from triphone state alignment.

The monophone model as used in Chapter 4 has 45 phones each being modelled by 3 emitting states, yielding a total of 135 emitting states with single Gaussian output. After expanding the monophone model to include all triphone models in the 368 training utterances, there are 5450 triphone models and a total 16350 emitting states, many of which only occur once or twice in the training data. Therefor triphone clustering and parameter tying are required to combat the data sparseness problem.

Different combinations of `RO` and `TB` thresholds were investigated to give varied number of tied triphone states. Then a few iteration of Baum-Welch estimation were carried out to update the triphone model parameters. The training, validation and testing split is the same as in Chapter 4, which is in accordance with the practice of other published result on the same data (Richmond, 2002).

Table 5.1 lists the number of emitting states of different clustered triphone models

#states	f_{RO}	f_{TB}	#states	f_{RO}	f_{TB}	#states	f_{RO}	f_{TB}	#states	f_{RO}	f_{TB}
194	295	280	510	100	60	840	60	20	1175	35	50
210	250	290	539	90	90	872	55	40	1197	30	80
240	205	260	570	75	130	900	30	170	1230	25	110
270	195	130	601	65	150	930	40	100	1262	40	0
300	165	190	629	85	20	958	30	150	1296	15	180
330	145	180	661	45	190	987	30	140	1322	30	50
360	125	200	690	75	30	1022	30	130	1359	20	120
390	125	140	720	55	120	1040	25	160	1408	20	110
420	105	170	751	40	170	1076	25	150	1440	25	60
450	85	200	799	45	130	1109	35	70	1493	25	50
480	115	30	808	60	40	1137	35	60			

Table 5.1: RO and TB thresholds and the resulting number of tied triphone states.

by varying RO and TB thresholds. The smallest model has 194 physical emitting states, only marginally bigger than the monophone system, and the largest one has 1493[1].

In the rest of this chapter I will present two triphone experiments. In section 5.4.1 triphone HMMs trained using standard Baum-Welch algorithm will be used as the inversion models to derive the articulatory estimate from the HMM state sequences returned by acoustic HMMs. The result will be compared with that of the trajectory updated monophone HMMs in chapter 4. The performance of trajectory updated triphone HMMs will be evaluated in section 5.4.2.

5.4.1 Triphone Baseline HMM

The triphone HMMs, trained using standard Baum-Welch algorithm on articulatory data, were used as inversion models to get the RMS generation error on training, validation and testing sets. This will give us an idea if the increase of model parameters (the number of physical emitting states) will result in better inversion performance, even without trajectory parameter updating.

The training started from a two-stream, monophone model with single Gaussian emitting states. Then the monophone units were expanded into triphone using the

[1] Although the trajectory trainer developed can update triphone tied-states, the training experiments were limited by the physical memory available, as triphone updates requires much more online storage than monophone models. I was able to train triphone models up to 1500 tied states on hardware accessible, and will report results based on that figure. Please note this is not a theoretical limitation but rather a practical one, and could be conquered by designing an offline training algorithm that utilises external storage.

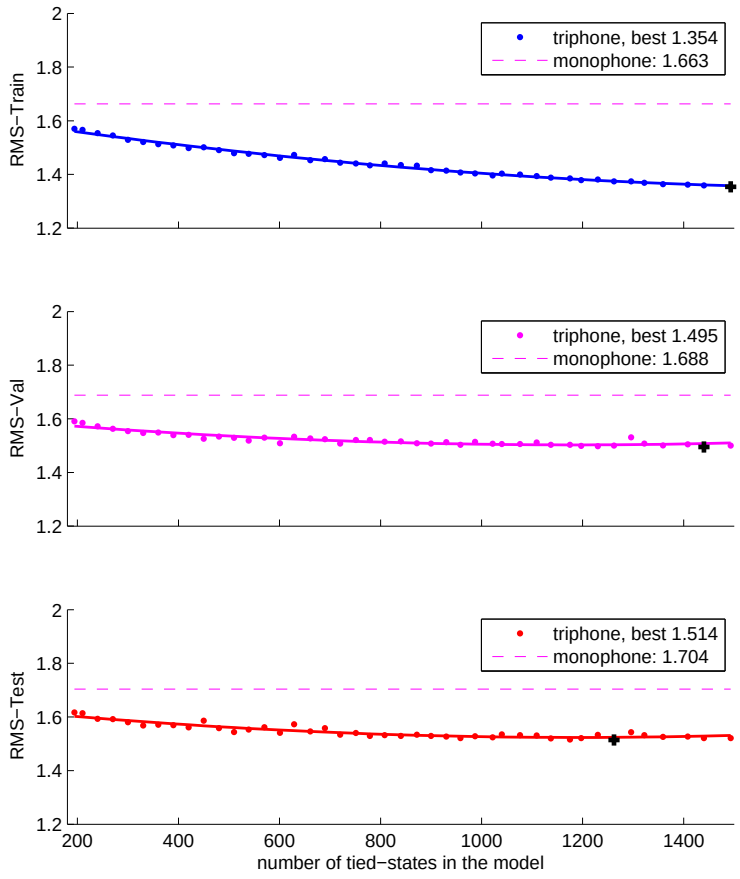

Figure 5.1: The averaged RMS error of baseline triphone models with varied number of tied states. Above: training data, middle: validation data, bottom: testing data. The alignment on val/test data were derived by force aligning the articulatory data using an 8-mixture component articulatory HMMs (A-Align). Points marked with + are the lowest triphone RMS error on corresponding data set.

HHEd tool, and state occupying statistics on training data were collected. Following that, the decision tree clustering was conducted with given RO, TB thresholds to produce a compact triphone model. Finally, the Gaussian mixture components in the acoustic stream was increased to 8-mixture for deriving triphone HMM state alignment for inversion.

Figure 5.1 plots the RMS errors of different baseline triphone models with varied number of emitting states on training, validation and testing data respectively. The triphone alignments used for calculating the mean trajectories were derived by force-aligning the articulatory data using the same triphone baseline HMM but with 8-mixture Gaussian output. The dashed lines give the corresponding RMS errors from the monophone HMMs tested under the same condition (with monophone alignment, see table 4.3 on page 68). The dots in the figure represent individual RMS errors, and the tick lines are the result of quadratic fitting the individual points. It is easy to see that all triphone models, regardless of the number of tied states, have a lower RMS error than monophone model on corresponding data set. And the improvement are statistically significant at $p < 0.005$. It seems models with more emitting states tend to have a lower RMS error than those with fewer. This trend is more obvious on training set but less so on validation and testing sets, suggesting overfitting occurred at around 1200 states, as seen in figure 5.2.

For the 43 different triphone models listed in table 5.1, the average RMS reduction compared to corresponding monophone model are 0.22mm, 0.17mm and 0.16mm on training, validation and testing set respectively. This suggests increasing model parameters is an effective way of enhancing the trajectory modelling power of Baum-Welch trained HMMs.

In a real-world scenario the alignments for generating mean trajectories are not obtained from articulatory HMMs, but by using acoustic HMMs in either force-aligning or decoding mode. It is more useful to look at figure 5.3, where RMS errors are computed by force aligning (S-Align) and decoding (S-Decode) using acoustic HMMs with 8-mixture Gaussian output. Again, dashed lines in figure 5.3 represent the corresponding RMS errors from monophone HMMs. (see table 4.6 on page 75). There prefix T- and M- are for triphone and monophone respectively. The results in figure 5.3 confirm that, even using alignment derived from speech, the triphone models still perform better than monophone models by a wide margin. However, overfitting in terms of number of model parameters seems to impact the result from S-Decode alignment earlier than S-Align. The RMS errors of S-Decode on both validation and testing set go up quicker

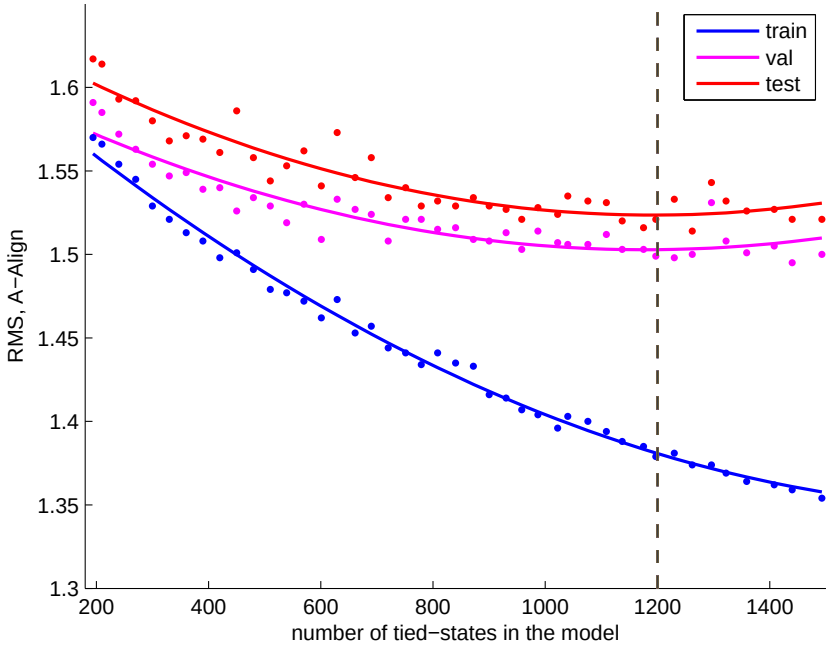

Figure 5.2: RMS errors per tied-state number on different data sets. The RMS errors on validation and testing sets stop to decrease at around 1200 states, indicating overfitting occured with Baum-Welch trained triphone HMMs.

than those of S-Align. A possible explanation could be that alignment from decoding speech, by its very nature, is inaccurate than force-alignment. And the overfitting impact from increasing the number of model parameters could kick in earlier. The best RMS errors obtained on test set are 1.603 using 779 emitting states, and 1.757 using 510 emitting states, for S-Align and S-Decode alignments respectively. Both are significantly lower (statistically confirmed) than Baum-Welch trained monophone model's 1.765 and 1.878 (table 4.6). And the averaged RMS reduction triphone models compared to corresponding monophone models on test set are 0.14mm for S-Align alignment and 0.09mm for S-Decode alignment, respectively.

Overall, the experimental results on Baum-Welch trained triphone systems confirm that increasing modelling detail can be an effective way of reducing RMS mean generation error of an HMM-based system, and the improvement is significant. This is attractive in situation where a complete trajectory parameter updating is not affordable.

5.4.2 Effect of Trajectory Parameter Update

Given the performance gain from the Baum-Welch trained triphone models, it is expected that similar improvement can be obtained on trajectory trained triphone models as well. In this section, let us look at the effect of trajectory training on triphone systems. Among the available training options, I choose to investigate rms-m update, based on its good performance in Chapter 4 and the saving of training time compared to variance update.

Figure 5.4 plots the RMS errors for different rms-m updated triphone models on training, validation and testing set respectively. The alignments used for calculating the RMS error figure were obtained by force-aligning the articulatory data with 8-mixture articulatory baseline triphone HMMs. As usual, the dashed lines give the performance of corresponding monophone rms-m updated models.

Looking on the top graph (training set), it is clear that more parameters result in significantly lower RMS errors on training set, compared to trajectory trained monophone model. The average RMS error drops to below 1.00mm when more than 1109 emitting states were employed. The graphs from validation and testing set, however, tell a different story. The trend is that more parameters actually increase the RMS errors, rather than reducing it, suggesting a sign of overfitting. Figure 5.5, where S-Align and S-Decode alignments were used, confirms the observation that trajectory updating triphone models tend to overfit the training data soon. This behaviour of trajectory

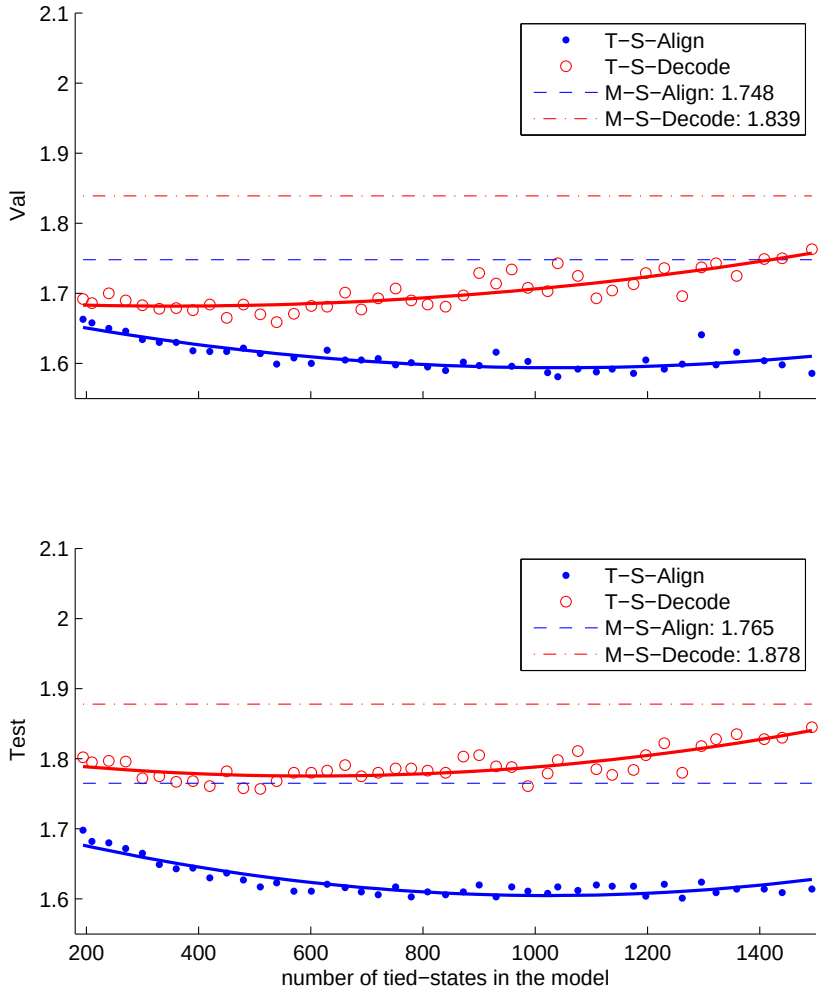

Figure 5.3: RMS performances from triphone baseline models (red lines with T prefix) with S-Align and S-Decode alignments. The dashed lines mark the result from corresponding monophone models (blue lines with M prefix).

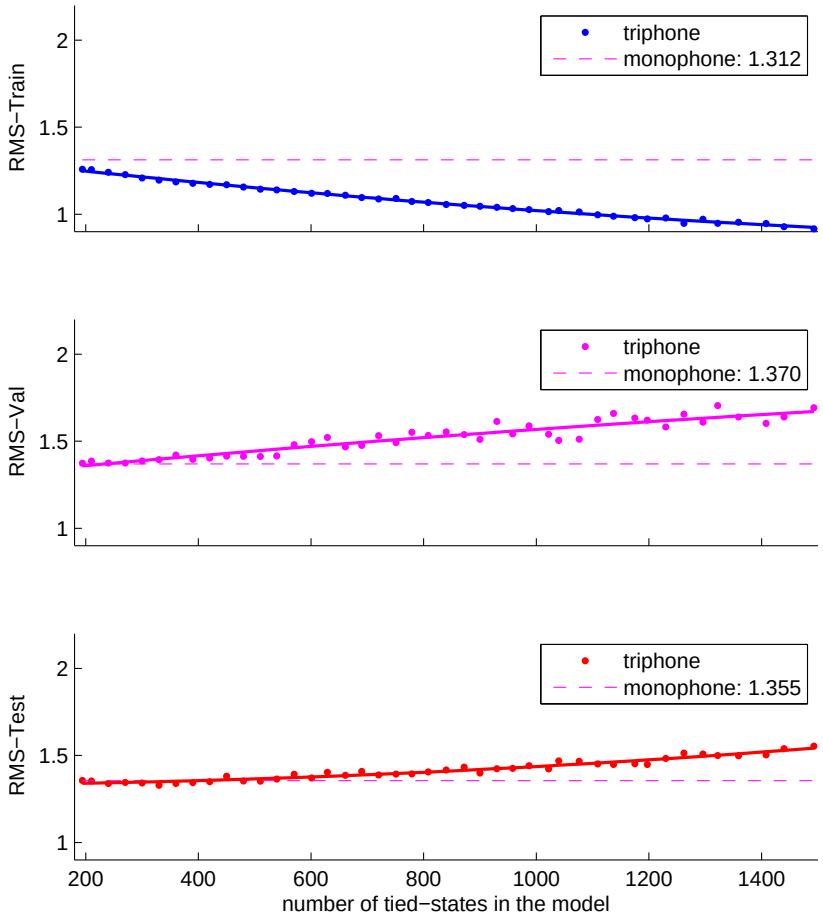

Figure 5.4: The averaged RMS error of rms-m updated triphone models with varied number of tied states compared to trajectory trained monophone model. Above: training data, middle: validation data, bottom: testing data. The alignment on val/test data were derived by force aligning the articulatory data using an 8-mixture articulatory HMMs (A-Align).

trained triphone models is rather unexpected, considering the observed improvement from Baum-Welch trained triphone models. Although, it is still significantly better than any Baum-Welch trained model, be it monophone or triphone, there is not much improvement over a trajectory trained monophone system.

5.5 Summary

Improving the performance of HMM system via detailed modelling units has a long-standing tradition in ASR community. In this chapter, I explored the possibility of enriching the modelling power of Trajectory-HMM through tree-clustered triphone unit with shared single Gaussian state output distribution. Using the HMM-based articulatory inversion method developed in Chapter 4, I was able to compare the triphone HMMs with their monophone counterparts under the same testing condition. The finding, however, is mixed.

Simply increasing the number of (shared) HMM states, even only being trained using Baum-Welch algorithm, will produce a much lower RMS inversion error compared to a conventional monophone model. This helps explain the practice of using contextual-dependent models in modern HMM-based speech synthesis system where trajectory training is not affordable: even without sophisticated trajectory training, the model will have overall better performance than monophone model.

Trajectory training of triphone models is possible with state tying technique. However, overfitting will be a major problem as demonstrated in the present experiment, especially when more than three times as many states as of monophone model are used. Compared to monophone system, modest improvement in RMS error was observed. Given the limited amount of data in the current MOCHA-TIMIT database, it may be the case that the data is too small for properly estimating all the triphone parameters. At the time of writing this, a new version of MOCHA-TIMIT is being constructed with many more recordings. It will be helpful to train the triphone inversion system on the new data when it is available to see if there is any overfitting there.

Another possible explanation for the close call between monophone and triphone result is that the monophone Trajectory-HMM already has enough modelling power to account for contextual variation so that going to more detailed triphone models does not help much. This is similar to neural network based speech recogniser where trained monophone network was found to give comparable performance to HMM triphone system, due to the increased modelling power for contextual variation.

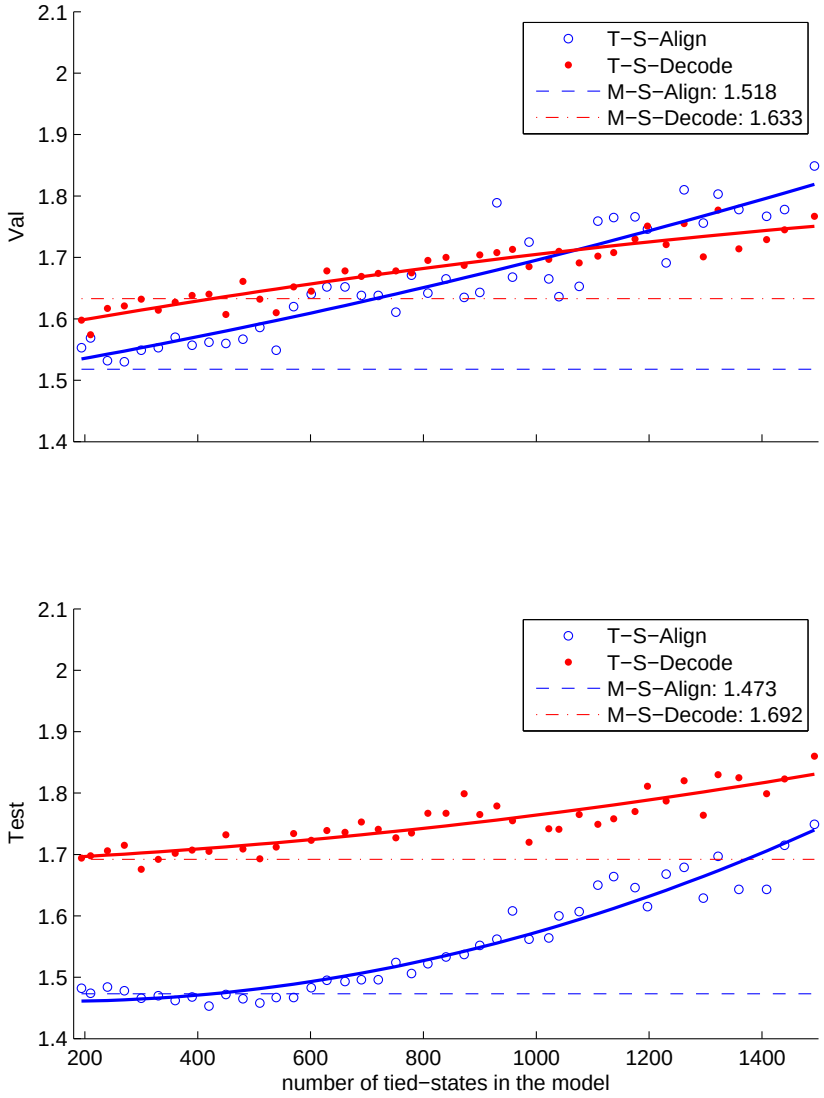

Figure 5.5: RMS performances from rms-m updated triphone models (red lines with T prefix) with S-Align and S-Decode alignments. The dashed lines mark the result from corresponding trajectory trained monophone models (blue lines with TM prefix).

Chapter 6

Phone Recognition Using Trajectory-HMMs

Chapter 4 and 5 demonstrate the usefulness of the Trajectory-HMM as a generative component in modelling continuous feature trajectories. In the present chapter we will look at the use of the Trajectory-HMM on a speech recognition task. The task is chosen for a few reasons. First is the evident importance of speech recognition, which has been the driving force for speech technology for decades since the early days of statistical speech modelling in 1970s (Jelinek, 1976). Second, the recognition task differs from articulatory inversion or speech synthesis in that recognition is essentially a discriminative problem: instead of seeking the similarity between the generated output and the training data, the goal of recognition is to maximise the discrimination between acoustically similar targets. While there is certainly a relation between the reduction of error rate and likelihood model fitness as demonstrated in conventional HMM modelling (Rabiner, 1989), model fitness in terms of likelihood is generally not required for a model to perform well on recognition. As a matter of fact, recent advances in large vocabulary speech recognition are mostly built using a discriminative framework, either by discriminatively estimating the acoustic model parameters (Povey and Woodland, 2001) or by introducing some discriminative feature transformation into the feature set (Hermansky et al., 2000).

Dynamic features in the form of first and second-order regression coefficients have been a standard part of the acoustic feature vector for nearly two decades (Furui, 1986). HMM ASR systems do not take full advantage of the added constraints from the dynamic feature the ASR community employed today, primarily because it is easy to implement and requires no changes to the HMM algorithms. Second, the delta and

delta-delta features do contain some added information, despite the fact that it can not be fully modelled by a conventionally trained HMM. Third, and most importantly, it works very well in practice.

Phone recognition experiments were carried out on the MOCHA-TIMIT data. I approached the recognition problem by directly decoding the acoustic signal using the trajectory decoding algorithm described in chapter 3. Section 6.1 covers the general difficulty of using Trajectory-HMM in recognition task. Section 6.2.2 discusses the proposed decoder in detail. The experimental set-up is given in section 6.2 and the results are presented and analysed in section 6.3. This chapter then finishes with a discussion in section 6.4.

6.1 Trajectory-HMM in Recognition

Applying the Trajectory-HMM to recognition involves two components. First, the model needs to be trained so that the Trajectory likelihood $p(\mathbf{c} \mid \mathbf{q})$ can reflect the dynamic constraints from the use of dynamic features. Second, a search algorithm needs to be developed to search for the most likely HMM state sequence corresponding to the acoustic representation in terms of trajectory likelihood. Training a Trajectory-HMM using the Viterbi approximation is straightforward, as shown in chapter 4. It is possible to improve the quality of training by iteratively deriving a better HMM training alignment utilising an aligning algorithm. Zen et al. (2004) shows such iterative training resulted in improved likelihood on the training data.

Although in previous chapters it has been shown that RMS training criterion has an edge over the MLE training objective, in the present work I opt to only use MLE training, as the decoder is designed to select the path with the highest log-likelihood. The main difficulty in decoding a Trajectory-HMM is that the correct use of the dynamic features will introduce inter-frame dependencies into the model. As a result, the number of hypotheses grows exponentially with the length of the utterance. Therefore, direct decoding is a formidable task without proper approximation.

Previously work has opted to rescore N-best lists produced by a conventional HMM (Tokuda et al., 2004)[1]. Using the proposed decoder described in section 3.4 it is possible to perform decoding on the phone recognition task.

[1] We were able to replicate the N-best list rescoring experiment of (Tokuda et al., 2004), and decided to focus on the use of a full scale decoder in the rest of this chapter.

6.2 Experiment Set-up

I choose to evaluate the model on a speaker-dependent phone recognition task for two reasons. First, it is a speaker-dependent task, so that we can isolate the potential effect of speaker variation. Second, the size of the task is small but still with reasonable complexity. The data contains around 450 utterances, and a single-Gaussian HMM system has a phone error rate (PER) around 50%, leaving plenty room for improvement.

To analyse the performance of the proposed decoding algorithm, I ran an English phone recognition task using the speech data from the MOCHA-TIMIT database, which contains one male speaker (msak0) and one female speaker (fsew0) with 460 TIMIT utterances for each. The first 400 sentences were used for training a monophone Trajectory-HMM, and the remaining 60 sentences were used for testing. The feature vector includes the standard 12 MFCC features plus the log energy, and their delta coefficients. The vocabulary consists of 45 phones, each of which was modelled by a 3-state left-to-right HMM with no skip. The output of HMM state was modelled by a single Gaussian with diagonal covariance. A phone loop grammar was used for decoding.

6.2.1 Trajectory-HMM for Phone Decoding

The speech data was first processed to extract acoustic features: the standard 12^{th} order MFCCs plus log-energy. Then the dynamic features were computed and appended to the static feature vectors. The monophone baseline HMMs were built using the Baum-Welch algorithm as implemented in HTK, from which Trajectory-HMMs can be built.

The trajectory parameter update algorithm uses a Viterbi style training procedure, and HMM state alignment is required. Similar to the articulatory inversion experiment in chapter 4, I derive the training alignment using the baseline HMM. When force aligning the training data, I experimented using different numbers of Gaussian mixture components for the baseline HMM, in the hope that a better alignment can be achieved with more mixture components.

6.2.2 The Trajectory Decoder

The decoder operates in a time-synchronous manner, using an extended token-passing algorithm. It works by first creating a searching lattice, either from a phone-loop grammar or from a Bigram language model, then propagating tokens carrying node infor-

mation around nodes in the lattice. There are a few factors that affect the speed of the decoder:

1. The type of dynamic coefficients.
   The delta and delta-delta features at each frame are calculated from the surrounding frames. At each frame during decoding, lattice nodes from the adjacent frames have to be inspected to collect statistics necessary for computing the partial trajectory likelihood. The more frames needed to calculate the dynamic features, the more computation is required, and the slower the decoding process will be. The two types of dynamic coefficients studied in chapter 4, namely dw3 and dw9, span a window of 3 and 9 frames respectively. This puts the dw9 type coefficient into considerable disadvantage in decoding. However, if the dynamic features include only delta features but not the delta-deltas, the number of frames for using the dw9 type coefficient reduces to 5 frames.

2. The number of delayed frames D.
   The decoding process is organised in a left-to-right time-recursive manner. The partial log-likelihood at frame t will need the intermediate statistics back to $t-D$ frames. As an approximation, the larger D is, the more accurate the resulting likelihood value is, and the slower the decoding process will be. The minimal value of D will be half the size of dynamic coefficient window, in order to correctly accumulate the path statistics.

3. The complexity of the Grammar.
   The decoder is capable of loading a bigram grammar expressed as a weighted network. In each frame of the decoding process, the tokens in all active network nodes need to be updated, which can be quite significant if a large network is used. More complex grammar (such as a trigram model) implies a bigger search lattice and slower decoding.

Apart from the listed factors, another practical constraint is the physical memory available. On a 32-bit machine the decoder has access to a maximum of 4GB memory. An initial experiment showed that the use of larger delay D on long utterance can easily exceed the 4GB memory limit.

6.3 Analysis of Decoding Result

I started from a conventional HMM trained using HTK. After that, the optimal state boundaries were obtained by running the decoder in alignment mode with a large delay ($D = 6$). Then the model's parameters were updated again based on the new state boundary information using trajectory mean update algorithm (3.43). This process was iterated a few times until the trajectory likelihood on the training data stabilised. For decoding, a merging window of 5 frames ($D = 2, L = 2$) was used with no pruning. The number of active nodes during decoding is of the order of 10^7 under this set-up. Note that it is possible to get a better result than what is presented here by using larger delays ($D \geq 3$), or employing delta-delta features (requires a merging window of 9 frames), although heavy pruning had to be employed so that the decoder can run on a machine with 2G RAM. Since the primary focus of this section is error analysis, we did not include results from larger D, which can be biased by search errors caused by pruning. Besides direct decoding, we also present the results from rescoring the N-best list ($N = 10, 20$ and 100) produced by the HTK-trained baseline HMMs. The rescoring was done by re-aligning and scoring the N-best hypotheses with trajectory trained HMMs (6-frame delay, $D = 6$). The phone error rates [2] from direct decoding are presented in table 6.1.

	HMM	Trajectory Decoding		Log-likelihood	
		$D = 1$	$D = 2$	$D = 1$	$D = 2$
fsew0	50.56%	55.48%	54.79%	-37.407	-37.210
msak0	52.68%	61.18%	58.96%	-35.786	-35.367

Table 6.1: Phone recognition error rates for two speakers on the MOCHA-TIMIT data. $D = 1, 2$ is the number look-back frames for decoding. Per-frame log-likelihood reported by the decoder is also presented.

Although using a larger delay ($D = 2$) helps achieve lower phone error rates compared to the results from setting D to 1, the phone error rates of the Trajectory-HMM

[2] The results for the HTK-trained baseline HMMs were obtained by tuning the `HVite` decoder's `-p` parameter (log inter model trans penalty) to balance the different error measures (insert/delete/substitution) and phone error rates, which was -1 for speaker fsew0, and -0.5 for msak0, respectively. The same settings were used for trajectory decoding experiments.

	HMM	N-Best			N-Best + reference		
		N=10	N=20	N=100	N=10	N=20	N=100
fsew0	50.56%	49.73%	49.35%	49.35%	49.73%	49.35%	49.35%
msak0	52.68%	52.63%	52.58%	52.00%	51.81%	51.81%	52.00%

Table 6.2: N-best list rescoring results for two speakers on the MOCHA-TIMIT data, with $N = 10, 20$ and 100.

(54.79% and 58.96%) are higher than those of a conventional HMM (50.56% and 52.68%). However, given the improvement obtained with $D = 2$, it is expected that larger delays ($D \geq 3$) would give more error reduction. The figures in the Log-likelihood columns confirm that hypotheses returned by decoding with a large delay ($D = 2$) give higher likelihood than those returned by running the decoder with $D = 1$.

While decoding directly does not give us a performance advantage over the HTK-trained HMMs, I was able to get modest improvement by rescoring the N-best list. Table 6.2 gives the results from N-best list rescoring. In all cases the error rates reported are lower than the baseline HMMs. And in general it can be observed that the larger N used, the lower phone error rates obtained. Including reference transcriptions in the rescoring is found to have no effect on the results for speaker `fsew0`, and gives modest improvement for speaker `msak0`. Manually inspecting the scores shows that the scores of most reference transcriptions are lower than those in the N-best list. This suggests that when direct decoding is performed, where the search space is much larger than rescoring an N-best list, the reference transcriptions will still not be selected. This partially explains why we could not reduce the phone error rate by direct decoding. A possible reason is that the training and decoding criterion, which is maximum trajectory likelihood $p(\mathbf{c} \mid \mathbf{q}, \lambda)$, is not optimal for recognition. In other words, a higher trajectory likelihood does not imply a lower recognition error. This is contrary to our hypothesis that by modelling the correlation between frame observations correctly, one is able to get better discrimination between recognition targets.

To help categorise the errors, I now introduce some notation to define the concept of phone level matching alignment. Let t_{w_i} denote the start frame of the i^{th} phone in a transcription, then the duration and the central position of the i^{th} phone can be written

as:

$$d(w_i) = t_{w_{i+1}} - t_{w_i} \tag{6.1}$$

$$c(w_i) = t_{w_i} + d(w_i)/2 \tag{6.2}$$

I define two phones w_i, w_j to have a matching alignment if:

$$\begin{cases} |d(w_i) - d(w_j)| & < \delta \\ |c(w_i) - c(w_j)| & < \varepsilon \end{cases} \tag{6.3}$$

where δ and ε are small integers, and are set to 2 and 3 in this experiment. Also I define $f(w_i)$ to be the per-frame likelihood of the i^{th} phone w_i averaged over its duration $d(w_i)$. Using the above definitions, I was able to classify phone recognition errors into three groups:

1. modelling error: A phone w_{rec} in the recognised transcription receives a much higher score than the correct phone w_{ref} with the matching alignment (as defined by (6.3)) in the reference transcription:

   $$f(w_{rec}) - f(w_{ref}) > \Delta \tag{6.4}$$

   where Δ is a positive number ($\Delta = 1.0$ in this experiment).

2. confusion error: A phone w_{rec} in the recognised transcription receives a score higher than the correct phone w_{ref} with the matching alignment in the reference transcription, but the difference in score is relatively small:

   $$f(w_{rec}) - f(w_{ref}) < \Delta \tag{6.5}$$

3. insert/delete error: Because the model is capable of generating a smoothed trajectory, sometimes a phone trajectory is smoothed excessively to give a good fit to the data, which is not the case in the reference transcription where the correct alignment corresponds to two phones w_{ref1}, w_{ref2}.

   $$f(w_{rec}) - f(w_{ref1}\ w_{ref2}) > 0 \tag{6.6}$$

   It is possible to find an over-smoothed phone spanning over more than two phones, although the inspection suggests that two phones are the most common case.

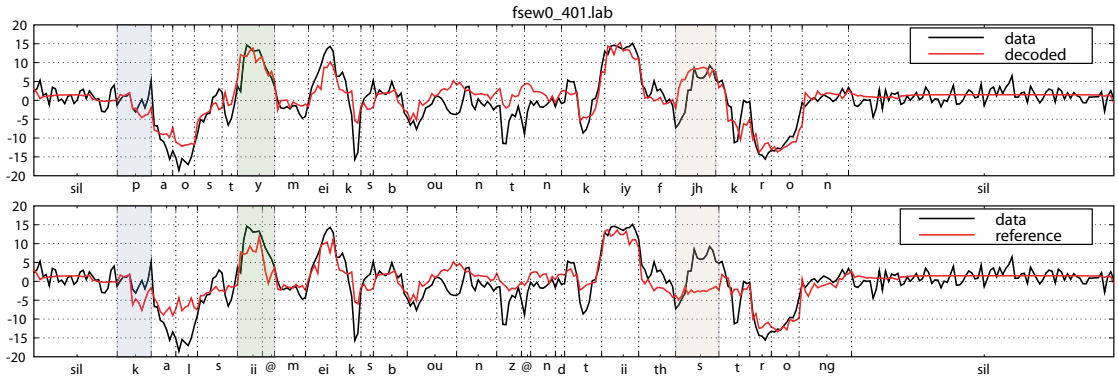

Figure 6.1: Smoothed mean trajectories for the first MFCC coefficient generated from transcription returned by the decoder (above) and the reference transcription (below) for one utterance in the test set. The shaded areas correspond to a confusion error, an insert/delete error and a modelling error, respectively.

Examples of all three kinds of errors are illustrated in figure 6.1, where smoothed trajectories are generated for both decoded transcription (above) and reference transcription (below) for one utterance in the testing set. The first shaded area highlights a confusion error where both phone /p/ and phone /y/ have a good fit to the data. The second shaded area shows an insert/delete error: phone /y/ is smoothed excessively to give a better fit to the data than the correct phone sequence /ii-@/, which explains why it is picked up by the decoder. A modelling error can be seen in the third shaded area where the wrong phone /jh/ has a much better fit to the data while the correct phone /s/ is not.

The distribution of the three error groups for the baseline HMM and the Trajectory-HMM is given in table 6.3. The majority of errors of both models comes from modelling error (53.72% and 62.42%), which is the mismatch between the learnt model and data. This type of error is difficult to get rid of unless a more complex output distribution, such as a Gaussian mixture model, is used to enhance the modelling power. The Trajectory-HMM, which in general has a better fit to the speech data, actually committed 8.7% more modelling errors compared to the baseline HMM. This suggests that the Trajectory-HMM may suffer a degradation of discrimination by favouring an over-smoothed phone trajectory that fits the data best among competing hypotheses. Both

models have a similar number of insert/delete errors (28.41% and 26.39%), which can be reduced by applying some sorts of penalty measure to control the length of phone alignment. For confusion errors, the Trajectory-HMM makes 6.7% fewer errors than the conventional HMM (11.19% to 17.88%), which is still a large portion. Training them discriminatively may provide better discrimination between acoustically similar phones in this case.

	modelling	confusion	insert/delete
HMM	53.72%	17.88%	28.41%
trajectory	62.42%	11.19%	26.39%

Table 6.3: Distribution of different error groups in the Trajectory-HMM decoding experiment.

6.4 Summary

The decoding result of the Trajectory-HMM brings out some interesting findings: First, the fact that the Trajectory-HMM can generate smoothed trajectory for speech data does not necessarily mean a lower recognition error rate. In contrast, we observed that under certain conditions it may be difficult for a model being too faithful to the data to have good discrimination between competing hypotheses. This partially explains why previously proposed alternative acoustic models that explicitly account for temporal correlation in speech, such as Segmental Models and Linear Dynamic Models, are not vastly superior to HMMs in practice. Second, like MLE trained HMMs, the MLE trained Trajectory-HMM also suffers from weak disambiguation between acoustically similar phones. Training the model discriminatively may help improve the ASR performance, albeit with more computational expense.

It is worth mentioning that the observation made here differs from that of an N-best list rescoring approach by (Tokuda et al., 2004). The following factors may contribute to the diverse results: first, our baseline system has a higher error rate than (Tokuda et al., 2004)'s 19.7%, which suggests the MOCHA data we used is more noisy; second, the search space of a full decoder is much larger than that of N-best re-scoring, and a short delay ($D = 2$) has to be used to make decoding tractable, which compromises the performance.

In closing this chapter, the author emphasises that when seeking alternative acoustic models for speech recognition, one should pay more attention on the discriminative power rather than the fitness between the learnt model and training data. The increase of likelihood on training data does not always lead to improved ASR performance, as is the case in this experiment.

Chapter 7

Conclusion

7.1 A Brief Review

The focus of this thesis has been the exploration of the use of Trajectory-HMMs in speech modelling, an area that has been dominated by conventional hidden Markov models for decades. The close relationship between a Trajectory-HMM and a normal HMM makes the trajectory approach stand out from other attempts in pursuing a different speech modelling method. While previous proposed "alternative" models are mostly based on an entirely different framework from HMM as reviewed in chapter 2, the Trajectory-HMM tries to "fix" the incorrect handling of dynamic features in HMM while retaining the fundamental HMM infrastructure. The obvious benefit is that the large repository of existing HMM algorithms can be re-used, and the time-honoured HMM architecture (phoneme-based HMM models with Gaussian outputs) is retained. However, being a model with a radically different output behaviour from HMMs, the Trajectory-HMM can be regarded as a new model in many ways: the stepwise mean output of HMMs for maximum likelihood generalisation is replaced by a smoothed continuous trajectory, the decoding is coupled with intra-frame dependencies, and the delta parameters are interpreted differently despite the fact that a Trajectory-HMM has exactly the same parametric form as a conventional HMM with the same network topology.

As the framework of Trajectory-HMM is still in a early stage of development, this thesis is devoted to obtaining a deeper understanding of the model in terms of: 1) the trainability of the model parameters, 2) the increase of modelling power by using detailed modelling unit and, 3) the use of Trajectory-HMM in decoding task, with a properly implemented decoding algorithm for Trajectory-HMM. The availability of the

parallel articulatory-acoustic corpus MOCHA-TIMIT provided us with the opportunity to test all these aspects on a single data set.

In chapter 4, the Trajectory-HMM was used in an articulatory inversion task as the generative component to produce recovered articulatory movement from the acoustic representation of speech. The two components (articulatory and acoustic HMMs), though seemingly serving different purposes, are connected by a common set of HMM states acting as the intermediate stage of the inversion process. A systematic study of the training behaviour was conducted using the articulatory inversion task. The commonly used MLE training criterion, while effective in reducing the mean RMS generation error on the task, was found to be less effective compared with an RMS training objective that seeks to directly minimise the RMS error on the training data. For the three kinds of individual parameter update methods, updating both mean and variance simultaneously was found to be the most effective, albeit also the most computationally expensive. The variance update, on the other hand, is the least effective parameter update method investigated. And there is evidence (decreased log-likelihood) that variance update moves the parameters towards a substantially different setting to other parameter update methods, and is therefore less stable. This finding suggests that, when limited by computational power, a recommended training strategy is to only update the mean parameter analytically, which is considerably faster than gradient-based approaches, with comparable performance. Furthermore, it has been found that optimal RMS reduction can be achieved by estimating the parameters of the two sets of HMMs in a joint manner. One weakness of the present framework is that the static output PDFs are limited to single Gaussians. While extending to GMMs is beyond the scope of this thesis, within the particular articulatory inversion task it has been shown that the system performance can be increased by obtaining more accurate HMM state alignment using mixture-based acoustic HMMs.

Two types of commonly used dynamic coefficients were compared: the three-frame dw3 type window as used in the HTS speech synthesis system, and the nine-frame dw9 type window as used in the HTK speech recognition system. The experiment reveals that dynamic coefficients spanning a longer window are not necessarily better than dynamic coefficients calculated from shorter window. In fact, the experimental result shows that the two types of window each have their own strengths and weaknesses. The dw9 type window is found to be superior for recognition, and also derives a more accurate HMM state alignment from acoustic data compared with the dw3 type window. However, the mean output trajectory generated from dw9 type coefficients is less

smooth and is therefore completely unusable for the intended purpose of articulatory inversion. The dw3 type window, on the other hand, yields a less accurate HMM state alignment but provides a smoothed mean output trajectory. The finding, though contrary to the common assumption that longer dynamic window should work better in all cases, gives us a hint of how to choose the dynamic coefficients. The articulatory inversion accuracy can be further improved by using both type of dynamic coefficients in a complementary way: the dw9 type coefficients were used for acoustic HMMs and for articulatory modelling, the dw3 type dynamic features were used.

A natural way of increasing the modelling power of a statistical model is to use more free parameters. However, more parameters do not always result in better performance, especially when overfitting occurs. Chapter 5 investigates the Trajectory-HMM using triphone modelling units as an alternative way to employ more parameters via Gaussian mixture models. The findings, however, were mixed. The Baum-Welch trained models were found to benefit greatly by moving the modelling unit from monophone to triphone, with a steady reduction in RMS mean generation errors on the testing set, as the number of triphone emitting states increases. This suggests that triphone modelling can be effective when trajectory training is not affordable. Such an observation helps explain the success of Baum-Welch trained HMMs in model-based speech synthesis where the modelling units are linguistically enhanced contextual features (Zen and Toda, 2005). On the other hand, the trajectory trained triphone models, while having a reduced RMS error on the articulatory task compared to the conventional monophone HMMs, are no better than monophone Trajectory-HMMs. A possible explanation could be that a properly trained monophone Trajectory-HMM already accounts well for the contextual variation in the speech. Moving from monophone to triphone thus has little gain, subject to the limited amount of data we have. Of course, such claim needs to be examined on larger data sets.

A practical difficulty of using Trajectory-HMM for speech recognition is the lack of a proper decoding algorithm: decoding Trajectory-HMMs always results in some trade-offs, due to the exponential growth of the search space. in Chapter 6 the proposed trajectory decoding algorithm was exercised on a monophone recognition task. The results on the MOCHA-TIMIT data suggest that the MLE trained Trajectory-HMM loses some discriminative power compared to Baum-Welch trained HMMs. Under certain conditions, it was observed that the Trajectory-HMM can be “too faithful” to the data to have good discrimination between competing hypothesises.

7.2 Future Research

The use of Trajectory-HMM in speech processing is still in its early stages. Based on the findings in this work, the author would like to highlight some possible directions for future research:

1. Multi-rate Modelling

   Most current ASR systems work on a frame-based representation of acoustic signal, usually sampled at a 10ms interval. With the help of dynamic features, short-term temporal information up to 100ms can be incorporated into an HMM system. However, acoustic variability that evolves slowly over longer time scales such as the coarticulation effect at the syllable level, is better modelled as a different scale. A multi-rate model fits nicely in scenario where speech patterns from different time scales, such as phones and syllables, are modelled at different levels. In a typical two-rate acoustic model (Cetin, 2005), the fine scale corresponds to the traditional phone HMMs with cepstral features, while the coarse scale can be used to model features extracted from a long-term window characterising either phones broadly, or syllable structure and stress (Chen et al., 2004).

   A multi-rate modelling paradigm is attractive for use with the trajectory framework because long-term dynamic patterns in speech (such as the articulatory features (Frankel et al., 2000, 2007)) can be captured and modelled in the same way as a phone-level Trajectory-HMM models short-term dynamics through the use of feature derivatives.

2. Alternative Decoding Objective Function

   The decoding algorithm proposed in this thesis operates on the maximum likelihood principle: the state sequence with the highest trajectory log-likelihood $\log p(\mathbf{c} \mid \mathbf{q}, \lambda)$, subject to decoding constraints, will be returned. This may not be the preferred criterion for recognition, where a model with good disambiguation between recognition targets is preferred. This concern can be seen by analysing the recognition result which suggests maximum likelihood training tend to reduce the discriminative power of the trained Trajectory-HMM model. Nevertheless, as demonstrated in chapter 4, it is possible to optimise the model parameter to directly minimise the output mean generation error on the data. If we change the decoding criterion to selecting the state sequence with the lowest RMS gen-

eration error on the observation, the objective function becomes:

$$E = \frac{1}{2}\frac{1}{T}(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})^T(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}}) \tag{7.1}$$

$\overline{\mathbf{c}}_{\mathbf{q}}$ in (7.1) is the trajectory mean vector given the state sequence $\mathbf{q}$, which is traditionally calculated by backward variable substitution in linear equations (see Appendix A.5). If $\overline{\mathbf{c}}_{\mathbf{q}}$ could be computed in a left-to-right, time recursive manner, maybe with some approximation, little modification is required to implement the RMS decoding objective within the token-passing trajectory decoding framework.

3. Development of More Efficient Algorithms
   The trajectory training algorithm is much slower than HMM's. The bottleneck is the calculation of covariance matrix $\mathbf{P}_{\mathbf{q}}$, which is the inverse of the precision matrix $\mathbf{R}_{\mathbf{q}}$. As high-dimensional matrix inversion method is very computationally demanding, it will be worth to investigate method to speed up the process. Because $\mathbf{P}_{\mathbf{q}}$ itself is a square matrix with the off diagonal elements being close to zero, it makes sense to develop special matrix inversion method that only calculate the central diagonal elements. The resulting matrix, though is an approximate, is likely to greatly speed up the training process with little impact on the training accuracy. Other areas of improvement include efficient approximation of probability of Gaussian mixture components in Trajectory-HMM, which would be a major breakthrough that opens the door for wider application of the model.

Appendix A

Appendix

A.1 Notational Conventions

Notation	Description	Dimensionality	Comments
T	length of a state sequence (frames)	scalar	
M	dimensionality of the static feature vector	scalar	usually 13 for MFCC frontend
N	number of Gaussians in a Trajectory-HMM	scalar	
L	half width of the dynamic window	scalar	
$\mathbf{q}$	HMM state sequence over T frames	$T \times 1$	
$\mathbf{c}$	static feature vector sequence over T frames	$TM \times 1$	
$\mathbf{o}$	augmented feature vector sequence over T frames	$3TM \times 1$	
$\boldsymbol{\mu}_{\mathbf{q}}$	mean vector for the augmented observation $\mathbf{o}$	$3TM \times 1$	
$\boldsymbol{\Sigma}_{\mathbf{q}}$	covariance matrix for the augmented observation $\mathbf{o}$	$3TM \times 3TM$	
$\mathbf{w}_t$	transforming matrix such that $\mathbf{o}_t = \mathbf{w}_t\mathbf{c}$	$3M \times TM$	
$\mathbf{W}$	transforming matrix such that $\mathbf{o} = \mathbf{W}\mathbf{c}$	$3TM \times TM$	not invertable
$\mathbf{m}$	mean vectors of the trajectory model	$3MN \times 1$	
$\boldsymbol{\phi}$	vector of diagonal covariances of the model	$3MN \times 1$	$\boldsymbol{\Sigma}_{\mathbf{q}}^{-1} = \mathrm{diag}(\mathbf{S}_{\mathbf{q}}\boldsymbol{\phi})$
$\boldsymbol{\Phi}$	global variance diagonal matrix of the model	$3MN \times 3MN$	$\Phi = \mathrm{diag}(\phi)$
$\mathbf{S}_{\mathbf{q}}$	transforming matrix so that $\boldsymbol{\mu}_{\mathbf{q}} = \mathbf{S}_{\mathbf{q}}\mathbf{m}$	$3TM \times 3MN$	not invertable but $\boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\mathbf{S}_{\mathbf{q}} = \mathbf{S}_{\mathbf{q}}\boldsymbol{\phi}$
$\overline{\mathbf{c}}_{\mathbf{q}}$	mean vector of $p(\mathbf{c} \mid \mathbf{q}, \lambda)$	$TM \times 1$	
$\mathbf{P}_{\mathbf{q}}$	covariance matrix of $p(\mathbf{c} \mid \mathbf{q}, \lambda)$	$TM \times TM$	full matrix
$\mathbf{R}_{\mathbf{q}}$	inverse covariance (precision) matrix of $p(\mathbf{c} \mid \mathbf{q}, \lambda)$	$TM \times TM$	$M(4L+1)$-band diagonal
$Z_{\mathbf{q}}$	normalisation term for state sequence $\mathbf{q}$	scalar	

A.2 Correctness of the Normalisation Term $Z_{\mathbf{q}}$

Since the probability density function of the Trajectory-HMM $p(\mathbf{c} \mid \mathbf{q}, \lambda)$ is obtained by multiplying a normalisation term $Z_{\mathbf{q}}^{-1}$ to $p(\mathbf{o} \mid \mathbf{q}, \lambda)$, the probability density function

of the conventional HMM, we can see $Z_{\mathbf{q}}$ is the correct term if it satisfies:

$$p(\mathbf{c} \mid \mathbf{q},\lambda) = \frac{1}{Z_{\mathbf{q}}} p(\mathbf{o} \mid \mathbf{q},\lambda) \tag{A.1}$$

Substituting $Z_{\mathbf{q}}$ as given in 3.25 into the right hand side of (A.1) we have:

$$\begin{aligned}\frac{1}{Z_{\mathbf{q}}} p(\mathbf{o} \mid \mathbf{q},\lambda) &= \frac{\frac{1}{\sqrt{(2\pi)^{3TM}|\boldsymbol{\Sigma}_{\mathbf{q}}|}} \exp\left\{-\frac{1}{2}(\mathbf{o}-\boldsymbol{\mu}_{\mathbf{q}})^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}(\mathbf{o}-\boldsymbol{\mu}_{\mathbf{q}})\right\}}{\frac{\sqrt{(2\pi)^{TM}|\mathbf{P}_{\mathbf{q}}|}}{\sqrt{(2\pi)^{3TM}|\boldsymbol{\Sigma}_{\mathbf{q}}|}} \exp\left\{-\frac{1}{2}(\boldsymbol{\mu}_{\mathbf{q}}^T \boldsymbol{\Sigma}_{\mathbf{q}} \boldsymbol{\mu}_{\mathbf{q}} - \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}})\right\}} \\ &= \frac{1}{\sqrt{(2\pi)^{TM}|\mathbf{P}_{\mathbf{q}}|}} \exp\Bigg\{-\frac{1}{2}\Big[(\mathbf{o}-\boldsymbol{\mu}_{\mathbf{q}})^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}(\mathbf{o}-\boldsymbol{\mu}_{\mathbf{q}}) \\ &\qquad -\boldsymbol{\mu}_{\mathbf{q}}^T \boldsymbol{\Sigma}_{\mathbf{q}} \boldsymbol{\mu}_{\mathbf{q}} + \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}}\Big]\Bigg\}\end{aligned} \tag{A.2}$$

The exponent of (A.2) is:

$$= \exp\left\{-\frac{1}{2}\left(\mathbf{o}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\mathbf{o} - \mathbf{o}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}} - \boldsymbol{\mu}_{\mathbf{q}}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\mathbf{o} + \boldsymbol{\mu}_{\mathbf{q}}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}} - \boldsymbol{\mu}_{\mathbf{q}}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}} + \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}}\right)\right\} \tag{A.3}$$

$$= \exp\left\{-\frac{1}{2}\left(\mathbf{c}^T \mathbf{W}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\mathbf{W}\mathbf{c} - \mathbf{c}^T \mathbf{W}^T \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\boldsymbol{\mu}_{\mathbf{q}} - \boldsymbol{\mu}_{\mathbf{q}} \boldsymbol{\Sigma}_{\mathbf{q}}^{-1}\mathbf{W}\mathbf{c} + \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}}\right)\right\} \tag{A.4}$$

$$= \exp\left\{-\frac{1}{2}\left(\mathbf{c}^T \mathbf{R}_{\mathbf{q}}\mathbf{c} - \mathbf{c}^T \mathbf{r}_{\mathbf{q}} - \mathbf{r}_{\mathbf{q}}^{\mathbf{T}}\mathbf{c} + \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}} \mathbf{r}_{\mathbf{q}}\right)\right\} \tag{A.5}$$

$$= \exp\left\{-\frac{1}{2}\left(\mathbf{c}^T \mathbf{R}_{\mathbf{q}}\mathbf{c} - \mathbf{c}^T \mathbf{R}_{\mathbf{q}}\mathbf{P}_{\mathbf{q}}\mathbf{r}_{\mathbf{q}} - \mathbf{r}_{\mathbf{q}}^{\mathbf{T}}\mathbf{P}_{\mathbf{q}}\mathbf{R}_{\mathbf{q}}\mathbf{c} + \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}}\mathbf{R}_{\mathbf{q}}\mathbf{P}_{\mathbf{q}}\mathbf{r}_{\mathbf{q}}\right)\right\} \tag{A.6}$$

$$= \exp\left\{-\frac{1}{2}(\mathbf{c}^T - \mathbf{r}_{\mathbf{q}}{}^T \mathbf{P}_{\mathbf{q}})\mathbf{R}_{\mathbf{q}}(\mathbf{c} - \mathbf{P}_{\mathbf{q}}\mathbf{r}_{\mathbf{q}})\right\} \tag{A.7}$$

$$= \exp\left\{-\frac{1}{2}(\mathbf{c}^T - \overline{\mathbf{c}}_{\mathbf{q}}^T)\mathbf{R}_{\mathbf{q}}(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})\right\} \tag{A.8}$$

$$= \exp\left\{-\frac{1}{2}(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})^T \mathbf{P}_{\mathbf{q}}{}^{-1}(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})\right\} \tag{A.9}$$

Therefore:

$$\frac{1}{Z_{\mathbf{q}}} p(\mathbf{o} \mid \mathbf{q},\lambda) = \frac{1}{\sqrt{(2\pi)^{TM}|\mathbf{P}_{\mathbf{q}}|}} \exp\left\{-\frac{1}{2}(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})^T \mathbf{P}_{\mathbf{q}}{}^{-1}(\mathbf{c} - \overline{\mathbf{c}}_{\mathbf{q}})\right\} \tag{A.10}$$

$$= \mathcal{N}(\mathbf{c} \mid \overline{\mathbf{c}}_{\mathbf{q}}, \mathbf{P}_{\mathbf{q}}) \tag{A.11}$$

$$= p(\mathbf{c} \mid \mathbf{q},\lambda) \tag{A.12}$$

A.3 Derivative of the Log-likelihood Objective Function

The derivative and gradient vectors of objective function play an important role in parameter optimisation. The log-likelihood objective function is:

$$\log p(\mathbf{c} \mid \mathbf{q}) = -\frac{1}{2}\left\{TM\log(2\pi) - \log|\mathbf{R_q}| + \mathbf{c}^T\mathbf{R_q}\mathbf{c} + \mathbf{r_q}^T\mathbf{P_q}\mathbf{r_q} - 2\mathbf{r_q}^T\mathbf{c}\right\} \quad \text{(A.13)}$$

The derivative of $\log(\mathbf{c} \mid \mathbf{q})$ w.r.t mean parameter vector $\mathbf{m}$ is:

$$\begin{aligned} d\log(\mathbf{c} \mid \mathbf{q}) =& -\frac{1}{2}(d(\mathbf{r_q}^T\mathbf{P_q}\mathbf{r_q}) - 2d(\mathbf{r_q}^T\mathbf{c})) \\ =& -\frac{1}{2}(\mathbf{r_q}^T\mathbf{P_q}d\mathbf{r_q} + \mathbf{r_q}^T\mathbf{P_q}d\mathbf{r_q} - 2\mathbf{c}^T d\mathbf{r_q}) \\ =& (\mathbf{c}^T - \mathbf{r_q}^T\mathbf{P_q})d\mathbf{r_q} \\ =& (\mathbf{c}^T - \mu_\mathbf{q}^T\Sigma_\mathbf{q}^{-1}\mathbf{W}\mathbf{P_q})\mathbf{W}^T\Sigma_\mathbf{q}^{-1}d\mu_\mathbf{q} \\ =& (\mathbf{c}^T - \mathbf{m}^T\mathbf{S_q}^T\Sigma_\mathbf{q}^{-1}\mathbf{W}\mathbf{P_q})\mathbf{W}^T\Sigma_\mathbf{q}^{-1}\mathbf{S_q}d\mathbf{m} \\ =& (\mathbf{c}^T - \mathbf{m}^T\mathbf{\Phi}\mathbf{S_q}^T\mathbf{W}\mathbf{P_q})\mathbf{W}^T\mathbf{S_q}\mathbf{\Phi}d\mathbf{m} \end{aligned} \quad \text{(A.14)}$$

Therefore the derivative w.r.t $\mathbf{m}$:

$$\boxed{\frac{d\log(\mathbf{c} \mid \mathbf{q})}{d\mathbf{m}} = (\mathbf{c}^T - \mathbf{m}^T\mathbf{\Phi}\mathbf{S_q}^T\mathbf{W}\mathbf{P_q})\mathbf{W}^T\mathbf{S_q}\mathbf{\Phi}} \quad \text{(A.15)}$$

Transpose to get the gradient vector w.r.t. $\mathbf{m}$:

$$\boxed{\frac{\partial\log(\mathbf{c} \mid \mathbf{q})}{\partial\mathbf{m}} = \mathbf{\Phi}\mathbf{S_q}^T\mathbf{W}(\mathbf{c} - \mathbf{P_q}\mathbf{W}^T\mathbf{S_q}\mathbf{\Phi}\mathbf{m})} \quad \text{(A.16)}$$

For the derivative w.r.t variance parameter vector:

$$\begin{aligned} d\log(\mathbf{c} \mid \mathbf{q}) &= \frac{1}{2}(tr(\mathbf{P_q}\mathbf{dR_q}) - \mathbf{c}^T(d\mathbf{R_q})\mathbf{c} - (d\mathbf{r_q}^T)\mathbf{P_q}\mathbf{r_q} - \mathbf{r_q}^T d(\mathbf{P_q})\mathbf{r_q} - \mathbf{r_q}^T\mathbf{P_q}d\mathbf{r_q} + 2(d\mathbf{r_q}^T)\mathbf{c}) \\ &= \frac{1}{2}(tr(\mathbf{P_q}d\mathbf{R_q}) - \mathbf{c}\mathbf{c}^T d\mathbf{R_q} - \mathbf{r_q}\mathbf{r_q}^T(-\mathbf{P_q}(d\mathbf{R_q})\mathbf{P_q}) - 2\mathbf{r_q}^T\mathbf{P_q}d\mathbf{r_q} + 2\mathbf{c}^T d\mathbf{r_q}) \\ &= \frac{1}{2}(tr(\mathbf{P_q}d\mathbf{R_q}) - \mathbf{c}\mathbf{c}^T d\mathbf{R_q} + \mathbf{P_q}\mathbf{r_q}\mathbf{r_q}^T\mathbf{P_q}d\mathbf{R_q} - 2\mathbf{r_q}^T\mathbf{P_q}d\mathbf{r_q} + 2\mathbf{c}^T d\mathbf{r_q}) \\ &= \frac{1}{2}(tr(\mathbf{P_q}(d\mathbf{W}^T\mathbf{\Sigma_q}^{-1}\mathbf{W})) - \mathbf{c}\mathbf{c}^T(d\mathbf{W}^T\mathbf{\Sigma_q}^{-1}\mathbf{W}) + \mathbf{P_q}\mathbf{r_q}\mathbf{r_q}^T\mathbf{P_q}(d\mathbf{W}^T\mathbf{\Sigma_q}^{-1}\mathbf{W}) \\ &\qquad - 2\mathbf{r_q}^T\mathbf{P_q}(d\mathbf{W}^T\mathbf{\Sigma_q}^{-1}\boldsymbol{\mu_q}) + 2\mathbf{c}^T(d\mathbf{W}^T\mathbf{\Sigma_q}^{-1}\boldsymbol{\mu_q})) \\ &= \frac{1}{2}(tr(\mathbf{W}\mathbf{P_q}\mathbf{W}^T d\mathbf{\Sigma_q}^{-1}) - \mathbf{W}\mathbf{c}\mathbf{c}^T\mathbf{W}^T d\mathbf{\Sigma_q}^{-1} + \mathbf{W}\mathbf{P_q}\mathbf{r_q}\mathbf{r_q}^T\mathbf{P_q}\mathbf{W}^T d\mathbf{\Sigma_q}^{-1} \\ &\qquad - 2\boldsymbol{\mu_q}\mathbf{r_q}^T\mathbf{P_q}\mathbf{W}^T d\mathbf{\Sigma_q}^{-1} + 2\boldsymbol{\mu_q}\mathbf{c}^T\mathbf{W}^T d\mathbf{\Sigma_q}^{-1}) \\ &= \frac{1}{2}(tr(\mathbf{W}\mathbf{P_q}\mathbf{W}^T d\mathbf{\Sigma_q}^{-1}) - \mathbf{W}\mathbf{c}\mathbf{c}^T\mathbf{W}^T d\mathbf{\Sigma_q}^{-1} + \mathbf{W}\bar{\mathbf{c}}_\mathbf{q}\bar{\mathbf{c}}_\mathbf{q}^T\mathbf{W}^T d\mathbf{\Sigma_q}^{-1} \\ &\qquad - 2\boldsymbol{\mu_q}\bar{\mathbf{c}}_\mathbf{q}^T\mathbf{W}^T d\mathbf{\Sigma_q}^{-1} + 2\boldsymbol{\mu_q}\mathbf{c}^T\mathbf{W}^T d\mathbf{\Sigma_q}^{-1}) \end{aligned} \quad \text{(A.17)}$$

Therefore the derivative w.r.t. Σ_q^{-1}:

$$\boxed{\frac{d\log(\mathbf{c}\mid\mathbf{q})}{d\boldsymbol{\Sigma}_{\mathbf{q}}^{-1}} = \frac{1}{2}\left(\mathbf{W}\mathbf{P}_{\mathbf{q}}\mathbf{W}^T - \mathbf{W}\mathbf{c}\mathbf{c}^T\mathbf{W}^T + \mathbf{W}\overline{\mathbf{c}}_{\mathbf{q}}\overline{\mathbf{c}}_{\mathbf{q}}^T\mathbf{W}^T - 2\boldsymbol{\mu}_{\mathbf{q}}\overline{\mathbf{c}}_{\mathbf{q}}^T\mathbf{W}^T + 2\boldsymbol{\mu}_{\mathbf{q}}\mathbf{c}^T\mathbf{W}^T\right)} \tag{A.18}$$

Because $\boldsymbol{\Sigma}_{\mathbf{q}}^{-1} = \mathrm{diag}(\mathbf{S}_{\mathbf{q}}\boldsymbol{\phi})$, Derivative w.r.t $\boldsymbol{\phi}$:

$$\begin{aligned}\frac{\partial\log p(\mathbf{c}\mid\mathbf{q})}{\partial\boldsymbol{\phi}} = \frac{1}{2}\mathbf{S}_{\mathbf{q}}{}^T\mathrm{diag}^{-1}\Big(&\mathbf{W}\mathbf{P}_{\mathbf{q}}\mathbf{W}^T - \mathbf{W}\mathbf{c}\mathbf{c}^T\mathbf{W}^T + \mathbf{W}\overline{\mathbf{c}}_{\mathbf{q}}\overline{\mathbf{c}}_{\mathbf{q}}^T\mathbf{W}^T \\ &-2\boldsymbol{\mu}_{\mathbf{q}}\overline{\mathbf{c}}_{\mathbf{q}}^T\mathbf{W}^T + 2\boldsymbol{\mu}_{\mathbf{q}}\boldsymbol{c}^T\mathbf{W}^T\Big)\end{aligned} \tag{A.19}$$

Here the diag^{-1} operator retrieves the diagonal elements of a matrix into a vector. Note it is common practice to optimise variance parameter in log domain ($\log\boldsymbol{\phi}$) to avoid numerical problem so and extra $\boldsymbol{\phi}$ term need to be multiplied to (A.19).

A.4 Derivative of the Sum Mean Square Error Objective Function

It is possible to directly minimise the RMS generation error on the training data. The equivalence measure of Sum Mean Square Error is used for mathematical convenience:

$$E_{sms} = \frac{1}{2}\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}})^T(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}}) \tag{A.20}$$

The derivative of E_{sms}:

$$\begin{aligned} d(E_{sms}) =& \frac{1}{2}\frac{1}{T}d\left[(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}})^T(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}})\right] && \text{(A.21)}\\ =& \frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}})^T d(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}}) && \text{(A.22)}\\ =& -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_{\mathbf{q}})^T d\overline{\mathbf{c}}_{\mathbf{q}} && \text{(A.23)}\end{aligned}$$

Therefore the derivative w.r.t. mean parameter vector $\mathbf{m}$ is:

$$\frac{d(E_{sms})}{d\mathbf{m}} = -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T d(\mathbf{P_q r_q}) \tag{A.24}$$

$$= -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T \mathbf{P q} d(\mathbf{W}^T\boldsymbol{\Sigma}_\mathbf{q}^{-1}\boldsymbol{\mu}_\mathbf{q}) \tag{A.25}$$

$$= -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T \mathbf{P_q W}^T\boldsymbol{\Sigma}_\mathbf{q}^{-1} d\boldsymbol{\mu}_\mathbf{q} \tag{A.26}$$

$$= -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T \mathbf{P_q W}^T\boldsymbol{\Sigma}_\mathbf{q}^{-1}\mathbf{S_q} d\mathbf{m} \tag{A.27}$$

$$= -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T \mathbf{P_q W}^T\mathbf{S_q}\boldsymbol{\Phi} d\mathbf{m} \tag{A.28}$$

$$= -\frac{1}{T}(\mathbf{c}-\mathbf{P_q W}^T\mathbf{S_q}\boldsymbol{\Phi}\mathbf{m})^T \mathbf{P_q W}^T\mathbf{S_q}\boldsymbol{\Phi} d\mathbf{m} \tag{A.29}$$

Transpose to get the gradient vector:

$$\boxed{\frac{\partial E_{sms}}{\partial \mathbf{m}} = -\frac{1}{T}\boldsymbol{\Phi}\mathbf{S}_\mathbf{q}^T\mathbf{W}\mathbf{P_q}(\mathbf{c}-\mathbf{P_q W}^T\mathbf{S_q}\boldsymbol{\Phi}\mathbf{m})} \tag{A.30}$$

The optimised m can be obtained by setting the gradient vector to zero and solve the set of linear equations:

$$\boxed{\boldsymbol{\Phi}\mathbf{S}_\mathbf{q}^T\mathbf{W}\mathbf{P_q}\mathbf{P_q W}^T\mathbf{S_q}\boldsymbol{\Phi}\mathbf{m} = \boldsymbol{\Phi}\mathbf{S}_\mathbf{q}^T\mathbf{W}\mathbf{P_q}\mathbf{c}} \tag{A.31}$$

For variance parameter, the derivative w.r.t. $\boldsymbol{\Sigma}_\mathbf{q}^{-1}$ is:

$$\frac{d(E_{sms})}{d\boldsymbol{\Sigma}_\mathbf{q}^{-1}} = -\frac{1}{T}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T d(\mathbf{P_q r_q}) \tag{A.32}$$

$$= -\frac{1}{T}\left(\mathbf{r_q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T d\mathbf{P_q} + (\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q} d\mathbf{r_q}\right) \tag{A.33}$$

$$= -\frac{1}{T}\left(\mathbf{r_q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T(-\mathbf{P_q}(d\mathbf{R_q})\mathbf{P_q}) + (\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q} d\mathbf{r_q}\right) \tag{A.34}$$

$$= -\frac{1}{T}\left(-\mathbf{P_q r_q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q} d\mathbf{R_q} + (\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q} d\mathbf{r_q}\right) \tag{A.35}$$

$$= -\frac{1}{T}\left(-\mathbf{P_q r_q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q} d(\mathbf{W}^T\boldsymbol{\Sigma}_\mathbf{q}^{-1}\mathbf{W}) + (\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q} d(\mathbf{W}^T\boldsymbol{\Sigma}_\mathbf{q}^{-1}\boldsymbol{\mu}_\mathbf{q})\right)$$

$$= \frac{1}{T}\left(\mathbf{W}\mathbf{P_q r_q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q W}^T d\boldsymbol{\Sigma}_\mathbf{q}^{-1} + \boldsymbol{\mu}_\mathbf{q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q W}^T d\boldsymbol{\Sigma}_\mathbf{q}^{-1}\right) \tag{A.36}$$

Because $\boldsymbol{\Sigma}_\mathbf{q}^{-1} = \mathrm{diag}(\mathbf{S_q}\boldsymbol{\phi})$, the derivative w.r.t $\boldsymbol{\phi}$ is:

$$\boxed{\frac{d(E_{sms})}{d\boldsymbol{\phi}} = \frac{1}{T}\mathbf{S}_\mathbf{q}^T\mathrm{diag}^{-1}\left(\mathbf{W}\mathbf{P_q r_q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q W}^T + \boldsymbol{\mu}_\mathbf{q}(\mathbf{c}-\overline{\mathbf{c}}_\mathbf{q})^T\mathbf{P_q W}^T\right)} \tag{A.37}$$

The gradient vectors A.30 and A.37 are mathematically equivalent to the parameter update rule given in (Wu and Wang, 2006), where the task was to minimise generation error on acoustic HMMs for speech synthesis.

A.5 Implementation Details

Directly implementing the trajectory algorithms as matrix computation manipulation is inefficient in practice, since the dimensionality of matrices involved is of the order of TM and most entries of the matrices are zeros. Several tricks are used to reduce the computation burden utilising the sparse structure of the matrices.

First notice that the precision matrix $\mathbf{R_q}$ and the vector $\mathbf{r_q}$ are defined as:

$$\mathbf{R_q} = \mathbf{W}^T \mathbf{\Sigma_q}^{-1} \mathbf{W} \tag{A.38}$$

$$\mathbf{r_q} = \mathbf{W}^T \mathbf{\Sigma_q}^{-1} \boldsymbol{\mu_q} \tag{A.39}$$

The elements in the t^{th} row of $\mathbf{R_q}$ and $\mathbf{r_q}$ can be fully determined by inspecting values of $\mathbf{W}, \mathbf{\Sigma_q}^{-1}$ and $\boldsymbol{\mu_q}$ within a window of $[t-L, t+L]$[1]. More specifically, it is possible to write out the general terms of the elements on the t^{th} row of $\mathbf{R_q}$ and $\mathbf{r_q}$ as [2]:

$$\mathbf{R_q}[t, t+k] = \sum_{n=0,1,2,} \sum_{j=-L}^{L} \sum_{k=0}^{2L} w^{(n)}(-j) \sigma_{t+j}^{(n)} w^{(n)}(k-j) \qquad k-j \leq L \tag{A.40}$$

$$\mathbf{r_q}[t] = \sum_{n=0,1,2,} \sum_{j=-L}^{L} w^{(n)}(-j) \sigma_{t+j}^{(n)} \mu_{t+j}^{(n)} \tag{A.41}$$

where $\mathbf{R_q}[t, t+k]$ denotes the element of $\mathbf{R_q}$ at index $(t, t+k)$ and $\mathbf{r_q}[t]$ denotes the t^{th} element of $\mathbf{r_q}$. $w^{(n)}(\gamma)\,(n = 0,1,2; \gamma = -L, \ldots, L)$ are the coefficients of the dynamic features for the static, delta and delta-delta features respectively. $\sigma_t^{(n)}$ and $\mu_t^{(n)}$ are the covariance and mean value of the t^{th} HMM state for the n^{th} order feature $(n = 0,1,2)$.

[1] Note in our implementation any entry of t beyond [0,T) is ignored.

[2] The notation here is for full matrix. The notation is slightly different for banded matrces: If $\mathbf{R_q}$ and $\mathbf{U_q}$ are T by $2L+1$ banded matrices then:

$$\mathbf{R_q}[t,k] = \sum_{n=0,1,2,} \sum_{j=-L}^{L} \sum_{k=0}^{2L} w^{(n)}(-j) \sigma_{t+j}^{(n)} w^{(n)}(k-j) \qquad k-j \leq L$$

$$\mathbf{U_q}[t,0] = \sqrt{\mathbf{R_q}[t,0] - \sum_{j=1}^{\min(t,2L)} \mathbf{U_q}[t-j,0]^2} \qquad k = 0$$

$$\mathbf{U_q}[t,k] = \frac{\mathbf{R_q}[t,k] - \sum_{j=1}^{\min(t,2L)} \mathbf{U_q}[t-j,j] \cdot \mathbf{U_q}[t-j,j+k]}{\mathbf{U_q}[t,0]} \qquad 0 < k \leq 2L$$

$$\mathbf{g_q}[t] = \frac{\mathbf{r_q}[t] - \sum_{j=1}^{\min(t,2L)} \mathbf{U_q}[t-j,j] \cdot \mathbf{g_q}[t-j]}{\mathbf{U_q}[t][t]} \qquad \text{for } t \text{ from 1 to}$$

$$\overline{\mathbf{c}}_\mathbf{q} = \frac{\mathbf{g_q}[t] - \sum_{j=1}^{2L} \mathbf{U_q}[t,j] \cdot \overline{\mathbf{c}}_\mathbf{q}[t+j]}{\mathbf{U_q}[t,t]} \qquad \text{for } t \text{ from T down to 1}$$

Similarly, the t^{th} row of the Cholesky-decomposition matrix $\mathbf{U_q}$ in (3.50) and the t^{th} element of the vector $\mathbf{g_q}$ in (3.57) can be computed from the partial values of $\mathbf{R_q}$ up to time t (by solving $\mathbf{U_q^T g_q} = \mathbf{r_q}$). $\overline{\mathbf{c}}_\mathbf{q}$ can be calculated in a similar manner. Therefore the general terms of the elements on the t^{th} row of $\mathbf{U_q}$, $\mathbf{g_q}$ and $\overline{\mathbf{c}}_\mathbf{q}$ can be written as:

$$\mathbf{U_q}[t,k] = \sqrt{\mathbf{R_q}[t,t] - \sum_{j=1}^{\min(t,2L)} \mathbf{U_q}[t-j,t]^2} \qquad k = t$$

$$\mathbf{U_q}[t,k] = \frac{\mathbf{R_q}[t,k] - \sum_{j=1}^{\min(t,2L)} \mathbf{U_q}[t-j,t] \cdot \mathbf{U_q}[t-j,k]}{\mathbf{U_q}[t,t]} \qquad t < k \leq t+2L$$

$$\mathbf{g_q}[t] = \frac{\mathbf{r_q}[t] - \sum_{j=1}^{\min(t,2L)} \mathbf{U_q}[t-j,t+j] \cdot \mathbf{g_q}[t-j]}{\mathbf{U_q}[t,t]} \qquad \text{for } t \text{ from 1 to T}$$

$$\overline{\mathbf{c}}_\mathbf{q} = \frac{\mathbf{g_q}[t] - \sum_{j=1}^{2L} \mathbf{U_q}[t,t+j] \cdot \overline{\mathbf{c}}_\mathbf{q}[t+j]}{\mathbf{U_q}[t,t]} \qquad \text{for } t \text{ from T down to 1}$$

Thus in the course of the decoding we only need to look ahead L frames in order to compute the t^{th} row of $\mathbf{U_q}$ and $\mathbf{g_q}$, which is needed in computing the factorised normalisation term $Z^{(t)}_{\mathbf{q}_{t+L}}$.

Bibliography

Theodore Wilbur Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley series in probability and Mathematical Statistics. John Wiley & Sons, New York, third edition, 2003.

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

S. Axelrod, R. Gopinath, and P. Olsen. Modeling with a subspace constraint on inverse covariance matrices. In *Proc. of ICSLP 2002*, pages 2177–2180, 2002.

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

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190, March 1983.

Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. of ICASSP 1986*, pages 49–52, Tokyo, 1986.

Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

Peter Bell and Simon King. Sparse gaussian graphical models for speech recognition. In *Proc. Interspeech 2007*, Antwerp, Belgium, August 2007.

Jeff A. Bilmes, Geoff Zweig, Karim Filali, Karen Livescu, Peng Xu, Kirk Jackson, Yigal Brandman, Eric Sandness Eva Holtz, Jerry Torres, and Bill Byrne. Discriminatively structured dynamic graphical models for speech recognition. Technical report, The Johns Hopkins University, 2001.

Alan W. Black and Paul A. Taylor. The Festival Speech Synthesis System: System documentation. Technical Report HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, 1997. Avaliable at http://www.cstr.ed.ac.uk/projects/festival.html.

Herve Bourlard and Nelson Morgan. Continuous speech recognition by connectionist statistical methods. *IEEE Transactions on Neural Networks*, 4(6):893–909, November 1993.

Matthew Brand, Nuria Oliver, and Alex Pentland. Coupled hidden Markov models for complex action recognition. *IEEE Computer Vision and Pattern Recognition*,, pages 994–999, 1997.

E. Bresch, Yoon-Chul Kim, K. Nayak, D. Byrd, and S. Narayanan. Seeing speech: capturing vocal tract shaping using real-time magnetic resonance imaging. *Signal Processing Magazine, IEEE*, 25(3):123–132, May 2008.

J. S. Bridle and M. P. Ralls. An approach to speech recognition using synthesis by rule. In Frank Fallside and William A. Woods, editors, *Computer speech processing*, pages 277–292. Prentice Hall International (UK) Ltd., 1985.

John Bridle. Towards better understanding of the model implied by the use of dynamic features in HMMs. In *Proc. of ICSLP 2004*, pages 725–728, 2004.

John S Bridle, Li Deng, Joseph Picone, Hywel B. Richards, Jeff Ma, Terri Kamm, Micheal Schuster, Sandi Pike, and Roland Regan. An investigation of segmental hidden dynamic models of speech coarticulation for automatic speech recognition. Technical report, The 1998 Workshop on Language Engineering, Center for Language and Speech Processing, The Johns Hopkins University, 1998.

Ozgur Cetin. Multi-rate and variable-rate modeling of speech at phone and syllable time scales. In *Proc. of ICASSP 2005*, Philadelphia, PA, USA, March 2005.

B. Chen, Q. Zhu, and N. Morgan. Learning long-term temporal features in LVCSR using neural networks. In *Proc. of ICSLP 2004*, Jeju, Korea, October 2004.

Robert A. J. Clark, Korin Richmond, and Simon King. Multisyn: Open-domain unit selection for the Festival speech synthesis system. *Speech Communication*, 49(4): 317–330, 2007.

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

Li Deng, Dong Yu, and A. Acero. Structured speech modeling. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1492–1504, September 2006.

V. Digalakis, J.R. Rohlicek, and M. Ostendorf. A dynamical system approach to continuous speech recognition. *Proc. of ICASSP 1991*, 1:289 – 292, April 1991.

H. Dudley. The vocoder. *Bell Laboratories Record*, 17:122–126, 1939.

Sorin Dusan and Li Deng. Acoustic-to-articulatory inversion using dynamical and phonological constraints. In *Proc. of the 5th Seminar on Speech Production: Models and Data*, pages 237–240, Kloster Seeon, Germany, May 2000.

G. Evermann, H.Y. Chan, M.J.F. Gales, T. Hain, X. Liu, D. Mrva, L. Wang, and P.C. Woodland. Development of the 2003 CU-HTK conversational telephone speech transcription system. In *Proc. of ICASSP 2004*, Montreal, May 2004.

Frank Fallside. On the acquisition of speech by machines, asm. *Speech Communication*, 11(2-3):247–260, June 1992.

Frank Fallside and William A. Woods, editors. *Computer Speech Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 1985.

J. Frankel, K. Richmond, S. King, and P. Taylor. An automatic speech recognition system using neural networks and linear dynamic models to recover and model articulatory traces. In *Proc. of ICSLP 2000*, 2000.

Joe Frankel. *Linear Dynamic Models for Automatic Speech Recognition*. PhD thesis, The Centre for Speech Technology Research, Edinburgh University, April 2003.

Joe Frankel, Mirjam Wester, and Simon King. Articulatory feature recognition using dynamic Bayesian networks. *Computer Speech & Language*, 21(4):620–640, October 2007.

S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans. ASSP*, 34(1):52–59, 1986.

M.J.F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(3):272–281, May 1999.

M.J.F. Gales and S.J. Young. The theory of segmental hidden Markov models. Technical Report TR 133, Cambridge University Engineering Department, June 1993.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions of Pattern Recognition and Machine Intelligence*, 6:721–741, November 1984.

Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for linear dynamical systems. Technical Report (Short Note) CRG-TR-96-2, Department of Computer Science, University of Toronto, 22 February 1996.

James R. Glass. A probabilistic framework for segment-based speech recognition. *Computer Speech & Language*, 17(2-3):137 – 152, 2003. ISSN 0885-2308. New Computational Paradigms for Acoustic Modeling in Speech Recognition.

Guillaume Gravier, Gerasimos Potamianos, and Chalapathy Neti. Asynchrony modeling for audio-visual speech recognition. In *Proc. of Human Language Technology Conference*, pages 1–6, San Diego, California, March 2002.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

H. Hermansky, D.P.W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional HMM systems. In *Proc. of ICASSP 2000*, volume 3, pages 1635–1638, 2000.

Sadao Hiroya. *Estimation of Articulatory Movements from Speech Signal Using an HMM-Based Speech Production Model*. PhD thesis, Department of Information Processing,Tokyo Institute of Technology, August 2006.

Michael M. Hochberg. *A Comparison of State-Duration Modeling Techniques for Connected Speech Recognition*. PhD thesis, Division of Engineering, Brown University, October 1992.

John Hogden, Anders Lofqvist, Vince Gracco, Igor Zlokarnik, Philip Rubin, and Elliot Saltzman. Accurate recovery of articulator positions from acoustics: New conclusions based on human data. *The Journal of the Acoustical Society of America*, 100 (3):1819–1834, September 1996.

J. N. Holmes, I. Mattingly, and J. Shearme. Speech synthesis by rule. *Language and Speech*, 7:127–143, 1964.

W. J. Holmes and M. J. Russell. Probabilistic-trajectory segmental HMMs. *Computer Speech & Language*, 13(1):3–37, January 1999.

R. Houde. *A Study of Tongue Body Motion during Selected Consonant Sounds*. PhD thesis, University Microfilms, Ann Arbor, Mich., 1967.

Frederick Jelinek. Continuous speech recognition by statistical methods. *Proc. of the IEEE*, 64(4):532–556, April 1976.

P. Kenny, M. Lennig, and P. Mermelstein. A linear predictive HMM for vector-valued observations with applications to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(2):220–225, Feb l990.

T. Kobayashi, M. Yagyu, and K. Shirai. Application of neural networks to articulatory motion estimation. In *Proc. of ICASSP 1991*, pages 489–492, April 1991.

L. F. Lamel and J. L. Gauvain. High performance speaker-independent phone recognition using CDHMM. In *Proc. Eurospeech*, pages 121–124, Berlin, Germany, September 1993.

Lori F. Lamel, Robert H. Kasel, and Stephanie Seneff. Speech database development: Design and analysis of the acoustic-phonetic corpus. In *Proc. of the DARPA Speech Recognition Workshop*, pages 100–109, February 1986.

S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech & Language*, 1(1):29–45, March 1986.

Min Li, Chandra Kambhamettu, and Maureen Stone. Automatic contour tracking in ultrasound images. *Clinical Linguistics & Phonetics*, 19(6-7):545–554, 2005.

Björn Lindblom, James Lubker, and Thomas Gay. Formant frequencies of some fixed-mandible vowels and a model of speech motor programming by predictive simulation. *The Journal of the Acoustical Society of America*, 62(S1):S15–S15, 1977.

Zhen-Hua Ling, Korin Richmond, Junichi Yamagishi, and ren Hua Wang. Integrating articulatory features into HMM-based parametric speech synthesis (submitted). *IEEE Transactions on Audio, Speech and Language Processing*, 2008.

X. Liu, M. J. F. Gales, K. C. Sim, and K. Yu. Investigation of acoustic modeling techniques for LVCSR systems. In *Proc. of ICASSP 2005*, pages 849–852, Philadelphia, PA, March 2005.

Karen Livescu, James Glass, and Jeff Bilmes. Hidden feature models for speech recognition using dynamic Bayesian networks. In *Proc. of Eurospeech 2003*, Geneva, Switzerland, 2003.

T. Masuko, Keiichi Tokuda, Takao Kobayashi, and S. Imai. Speech synthesis using HMMs with dynamic features. In *Proc. of ICASSP 1996*, pages 389–392, Atlanta, GA, May 1996.

Yasuhiro Minami, Erik McDermott, Atsushi Nakamura, and Shigeru Katagiri. A theoretical analysis of speech recognition based on feature trajectory models. In *Proc. of Interspeech 2004*, pages 549–552, 2004.

Thomas P. Minka. From hidden Markov models to linear dynamical systems. Technical report, The MIT Media Laboratory, 1998.

K.G. Munhall, E. Vatikiotis-Bateson, and Y. Tohkura. X-ray film database for speech research. *Journal of the Acoustical Society of America*, pages 1222–1224, 1998.

Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, July 2002.

A. Nádas, D. Nahamoo, and M. A. Picheny. On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, 36(9):1432–1436, 1988.

Julian James Odell. *The Use of Context in Large Vocabulary Speech Recognition*. PhD thesis, University of Cambridge, 1995.

Peder A. Olsen and Ramesh A. Gopinath. Modeling inverse covariance matrices by basis expansion. *IEEE Transactions on Speech and Audio Processing*, 12(1):37–46, January 2004.

M. Ostendorf, A. Kannan, O. Kimball, and J.R. Rohlicek. Continuous word recognition based on the stochastic segment model. In *Proc. of DARPA Workshop CSR*, 1992.

Mari Ostendorf, Vassilios V. Digalakis, and Owen A. Kimball. From HMM's to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5):360–378, 1996.

George Papcun, Judith Hochberg, Timothy R. Thomas, Francois Laroche, Jeff Zacks, and Simon Levy. Inferring articulation and recognizing gestures from acoustics with a neural network trained on x-ray microbeam data. *Journal of the Acoustical Society of America*, 92(2):688–700, August 1992.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, inc., San mateo, California, 1988.

J. Picone, S. Pike, R. Regan, T. Kamm, J. Bridle, L. Deng, Z. Ma, H. Richards, and M. Schuster. Initial evaluation of hidden dynamic models on conversational speech. In *Proc. of ICASSP 1999*, Phoenix, Arizona, USA, May 1999.

A. B. Poritz. Hidden Markov models: a guided tour. *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 7–13 vol.1, April 1988.

D. Povey and P.C. Woodland. Minimum phone error and I-smoothing for improved discriminative training. In *Proc. of ICASSP 2002*, volume 1, pages 105–108, Orlando, Florida, May 2001.

Chao Qin and M. A. Carreira-Perpiñán. An empirical investigation of the nonuniqueness in the acoustic-to-articulatory mapping. In *Proc. of Interspeech 2007*, pages 74–77, 2007.

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

M. G. Rahim, W. B. Keijn, J. Schroeter, and C. C. Goodyear. Acoustic to articulatory parameter mapping using an assembly of neural networks. In *Proc. of ICASSP 1991*, pages 485–488, April 1991.

Hywel B. Richards and John S. Bridle. The HDM: A segmental hidden dynamic model of coarticulation. In *Proc. of ICASSP 1999*, Phoenix, Arizona, USA, May 1999.

K. Richmond. *Estimating Articulatory Parameters from the Acoustic Speech Signal*. PhD thesis, The Centre for Speech Technology Research, Edinburgh University, 2002.

K. Richmond, S. King, and P. Taylor. Modelling the uncertainty in recovering articulation from acoustics. *Computer Speech & Language*, 17:153–172, 2003.

Korin Richmond. Trajectory mixture density networks with multiple mixtures for acoustic-articulatory inversion. In M. Chetouani, A. Hussain, B. Gas, M. Milgram, and J.-L. Zarader, editors, *Advances in Nonlinear Speech Processing, International Conference on Non-Linear Speech Processing, NOLISP 2007*, volume 4885 of *Lecture Notes in Computer Science*, pages 263–272. Springer-Verlag Berlin Heidelberg, December 2007.

A. J. Robinson, G. D. Cook, D. P. W. Ellis, E. Fosler-Lussier, S. J. Renals, and D. A. G. Williams. Connectionist speech recognition of broadcast news. *Speech Communication*, 37(1):27–45, May 2002.

Anthony J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, March 1994.

Antti-Veikko I. Rosti. *Linear Gaussian Models for Speech Recognition*. PhD thesis, University of Cambridge, 2004.

M. Russell. A segmental HMM for speech pattern modelling. In *Proc. of ICASSP 1993*, volume 2, pages 499 – 502, Minneapolis, April 1993.

M. Russell and R. Moore. Explicit modelling of state occupancy in hidden Markov models for automatic speech recognition. In *Proc. of ICASSP 1985*, volume 10, pages 5–8, Apr 1985.

Martin J. Russell and Philip J.B. Jackson. A multiple-level linear/linear segmental hmm with a formant-based intermediate layer. *Computer Speech & Language*, 19 (2):205 – 225, 2005. ISSN 0885-2308. doi: DOI:10.1016/j.csl.2004.08.001.

M. R. Schroeder. Determination of the geometry of the human vocal tract by acoustic measurements. *The Journal of the Acoustical Society of America*, 1967.

Juergen Schroeter and M. Mohan Sondhi. Techniques for estimating vocal-tract shapes from the speech signal. *IEEE Transactions on Acoustics and Speech Signal Processing*, 2(1, Part II):133–150, 1994.

Tomoki Toda and Keiichi Tokuda. Speech parameter generation algorithm considering global variance for HMM-based speech synthesis. In *Proc. of Interspeech 2005*, Portugal, Lisbon, September 2005.

Tomoki Toda, Alan W Black, and Keiichi Tokuda. Acoustic-to-articulatory inversion mapping with gaussian mixture model. In *Proc. of ICSLP 2004*, pages 1129–1132, Jeju, Korea, 2004.

Keiichi Tokuda, Takao Kobayashi, and S. Imai. Speech parameter generation from HMM using dynamic features. In *Proc. of ICASSP 1995*, pages 660–663, Detroit, MI, May 1995.

Keiichi Tokuda, Takayoshi Yoshimura, Takao Kobayashi Takashi Masuko, and Tadashi Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proc. of ICASSP 2000*, pages 1315–1318, Istanbul, Turkey, June 2000.

Keiichi Tokuda, Heiga Zen, and Tadashi Kitamura. Reformulating the HMM as a trajectory model. In *Proc. of Beyond HMM*, December 2004.

V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. MMIE training of large vocabulary recognition systems. *Speech Communication*, 22(4):303–314, 1997.

C. Wellekens. Explicit time correlation in hidden Markov models for speech recognition. In *Proc. of ICASSP 1987*, volume 12, pages 384 – 386, Apr 1987.

Christopher K. I. Williams. How to pretend that correlated variables are independent by using difference observations. *Neural Computation*, 17(1):1–6, 2005.

P. C. Woodland and C. J. Leggetter. Speaker adaptation of continuous density HMMs using multivariate linear regression. In *Proc. of ICSLP 1994*, pages 451–454, 1994.

Yi-Jian Wu and Ren-Hua Wang. Minimum generation error training for HMM-based speech synthesis. In *Proc. of ICASSP 2006*, volume 1, pages I–I,14–19, 2006.

J. Yamagishi, K. Ogata, Y. Nakano, J. Isogai, and T. Kobayashi. HSMM-based model adaptation algorithms for average-voice-based speech synthesis. In *Proc. of ICASSP 2006*, volume 1, 2006.

Junichi Yamagishi and Takao Kobayashi. Average-voice-based speech synthesis using hsmm-based speaker adaptation and adaptive training. *IEICE Trans. Information and Systems*, E90-D(2):533–543, February 2007.

S. J. Young and P. C. Woodland. State clustering in hidden Markov model-based continuous speech recognition. *Computer Speech & Language*, 8(4):369–383, October 1994.

SJ Young, NH Russell, and JHS Thornton. Token passing: a conceptual model for connected speech recognition systems'. Technical Report TR38, CUED Technical Report, Cambridge University, 1989.

S.J. Young, J.J. Odell, and P.C. Woodland. Tree-based state tying for high accuracy modelling. In *Human Language Technology Workshop*, pages 307–312, Plainsboro, NJ, 1994.

Steve Young, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. *The HTK Book (for HTK Version 3.4)*. Cambridge University, Cambridge, England, 2006.

Dong Yu, Li Deng, and Alex Acero. A lattice search technique for a long-contextual-span hidden trajectory model of speech. *Speech Communication*, 48(9):1214–1226, September 2006.

Heiga Zen and Tomoki Toda. An overview of Nitech HMM-based speech synthesis system for Blizzard challenge 2005. In *Proc. of Interspeech 2005*, pages 93–96, Portugal, Lisbon, September 2005.

Heiga Zen, Keiichi Tokuda, and Tadashi Kitamura. A Viterbi algorithm for a trajectory model derived from HMM with explicit relationship between static and dynamic features. In *Proc. of ICASSP 2004*, pages 837–840, Montreal, May 2004.

Heiga Zen, Tomoki Toda, and Keiich. Tokuda. The Nitech-NAIST HMM-based speech synthesis system for the blizzard challenge 2006. In *Proc. of Blizzard Challenge 2006 workshop*, September 2006.

Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan W. Black, and Keiichi Tokuda. The HMM-based speech synthesis system version 2.0. In *Proc. of ISCA SSW6*, pages 294–299, Bonn, Germany, August 2007.

Le Zhang and Steve Renals. Acoustic-articulatory modelling with the trajectory HMM. *IEEE Signal Processing Letters*, 15:245–248, 2008.

Ciyou Zhu, Richard Byrd, and Jorge Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

Igor Zlokarnik. Adding articulatory features to acoustic features for automatic speech recognition. *The Journal of the Acoustical Society of America*, 97(5):3246, May 1995.

Geoffrey Zweig and Stuart J. Russell. Speech recognition with dynamic Bayesian networks. In *AAAI*, 1998.