PROBABILISTIC ACOUSTIC MODELLING FOR PARAMETRIC SPEECH SYNTHESIS

Sean Matthew Shannon
Trinity College

Department of Engineering
University of Cambridge
Trumpington Street
Cambridge, CB2 1PZ, U.K.

September 2014

This dissertation is submitted for the degree of Doctor of Philosophy

PROBABILISTIC ACOUSTIC MODELLING FOR PARAMETRIC SPEECH SYNTHESIS

Sean Matthew Shannon

Summary

This thesis proposes a new probabilistic model and speech parameter generation method for statistical parametric text-to-speech synthesis. These are designed to address drawbacks with the conventional approach to statistical parametric speech synthesis, in which a form of hidden Markov model (HMM) is used for modelling and one of two methods, known as standard speech parameter generation and parameter generation considering global variance, is used for speech parameter generation.

The new probabilistic model is a form of autoregressive HMM, and addresses an inconsistency in the treatment of the dynamics of speech parameter sequences present in the conventional approach. Compared to previous attempts to address this inconsistency such as the trajectory HMM, our model has the advantage of supporting efficient parameter estimation using expectation maximization and decision tree clustering.

The new speech parameter generation method is informed by a mathematical analysis of parameter generation considering global variance and an investigation into why it sometimes introduces artifacts into the synthesized speech. The proposed method is as fast as standard speech parameter generation but improves naturalness as much as parameter generation considering global variance, without introducing artifacts. This makes it an attractive generation method when fast, high-quality generation is desired.

As we develop the new model and generation method we also investigate and develop a better understanding of several aspects of existing approaches. We show that the conventional approach, due its lack of consistency, greatly underestimates the variance present in speech parameter sequences. We present a view of the trajectory HMM as a directed graphical model, which serves to highlight both its similarities to and differences from the autoregressive HMM. Finally we investigate the causes of the artifacts sometimes introduced by parameter generation considering global variance. We find that excursions in the generated speech parameter sequence are associated with artifacts, and identify the source of these excursions as a pathology in parameter generation considering global variance.

Contents

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

Acknowledgements

This thesis owes a lot to many people. First and foremost, I would like to thank my supervisor Bill Byrne. His insight and no-nonsense approach have transformed the way I work and write, and I found our meetings extremely rewarding. I would also like to thank Carl Rasmussen for being my advisor. Thanks also to Catherine Breslin, Rogier van Dalen, Milica Gasic, Pirros Tsiakoulis and of course Bill Byrne for their feedback on this thesis at various stages of its development.

Thanks to Mark Gales, Anton Ragni, Matt Gibson, Kai Yu, Pirros Tsiakoulis, Simon King, Junichi Yamagishi, Gustav Henter, Keiichi Tokuda, Yoshihiko Nankaku and above all Heiga Zen for many fascinating chats about speech synthesis; and to Jurgen van Gael, Yee Whye Teh, Andrew Wilson, Matt Seigel, Zoi Roupakia, Milica Gasic, Juan Pino, Rory Waite, Chao Zhang, Pierre Lanchantin and Penny Karanasou for fascinating discussions about a myriad of other topics. Special thanks must go to Rogier van Dalen for providing endless interesting discussions which evolved in ways that I could never have imagined; it is always a surprise and a delight to talk to him.

Thanks must also go to David MacKay, both because without him I may never have studied speech at Cambridge, and because his excellent text book opened my eyes to the wonders of probabilistic modelling. Yee Whye Teh and Rich Turner also deserve special mention for restoring my faith, unbeknownst to them, in probabilistic modelling with their excellent work on Bayesian language modelling and probabilistic amplitude demodulation.

I would also like to say thank you for the financial support I received. The work in this thesis was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 213845 (EMIME) and in part by EPSRC Programme Grant EP/I031022/1 (Natural Speech Technology). I will be forever grateful for the generous way I was able to pursue the course I wanted to within the EMIME and NST projects, and thank you in particular to Bill Byrne and Phil Woodland in Cambridge and Simon King in Edinburgh for making this possible. Thanks also to Trinity College for making it possible for me to attend a conference.

Finally thank you to parents, for being unfailingly supportive of everything I have done, and for always being there when I have needed them.

Notation and terminology

In this section we review the notational conventions and terminology used in this thesis. We first review basic notation. A set consisting of elements 1, 2 and 3 is denoted $\{1, 2, 3\}$. A pair is denoted $(5, 2)$, a 3-tuple $(5, 2, 3)$, etc. We denote the natural numbers $\{1, 2, \ldots\}$ by $\mathbb{N}$, the integers by $\mathbb{Z}$ and the real numbers by $\mathbb{R}$.

We now review our notation and terminology for functions. A function has a *domain* and a *range*, both of which are sets, and maps each *input* element x in the domain to a unique *output* element, denoted $f(x)$, in the range. In the expression $f(x)$ we refer to x as the *argument* of f. We write $f : A \to B$ to indicate that f is a function with domain A and range B. We write $x \mapsto x^2$ to denote a function which maps the element x to x^2, leaving the domain and range of the function unspecified. Thus $f : \mathbb{R} \to \mathbb{R}$, $x \mapsto x^2$ fully specifies a simple quadratic function. For functions where the domain is a Cartesian product $A \times B$ we often write $f(x, y)$ instead of $f((x, y))$ and consider f to be a function of two arguments. Sometimes, particularly when the domain is a "discrete" set, it is convenient to use the subscript notation f_i as an alternative to $f(i)$, in which case we refer to i as an *index*. Similarly we may write f_{ij} as an alternative to $f(i, j)$. We use superscripted expressions such as f^r to denote taking a power rather than indexing.

We now discuss our notation and terminology for sequences. A *sequence* is a function whose domain is a totally ordered set, for example $\{1, \ldots, T\}$ or $\mathbb{Z}$. We refer to the domain of a sequence as its *index set*, and the cardinality of the index set as the *length* of the sequence. We use $\text{len}(a)$ to denote the length of a sequence a. Most of the sequences we consider in this thesis have index set $\{1, \ldots, T\}$ for some length T. We generally use the indexing notation for sequences, for example a_t rather than $a(t)$. If a is a sequence, we sometimes use $a_{1:T}$ to denote the subsequence $[a_t]_{t=1}^T$, where the notation $[\cdot]_{t=1}^T$ is described below.

We now discuss our terminology for tensors. A *tensor* is a function whose domain is a Cartesian product of R totally ordered sets. We refer to R as the *rank* of the tensor. Thus a rank 1 tensor is just a sequence. We refer to a rank 0 tensor as a *scalar* and a rank 2 tensor as a *matrix*. We do not use the geometric notion of a tensor from multilinear algebra in this thesis. We often refer to a rank 1 tensor as a *vector*. However we also use the term

vector to mean an element of a *vector space*, i.e. a set for which there is a notion of addition and scalar multiplication that satisfies certain axioms. Which meaning of *vector* is intended should be clear from context. It is sometimes helpful to consider a T-dimensional rank 1 tensor a as a $T \times 1$ matrix or *column vector*, in which case its transpose a^T is a $1 \times T$ matrix or *row vector*. We generally use the indexing notation for tensors, for example writing a_{ij} rather than $a(i, j)$.

We will often find an alternative notation for constructing functions convenient. The expression $[\cdot]_{x \in A}$, where $\cdot$ is an expression that typically involves x as a free variable, denotes a function f with domain A where $f(u)$ is given by the expression $\cdot$ evaluated with x set to u. For example: $[x^2]_{x \in \mathbb{R}}$ denotes a simple quadratic function; and $[t^2]_{t \in \mathbb{N}}$ denotes the sequence $1, 4, 9, \ldots$. We refer to this as *function builder notation*. The range of the constructed function is left unspecified by this notation. We use $[\cdot]_{t=a}^{b}$ as a concise notation for $[\cdot]_{t \in A}$ where $A = \{a, \ldots, b\}$ and $a, b \in \mathbb{Z}$. We sometimes use $[\cdot]_x$ if we wish to leave the domain implied. For example trivially for any sequence a we have $a = [a_t]_t$. When the expression $\cdot$ above is a tuple, for example $[(a_t, b_t)]_{t \in A}$, we often drop the nested brackets and write $[a_t, b_t]_{t \in A}$. We also allow multiple indices in function builder notation. For example $[\cdot]_{i \in I, j \in J}$, or simply $[\cdot]_{i,j}$, denotes a function with domain $I \times J$.

Our notation for probability distributions is as follows. We denote the probability of a "discrete" random variable x taking a value a by $\mathbb{P}(x = a)$. We also denote the probability density of a "continuous" random variable x at a by $\mathbb{P}(x = a)$. The equals sign here could be considered misleading but the notation has the advantage of being concise and precise. From a measure theoretic perspective there is no inherent distinction between the discrete and continuous cases, and in the discrete case $\mathbb{P}(x = a)$ can be viewed as a density with respect to the counting measure. Where it is not likely to cause confusion we follow the common convention of using the same symbol to denote a random variable and its value, and of writing $\mathbb{P}(x)$ as shorthand for $\mathbb{P}(x = x)$.

Finally, in following the text it may be helpful to know how we use the overline or overbar. If a is a quantity of interest, e.g. the area of a square, and there is a function of interest which returns the value of a in a particular situation, e.g. the function $l \mapsto l^2$ which gives the area of a square from its side length, then this function is often denoted $\overline{a}$.

Abbreviations

The following abbreviations are used in this thesis:

CRF	conditional random field
DTW	dynamic time warping
EM	expectation maximization
GAM	Gaussian acoustic model
GMSD	global mean squared deviation
GSTM	Gaussian state transition model
GV	global variance
HMM	hidden Markov model
HTS	HMM-based speech synthesis system (software)
LGLAR	linear Gaussian linear autoregressive
LSPA	local static parameter adjustment
MCD	mel cepstral distortion
MDL	minimum description length
MEMM	maximum entropy Markov model
MSE	mean squared error
TSLP	test set log probability

Chapter 1

Introduction

In recent years *statistical parametric speech synthesis* has been widely adopted for building text-to-speech synthesis systems, offering the ability to build a synthesis system from a corpus of natural speech, output of predictable and reasonably high quality, a relatively low memory footprint, and the ability to mimic and control voice characteristics (Zen et al., 2009; Taylor, 2009). This approach involves the use of a probabilistic model defined over sequences of *speech parameters* representing the speech audio. The parameters of the probabilistic model are first learned from the corpus of natural speech in a process referred to as *model parameter estimation*, then the trained probabilistic model is used to synthesize speech parameter sequences for new text in a process referred to as *speech parameter generation*. The field has largely converged on a standard approach to statistical parametric speech synthesis based on *hidden Markov models (HMMs)* (Zen et al., 2009), and we refer to this as the *standard HMM synthesis framework*.

Despite its success, there are many potential areas for improvement in the standard framework, and we consider two of these in detail in this thesis. The first issue we consider is the fact that the standard HMM synthesis framework is inconsistent in its treatment of the dynamics of speech parameter sequences, with one approach to modelling these dynamics being used during parameter estimation and another approach used during speech parameter generation. As we will see, this inconsistency means that the probabilistic model trained by the standard framework is a terrible probabilistic model of speech parameter sequences, in the sense of having a very low *test set log probability*. Thankfully typical speech parameter generation techniques, such as *standard speech parameter generation* and *parameter generation considering global variance*, do not rely strongly on aspects of the probabilistic model that are deficient due to this inconsistency, and it is still possible to obtain high quality synthesized speech. Nevertheless it may be of interest, both theoretically and practically, to address this inconsistency. Previous work addressing this inconsistency has resulted in consistent models at the cost of more complicated parameter estimation (van

Horn, 2002; Zen et al., 2007b). For example the *trajectory HMM* (Zen et al., 2007b) results in more natural synthesized speech, but does not support efficient parameter estimation methods such as *expectation maximization* or *decision tree clustering*, and gradient ascent-based parameter estimation using a fixed alignment and fixed decision trees are typically used instead. The challenge remains to find a model which can efficiently and consistently be used for both parameter estimation and speech parameter generation.

The second issue with the standard framework which we consider is the fact that there are trade-offs involved in the choice of speech parameter generation method. Standard speech parameter generation optimizes a quadratic utility function which has an analytic solution, whereas parameter generation considering global variance optimizes a more complicated utility function designed to increase the *global variance (GV)* of generated trajectories, which is a measure of how spread out the values in a trajectory are around their mean value. The two techniques have different strengths and weaknesses: parameter generation considering global variance results in clearer and less "muffled"-sounding speech than standard generation; the conventional algorithm for standard generation is fast and exact, whereas the conventional gradient ascent-based algorithm for parameter generation considering global variance is slower and approximate; there is a low latency approximate algorithm available for standard generation but not for parameter generation considering global variance; and parameter generation considering global variance sometimes results in *artifacts* in the synthesized speech, though using *early stopping* during gradient ascent is an effective way to reduce the occurrence of artifacts. It would be preferable to have a generation method which achieves both speed and quality and does not introduce artifacts.

In this thesis we address the above two issues. To address the first issue, we propose using a form of *autoregressive HMM*, specifically the *linear Gaussian linear autoregressive HMM (LGLAR HMM)*, for parametric speech synthesis. The proposed model is consistent, supports efficient model parameter estimation, supports existing effective speech parameter generation methods, and supports a simple and exact low latency parameter generation method. To address the second issue, we introduce a new GV-like generation method which is as fast as standard generation but improves naturalness as much as parameter generation considering global variance, without introducing artifacts. This makes it a very attractive generation method when fast, high-quality generation is desired.

In addition we attempt to contribute to a better understanding of various aspects of existing and new models and generation methods. For example:

- We show that one of the major practical effects of the lack of consistency present in the standard framework is that it greatly underestimates predictive variance.
- We identify a weakness in the autoregressive HMM which we refer to as *future state blindness*. This weakness must be alleviated by any practical autoregressive HMM

system in order to obtain good performance, and we explore the implications of this weakness for the LGLAR HMM.

- We describe two ways to view the trajectory HMM and the LGLAR HMM within a common framework, including a view of the trajectory HMM as a directed graphical model. These views serve to highlight both the similarities and differences between the two models.

- Parameter generation considering global variance is often thought of as "sharpening" the *trajectory* produced by standard generation. We make the sense in which this happens precise, showing that it involves subtracting a constant from the elements on the diagonal of the *precision matrix*.

- We show theoretically and experimentally that consistent models such as the trajectory HMM and autoregressive HMM model global variance very well. This implies that, for consistent models, parameter generation considering global variance is perhaps best viewed as compensating for a deficiency in standard generation, not a deficiency in the model.

- We investigate the causes of artifacts in some detail. We show that parameter generation considering global variance without early stopping suffers from a pathology which can result in *excursions* in the generated trajectory, and show that excursions in different cepstral components often occur simultaneously. We provide evidence that these excursions are the cepstral-level phenomenon responsible for the perceptual-level phenomenon of artifacts. We also provide a partial explanation of why early stopping helps to prevent artifacts, and show that a previously suggested hypothesis about the causes of artifacts has only a small effect for our experimental systems. These considerations provide us with a better understanding of when artifacts are likely to occur and how to prevent them.

The remainder of this thesis is organized as follows. In Chapter 2 we review some of the fundamentals of probabilistic modelling. In Chapter 3 we review some of the fundamentals of statistical parametric speech synthesis, and define the standard HMM synthesis framework and the trajectory HMM. In Chapter 4 we propose using a form of autoregressive HMM, the LGLAR HMM, for statistical parametric speech synthesis, and compare it to the standard framework and the trajectory HMM in subjective and objective evaluations. In Chapter 5 we examine a variety of connections between the standard framework, the trajectory HMM and the LGLAR HMM, and discuss the effect of the inconsistency present in the standard framework. In Chapter 6 we investigate parameter generation considering global variance in detail and propose a new generation method which addresses some of its drawbacks.

Chapter 2

Probabilistic modelling

In this chapter we review some of the fundamentals of *probabilistic modelling*, by which we mean the use of probability distributions to model aspects of the world. Using probabilistic models allows the broad form of a model to be specified by the investigator while the details are learned automatically from data. The ability to incorporate domain-specific knowledge in the design of probabilistic models while allowing specific details to be learned in a data-driven and flexible way is a powerful asset of probabilistic modelling.

The layout of this chapter is as follows. We first describe how probabilistic models can be used for *prediction*. We discuss both probabilistic models and conditional probabilistic models, and review fundamental notions such as *parameter estimation*. We also discuss how to evaluate probabilistic models. We then review some basic probabilistic models that will be used as building blocks for many of the more complicated models considered in this thesis. We also review the concept of an *exponential family* and a *conditional exponential family*. These concepts will be central to much of our exposition and will provide us with a number of insights about specific models. We then review two parameter estimation methods, *decision tree clustering* and *expectation maximization*, that are widely used and broadly applicable to certain classes of probabilistic model. Finally we review how the basic probabilistic models can be combined to form a sequential probabilistic model known as the *hidden Markov model* and discuss how to perform parameter estimation for this model.

2.1 Probabilistic models and prediction

Consider an infinite sequence $[Y_r]_{r\in\mathbb{N}}$ of random variables, where each *observation* Y_r takes its value in a set $\mathcal{Y}$. In many situations we wish to make some form of prediction about the value of a *test corpus* $Y^* = [Y_r]_{r\in R^*}$ of future observations given the value of a *training corpus* $Y^\circ = [Y_r]_{r\in R^\circ}$ of past observations, where R° and R^* are disjoint subsets of $\mathbb{N}$. We refer to r as the *utterance index*; this terminology is speech-focused but the concept is

general. We refer to $[Y_r]_{r\in\mathbb{N}}$ as the *potential corpus*. For example for a series of coin tosses the set $\mathcal{Y}$ might consist of heads and tails, with the potential corpus consisting of an infinite sequence of tosses.

Often it is reasonable to assume that the potential corpus is generated by sampling each observation independently from a common distribution, where the common distribution is known to belong to some specified family but the precise member of this family, as specified by a random variable λ, is unknown and to be inferred from data. In this case we have

$$\mathbb{P}(Y_{1:N} \mid \lambda) = \prod_{r=1}^{N} \mathbb{P}(Y_r \mid \lambda) \tag{2.1}$$

for any $N \in \mathbb{N}$. The conditional distribution $\mathbb{P}(Y_r \mid \lambda)$ above is the same for all r, and so when specifying this distribution we often simply write $\mathbb{P}(y \mid \lambda)$, where y is an unspecified observation in the potential corpus and takes values in $\mathcal{Y}$. For simplicity we typically assume λ is a member of a finite set, a finite-dimensional Euclidean space, or a combination of the two. In this case we refer to λ as the collection of *model parameters*, and refer to the family of probability distributions parameterized by λ as a *parametric family* or *probabilistic model*. When we write "the probabilistic model $\mathbb{P}(y \mid \lambda)$" we therefore intend to denote a family of probability distributions over y, where the family is parameterized by λ.

Sometimes the desired form of the prediction is a probability distribution over possible test corpus values. One approach to producing predictions of this form is to use $\mathbb{P}(Y^* \mid \lambda = \hat{\lambda}(Y^\circ))$ as the desired distribution, where $\hat{\lambda}$ is a function mapping each possible value of the training corpus to a value of the model parameters. We refer to $\hat{\lambda}(Y^\circ)$ as a *point estimate* of the model parameters λ, and refer to the process of choosing the value of the model parameters as *parameter estimation*, *training* or *learning*. For *maximum likelihood estimation* the point estimate is chosen to maximize the training corpus likelihood, that is

$$\hat{\lambda}(Y^\circ) = \arg\max_{\lambda} \mathbb{P}(Y^\circ \mid \lambda) \tag{2.2}$$

Maximum likelihood estimation is susceptible to *overfitting* (MacKay, 2003; Bishop, 2006), where the estimated model parameters encode happenstances of the training corpus rather than facts about the underlying distribution, leading to good training corpus performance but bad test corpus performance. If there is not substantial overfitting then we say the parameters have been estimated *robustly*. Despite this disadvantage, maximum likelihood estimation has the advantage of often being slightly simpler to implement than alternatives with better properties such as Bayesian (MacKay, 2003; Bishop, 2006) approaches, and we will use maximum likelihood estimation in this thesis.

2.1.1 Latent variable models

Often $\mathbb{P}(y \mid \lambda)$ is of the form

$$\mathbb{P}(y \mid \lambda) = \sum_x \mathbb{P}(y \mid x, \lambda)\, \mathbb{P}(x \mid \lambda) \tag{2.3}$$

In this case we refer to the random variable x as a *hidden* or *latent* variable, and the corresponding model as a *latent variable model*. Just as y above denotes the observation Y_r for some unspecified utterance r, x above denotes the latent variable X_r for the same utterance. The latent variable takes values $X^\circ = [X_r]_{r \in R^\circ}$ on training corpus utterances and values $X^* = [X_r]_{r \in R^*}$ on test corpus utterances. The training corpus and test corpus themselves are still Y° and Y^* since the latent variable is unobserved.

2.1.2 Conditional probabilistic models

The conceptual framework described at the start of this section can be extended to the case of conditional prediction. Each utterance index r is now associated with a *conditioned-on variable* $X_r \in \mathcal{X}$, whose value is known at prediction time even for the test corpus. We wish to make some form of prediction about the value of future observations $Y^* = [Y_r]_{r \in R^*}$ given the value of future conditioned-on variables $X^* = [X_r]_{r \in R^*}$, the value of past observed variables $Y^\circ = [Y_r]_{r \in R^\circ}$, and the value of past conditioned-on variables $X^\circ = [X_r]_{r \in R^\circ}$. Now the training corpus is (X°, Y°) and the test corpus is (X^*, Y^*). For example we might have a red coin and a blue coin, where for each utterance r we toss the coin with colour X_r, specified by some external source, and record the result Y_r. Here $\mathcal{X} = \{\text{red}, \text{blue}\}$.

Sometimes it is reasonable to assume that

$$\mathbb{P}(Y_{1:N} \mid X_{1:N}, \lambda) = \prod_{r=1}^{N} \mathbb{P}(Y_r \mid X_r, \lambda) \tag{2.4}$$

for any $N \in \mathbb{N}$. Again for simplicity we typically assume λ is a member of a finite set, a finite-dimensional Euclidean space, or a combination of the two. We refer to a family of conditional probability distributions parameterized by λ as a *conditional parametric family* or *conditional probabilistic model*. When we write "the conditional probabilistic model $\mathbb{P}(y \mid x, \lambda)$" we therefore intend to denote a family of conditional probability distributions over y given x, where the family is parameterized by λ. Maximum likelihood estimation is now defined by

$$\hat{\lambda}(X^\circ, Y^\circ) = \arg\max_\lambda \mathbb{P}(Y^\circ \mid X^\circ, \lambda) \tag{2.5}$$

2.1.3 Statistics

A *statistic* is a real-valued function of the observed data. Its value can be seen as a one-dimensional summary of the observed data, paying particular attention to certain aspects

of the data and ignoring others. Thinking in terms of statistics will prove helpful in various sections below, and in this section we establish terminology.

In the context of a probabilistic model $\mathbb{P}(y \mid \lambda)$ where y takes values in $\mathcal{Y}$, an *utterance-level statistic* is a function $\mathcal{Y} \to \mathbb{R}$, and a *corpus-level statistic* is a function $\mathcal{S}(\mathcal{Y}) \to \mathbb{R}$, where $\mathcal{S}(\mathcal{Y})$ denotes the set of finite sequences with values in $\mathcal{Y}$. Often a corpus-level statistic is obtained by summing an utterance-level statistic across all utterances in the corpus. In the context of a conditional probabilistic model $\mathbb{P}(y \mid x, \lambda)$ where y takes values in $\mathcal{Y}$ and x takes values in $\mathcal{X}$, an *utterance-level statistic* is a function $\mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and a *corpus-level statistic* is a function $\mathcal{S}(\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$.

Since the observed data is a random variable, so is the value of a statistic. We refer to the marginal distribution over the value of a statistic under a given probabilistic model as the *implied distribution* of that statistic. For a probabilistic model $\mathbb{P}(y \mid \lambda)$, the implied distribution for an utterance-level statistic f is a distribution over the value of $f(y)$ given λ, and the implied distribution for a corpus-level statistic f for a corpus Y°, say, is a distribution over the value of $f(Y^\circ)$ given λ. For a conditional probabilistic model $\mathbb{P}(y \mid x, \lambda)$, the implied distribution for an utterance-level statistic f is a distribution over the value of $f(x, y)$ given x and λ, and the implied distribution for a corpus-level statistic f for a corpus (X°, Y°), say, is a distribution over the value of $f(X^\circ, Y^\circ)$ given X° and λ.

We have defined statistics as real-valued functions of the observed data above, but we sometimes also refer to a vector-valued function of the observed data as a statistic. A vector-valued statistic may be thought of as a finite collection of real-valued statistics.

2.1.4 Evaluating probabilistic models

There is a natural, simple and powerful way to evaluate the combination of a probabilistic model $\mathbb{P}(y \mid \lambda)$ and a point estimation procedure $\hat{\lambda}$. The *test set log probability (TSLP)* is defined as

$$\log \mathbb{P}(Y^* \mid \lambda = \hat{\lambda}(Y^\circ)) \tag{2.6}$$

where Y^* is the value of the test corpus actually observed and Y° is the value of the training corpus observed. In the case of a conditional probabilistic model the TSLP is defined as

$$\log \mathbb{P}(Y^* \mid X^*, \lambda = \hat{\lambda}(X^\circ, Y^\circ)) \tag{2.7}$$

Since the probability distribution is determined before observing the test corpus, there is no way to "cheat" the metric: the only way to achieve a high expected TSLP score is for the trained model to model unseen data well. TSLP is measured in *nats*, which is a unit of information bearing the same relationship to the natural logarithm as the bit bears to the base-2 logarithm (MacKay, 2003).

2.2 Some probabilistic models

In this section we describe some of the basic probabilistic models and conditional probabilistic models that we will make use of in this thesis.

2.2.1 Exponential families

An *exponential family* (Bishop, 2006) is a probabilistic model of a particular form. They are a useful concept for two reasons. Firstly many commonly used probabilistic models are exponential families, and this conceptual framework allows them to be treated in a unified way. Secondly each exponential family admits simple finite-dimensional *sufficient statistics*, defined below, which simplifies parameter estimation techniques such as maximum likelihood, expectation maximization and decision tree clustering. In fact, exponential families are the only non-conditional probabilistic models (of a certain broad class) which admit finite-dimensional sufficient statistics, a result known as the Pitman-Koopman theorem or Fisher-Darmois-Koopman-Pitman theorem (Barankin and Maitra, 1963).

We give a mathematically loose definition of an exponential family, ignoring issues such as integrability. An *exponential family* is a probabilistic model $\mathbb{P}(y \,|\, \eta)$ specified by a set $\mathcal{Y}$, a function $f : \mathcal{Y} \to \mathbb{R}^K$ for some non-negative integer K, a function $h : \mathcal{Y} \to \mathbb{R}$, and a convex set $\Xi \subset \mathbb{R}^K$. For any $\eta \in \Xi$ the random variable y takes values in $\mathcal{Y}$. Define a function $Z : \mathbb{R}^K \to \mathbb{R}$ by

$$Z(\eta) = \int \exp\left(h(y) + \sum_k \eta_k f_k(y)\right) \mathrm{d}y \tag{2.8}$$

where we have opted to view f as a finite collection $[f_k]_{k=1}^K$ of functions where each f_k is a function $\mathcal{Y} \to \mathbb{R}$. To define a valid exponential family, $Z(\eta) < \infty$ must hold for all η in Ξ. For each $\eta \in \Xi$ the corresponding probability distribution over $\mathcal{Y}$ is defined by

$$\mathbb{P}(y \,|\, \eta) = \frac{1}{Z(\eta)} \exp\left(h(y) + \sum_k \eta_k f_k(y)\right) \tag{2.9}$$

$$\text{so } \log \mathbb{P}(y \,|\, \eta) = h(y) + \sum_k \eta_k f_k(y) - \log Z(\eta) \tag{2.10}$$

The parameterization of the probabilistic model in terms of η is referred to as the *natural parameterization* (Bishop, 2006), and we use the phrase "the natural parameters η" to indicate that we are considering the natural parameterization with parameters η. The function h can be seen as specifying a base measure with respect to which the exponential family is defined, and we refer to it as the *base measure function*. In a machine learning context each f_k is sometimes referred to as a *feature function*. Note that $\mathcal{Y}$, K, f, h and Ξ determine the exponential family, while η specifies the member of the exponential family.

Exponential families have a useful sufficiency property. The log likelihood after observing a training corpus $Y^\circ = [Y_r]_{r \in R^\circ}$ is

$$\log \mathbb{P}\left(Y^\circ \,|\, \eta\right) = \sum_r h(Y_r) + \sum_k \eta_k \sum_r f_k(Y_r) - |R^\circ| \log Z(\eta) \tag{2.11}$$

$$= H(Y^\circ) + \sum_k \eta_k F_k(Y^\circ) - \mathrm{len}(Y^\circ) \log Z(\eta) \tag{2.12}$$

$$= \tilde{h} + \sum_k \eta_k \tilde{f}_k - \tilde{R} \log Z(\eta) \tag{2.13}$$

where

$$H(Y^\circ) = \sum_r h(Y_r) \tag{2.14}$$

$$F(Y^\circ) = \sum_r f(Y_r) \tag{2.15}$$

$$\tilde{h} = H(Y^\circ) \tag{2.16}$$

$$\tilde{f} = F(Y^\circ) \tag{2.17}$$

$$\tilde{R} = \mathrm{len}(Y^\circ) \tag{2.18}$$

Here $\tilde{h}$ is a scalar, $\tilde{f}$ is a K-dimensional vector, and the *occupancy* $\tilde{R}$ is a scalar. The functions H, F and len are corpus-level statistics obtained by summing the utterance-level statistics h, f and $y \mapsto 1$. For later reference it will be helpful to define

$$\hat{\eta}(\tilde{f}, \tilde{R}) = \arg\max_\eta \left\{ \sum_k \eta_k \tilde{f}_k - \tilde{R} \log Z(\eta) \right\} \tag{2.19}$$

$$\hat{L}(\tilde{h}, \tilde{f}, \tilde{R}) = \max_\eta \left\{ \tilde{h} + \sum_k \eta_k \tilde{f}_k - \tilde{R} \log Z(\eta) \right\} \tag{2.20}$$

where $\tilde{f}$ is a K-dimensional vector. The functions F and len are *sufficient statistics* for η, meaning that any inference that can be made about the value of η depends only on $\tilde{f} = F(Y^\circ)$ and $\tilde{R} = \mathrm{len}(Y^\circ)$ and not on the precise details of Y° (Bishop, 2006). In particular the maximum likelihood estimate of η for a given Y° is given by $\hat{\eta}(\tilde{f}, \tilde{R})$, and this depends only on the values of the sufficient statistics. We will see in more detail how this sufficiency property is useful for parameter estimation in §2.4 and §2.5.2.

Exponential families have a fairly strong statistics-matching property: maximum likelihood estimation typically matches the expected value of the statistic F to its value on the training corpus. It can be verified that

$$\frac{\partial}{\partial \eta_k} \log Z(\eta) = \int \mathbb{P}(y \,|\, \eta) f_k(y) \,\mathrm{d}y \tag{2.21}$$

$$= \mathbb{E} f_k(y) \tag{2.22}$$

where the distribution over y used for the expectation is $\mathbb{P}(y \mid \eta)$. Thus the derivative of the log likelihood is

$$\frac{\partial}{\partial \eta_k} \log \mathbb{P}(Y^\circ \mid \eta) = \sum_r f_k(Y_r) - \tilde{R}\, \mathbb{E} f_k(y) \tag{2.23}$$

$$= F_k(Y^\circ) - \mathbb{E} F_k(Y) \tag{2.24}$$

where the expectation in (2.24) is taken assuming Y is a sequence with index set R° where each Y_r is sampled independently from $\mathbb{P}(Y_r \mid \eta)$. If the maximum likelihood estimate is in the interior of Ξ then (2.23) must be equal to zero here, and so the expected value of the utterance-level statistic f_k under the trained model is equal to its average value on the training corpus, or equivalently the expected value of the corpus-level statistic F_k under the trained model is equal to its value on the training corpus. This statistics-matching property is often insightful when thinking about which aspects of a real world situation are captured by a given exponential family. It also means that if the trained probabilistic model is found to be deficient in some way, and that deficiency can be distilled into a specific statistic which has an inappropriate implied distribution, then this deficiency can be fixed in a conceptually simple way: by adding the statistic, or more generally powers of the statistic, to the list of feature functions defining the exponential family, we can ensure that the expected value, or more generally the moments, of the implied distribution of that statistic matches its value on the training corpus. This gives the exponential family framework flexibility from the point of view of designing probabilistic models.

Exponential families also have a useful convexity property. The function $\eta \mapsto \log Z(\eta)$ is convex, so the log likelihood function is concave. In particular this means that any local maximum in the likelihood function is a global maximum.

2.2.2 The Gaussian distribution

A multivariate Gaussian distribution is a probability distribution over $\mathbb{R}^D$ for some *dimensionality* D (Bishop, 2006). It is most commonly parameterized in terms of a *mean vector* $\mu \in \mathbb{R}^D$ and a symmetric positive definite *covariance matrix* $\Sigma \in \mathbb{R}^{D \times D}$, and has probability density function

$$\mathbb{P}(y \mid \mu, \Sigma) = \mathcal{N}(y; \mu, \Sigma) \tag{2.25}$$

where

$$\log \mathcal{N}(y; \mu, \Sigma) = -\tfrac{D}{2} \log 2\pi - \tfrac{1}{2} \log \det \Sigma - \tfrac{1}{2} (y - \mu)^\mathsf{T} \Sigma^{-1} (y - \mu) \tag{2.26}$$

for $y \in \mathbb{R}^D$. If y is a random variable distributed normally with mean μ and covariance Σ we write

$$y \mid \mu, \Sigma \sim \mathcal{N}(\mu, \Sigma) \tag{2.27}$$

We have $\mathbb{E}y = \mu$ and $\mathbb{E}(y-\mu)(y-\mu)^\mathsf{T} = \Sigma$.

The collection of Gaussian distributions of a particular dimensionality forms an exponential family. We can see this by rewriting the Gaussian probability density function in (2.25) in terms of its natural parameters, which are the inverse covariance matrix or *precision matrix* $P = \Sigma^{-1}$ and a precision-times-mean or *b-value*[1] $b = P\mu$. In this parameterization the probability density function is

$$\log \mathbb{P}(y \,|\, b, P) = \log \mathcal{N}_{\mathrm{NP}}(y; b, P) \tag{2.28}$$

where

$$\log \mathcal{N}_{\mathrm{NP}}(y; b, P) = -\tfrac{1}{2} y^\mathsf{T} P y + b^\mathsf{T} y - \log Z(b, P) \tag{2.29}$$

$$\log Z(b, P) = \tfrac{D}{2} \log 2\pi - \tfrac{1}{2} \log\det P + \tfrac{1}{2} b^\mathsf{T} P^{-1} b \tag{2.30}$$

Here P is a symmetric positive definite matrix. If y is a random variable distributed normally with b-value b and precision matrix P we write

$$y \,|\, b, P \sim \mathcal{N}_{\mathrm{NP}}(b, P) \tag{2.31}$$

We briefly show explicitly how (2.28) defines an exponential family. In the notation of (2.9), h for the Gaussian exponential family is just the constant function $y \mapsto 1$, the finite collection of feature functions $[f_k]_{k=1}^K$ consists of $y \mapsto y_d$ for $d = 1, \dots, D$ together with $y \mapsto -\frac{1}{2} y_d y_e$ for $d, e = 1, \dots, D$, and the natural parameters η above consist of the b-value b and the precision matrix P. Here the parameter corresponding to $y \mapsto y_d$ is the entry b_d of the b-value b, and the parameter corresponding to $y \mapsto -\frac{1}{2} y_d y_e$ is the entry P_{de} of the precision matrix P. In this general view of the Gaussian family as an exponential family, P need not be symmetric, but only the symmetric part of P affects the resulting distribution so without loss of generality we may assume it is symmetric. The constraint that P is positive definite corresponds to $Z(\eta) < \infty$ above.

For the Gaussian exponential family, the sufficient statistics are $Y^\circ \mapsto \sum_r Y_r$ and $Y^\circ \mapsto -\frac{1}{2}\sum_r Y_r Y_r^\mathsf{T}$. The maximum likelihood estimate of the parameters is given by

$$\hat{\mu} = \frac{1}{|R^\circ|} \sum_r Y_r \tag{2.32}$$

$$\hat{\Sigma} = \frac{1}{|R^\circ|} \sum_r Y_r Y_r^\mathsf{T} - \hat{\mu}\hat{\mu}^\mathsf{T} \tag{2.33}$$

as long as $|R^\circ| > 0$ and the right side of (2.33) is invertible. If the maximum likelihood natural parameters $\hat{b}$ and $\hat{P}$ are desired, these may be computed from $\hat{\mu}$ and $\hat{\Sigma}$. If the right side of (2.33) is not invertible then the likelihood function has a *singularity* and

[1] There appears to be no standard short term for this quantity, so we chose b-value.

can be made arbitrarily large. Parameters (b, P) with an extremely large likelihood may seem desirable, but such parameters typically do not generalize well. This is a well-known pathology in maximum likelihood estimation (MacKay, 2003). One way to ensure there are no singularities in the likelihood function when $D = 1$ is to restrict P_{11} to have some maximum allowed value. This is referred to as *variance flooring*. Restricting the set of allowed parameters of a one-dimensional Gaussian using variance flooring still yields an exponential family, since the set of allowed natural parameters is convex.

2.2.3 Conditional exponential families

There is an extension of the concept of an exponential family to the case of conditional probabilistic models. The theory in this case is not quite as neat as in the non-conditional case, but we will nonetheless find it useful.

A *conditional exponential family* (Feigin, 1981) is a conditional probabilistic model $\mathbb{P}(y \mid x, \eta)$ specified by sets $\mathcal{X}$ and $\mathcal{Y}$, a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^K$ for some non-negative integer K, a function $h : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and a convex set $\Xi \subset \mathbb{R}^K$. For any $\eta \in \Xi$ the random variable y takes values in $\mathcal{Y}$ and x takes values in $\mathcal{X}$. The log pdf is

$$\log \mathbb{P}(y \mid x, \eta) = h(x, y) + \sum_k \eta_k f_k(x, y) - \log Z(x, \eta) \tag{2.34}$$

$$\text{where } Z(x, \eta) = \int \exp\left(h(x, y) + \sum_k \eta_k f_k(x, y)\right) \mathrm{d}y \tag{2.35}$$

To define a valid conditional exponential family, $Z(x, \eta) < \infty$ must hold for all η in Ξ and x in $\mathcal{X}$. We may again refer to the parameterization of the probabilistic model in terms of η as the *natural parameterization*. The function $\eta \mapsto \log Z(x, \eta)$ is convex for any $x \in \mathcal{X}$, and so the log likelihood function $\eta \mapsto \log \mathbb{P}(Y^\circ \mid X^\circ, \eta)$ is concave. Here $X^\circ = [X_r]_{r \in R^\circ}$. It should be noted that our terminology differs from that used by Feigin (1981), who instead used the term conditional exponential family to refer to a family of Markov processes with transition probabilities of the form (2.34).

In general conditional exponential families do not have the nice sufficiency properties of exponential families. The log likelihood after observing a training corpus (X°, Y°) is

$$\log \mathbb{P}(Y^\circ \mid X^\circ, \eta) = \sum_r h(X_r, Y_r) + \sum_k \eta_k \sum_r f_k(X_r, Y_r) - \sum_r \log Z(X_r, \eta) \tag{2.36}$$

$$= H(X^\circ, Y^\circ) + \sum_k \eta_k F_k(X^\circ, Y^\circ) - \sum_r \log Z(X_r, \eta) \tag{2.37}$$

$$= \tilde{h} + \sum_k \eta_k \tilde{f}_k - \sum_r \log Z(X_r, \eta) \tag{2.38}$$

where

$$H(X^\circ, Y^\circ) = \sum_r h(X_r, Y_r) \tag{2.39}$$

$$F(X^\circ, Y^\circ) = \sum_r f(X_r, Y_r) \tag{2.40}$$

$$\tilde{h} = H(X^\circ, Y^\circ) \tag{2.41}$$

$$\tilde{f} = F(X^\circ, Y^\circ) \tag{2.42}$$

Here $\tilde{h}$ is a scalar and $\tilde{f}$ is a K-dimensional vector. Thus X° and $\tilde{f}$ are sufficient. However the extra term $\sum_r \log Z(X_r, \eta)$ in the log likelihood function, which may involve a complicated interaction between X_r and η, means that $\tilde{f}$ is not sufficient on its own.

Conditional exponential families have a fairly strong statistics-matching property similar to that of exponential families: maximum likelihood estimation typically matches the expected value of the statistic F to its value on the training corpus. It can be verified that

$$\frac{\partial}{\partial \eta_k} \log Z(x, \eta) = \int \mathbb{P}(y \,|\, x, \eta) f_k(x, y) \,\mathrm{d}y \tag{2.43}$$

$$= \mathbb{E} f_k(x, y) \tag{2.44}$$

where the distribution over y used for the expectation is $\mathbb{P}(y \,|\, x, \eta)$. Thus the derivative of the log likelihood is

$$\frac{\partial}{\partial \eta_k} \log \mathbb{P}(Y^\circ \,|\, X^\circ, \eta) = \sum_r f_k(X_r, Y_r) - \mathbb{E} \sum_r f_k(X_r, Y_r) \tag{2.45}$$

$$= F_k(X^\circ, Y^\circ) - \mathbb{E} F_k(X^\circ, Y) \tag{2.46}$$

where the distribution over Y_r used for the first expectation is $\mathbb{P}(Y_r \,|\, X_r, \eta)$, and the second expectation is taken assuming Y is a sequence with index set R° where each Y_r is sampled independently from $\mathbb{P}(Y_r \,|\, X_r, \eta)$. If the maximum likelihood estimate is in the interior of Ξ then (2.45) must be equal to zero here, and so the expected value of the corpus-level statistic F given inputs X° is equal to its value on the training corpus. As for exponential families, this statistics-matching property is often insightful when thinking about which aspects of a real world situation are captured by a given conditional exponential family, and also means that if a deficiency in the model can be distilled to a specific statistic with the wrong implied distribution then this deficiency can be fixed in a conceptually simple way.

There is a special type of conditional exponential family for which simple finite-dimensional sufficient statistics do exist. Following Feigin (1981) we refer to a conditional exponential family as a *conditionally additive exponential family* if the log normalization constant is of the form

$$\log Z(x, \eta) = \sum_{j=1}^{J} g_j(x) \xi_j(\eta) \tag{2.47}$$

where J is a non-negative integer, $g : \mathcal{X} \to \mathbb{R}^J$ and $\xi : \Xi \to \mathbb{R}^J$. This is a slight generalization of the definition used by Feigin (1981), who focused on the case $J = 1$. In this case the log normalization term behaves nicely under summation over r, and if we define

$$G(X^\circ) = \sum_r g(X_r) \tag{2.48}$$

then F and G are sufficient statistics for η.

It is natural to ask whether the conditional probabilistic models with finite-dimensional sufficient statistics can be characterized in the way the Pitman-Koopman theorem does for non-conditional probabilistic models. We are not aware of any work addressing this. Another natural question is how rare the existence of finite-dimensional sufficient statistics is for conditional models, and relatedly how rare the conditional additivity property is. We will see one example of a conditionally additive exponential family below, but we do not know of any work addressing this question in general.

2.2.4 Linear regression models

In this section we describe a simple tractable form of conditional probabilistic regression model, following Bishop (2006).

Consider a conditional probabilistic model $\mathbb{P}(y \,|\, x, \lambda)$ where the observed value y is real-valued, the value x is an element of an arbitrary set $\mathcal{X}$, and where the observed value y is assumed to be the value of some underlying function $\mathcal{X} \to \mathbb{R}$ at x corrupted by Gaussian noise. If the underlying function is assumed to be a linear combination of a fixed collection $[\varphi_d]_{d=1}^D$ of *basis functions*, where $\varphi_d : \mathcal{X} \to \mathbb{R}$, then we have

$$y \,|\, x, \lambda \sim \mathcal{N}\left(\sum_{d=1}^D a_d \varphi_d(x), \sigma^2\right) \tag{2.49}$$

where a_d is the *regressive coefficient* of the basis function φ_d, σ^2 is referred to as the *conditional variance*, and $\lambda = ([a_d]_{d=1}^D, \sigma^2)$. The collection of basis functions may equivalently be considered as a single function $\varphi : \mathcal{X} \to \mathbb{R}^D$ with $\varphi(x) = [\varphi_d(x)]_{d=1}^D$. This is a *linear regression model* (Bishop, 2006). Note that here "linear" means "linear in the regressive coefficients".

Given a training corpus (X°, Y°), the log likelihood is given by

$$\sum_r \log \mathbb{P}(Y_r \mid X_r, \lambda) = -\frac{1}{2}|R^\circ| \log 2\pi - \frac{1}{2}|R^\circ| \log(\sigma^2) - \frac{1}{2}\frac{1}{\sigma^2} \sum_r \left(Y_r - \sum_d a_d \varphi_d(X_r) \right)^2 \tag{2.50}$$

$$= -\frac{1}{2}\tilde{R} \log 2\pi - \frac{1}{2}\tilde{R} \log(\sigma^2) - \frac{1}{2}\frac{1}{\sigma^2} \left(\tilde{u} - 2\sum_d \tilde{s}_d a_d + \sum_{d,e} \tilde{S}_{de} a_d a_e \right) \tag{2.51}$$

where

$$\tilde{u} = \sum_r Y_r^2 \tag{2.52}$$

$$\tilde{s} = \sum_r \varphi(X_r) Y_r \tag{2.53}$$

$$\tilde{S} = \sum_r \varphi(X_r) \varphi^\mathsf{T}(X_r) \tag{2.54}$$

$$\tilde{R} = |R^\circ| = \sum_r 1 \tag{2.55}$$

Here $\tilde{s}$ is a D-dimensional vector and $\tilde{S}$ is a $D \times D$ matrix. Thus $(\tilde{u}, \tilde{s}, \tilde{S}, \tilde{R})$, or rather the functions which compute these values from (X°, Y°), are sufficient statistics. The maximum likelihood estimate of the parameters is given by

$$\hat{a} = \tilde{S}^{-1} \tilde{s} \tag{2.56}$$

$$\hat{\sigma}^2 = \frac{1}{\tilde{R}} \left(\tilde{u} - \tilde{s}^\mathsf{T} \tilde{S}^{-1} \tilde{s} \right) \tag{2.57}$$

as long as $\tilde{R} > 0$, $\tilde{S}$ is invertible and the right side of (2.57) is greater than zero. If $\tilde{S}$ is not invertible then there are multiple optimal values of a. In this case we might choose to select the value of a with smallest Euclidean norm, which corresponds to using the pseudo-inverse instead of the inverse in (2.56). If the right side of (2.57) is equal to zero then the likelihood function has a singularity and can be made arbitrarily large. As in the case of the Gaussian family, we can ensure such singularities do not occur by using variance flooring, i.e. restricting σ^2 to have some minimum allowed value.

The linear regression model is a conditionally additive exponential family. The natural parameters are $(\tau, -\tau a)$, where $\tau = 1/\sigma^2$ and $a = [a_d]_{d=1}^D$, with corresponding feature functions $(x, y) \mapsto -\frac{1}{2}y^2$ and $(x, y) \mapsto -\varphi(x)y$. The only term in the log normalization constant involving both x and the parameters is $\frac{1}{2}\tau(a^\mathsf{T}\varphi(x))^2$, which may be rewritten as $\frac{1}{2}\mathrm{tr}(\tau a a^\mathsf{T} \varphi(x)\varphi^\mathsf{T}(x))$ and is therefore of the form (2.47). Viewing the linear regression model

as a conditionally additive exponential family provides another way to see that it has finite-dimensional sufficient statistics, and shows that the log likelihood is concave in the above natural parameterization.

There is another parameterization of the linear regression model that we sometimes find useful. Define a vector $\tilde{a} = [\tilde{a}_d]_{d=0}^D$ by

$$\tilde{a}_0 = \frac{1}{\sigma} \tag{2.58}$$

$$\tilde{a}_d = -\frac{a_d}{\sigma}, \quad d \in \{1, \dots, D\} \tag{2.59}$$

We refer to this as the $\tilde{a}$ *parameterization*. It differs from the natural parameterization by a factor of σ. In this parameterization the log pdf is

$$\log \mathbb{P}(y \,|\, x, \tilde{a}_{0:D}) = -\tfrac{1}{2} \log 2\pi + \tfrac{1}{2} \log(\tilde{a}_0^2) - \tfrac{1}{2} v^\mathsf{T} \left(\tilde{a}_{0:D} \tilde{a}_{0:D}^\mathsf{T} \right) v \tag{2.60}$$

where $v \in \mathbb{R}^{D+1}$ is the vector $\varphi(x) \in \mathbb{R}^D$ with the value $y \in \mathbb{R}$ prepended. Perhaps surprisingly the log likelihood is also concave in this parameterization. As defined above the $\tilde{a}$ parameterization has the constraint that $\tilde{a}_0 > 0$. However we may allow $\tilde{a}_0 < 0$ by specifying that we still use (2.60) in this case. This introduces a redundancy of a factor of two into our parameterization, since replacing $\tilde{a}_{0:D}$ by $-\tilde{a}_{0:D}$ does not change the distribution in (2.60). We are not free to allow $\tilde{a}_0 = 0$, since this corresponds to $\tau = 0$ and $\sigma = \infty$ and does not define a valid probability distribution, but we can view this as a limiting case that we can get arbitrarily close to but not reach. Thus we can view essentially any value $\tilde{a}_{0:D} \in \mathbb{R}^{D+1}$ as specifying a linear regression model, and any linear regression model as being specified by some $\tilde{a}_{0:D} \in \mathbb{R}^{D+1}$.

The tractability of the linear regression model presented above depends on the underlying function being linear in the regressive coefficients, but allows it to be non-linear in the input. Nevertheless it is common, in the case where $\mathcal{X}$ is a vector space, to use *affine* (linear plus a bias) basis functions so that the underlying function is affine in the input. A particularly simple instance of this when $\mathcal{X} = \mathbb{R}^K$ is to use the *canonical* basis functions $[\varphi_k]_{k=1}^{K+1}$, where $\varphi_k(x) = x_k$ for $k = 1, \dots, K$ and $\varphi_{K+1}(x) = 1$. This corresponds to allowing the underlying function to be a general affine function of the input. We refer to a linear regression model $\mathbb{P}(y \,|\, x, \lambda)$ with canonical basis functions as a *linear Gaussian linear regression model*. Note that the "linear" in "linear Gaussian" refers to being linear in the input. Such a model defines a family of conditional distributions over y given x, and we refer to a member of this family as a *linear Gaussian distribution* (Roweis and Ghahramani, 1999; Bishop, 2006). To say that a conditional distribution $\mathbb{P}(y \,|\, x)$ where $x \in \mathbb{R}^K$ is linear Gaussian thus means that $\mathbb{P}(y \,|\, x) = \mathcal{N}(y; \sum_{k=1}^K a_k x_k + a_{K+1}, \sigma^2)$ for some $[a_k]_{k=1}^K \in \mathbb{R}^K$, $a_{K+1} \in \mathbb{R}$ and $\sigma^2 \in \mathbb{R}$ with $\sigma^2 > 0$.

The above linear regression model may trivially be extended to a model $\mathbb{P}(y \,|\, x, \lambda)$ where y is a vector by using a separate linear regression model $\mathbb{P}(y_i \,|\, x, \lambda_i)$ for each component i,

that is

$$y_i \,|\, x, \lambda \sim \mathcal{N}\left(\sum_{d=1}^{D} a_{id}\varphi_{id}(x), \sigma_i^2\right) \tag{2.61}$$

where $\lambda = [\lambda_i]_i$. The distribution of $y - \mathbb{E}y$ given x is Gaussian with a diagonal covariance matrix. We also refer to this extended model as a linear regression model, and again it has sufficient statistics. Alternatively we can assume that the distribution of $y - \mathbb{E}y$ given x is Gaussian with a full covariance matrix, or equivalently allow $y_{1:i-1}$ to form part of the input for the linear regression model for y_i. We refer to this as the *full-covariance* linear regression model.

2.2.5 Composite linear Gaussian distributions

In this section we define *composite linear Gaussian distributions* and show that every Gaussian distribution can be written in this form. The concepts and properties established in this section will be used in various sections throughout the thesis.

We first define a variant of the *Cholesky decomposition*. It is well known that for a given positive definite matrix P, there exists a unique lower triangular matrix M with positive entries along its diagonal such that $P = MM^\mathsf{T}$. The *Cholesky factor* M can be computed in $O(T^3)$ time where T is the size of P. Similarly there exists a unique lower triangular matrix L with positive entries along its diagonal such that $P = L^\mathsf{T}L$, and we refer to this as the *alternative Cholesky decomposition*. The *alternative Cholesky factor* L can be computed in $O(T^3)$ time using a variant of the algorithm used to compute the conventional Cholesky factor, or alternatively by using the fact that $L = R\tilde{M}^\mathsf{T}R$ where $\tilde{M}$ is the conventional Cholesky factor of RPR and R is a matrix with ones along its anti-diagonal, so that Ry is the vector y with the components reversed.

A *canonically ordered composite linear Gaussian distribution* is a distribution $\mathbb{P}(y)$ over a vector $y \in \mathbb{R}^T$ of the form $\mathbb{P}(y) = \prod_{t=1}^{T} \mathbb{P}(y_t \,|\, y_{1:t-1})$ where each factor $\mathbb{P}(y_t \,|\, y_{1:t-1})$ is linear Gaussian. This implies the overall distribution $\mathbb{P}(y)$ is Gaussian. Since the conditional distribution $\mathbb{P}(y_t \,|\, y_{1:t-1})$ is linear Gaussian, it can be written as

$$y_t = \sum_{k=1}^{t-1} a_{tk} y_{t-k} + a_{tt} + \sigma_t z_t \tag{2.62}$$

where $[a_{tk}]_{k=1}^{t-1}$, a_{tt} and σ_t^2 are the parameters of a linear regression model and we have introduced a vector $z = [z_t]_{t=1}^{T}$ where $z \sim \mathcal{N}(0, I)$. As we saw in §2.2.4, there are a number of ways to parameterize a linear regression model. By using the $\tilde{a}$ parameterization we can

rewrite (2.62) as a matrix equation

$$Ly = \xi + z \tag{2.63}$$

$$\text{where } L_{tt} = \frac{1}{\sigma_t} = \tilde{a}_{t0} \tag{2.64}$$

$$L_{t(t-k)} = -\frac{a_{tk}}{\sigma_t} = \tilde{a}_{tk} \text{ for } k = 1, \ldots, t-1 \tag{2.65}$$

$$\xi_t = \frac{a_{tt}}{\sigma_t} = -\tilde{a}_{tt} \tag{2.66}$$

Here the t^{th} element of ξ and the t^{th} row of L give the linear regression model parameters, in the $\tilde{a}$ parameterization, for the distribution $\mathbb{P}(y_t \,|\, y_{1:t-1})$. We have $\mu = \mathbb{E}y = L^{-1}\xi$, so $y - \mu = L^{-1}z$, so $\mathbb{E}(y-\mu)(y-\mu)^{\mathsf{T}} = L^{-1}IL^{-\mathsf{T}} = (L^{\mathsf{T}}L)^{-1}$. Thus $\mathbb{P}(y)$ is a Gaussian distribution with precision matrix $P = L^{\mathsf{T}}L$ and b-value $b = P\,\mathbb{E}y = L^{\mathsf{T}}\xi$, where L is lower triangular. It is easy to verify that any Gaussian distribution can be written as a canonically ordered composite linear Gaussian distribution. Indeed given natural parameters b and P, the corresponding L is given by the alternative Cholesky factor of P and the corresponding ξ is obtained by solving $b = L^{\mathsf{T}}\xi$ using a triangular matrix solve.

A distribution $\mathbb{P}(y)$ is a *composite linear Gaussian distribution* if it can be written as a canonically ordered composite linear Gaussian distribution by permuting the components of y. This definition is equivalent to saying that $\mathbb{P}(y)$ is given by a *directed graphical model* (Bishop, 2006, chapter 8) where each y_t is a node which has linear Gaussian distribution given its parents, and this is closely related to the definition of a linear Gaussian model given by Roweis and Ghahramani (1999). If the identity permutation is used then we have $Ly = \xi + z$ where L is lower triangular with $P = L^{\mathsf{T}}L$, and the corresponding decomposition of $\mathbb{P}(y)$ is of the form $\prod_t \mathbb{P}(y_t \,|\, y_{1:t-1})$ as above. If the reversing permutation $t \mapsto T+1-t$ is used then it can be verified that $Uy = \xi + z$ where U is upper triangular with $P = U^{\mathsf{T}}U$, and the corresponding decomposition of $\mathbb{P}(y)$ is of the form $\prod_t \mathbb{P}(y_t \,|\, y_{t+1:T})$. This clarifies the relationship between the conventional and alternative Cholesky decompositions of a positive definite matrix P: the conventional Cholesky decomposition computes the parameters of the composite linear Gaussian distribution equivalent to $\mathcal{N}_{\text{NP}}(0, P)$ where the regressions are directed "backwards in time", whereas for the alternative Cholesky decomposition the regressions are directed "forwards in time".

We now consider a specialization of the above definitions. Suppose $\mathbb{P}(y)$ is a canonically ordered composite linear Gaussian distribution. If $\mathbb{P}(y_t \,|\, y_{1:t-1}) = \mathbb{P}(y_t \,|\, y_{t-K:t-1})$ for $t > K$ where K is a non-negative integer, then we refer to $\mathbb{P}(y)$ as a *canonically ordered composite linear Gaussian autoregressive distribution* and refer to K as the *depth*. For uniformity it is often convenient to specify a fixed *initial context* $y_{-(K-1):0}$ so that $\mathbb{P}(y_t \,|\, y_{1:t-1}) = \mathbb{P}(y_t \,|$

$y_{t-K:t-1}$) for all t. In this case (2.62) becomes

$$y_t = \sum_{k=1}^{K} a_{tk} y_{t-k} + a_{t(K+1)} + \sigma_t z_t \tag{2.67}$$

If the initial context is zero then we have $Ly = \xi + z$ as before, where L is now a banded lower triangular matrix with subdiagonal width at most K, and the precision matrix $P = L^\mathsf{T} L$ is banded with subdiagonal and superdiagonal widths at most K. Since the alternative Cholesky factor of a banded matrix is banded, any Gaussian distribution with banded precision matrix can be written as a canonically ordered composite linear Gaussian autoregressive distribution. A distribution $\mathbb{P}(y)$ is a *composite linear Gaussian autoregressive distribution* if it can be written as a canonically ordered composite linear Gaussian autoregressive distribution by permuting the components of y. By default we consider the identity permutation where the autoregressions are directed forwards in time, but occasionally we consider the reversing permutation where the autoregressions are directed backwards in time.

2.3 Decision tree clustering

Suppose we wish to model the conditional distribution of an observation y given a *state* ψ, where ψ belongs to a finite set referred to as the *state space*. One way to specify such a model is to use a *basic* probabilistic model $\tilde{\mathbb{P}}(y \,|\, \tilde{\eta})$ as a building block, setting

$$\mathbb{P}\left(y \,|\, \psi, [\eta_\psi]_\psi\right) = \tilde{\mathbb{P}}(y \,|\, \tilde{\eta} = \eta_\psi) \tag{2.68}$$

In other words, for each state in the state space the distribution over y is modelled using a separate "copy" of the basic model. However if the state space is very large then there is often not enough data to robustly estimate separate parameters for each state using maximum likelihood; indeed for many states there may be no data at all. One solution to this problem is to partition state space into *clusters* and constrain the states in each cluster to have identical parameters, that is

$$\mathbb{P}\left(y \,|\, \psi, [\eta_C]_{C \in \mathcal{C}}, \mathcal{C}\right) = \tilde{\mathbb{P}}(y \,|\, \tilde{\eta} = \eta_C) \tag{2.69}$$

where $\mathcal{C}$ is a partition of state space, and C on the right side is the cluster which contains ψ. If we choose the partition $\mathcal{C}$ judiciously then it may be coarse enough to allow robust estimation of the parameters for each cluster but fine enough to achieve good modelling accuracy. In this section we describe a particular approach to choosing a partition based on data known as *decision tree clustering* (Young et al., 1994). We focus on the case where the basic model is an exponential family. We first describe maximum likelihood estimation

of the parameters $[\eta_C]_{C\in\mathcal{C}}$ given a partition $\mathcal{C}$, then define the concept of a decision tree, and finally describe a typical form of decision tree clustering.

Suppose the basic model above is an exponential family specified by h and $[f_k]_{k=1}^K$, and a partition $\mathcal{C}$ of state space has been chosen. Then we have

$$\log \mathbb{P}\left(y \mid \psi, [\eta_C]_{C\in\mathcal{C}}, \mathcal{C}\right) = h(y) + \sum_{k=1}^{K} \eta_{Ck} f_k(y) - \log Z(\eta_C) \tag{2.70}$$

where C is the cluster containing ψ and $\eta_C = [\eta_{Ck}]_{k=1}^K$ is the collection of model parameters for cluster C. From (2.13) the overall log likelihood given data (Ψ°, Y°), where $\Psi^\circ = [\Psi_r]_{r\in R^\circ}$, is

$$\log \mathbb{P}\left(Y^\circ \mid \Psi^\circ, [\eta_C]_{C\in\mathcal{C}}, \mathcal{C}\right) = \sum_{C\in\mathcal{C}} \tilde{h}_C + \sum_{C\in\mathcal{C}} \sum_{k=1}^{K} \eta_{Ck} \tilde{f}_{Ck} - \sum_{C\in\mathcal{C}} \tilde{R}_C \log Z(\eta_C) \tag{2.71}$$

where

$$\tilde{h}_C = \sum_{\psi\in C} \tilde{h}_\psi \tag{2.72}$$

$$\tilde{f}_{Ck} = \sum_{\psi\in C} \tilde{f}_{\psi k} \tag{2.73}$$

$$\tilde{R}_C = \sum_{\psi\in C} \tilde{R}_\psi \tag{2.74}$$

and

$$\tilde{h}_\psi = \sum_r \delta(\Psi_r, \psi) h(Y_r) \tag{2.75}$$

$$\tilde{f}_{\psi k} = \sum_r \delta(\Psi_r, \psi) f_k(Y_r) \tag{2.76}$$

$$\tilde{R}_\psi = \sum_r \delta(\Psi_r, \psi) \tag{2.77}$$

where δ is the discrete delta function defined by $\delta(a, b) = 1$ if $a = b$ and $\delta(a, b) = 0$ otherwise. The sufficient statistic values, e.g. $\tilde{h}_C$, for a cluster C are obtained by simply summing the sufficient statistic values, e.g. $\tilde{h}_\psi$, for each state $\psi \in C$ in that cluster. The maximum likelihood estimate of the parameters $[\eta_C]_{C\in\mathcal{C}}$ is $[\hat{\eta}(\tilde{f}_C, \tilde{R}_C)]_{C\in\mathcal{C}}$, where $\hat{\eta}$ is the function for the basic model given by (2.19). The overall log likelihood value at this maximum is

$$\sum_{C\in\mathcal{C}} \hat{L}\left(\tilde{h}_C, \tilde{f}_C, \tilde{R}_C\right) \tag{2.78}$$

where $\hat{L}$ is the function for the basic model given by (2.20).

We now define the concept of a decision tree. For a given state space, a *question* is a function from the state space to $\{\text{yes}, \text{no}\}$. A *decision tree* is a binary tree where each

internal node is labelled by a question. Each state ψ has an associated leaf obtained by starting at the root node and recursively descending the tree, at each internal node with question ζ passing to the left child if $\zeta(\psi)$ is no and to the right child if $\zeta(\psi)$ is yes. The set of states which end up in the same leaf under this procedure specifies a cluster, and so the decision tree as a whole specifies a partition of state space.

The tree is typically learned from data by greedily maximizing a given *objective function* over trees with questions taken from a given *question set*. This is a tractable approximation to maximizing the objective function over all such trees. The objective function typically used is the maximum log likelihood given the partition minus a penalty term linear in the number of leaves, that is

$$\sum_{C \in \mathcal{C}} \hat{L}\left(\tilde{h}_C, \tilde{f}_C, \tilde{R}_C\right) - \xi \sum_{C \in \mathcal{C}} 1 \tag{2.79}$$

The constant penalty for a new leaf is referred to as the *clustering threshold* ξ. To greedily optimize this objective function we recursively split each leaf using the question that maximizes the change in log likelihood, unless the maximum achievable change is less than ξ in which case we do not split that leaf. The change in log likelihood for splitting a cluster C into two clusters D and E is

$$\hat{L}\left(\tilde{h}_D, \tilde{f}_D, \tilde{R}_D\right) + \hat{L}\left(\tilde{h}_E, \tilde{f}_E, \tilde{R}_E\right) - \hat{L}\left(\tilde{h}_C, \tilde{f}_C, \tilde{R}_C\right) \tag{2.80}$$

Note that the change in log likelihood for a potential split only depends on the clusters involved in the potential split and not on other details of the partition $\mathcal{C}$. This means the order in which we choose to split leaves makes no difference. Typically a hard minimum occupancy constraint $\tilde{R}_C \geq \tilde{R}_{\min}$ is also imposed on each leaf C, which can be incorporated above by setting the objective function to $-\infty$ for any tree that violates this constraint.

The *minimum description length (MDL)* criterion (Shinoda and Watanabe, 2000) is an automated way to choose the clustering threshold ξ. It sets

$$\xi = \tfrac{1}{2} \rho K \log \tilde{R}_{\text{root}} \tag{2.81}$$

where K is the number of parameters per leaf, $\tilde{R}_{\text{root}}$ is the total occupancy of the root node, and ρ is a heuristic scaling factor which arguably should theoretically be 1, and is often set to 1 in practice. We refer to ρ as the *MDL tuning factor*.

A partition is defined as a set of sets with certain properties, but sometimes it is helpful to instead think about it as a surjective function $\overline{q}$ from state space to $\{1, \ldots, Q\}$ for some $Q \in \mathbb{N}$. We refer to $\overline{q}$ as a *clustering function*. These two representations are equivalent: given a partition of state space we may arbitrarily order the clusters and specify $\overline{q}$ to be the function taking a state to the index of its corresponding cluster under this order; given a surjective function $\overline{q}$ from state space to $\{1, \ldots, Q\}$ the set $\mathcal{C} = \{\, \overline{q}^{-1}(\{q\}) : q \in \{1, \ldots, Q\}\}$ is a partition. We refer to the integer $q = \overline{q}(\psi)$ assigned to a given state ψ as its *cluster*

index or *leaf index*. The parameters of cluster q may thus be written $\eta_q = [\eta_{qk}]_k$ instead of $\eta_C = [\eta_{Ck}]_k$, and the sufficient statistic values as, for example $\tilde{h}_q$ instead of $\tilde{h}_C$. We will find the representation of a partition as a function $\overline{q}$ helpful in describing various models in the remainder of this thesis.

The purpose of the term involving ξ in (2.79) is to help prevent overfitting. Because the parameters for each leaf are estimated independently only using the data assigned to that leaf, adding more leaves means that each leaf is estimated using less, but more specific, data. Performance on the test corpus is therefore a trade-off: using more leaves gives a finer-grained, more powerful model, but its parameters are estimated less robustly. The term involving ξ acts as a simple penalty on adding leaves, restricting the size of the tree and encouraging robust estimation of the parameters for each leaf. It should be noted that the ξ term is only needed because maximum likelihood estimation is susceptible to overfitting. When estimation of the leaf model parameters is done in a Bayesian way, even using a simple independent global prior for all leaves, then we may effectively set $\xi = 0$ and obtain very large trees which nevertheless have good generalization performance (Hashimoto et al., 2009).

2.4 Expectation maximization

The *expectation maximization (EM)* framework (Dempster et al., 1977) is an approach to approximate maximum likelihood parameter estimation applicable to a wide range of latent variable models. Our statement of expectation maximization follows Bishop (2006).

Suppose we have a model $\mathbb{P}(Y, X \mid \lambda)$ over an observed variable Y and a latent variable X. For example we might have that $Y = [Y_r]_{r \in R^\circ}$ and $X = [X_r]_{r \in R^\circ}$ are collections of independent draws of the observed and hidden variables in a latent variable model as in §2.1.1. However we do not assume this form here, because we want our derivation to also apply to the sequential models considered below. We assume X is discrete-valued here, but the theory applies very similarly in the continuous case. It is often the case for such latent variable models that the *overall log likelihood* $\log \mathbb{P}(Y, X \mid \lambda)$ has a simple form as a function of the parameters λ and can be analytically maximized, while the true, *marginal log likelihood* $\log \mathbb{P}(Y \mid \lambda)$ we would like to maximize is a complicated function of λ. Expectation maximization is a way to use the ease of maximizing the overall log likelihood to iteratively and approximately maximize the marginal log likelihood.

We now give a terse derivation of expectation maximization. Full details are given in Bishop (2006). For any function Q which is the probability mass function of a distribution

over X we have

$$\log \mathbb{P}(Y \mid \lambda) = \log \sum_X \mathbb{P}(Y, X \mid \lambda) \tag{2.82}$$

$$= \sum_X Q(X) \log \mathbb{P}(Y, X \mid \lambda) + H(Q) + \mathrm{KL}\left(Q; X \mapsto \mathbb{P}(X \mid Y, \lambda)\right) \tag{2.83}$$

$$\geq \sum_X Q(X) \log \mathbb{P}(Y, X \mid \lambda) + H(Q) \tag{2.84}$$

$$= \mathcal{F}(Q, \lambda) \tag{2.85}$$

where

$$H(Q) = -\sum_X Q(X) \log Q(X) \tag{2.86}$$

$$\mathrm{KL}(Q; R) = \sum_X Q(X) \log \frac{Q(X)}{R(X)} \tag{2.87}$$

$$\mathcal{F}(Q, \lambda) = \sum_X Q(X) \log \mathbb{P}(Y, X \mid \lambda) + H(Q) \tag{2.88}$$

That is, H is the *entropy* and KL is the Kullback-Leibler divergence. Here $X \mapsto \mathbb{P}(X \mid Y, \lambda)$ is the probability mass function of the conditional distribution of X given Y and λ, and R is the probability mass function of a distribution over X. Note that the dependence of $\mathcal{F}$ on the training corpus Y is left implicit. The inequality (2.84) follows from the fact that the Kullback-Leibler divergence is non-negative. The basic properties of the Kullback-Leibler divergence also show that we have equality if and only if $Q(X) = \mathbb{P}(X \mid Y, \lambda)$.

Expectation maximization conceptually proceeds by iterating two steps. Firstly given a current estimate of the parameters λ we find the Q which maximizes $\mathcal{F}(Q, \lambda)$, denoted $\hat{Q}(\lambda)$. This is called the *E step*. Secondly given a current value of Q we find the λ which maximizes $\mathcal{F}(Q, \lambda)$. This is called the *M step*. In the expectation maximization framework conceptually we repeat these two steps, thus doing a form of coordinate ascent on $\mathcal{F}$. This view of EM as coordinate ascent on $\mathcal{F}$ is described by Neal and Hinton (1998).

As we saw above $\hat{Q}(\lambda)$ is just the posterior $\hat{Q}(\lambda)(X) = \mathbb{P}(X \mid Y, \lambda)$, and with this value we have equality, that is $\log \mathbb{P}(Y \mid \lambda) = \mathcal{F}(\hat{Q}(\lambda), \lambda)$. This shows that expectation maximization does not decrease the likelihood, since it does not decrease $\mathcal{F}$ and $\mathcal{F}$ is equal to the log likelihood after each E step (Neal and Hinton, 1998).

The function $\lambda \mapsto \mathcal{F}(Q, \lambda)$ is referred to as the EM *auxiliary function*. We include the $H(Q)$ term as part of the auxiliary function, though some authors omit it.

2.5 Sequential models

We refer to a probabilistic model $\mathbb{P}(y \mid \lambda)$ where y is a sequence, or a conditional probabilistic model $\mathbb{P}(y \mid x, \lambda)$ where y is a sequence, as a *sequential model*. In this section we review a

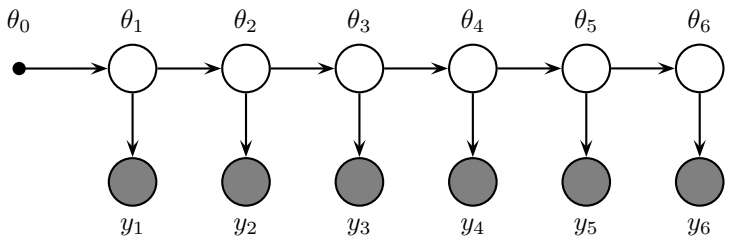


Figure 2.1: A directed graphical model for a hidden Markov model. Here $\theta = \theta_{1:6}$ is the state sequence and $y = y_{1:6}$ is the sequence of observations. The value θ_0 is a deterministic initial state. Nodes also depend on the model parameters ν and λ (not shown). We consider θ_t hidden and y_t observed.

very commonly used sequential model, the hidden Markov model, and describe parameter estimation for it.

Following standard convention when dealing with sequential models (Rabiner, 1989; Bishop, 2006), we drop the utterance index r in our notation and effectively assume the training corpus and test corpus each consist of just one utterance. The multiple-utterance case is usually easy to recover by replacing sums, products, etc over time t with the same operation over utterance index r and time t. It is often possible to make this connection fully rigorous by considering a collection of multiple sequences as one long sequence and making appropriate adjustments to the model.

2.5.1 Hidden Markov models

A *hidden Markov model (HMM)* (Rabiner, 1989; Bishop, 2006) is a sequential latent variable probabilistic model. HMMs are frequently used as models of speech due to their combination of reasonable modelling accuracy and tractability.

An HMM is a probabilistic model $\mathbb{P}(y, \theta \,|\, \nu, \lambda)$ where $\theta = [\theta_t]_{t=1}^T$ and $y = [y_t]_{t=1}^T$ are sequences. At each discrete time t there is a *state* θ_t and an *observation* y_t. Each state θ_t is a member of a finite set referred to as the *state space*. Each observation y_t is a member of some *observation space*, which may be discrete or continuous or something more general. Conventionally the sequence θ of states is *hidden* or *latent* with only the sequence y of observations observed. The generative model is that the state evolves over time according to a Markov process, and each observation y_t at time t "depends" only on the state θ_t at that time. This is most clearly expressed as the following *directed graphical model* (Bishop, 2006), also shown in Figure 2.1

$$\mathbb{P}(y, \theta \,|\, \nu, \lambda) = \prod_{t=1}^{T} \mathbb{P}(y_t \,|\, \theta_t, \lambda) \mathbb{P}(\theta_t \,|\, \theta_{t-1}, \nu) \tag{2.89}$$

where we assume there is some deterministic *initial state* θ_0. We have assumed that the parameters ν of the *state transition model* $\mathbb{P}(\theta_t \,|\, \theta_{t-1}, \nu)$ are distinct from the parameters λ of $\mathbb{P}(y_t \,|\, \theta_t, \lambda)$. When visualizing directed graphical models such as in Figure 2.1, each large circle represents a random variable and is shaded if the variable is observed or unshaded if it is latent. The arrows represent the conditional dependency structure between variables. A small black circle represents variables which are conditioned on, such as θ_0 here.

The form of the HMM model allows efficient inference. For a given $y_{1:T}$ we can compute the most likely state sequence $\arg\max_{\theta_{1:T}} \mathbb{P}(\theta_{1:T}, y_{1:T} \,|\, \nu, \lambda)$ efficiently using a recursive dynamic programming algorithm known as the *Viterbi algorithm* (Rabiner, 1989). Similarly we can compute $\mathbb{P}(y_{1:t}, \theta_t = \psi \,|\, \nu, \lambda)$ for all times t and all possible states ψ efficiently using the *forward algorithm* and $\mathbb{P}(y_{t+1:T} \,|\, \theta_t = \psi, \nu, \lambda)$ for all t and ψ efficiently using the *backward algorithm*. By combining the results of the forward and backward algorithms we can efficiently compute various useful quantities for a given $y_{1:T}$ such as the likelihood $\mathbb{P}(y_{1:T} \,|\, \nu, \lambda)$ and the *state occupancies* $\gamma_{t\psi} = \mathbb{P}(\theta_t = \psi \,|\, y_{1:T}, \nu, \lambda)$ for every time t and state ψ (Rabiner, 1989). The Viterbi and forward-backward algorithms are special cases of a general efficient exact inference algorithm for graphical models with tree-structured *factor graphs* and discrete latent variables (Bishop, 2006).

The simple form of the HMM also often allows efficient estimation of the model parameters ν and λ using expectation maximization. The details depend on the way the distributions $\mathbb{P}(\theta_t \,|\, \theta_{t-1}, \nu)$ and $\mathbb{P}(y_t \,|\, \theta_t, \lambda)$ are parameterized. We discuss one fairly general case of using expectation maximization to estimate λ in the next section.

The HMMs typically used for speech both specialize and generalize the above framework in a number of ways. Firstly the type of HMM model used for speech typically has a very sparse transition structure, where each state often has only a handful of possible next states. We refer to the average number of next states for a given state as the *branching factor*. If we assume a fixed branching factor, then exploiting the sparse transition structure changes the forward-backward and Viterbi algorithms from $O(TM^2)$, where T is the number of frames and M the size of the state space, to $O(TM)$. This is important for typical speech models, particularly for their use in speech recognition since the state space is typically very large, but also for their use in speech synthesis where the state space under consideration can usually be restricted on a per-utterance basis but where a long utterance still contains many different states. Secondly the HMMs typically used for modelling speech assume that certain states may be *non-emitting*, i.e. that generatively they produce no observation and leave the observation sequence so far untouched (Young et al., 2006, section 1). Allowing non-emitting states can sometimes complicate algorithms. We follow standard practice and usually ignore this issue in the statement of our algorithms. Finally the length T of the generated sequence $y_{1:T}$ is not specified by the model by default, but for certain purposes such as speech synthesis must be chosen. Conceptually this can be incorporated into the

above framework by including a special *end* state which only ever transitions to itself and emits a dummy observation.

There are two important conditional independence assumptions encoded by the HMM: that $\mathbb{P}(\theta_t \,|\, \theta_{1:t-1}) = \mathbb{P}(\theta_t \,|\, \theta_{t-1})$; and that $\mathbb{P}(y_t \,|\, \theta_{1:T}) = \mathbb{P}(y_t \,|\, \theta_t)$. These conditional independence assumptions are to a large extent responsible for the efficient inference and learning described above. These may appear to be very restrictive assumptions in terms of the range of phenomena which may be modelled, but it is often possible to obtain a more accurate model by *augmenting* the state space. We will see examples below of the use of this trick for building more accurate probabilistic models of speech: it allows more accurate modelling of the duration of each phoneme (§3.1.5), and modelling of coarticulation effects where the acoustics at a given time depend not only on the current phoneme but also on neighbouring phonemes (§3.1.1 and §3.2.1). In general coming up with a good probabilistic model is often a trade-off between modelling accuracy and tractability of learning and inference. By augmenting the state as described above, we can improve modelling accuracy while staying within the tractable HMM framework. These conditional independence assumptions mean that in practical systems the state must serve two roles: encoding all the information about the past state history which is relevant to deciding where to transition to next; and encoding all the information about the state sequence which is relevant to producing the current observation.

2.5.2 Parameter estimation for HMMs

In this section we consider how to estimate the parameters of a fairly general form of hidden Markov model using expectation maximization and decision tree clustering. This form is closely related to the form used for modelling speech later. We focus on estimating the parameters λ of the distribution $\mathbb{P}(y \,|\, \theta, \lambda)$ since the state transition model is not a focus of this thesis. We first describe expectation maximization for HMMs in general, then describe expectation maximization for our particular model, then describe decision tree clustering for this model, and finally outline the complete parameter estimation procedure.

When applying expectation maximization to the HMM the hidden variable is the state sequence θ. For any HMM $\mathbb{P}(y, \theta \,|\, \nu, \lambda)$ the auxiliary function (2.88) used in expectation maximization takes the form

$$\mathcal{F}\left(Q, (\nu, \lambda)\right) = \mathbb{E}_Q \log \mathbb{P}(y \,|\, \theta, \lambda) + \mathbb{E}_Q \log \mathbb{P}(\theta \,|\, \nu) + H(Q) \tag{2.90}$$

where we have used the notation $\mathbb{E}_Q f(\theta)$ to denote $\sum_\theta Q(\theta) f(\theta)$. To re-estimate λ by maximizing the auxiliary function we may therefore consider only the first term above.

This has the form

$$\mathbb{E}_Q \log \mathbb{P}(y \mid \theta, \lambda) = \mathbb{E}_Q \sum_t \log \mathbb{P}(y_t \mid \theta_t, \lambda) \tag{2.91}$$

$$= \mathbb{E}_Q \sum_{t,\psi} \mathbb{I}\{\theta_t = \psi\} \log \mathbb{P}(y_t \mid \theta_t = \psi, \lambda) \tag{2.92}$$

$$= \sum_{t,\psi} \gamma_{t\psi} \log \mathbb{P}(y_t \mid \theta_t = \psi, \lambda) \tag{2.93}$$

where $\mathbb{I}A$ denotes the indicator function of a set A, and the state occupancy $\gamma_{t\psi} = \mathbb{E}_Q \mathbb{I}\{\theta_t = \psi\}$ is the probability under Q that the state θ_t at time t is equal to ψ. As we mentioned above these may be efficiently computed by the forward-backward algorithm.

We now consider an HMM where the distribution $\mathbb{P}(y_t \mid \theta_t = \psi, \lambda)$ for each state ψ is from an exponential family, and where a partition has been defined on state space and the model parameters are restricted to be shared across all states in a given cluster. The partition is specified by a function $\overline{q}$ from state space to a finite set of integers as described in §2.3. Let h and $[f_k]_k$ be the functions defining the exponential family. Then we assume

$$\log \mathbb{P}(y_t \mid \theta_t, \lambda) = h(y_t) + \sum_k \eta_{qk} f_k(y_t) - \log Z(\eta_q) \tag{2.94}$$

where $q = \overline{q}(\theta_t)$, $\eta_q = [\eta_{qk}]_k$ and $\lambda = ([\eta_q]_q, \overline{q})$. In this case we have

$$\mathbb{E}_Q \log \mathbb{P}(y \mid \theta, \lambda) = \sum_{q,t} \gamma_{tq} h(y_t) + \sum_{q,k} \eta_{qk} \sum_t \gamma_{tq} f_k(y_t) - \sum_q \tilde{R}_q \log Z(\eta_q) \tag{2.95}$$

$$= \sum_q \tilde{h}_q + \sum_{q,k} \eta_{qk} \tilde{f}_{qk} - \sum_q \tilde{R}_q \log Z(\eta_q) \tag{2.96}$$

$$= \sum_q \left(\tilde{h}_q + \sum_k \eta_{qk} \tilde{f}_{qk} - \tilde{R}_q \log Z(\eta_q) \right) \tag{2.97}$$

where

$$\gamma_{tq} = \sum_{\psi : \overline{q}(\psi) = q} \gamma_{t\psi} \tag{2.98}$$

$$\tilde{h}_q = \sum_t \gamma_{tq} h(y_t) \tag{2.99}$$

$$\tilde{f}_{qk} = \sum_t \gamma_{tq} f_k(y_t) \tag{2.100}$$

$$\tilde{R}_q = \sum_t \gamma_{tq} \tag{2.101}$$

Note that (2.97) is a sum over q of terms of the form (2.13). Thus the values $\tilde{f}_q$ and $\tilde{R}_q$ are sufficient for estimating the parameters η_q for cluster q, and we may re-estimate the parameters of each cluster separately.

Thus one iteration of expectation maximization in the above model divides naturally into the following stages. Given a current estimate of the parameters $\eta = [\eta_q]_q$:

- **initialize**: zero *accumulators* $\tilde{f}_q$ and $\tilde{R}_q$ for each cluster q
- for each utterance in the training corpus:
  - **infer**: use forward-backward to compute the state occupancies $\gamma_{t\psi}$ and so the cluster occupancies γ_{tq}.
  - **accumulate**: for each time t and cluster q, add $\gamma_{tq} f_k(y_t)$ to $\tilde{f}_{qk}$ and add γ_{tq} to $\tilde{R}_q$
- **re-estimate**: for each cluster q, use the accumulated values $\tilde{f}_q$ and $\tilde{R}_q$ to re-estimate the parameters η_q (using the general procedure for finding the maximum likelihood in the given exponential family)

We may estimate the partition $\overline{q}$ using decision tree clustering. Instead of using the maximum likelihood given a partition as the first term in the objective function as in (2.79), we use the maximum of the EM auxiliary function (2.90) given the partition. The use of a fixed distribution Q over state sequences makes clustering tractable. In fact we may use just the maximum of the first term $\mathbb{E}_Q \log \mathbb{P}(y \mid \theta, \lambda)$ of the auxiliary function since the other terms do not depend on λ. As we saw in (2.96) this function has the same functional form as the log likelihood for an exponential family. We may therefore use the procedure described in §2.3 to do decision tree clustering in this model, with the modification that the sufficient statistic values for each state ψ which form the basis of decision tree clustering are now computed as

$$\tilde{h}_\psi = \sum_t \gamma_{t\psi} h(y_t) \tag{2.102}$$

$$\tilde{f}_{\psi k} = \sum_t \gamma_{t\psi} f_k(y_t) \tag{2.103}$$

$$\tilde{R}_\psi = \sum_t \gamma_{t\psi} \tag{2.104}$$

A complete parameter estimation procedure might operate as follows. First an initial model which uses some fixed, simple form of tree is computed. The details of this initialization depend on the precise form of the model. Several iterations of EM may then be performed. Then decision tree clustering is performed, followed by several iterations of EM. This last step, consisting of clustering following by EM, may then be repeated zero or more times.

2.6 Summary of contributions

This chapter contains only a very minor novel contribution: a slight generalization of conditionally additive exponential families to the case $J > 1$ (§2.2.3).

Chapter 3

Speech synthesis

In this chapter we review some of the fundamentals of text-to-speech synthesis. Text-to-speech synthesis aims to synthesize speech from text. Modern approaches to speech synthesis are typically *corpus-based*, meaning that aspects of the system are learned from a training corpus of speech from a human speaker. The training corpus consists of a collection of (text, speech) pairs, where the speech is what the human speaker produced when asked to read the text aloud. There are two main current approaches to speech synthesis: *unit selection* synthesis and *statistical parametric speech synthesis*. Unit selection systems synthesize speech for previously unseen text by concatenating small segments of audio from the training corpus. A detailed overview of the unit selection approach, as well as earlier formant-based systems and other approaches, is given by Taylor (2009, chapters 13, 14 and 16). In this thesis we focus on statistical parametric speech synthesis. This approach involves the use of a probabilistic model defined over sequences of *speech parameters* representing the speech audio.

The layout of this chapter is as follows. We first review the extra steps of processing which surround the probabilistic model itself, converting input text to be read aloud to a *label sequence* and converting between a speech waveform and a sequence of speech parameters. We then describe the standard approach to using an HMM to model sequences of speech parameters, which we refer to as the *standard HMM synthesis framework*. There is a known inconsistency in the standard framework, and we describe a model known as the *trajectory HMM* (Zen et al., 2007b) which has previously been used to address this. Finally we discuss how to evaluate speech synthesis systems, and give details of the experimental set-up used for the experimental evaluations in this thesis.

3.1 Overview of statistical parametric speech synthesis

In this section we provide an overview of a typical statistical parametric speech synthesis system. This thesis focuses on the probabilistic model at the heart of this approach, and so we review the other components of the system only briefly. A more complete overview is given by Zen et al. (2009).

We first describe the *source-filter* representation of speech. A speech signal may be fairly accurately described as a quickly varying *source* or *excitation* signal passed through a slowly varying linear *filter* (Taylor, 2009). For example for the *STRAIGHT vocoder* (Kawahara et al., 1999), the source is treated as having both a periodic component and an aperiodic component. The periodic component essentially consists of a pulse train and is specified by a real value giving the *fundamental frequency* F_0 of this pulse train, or a special "unvoiced" value if there is no periodic component. The aperiodic component consists of coloured noise and is typically specified as an *aperiodicity spectrum* giving the relative power of the periodic and aperiodic components of the source at each frequency.

We now specify the form of a typical statistical parametric speech synthesis system. The text is represented as a sequence of *labels* encoding salient phonemic and linguistic aspects of the text, and the speech audio is represented as a sequence of *speech parameters* (not to be confused with the *model parameters* present in the probabilistic model) encoding information about the source and filter (Zen et al., 2009). This approach is referred to as *parametric* due to the use of speech parameters as the representation of audio. The system consists of three main components:

- a *front end* text analysis module which converts text to a label sequence
- a conditional probabilistic model of the speech parameter sequence given the label sequence, which we refer to as *the statistical model*
- a *back end* module consisting of an *analysis* component which takes a waveform and converts it to a speech parameter sequence, and a *synthesis* component which takes a speech parameter sequence and converts it to a waveform

The purpose of the front end and back end is to encode the raw text and speech in a way that is more amenable to being modelled using simple probabilistic models. To train the statistical model, each utterance in the training corpus is converted from a (text, speech) pair to a (label sequence, speech parameter sequence) pair by using the front end and the analysis component of the back end. These (label sequence, speech parameter sequence) pairs may then be used to learn the parameters of the statistical model using standard techniques such as maximum likelihood estimation. Once the statistical model has been trained, speech audio may be synthesized for previously unseen text by using the front

```
No-sir-ee!
```

Figure 3.1: Example of some text to be read aloud represented as a sequence of characters.

end to convert the input text to be read aloud to a label sequence, using the statistical model to generate a speech parameter sequence given this label sequence, and then using the synthesis component of the back end to generate a waveform.

In the remainder of this section we describe the front end and back end modules in more detail, and specify aspects of the statistical model that are common to almost all the models considered in this thesis.

3.1.1 Front-end text analysis

The front end or text analysis module converts text to a label sequence. The text is typically represented as a sequence of characters. An example of some text to be read aloud is given in Figure 3.1.

A simple front end would first tokenize the character sequence into a sequence of *words*, then look up a pronunciation for each word, consisting of a sequence of *phonemes*, in a *pronunciation dictionary* (Taylor, 2009, chapter 5). If no pronunciations exist for a given word then one is guessed based on a simple pronunciation model. If more than one pronunciation exists for a given word then one is chosen somehow. In this way the character sequence can be converted to a sequence of phonemes. Special "pause" or "silence" phonemes can also be inserted into the phoneme sequence to emulate *phrase breaks* where a speaker pauses momentarily at the end of a linguistic *phrase*. The resulting phoneme sequence is a simple form of label sequence, where each label, referred to as a *monophone label*, consists of a single phoneme, referred to as the *base phoneme*.

In practice, to obtain higher quality synthesis, more complicated text analysis is typically used and additional information is attached to each label. Since the text analysis stage is not the focus of this thesis, we only describe the format of the label sequence output. A good description of the challenges involved in text analysis is given by Taylor (2009, chapter 5). An example of a label for a typical statistical parametric speech synthesis system is shown in Figure 3.2. This label is a 53-tuple of various phonemic and linguistic features. We provide a few examples of the phonemic and linguistic features encoded in this label to give the reader some idea about the types of contextual information typically considered. A full list of the contextual features often used in statistical parametric speech synthesis systems for English is given by a number of authors (Zen et al., 2009; Tokuda et al., 2002b; HTS working group, 2013).

```
pau^n-ow+s=er@2_1/A:0_0_0/B:1-0-2@1-1&1-3#1-3$1-3!0-1;0-1|ow/C:1+1+2
    /D:0_0/E:det+1@1+3&1+2#0+1/F:content_1/G:0_0/H:3=3@1=1|L-L%
    /I:0=0/J:3+3-1
```

Figure 3.2: Example of a full-context label. This label encodes a 53-tuple of phonemic and linguistic context for a base phoneme `ow`. The format used is described in §3.1.1. The line wrapping shown is not present in the original.

- The string `pau^n-ow+s=er` specifies a tuple (`pau`, `n`, `ow`, `s`, `er`) of the previous-previous, previous, current, next and next-next phoneme in the phoneme sequence. That is, it specifies the base phoneme `ow` and surrounding phonemic context. Such a 5-tuple of phonemic context is referred to as a *quinphone*.
- The string `@2_1` encodes the fact that the current phoneme is the second and the last phoneme in the current *syllable*.
- The string `A:0` encodes the fact that the previous syllable was not stressed.
- The string `/E:det` encodes a guessed *part-of-speech tag* for the current word.
- The string `H:3=3@1=1|L-L%` encodes the fact that there are 3 syllables and 3 words in the current phrase, that the current phrase is the first and last, and that the *ToBI end tone* (Silverman et al., 1992) of the current phrase is L-L%.

A label thus encodes a base phoneme together with phonemic and linguistic context which may be relevant to the acoustic realization of the base phoneme. This contextual information can have a significant effect on a phoneme's acoustic realization due to effects such as *coarticulation* (Taylor, 2009). We refer to labels with a large amount of contextual information such as the one in Figure 3.2 as *full-context labels*. For brevity from now on we use the term *phonemic context* to refer to the quinphone part of a full-context label, and the term *linguistic context* to refer to the remaining parts, which are mostly non-phonemic.

It should be noted that for the text in the training corpus, text analysis typically contains an additional step. This is to cope with some of the potential mismatches between the phoneme sequence predicted by the process described above and the phoneme sequence actually used by the human speaker. For example the speaker may have chosen to insert phrase breaks at different positions than those predicted by the text analysis process, and may have used reduced vowels in places where the text analysis predicts non-reduced vowels. This is typically handled by iteratively training a simple monophone speech recognizer on the training data, where the network used by the speech recognizer largely follows the phoneme sequence predicted by text analysis but allows pauses to be inserted between

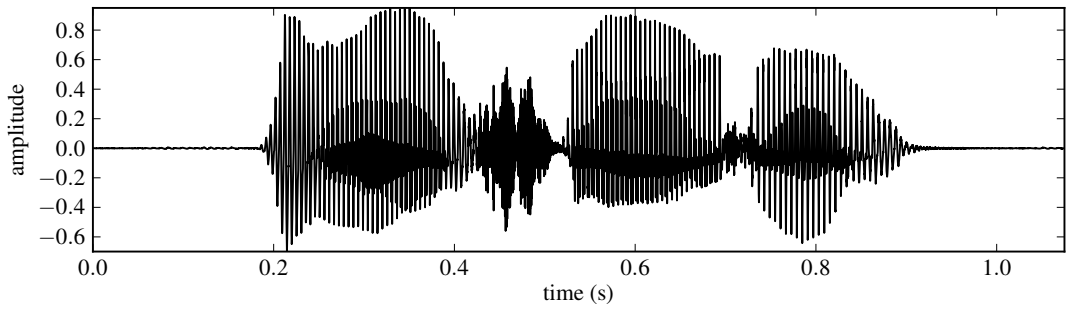


Figure 3.3: Example of a synthesized speech waveform. The value at each moment in time is the amplitude on an arbitrary scale. Time is discretized at a sampling rate of 16 kHz.

words and vowel reductions to be altered. The most likely phoneme sequence produced by the trained recognizer is then used in place of the original phoneme sequence produced by the process described above. Thus for utterances in the training corpus the speech audio is used to inform the text analysis process.

3.1.2 Back-end waveform analysis and synthesis

The back end converts speech audio to a sequence of speech parameters and vice versa. It consists of an *analysis* component which takes a waveform and converts it to a speech parameter sequence, and a *synthesis* component which takes a speech parameter sequence and converts it to a waveform.

The speech audio is represented as a *waveform*, which for the purposes of this thesis we take to be a sequence of real numbers representing (an affine transform of) the air pressure at a given position in space over a sequence of successive evenly-spaced moments in time. The number of such moments per second is referred to as the *sampling frequency* of the waveform. An example of a waveform is shown in Figure 3.3.

The representation of the speech audio seen by the statistical model is a sequence $[C_t]_{t=1}^{T}$ of *speech parameters*. Typically there is approximately one entry in this sequence per 5 ms of speech audio. The index t of this sequence is referred to as the *frame*. The speech parameters C_t for frame t encode information about the source-filter representation of the waveform near the time corresponding to frame t. For example for a typical statistical parametric speech synthesis system based on the STRAIGHT vocoder, the speech parameters C_t for frame t consist of three *portions*:

- a 40-dimensional real vector $[C_{t1i}]_{i=0}^{39}$ encoding the filter power spectrum at frame t in the form of a *mel cepstrum* (Fukada et al., 1992)

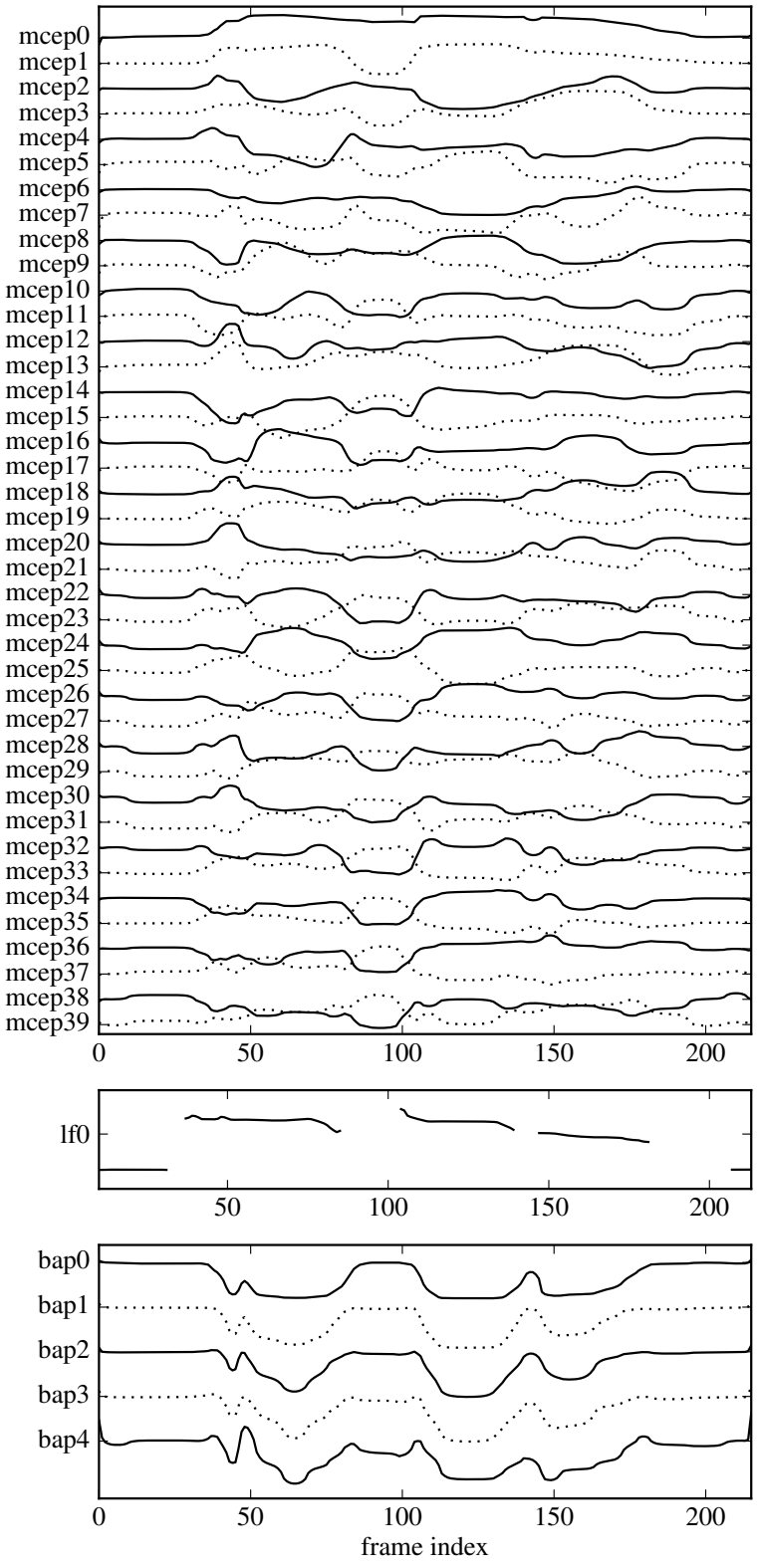


Figure 3.4: Example of a sequence of speech parameters. For each 5 ms frame the speech parameters consist of a 40-dimensional mel-cepstral vector (upper pane), a $\log F_0$ value (middle pane), and a 5-dimensional band aperiodicity vector (lower pane). For visual clarity the trajectories are offset from each other and are scaled to have unit global variance.

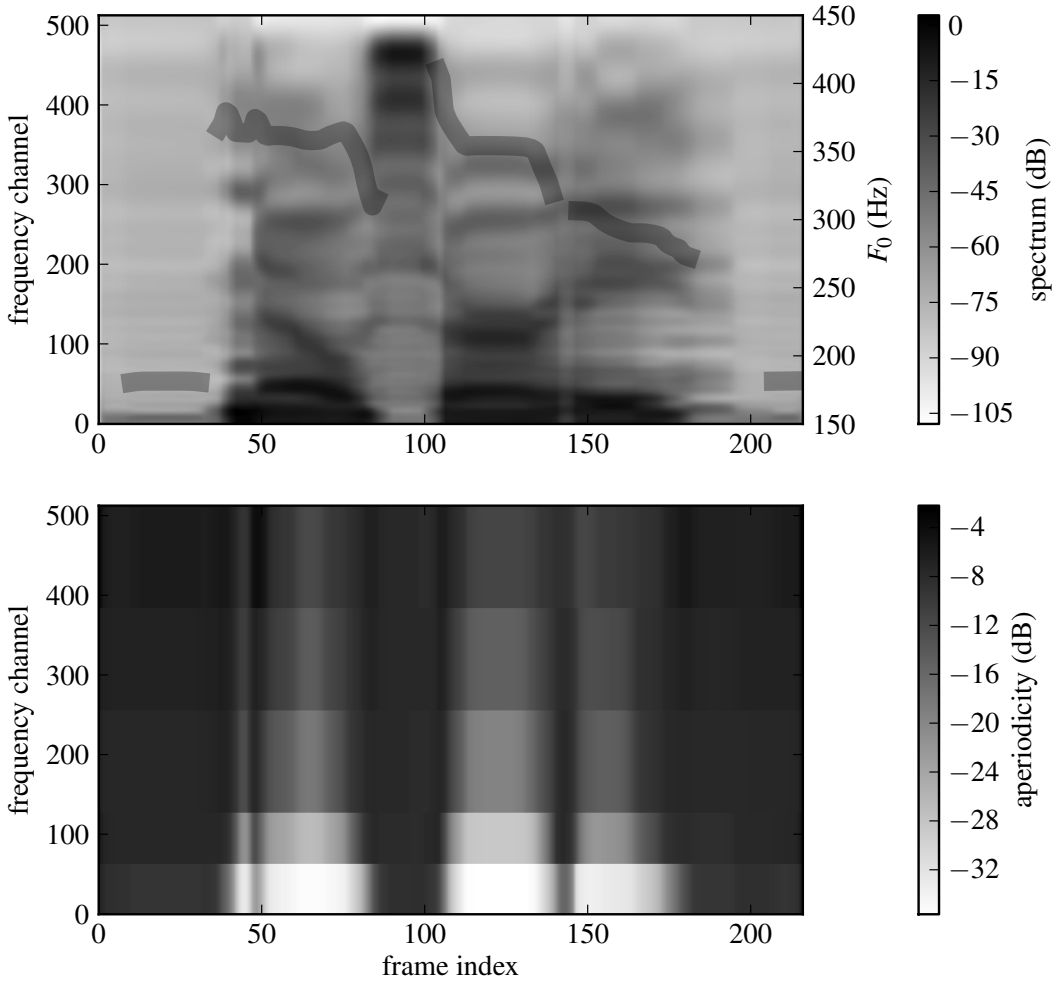


Figure 3.5: Example of a sequence of source-filter parameters for the STRAIGHT vocoder. For each 5 ms frame the parameters consist of a 513-dimensional vector specifying the power spectrum of the filter to be used for that frame (upper panel), an F_0 value specifying the fundamental frequency of the periodic part of the source excitation for that frame (shown by the solid line on the upper panel), and a 513-dimensional vector specifying the relative power of the aperiodic part of the source excitation at each frequency (lower panel).

- a 0/1-dimensional real vector consisting of the log fundamental frequency of the source excitation (0-dimensional for unvoiced frames, 1-dimensional for voiced frames)
- a 5-dimensional *band aperiodicity* vector $[C_{t3i}]_{i=0}^{4}$ encoding the aperiodicity spectrum as a piecewise constant function of frequency, constant on a set of frequency bands

The speech parameter for frame t, portion p and vector component i will be denoted C_{tpi}. An example of a sequence of speech parameters is shown in Figure 3.4. The corresponding filter power spectrum, fundamental frequency and aperiodicity spectrum for each frame are shown in Figure 3.5. The effect of using a piecewise constant aperiodicity spectrum can be clearly seen. While this banding may appear a crude model, the aperiodicity is generally regarded as having the weakest influence on the acoustics of the three portions of the speech parameters and so this is often sufficient to obtain reasonably natural speech.

We note in passing that the approach to statistical parametric speech synthesis described above suffers from a mismatch: the probabilistic model is trained to model speech parameter sequences extracted from natural speech using the analysis component of the back end, but its performance is judged on the results of applying the synthesis component of the back end. The analysis and synthesis components of the back end are designed to be approximate inverses, so that if we take a waveform, analyze it to obtain a sequence of speech parameters and then synthesize from these, we end up with something that is perceptually similar to the original. Running the analysis then the synthesis component of the back end on natural speech still results in fairly natural-sounding speech, so if the statistical model was perfect then this mismatch would not pose a substantial problem. However it is conceivable that weaknesses in the statistical model might interact with the lack of invertibility of the back end in deleterious ways. One solution to this mismatch would be to specify a conditional probabilistic model of the waveform given the speech parameter sequence, and train the overall probabilistic model of the waveform given the text as one big probabilistic model. There has been some work exploring this approach (Maia et al., 2010; Nakamura et al., 2013). We do not consider the above mismatch further in this thesis.

3.1.3 Alignments

The statistical model is at the heart of a statistical parametric speech synthesis system, and bridges the gap between the symbolic label sequence with no explicit notion of time and the continuous-valued, time-domain speech parameter sequence. To do so it makes use of a further intermediate representation called an *alignment* which contains timing information. A probabilistic model known as the *duration model* provides a conditional model of the alignment given the label sequence, and a probabilistic model known as the *speech parameter-level acoustic model* or the *acoustic model* provides a conditional model

```
0 37 pau
37 49 n
49 81 ow
81 104 s
104 140 er
140 178 iy
178 216 pau
```

Figure 3.6: Example of a label-level alignment with monophone labels. The first two columns are the start and end times of a segment, and the third column is the phoneme. The timings are in frames, where each frame spans 5 ms. For example `ow` lasts for 160 ms, starting at 245 ms from the start of the utterance and ending at 405 ms from the start of the utterance.

of the speech parameter sequence given the alignment. We now describe the alignment representation in detail.

A *label-level alignment* consists of a label sequence together with a *duration* in frames for each label, or equivalently a start and end frame for each label. An example of a label-level alignment with monophone labels is given in Figure 3.6. A label-level alignment thus associates each label with a certain segment of frames, and so specifies a *current* label for each frame t.

For extra modelling accuracy it is found to be helpful to consider each label as composed of a number of *sublabels* (slightly confusingly often called *states*) ordered in time. Each sublabel for a given label is current in turn for some number of frames. The number of sublabels for a given label could depend on the label, but we assume it is fixed and denote it as S, e.g. typically $S = 5$. We will assume the sublabel takes values in $\{1, \ldots, S\}$. Using sublabels is a simple and effective way to allow the acoustic realization of a given label to vary systematically over the course of that label while using relatively simple probabilistic models.

Given a label sequence we can form the corresponding sequence of (label, sublabel) pairs. To take a simple example with monophone labels, if the label sequence was (`k`, `ae`) then the corresponding (label, sublabel) sequence would be ((`k`, 1), (`k`, 2), (`k`, 3), (`k`, 4), (`k`, 5), (`ae`, 1), (`ae`, 2), (`ae`, 3), (`ae`, 4), (`ae`, 5)). An alignment specifying when each (label, sublabel) pair is current is referred to as a *sublabel-level alignment*. An example of a sublabel-level alignment with monophone labels is given in Figure 3.7.

Note that the fact we assume a hard boundary between successive labels does not mean the synthesized waveform contains discernible boundaries: we will see that the systems we build have various mechanisms for modelling effects such as coarticulation. We will treat

```
0 1 pau 1
1 2 pau 2
2 10 pau 3
10 32 pau 4
32 37 pau 5
37 40 n 1
40 43 n 2
43 46 n 3
46 47 n 4
47 49 n 5
49 52 ow 1
52 61 ow 2
61 69 ow 3
69 76 ow 4
76 81 ow 5
...
```

Figure 3.7: Example of a sublabel-level alignment with monophone labels. The first two columns are the start and end times of a segment, and the third and fourth columns are the phoneme and sublabel. The timings are in frames, where each frame spans 5 ms.

label boundaries as latent variables which have some relationship to physical reality but which are ultimately just a useful intermediate representation for specifying a model.

3.1.4 Duration model

A *duration model* is a conditional probabilistic model over the sublabel-level alignment given the label sequence. Given a label sequence $l = [l_j]_{j=1}^{J}$, the alignment may be thought of as a specification of the duration d_{js} in frames for each (label, sublabel) pair in the (label, sublabel) sequence, where j is the index of the label in the label sequence and s is the sublabel. Thus the duration model is a probabilistic model $\mathbb{P}(d \,|\, l, \nu)$ over the sequence $d = [d_{js}]_{j,s}$ of durations, where $j = 1, \ldots, J$ and s ranges over sublabels, given the label sequence $l = [l_j]_{j=1}^{J}$. Here ν is a collection of model parameters. Typical models used in statistical parametric speech synthesis assume the duration of each (label, sublabel) pair is independent of the duration of the other (label, sublabel) pairs. Typically it is assumed that the minimum duration of each sublabel is one frame. The duration model parameters ν are learned from data, often at the same time as the acoustic model parameters λ. We briefly mention how this can be done for one form of acoustic model in §3.2.3. At synthesis time we use the duration model to generate a sublabel-level alignment given the label sequence, i.e. to generate a duration d_{js} for each label l_j in the label sequence and each sublabel s.

This is referred to as *duration generation*. One way to do this is to use the most likely duration under the duration model for each (label, sublabel) pair.

We now specify the conventional duration model used in statistical parametric speech synthesis (Zen et al., 2009) and the one used throughout this thesis. The *Gaussian duration model* assumes that (Yoshimura et al., 1998)

$$d_{js} \,|\, l_j, \nu \sim \mathcal{N}(\mu_q, \sigma_q^2), \quad j \in \{1, \dots J\}, s \in \{1, \dots, S\} \tag{3.1}$$

$$\text{where } q = \overline{q}(l_j, s) \tag{3.2}$$

$$\nu = \left([\mu_q, \sigma_q^2]_{q=1}^{Q}, \overline{q} \right) \tag{3.3}$$

This specifies a probabilistic model $\mathbb{P}(d \,|\, l, \nu)$. The model parameters ν are a finite collection of means and variances $[\mu_q, \sigma_q^2]_{q=1}^{Q}$ and a clustering function $\overline{q}$ which takes a label m and sublabel s and returns a value $q \in \{1, \dots, Q\}$.

At first glance the Gaussian, which has support $\mathbb{R}$, is not well-suited to modelling the duration, which typically takes values in $\mathbb{N}$, but this mismatch is not found to cause major problems in practice. At synthesis time a duration must be chosen for each (label, sublabel) pair in the (label, sublabel) sequence. One way to do this is to choose the most likely value subject to the constraint that it be a positive integer, or equivalently to choose the closest integer to the mean of the Gaussian distribution for that pair or 1 if this is not positive. In principle the mismatch between $\mathbb{R}$ and $\mathbb{N}$ could be removed by using a probability distribution of the form

$$\mathbb{P}(d \,|\, \mu, \sigma^2) = \frac{\mathcal{N}(d; \mu, \sigma^2)}{\sum_{d \in \mathbb{N}} \mathcal{N}(d; \mu, \sigma^2)} \tag{3.4}$$

instead of the unnormalized distribution $\mathbb{P}(d \,|\, \mu, \sigma^2) = \mathcal{N}(d; \mu, \sigma^2)$. This distribution is sometimes known as the *discrete Gaussian distribution* (Gentry et al., 2008), though it is typically defined over $\mathbb{Z}$ rather than $\mathbb{N}$. It forms an exponential family with the same sufficient statistics as the conventional Gaussian distribution.

3.1.5 State transition model

It is possible to re-work many duration models into a Markovian form. When combined with certain types of acoustic model, this has the advantage of allowing efficient inference and learning (Zen et al., 2007c). We give a brief overview of this process. Full details, including how this affects inference and learning, can be found in a technical report by Zen (2007).

A *semi-Markov process* is an extension of the concept of a Markov process where each state in the state space has an explicit duration distribution associated with it. There is a simple trick to convert a discrete-time semi-Markov process on one state space into a

discrete-time Markov process on an enlarged state space: we simply augment the state with an integer specifying the number of steps remaining in the current original state and update the transition structure accordingly. Strictly speaking this trick results in a countably infinite state space unless each state in the original semi-Markov process has a maximum duration, but we ignore this technicality here.

For a given label sequence $[l_j]_{j=1}^J$ a duration model which assumes the duration of each (label, sublabel) pair is independent of the duration of the other such pairs is a simple example of a semi-Markov process. If we apply the above trick we obtain a Markovian *state transition model*. A state ψ in the state space of this new process is a tuple (m, j, s, d) consisting of the current label m, the index j of the current label in the label sequence, the current sublabel s, and the number of frames d remaining in the current sublabel. The state at frame t is denoted θ_t, and the sequence of states is referred to as the *state sequence* $\theta = [\theta_t]_{t=1}^T$. Thus the state transition model is a probabilistic model

$$\mathbb{P}\left(\theta \,|\, l, \nu\right) \tag{3.5}$$

over the state sequence $\theta = [\theta_t]_{t=1}^T$ given the label sequence $l = [l_j]_{j=1}^J$ with model parameters ν. Note that the length T of θ is a random variable. Assuming the minimum duration of each sublabel is one frame, there is a simple one-to-one correspondence between state sequences and sublabel-level alignments, so we are free to think in terms of whichever is most helpful.

If we apply the above procedure to the Gaussian duration model defined in §3.1.4 we obtain a Markovian state transition model $\mathbb{P}(\theta \,|\, l, \nu)$ which we refer to as the *Gaussian state transition model (GSTM)*. If a state sequence $\theta = [\theta_t]_{t=1}^T$ given the label sequence $l = [l_j]_{j=1}^J$ is distributed according to the Gaussian state transition model with parameters $([\mu_q, \sigma_q^2]_{q=1}^Q, \overline{q})$ then we write

$$\theta \,|\, l, [\mu_q, \sigma_q^2]_{q=1}^Q, \overline{q} \sim \mathrm{GSTM}\left(l; [\mu_q, \sigma_q^2]_{q=1}^Q, \overline{q}\right) \tag{3.6}$$

3.1.6 Acoustic model

A *speech parameter-level acoustic model* or simply *acoustic model* is a conditional probabilistic model $\mathbb{P}(C \,|\, \theta, \lambda)$ over the speech parameter sequence $C = [C_{tpi}]_{t,p,i}$ given the state sequence $\theta = [\theta_t]_t$ with model parameters λ, where time t ranges over $\{1, \ldots, T\}$, portion p ranges over $\{1, 2, 3\}$ and vector component i ranges over $\{0, \ldots, I_p - 1\}$. We refer to a sequence of real numbers as a *trajectory*. Most of the speech parameter-level acoustic models we consider in this thesis assume that the trajectory $[C_{tpi}]_{t=1}^T$ for portion p and vector component i is conditionally independent of the trajectories for all the other portions and vector components. We refer to a conditional probabilistic model $\mathbb{P}(c \,|\, \theta, \lambda)$ over a single trajectory $c = [c_t]_{t=1}^T$ given the state sequence as a *trajectory-level acoustic model*.

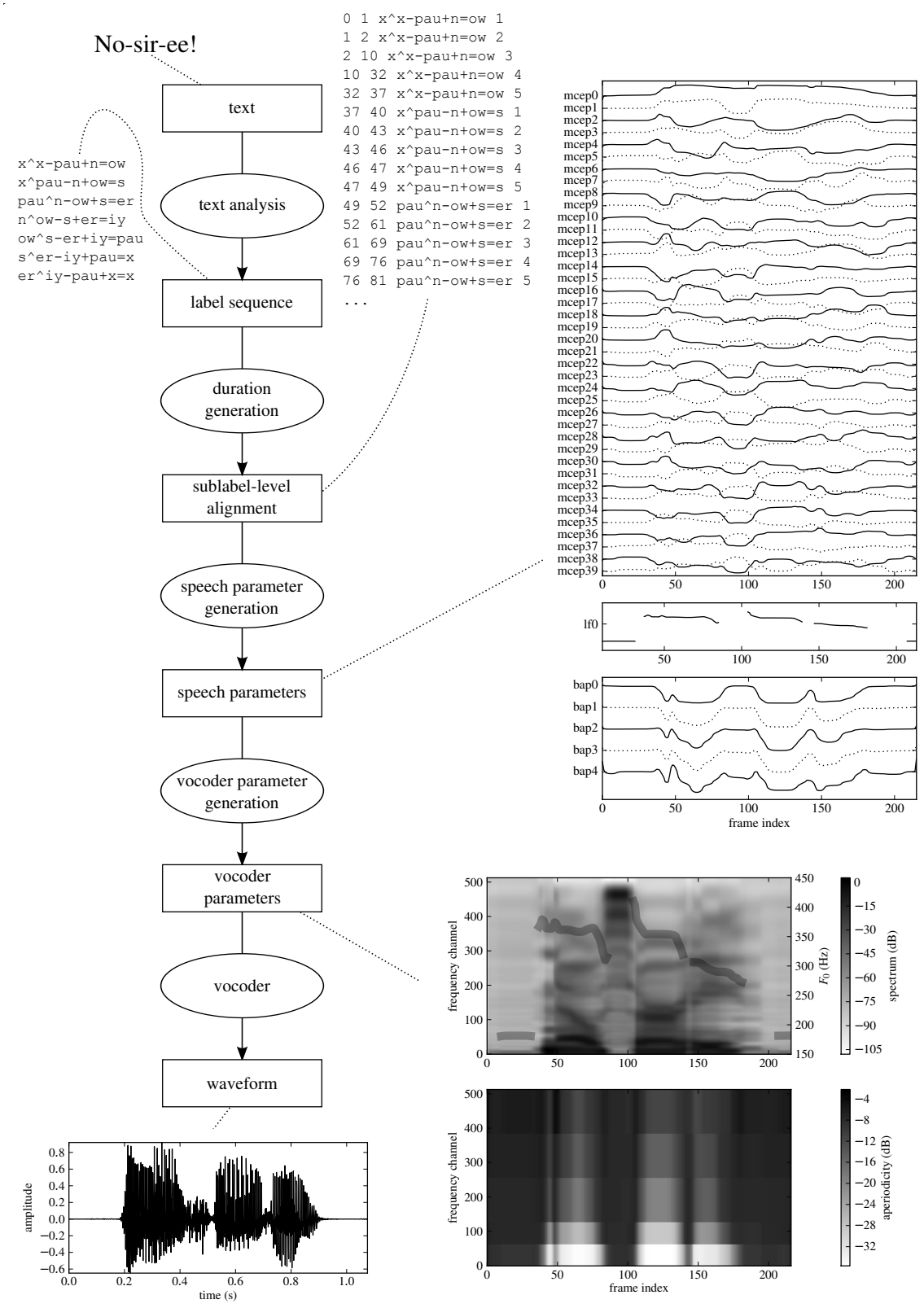


Figure 3.8: Example of a statistical parametric speech synthesis pipeline, which takes text to be read aloud as input and produces a waveform as output. Rectangles indicate representations of data, and ovals indicate the processes that map between these representations at synthesis time. For each representation a dotted line leads to an example of the representation. For visual clarity only the quinphone part of each label is shown.

The parameters λ of the speech parameter-level acoustic model are learned from data. Typically they are learned jointly with the duration model parameters ν. The details of parameter estimation depend on the form of acoustic model.

At synthesis time we use the speech parameter-level acoustic model to generate a speech parameter sequence C given a state sequence θ. This is referred to as *speech parameter generation*. One way to do this is to use the most likely speech parameter sequence given the state sequence. We refer to this as *maximum probability speech parameter generation*. The term *maximum likelihood speech parameter generation* is more widely in use for this parameter generation method, but to us maximum likelihood denotes a particular approach to estimating the parameters of a probabilistic model, which has nothing to do with speech parameter generation, and the likelihood function $\lambda \mapsto \mathbb{P}(C \mid \theta, \lambda)$ for the acoustic model is not involved in any way in speech parameter generation. We will see how to apply this and other speech parameter generation methods to various acoustic models in the remainder of this thesis.

A summary of the different processes used at synthesis time to successively transform text into speech audio is given in Figure 3.8.

3.2 The standard HMM synthesis framework

In this section we describe one particular specialization of the general approach presented in §3.1 which we call the *standard HMM synthesis framework*. It operates as follows. At training time, for each trajectory in the training corpus, we compute its approximate derivative and second derivative with respect to time to obtain a *delta trajectory* and *delta-delta trajectory*. We then train a statistical model, combining the Gaussian duration model described in §3.1.4 with a simple acoustic model, on this augmented data. The acoustic model assumes the original, delta and delta-delta trajectories are conditionally independent of each other given the state sequence, even though they are in fact deterministically related. At synthesis time we perform duration generation by finding the most likely duration for each sublabel under the trained Gaussian duration model. We perform speech parameter generation by finding the trajectory such that the combination of this trajectory, the corresponding delta trajectory and the corresponding delta-delta trajectory has the highest probability under the trained acoustic model.

This approach has the advantage of using a simple and tractable acoustic model during training, while allowing the acoustic model to capture some information about the dynamics of the speech parameters and to use this information at synthesis time. It has the disadvantage that it is inconsistent in its treatment of the relationship between the trajectory and its corresponding delta and delta-delta trajectories, taking this relationship into account at synthesis time but ignoring it during training.

The remainder of this section specifies the standard HMM synthesis framework in detail. A commonly used software implementation of the standard HMM synthesis framework is the *HMM-based speech synthesis system (HTS)* (HTS working group, 2012), based on the *hidden Markov model toolkit (HTK)* (Young et al., 2006).

3.2.1 Gaussian acoustic model

In this section we describe the *Gaussian acoustic model (GAM)*, which is a very simple trajectory-level acoustic model $\mathbb{P}(c \,|\, \theta, \lambda)$ used as a building block in the statistical model used during training by the standard framework. Its model parameters λ are a finite collection of means and variances $[\mu_q, \sigma_q^2]_{q=1}^Q$ and a clustering function $\overline{q}$ which takes a state ψ and returns a value $q \in \{1, \ldots, Q\}$. The trajectory value c_t at time t is assumed to be conditionally independent of the trajectory value at other times and Gaussian distributed with mean and variance depending on the corresponding value of q, that is

$$c_t \,|\, \theta_t, \lambda \sim \mathcal{N}(\mu_q, \sigma_q^2), \quad t \in \{1, \ldots, T\} \tag{3.7}$$

$$\text{where } q = \overline{q}(\theta_t) \tag{3.8}$$

$$\lambda = \left([\mu_q, \sigma_q^2]_{q=1}^Q, \overline{q} \right) \tag{3.9}$$

Instead of using the mean and variance, we will find it convenient to parameterize the Gaussian in terms of its natural parameters, so we have

$$c_t \,|\, \theta_t, \lambda \sim \mathcal{N}_{\mathrm{NP}}(b_q, \tau_q), \quad t \in \{1, \ldots, T\} \tag{3.10}$$

$$\text{where } q = \overline{q}(\theta_t) \tag{3.11}$$

$$\lambda = \left([b_q, \tau_q]_{q=1}^Q, \overline{q} \right) \tag{3.12}$$

For generality and notational convenience we have made $\overline{q}$ a function of the state $\psi = (m, j, s, d)$, but in practice it typically only depends on the label m and sublabel s. If a trajectory $c = [c_t]_{t=1}^T$ given the state sequence $\theta = [\theta_t]_{t=1}^T$ is distributed according to the Gaussian acoustic model with parameters $([b_q, \tau_q]_{q=1}^Q, \overline{q})$ then we write

$$c \,|\, \theta, [b_q, \tau_q]_{q=1}^Q, \overline{q} \sim \mathrm{GAM}\left(\theta; [b_q, \tau_q]_{q=1}^Q, \overline{q} \right) \tag{3.13}$$

It can be shown that the Gaussian acoustic model is a conditionally additive exponential family, as long as we allow the base measure function $h(c, \theta)$ in (2.34) to take on the value $-\infty$ when c and θ are of different lengths.

We can also write the Gaussian acoustic model explicitly as a multivariate Gaussian distribution over c. We have

$$c \,|\, \theta, [b_q, \tau_q]_{q=1}^Q, \overline{q} \sim \mathcal{N}_{\mathrm{NP}}(b^{\mathrm{win}}, P^{\mathrm{win}}) \tag{3.14}$$

where

$$b^{\text{win}} = \left[b_{\overline{q}(\theta_t)}\right]_{t=1}^{T} \tag{3.15}$$

$$P^{\text{win}} = \operatorname{diag}\left(\left[\tau_{\overline{q}(\theta_t)}\right]_{t=1}^{T}\right) \tag{3.16}$$

Note that the precision matrix P^{win} is diagonal since each the trajectory value at each frame is independent of the trajectory values at other frames.

3.2.2 Windows

In this section we describe the concept of a *window* and introduce related terminology. In the standard HMM synthesis framework windows are used to specify how the approximate first and second time derivatives of a trajectory are computed.

Given a trajectory $c = [c_t]_{t=1}^{T}$ we may define a new trajectory $\tilde{c} = [\tilde{c}_t]_{t=1}^{T}$ by taking the cross-correlation with a *window* $[w_k]_{k=-K^{\text{L}}}^{K^{\text{R}}}$, that is

$$\tilde{c}_t = \sum_{k=-K^{\text{L}}}^{K^{\text{R}}} w_k c_{t+k} \tag{3.17}$$

Here $w_k \in \mathbb{R}$ is referred to as a *window coefficient*, and the *left extent* K^{L} and *right extent* K^{R} are typically non-negative. The function which takes c and returns $\tilde{c}$ is linear, and is local in the sense that the value of $\tilde{c}_t$ is only affected by values of c_u for u close to t. The matrix W for this linear function has entries $W_{st} = w_{t-s}$, and so is Toeplitz, and is banded with subdiagonal width at most K^{L} and superdiagonal width at most K^{R}. We refer to the $T \times T$ matrix W as the *window matrix* for the given window. The window matrix implicitly depends on T.

Strictly speaking the value of $\tilde{c}_t$ for the first few and the last few frames is not defined by (3.17), since the value of c_t is not specified for $t < 1$ or $t > T$. One way to specify these values of $\tilde{c}_t$ is to say that c_t is assumed zero for $t < 1$ or $t > T$, which is equivalent to making the matrix W exactly Toeplitz. We refer to this approach as using *zero-input windows*. Another way to specify these values is to say that $\tilde{c}_t$ is set to zero for the first few and last few frames for which it would otherwise not be well-defined, and this is equivalent to setting the first few and last few rows of the matrix W to zero. We refer to this approach as using *zero-output windows*. More general choices where entirely different window coefficients are used for the first few or last few rows of W are also possible. We will largely ignore such issues, but it will sometimes be useful to consider the effect of such choices, which we term *end effects*. Some of the issues surrounding end effects are discussed in more detail in Appendix B.

The window coefficients can be chosen such that $\tilde{c}$ is a finite-difference approximation to the first or second derivative with respect to time of a continuous-time function which

window	offset (k)		
	-1	0	$+1$
static ($d = 0$)		1.0	
delta ($d = 1$)	-0.5	0.0	0.5
delta-delta ($d = 2$)	1.0	-2.0	1.0

Table 3.1: Windows often used with the standard HMM synthesis framework. Each entry specifies a window coefficient w_{dk} for window d, offset k.

has value c_t at time $t \in \mathbb{Z}$. We refer to the original, approximate first derivative and approximate second derivative trajectories as the *static*, *delta* and *delta-delta* trajectories, and the windows used to compute these as the static, delta and delta-delta windows. For example the windows shown in Table 3.1 are often used with the standard HMM synthesis framework to compute a static, delta and delta-delta trajectory. We refer to the trajectory obtained by applying a non-trivial window to the static trajectory as a *dynamic trajectory*.

We now review some terminology that will be useful below when we have a collection of D windows. We refer to the maximum left extent of the D windows as the left extent of the collection, and similarly for the right extent. We refer to a D-tuple of trajectories as a *trajectory tuple*. A trajectory tuple is naturally viewed as a $D \times T$ matrix, but by taking the transpose can also be viewed as a $T \times D$ matrix $o = [o_{td}]_{t,d}$, or by flattening can be viewed as a (DT)-dimensional vector. The function which takes a trajectory and applies each of the D windows to obtain a trajectory tuple can thus be seen as a linear map $\overline{o} : \mathbb{R}^T \to \mathbb{R}^{DT}$. For typical collections of windows such as the collection in Table 3.1, $\overline{o}$ is injective, meaning we can recover the original trajectory from its image under the map, and we will assume injectivity from now on. However $\overline{o}$ is typically not surjective, since $DT > T$ if $D \geq 2$ and $T \geq 1$. We refer to the image of $\overline{o}$ as the *realizable subspace*, and we refer to a trajectory tuple in the image as *realizable*. The realizable subspace forms a T-dimensional subspace of the (DT)-dimensional space of possible trajectory tuples. Thus most trajectory tuples are not realizable. Saying a trajectory tuple is realizable means that the different trajectories which make up the trajectory tuple are in some sense compatible with each other. For example for the windows in Table 3.1 it means that the $d = 1$ element in the trajectory tuple is the delta trajectory derived from the $d = 0$ element in the trajectory tuple, and the $d = 2$ element in the trajectory tuple is the delta-delta trajectory derived from the $d = 0$ element.

3.2.3 Statistical model used during training

In this section we describe the statistical model used by the standard HMM synthesis framework during training and discuss estimation of its parameters. As mentioned above,

the procedure used during training is to model the speech parameter sequence after it has been augmented with "dynamic" information using the approach in §3.2.2. The statistical model is therefore defined over a space containing this augmented speech parameter sequence rather than over the speech parameter sequence itself.

The statistical model used during training is a conditional probabilistic model $\mathbb{P}(O \mid l, \nu, \lambda)$ over an *observation sequence* $O = [O_{tpid}]_{t,p,i,d}$ where time t ranges from 1 to T, portion p ranges from 1 to 3, vector component i ranges from 0 to $I_p - 1$, and window d ranges from 0 to $D_p - 1$. The statistical model is composed of a duration or state transition model and an acoustic model. The duration model is the Gaussian duration model. The acoustic model assumes the trajectory $[O_{tpid}]_{t=1}^{T}$ for each portion p, vector component i and window d is conditionally independent of the other trajectories, and each trajectory is modelled using the Gaussian acoustic model. The complete statistical model $\mathbb{P}(O \mid l, \nu, \lambda)$ used during training by the standard HMM synthesis framework is therefore:

$$\theta \mid l, \nu, \overline{q}^{\mathrm{DUR}} \sim \mathrm{GSTM}\left(l; \left[\mu_q^{\mathrm{DUR}}, (\sigma^2)_q^{\mathrm{DUR}}\right]_{q=1}^{Q^{\mathrm{DUR}}}, \overline{q}^{\mathrm{DUR}}\right) \tag{3.18}$$

$$[O_{tpid}]_{t=1}^{T} \mid \theta, \lambda \sim \mathrm{GAM}\left(\theta; [b_{pqid}, \tau_{pqid}]_{q=1}^{Q_p}, \overline{q}_p\right), \quad \begin{cases} p \in \{1,2,3\} \\ i \in \{0, \ldots, I_p - 1\} \\ d \in \{0, \ldots, D_p - 1\} \end{cases} \tag{3.19}$$

$$\text{where } \nu = \left(\left[\mu_q^{\mathrm{DUR}}, (\sigma^2)_q^{\mathrm{DUR}}\right]_q, \overline{q}^{\mathrm{DUR}}\right) \tag{3.20}$$

$$\lambda = \left[[b_{pqid}, \tau_{pqid}]_{q,i,d}, \overline{q}_p\right]_p \tag{3.21}$$

Since the Gaussian state transition model is Markovian and the Gaussian acoustic model used for each trajectory assumes the trajectory values over time are conditionally independent, the above statistical model is an HMM. As discussed above, each of the functions $\overline{q}^{\mathrm{DUR}}$ and $[\overline{q}_p]_p$ determines a partition of the set of all possible (label, sublabel) pairs. A separate partition is assumed for each portion p in the acoustic model and for the duration model.

The observation sequence O used for each utterance during training is derived from the speech parameter sequence C. Given a speech parameter sequence $C = [C_{tpi}]_{t,p,i}$, define a corresponding observation sequence $O = [O_{tpid}]_{t,p,i,d}$ by specifying that each trajectory $[O_{tpid}]_{t=1}^{T}$ is the result of applying the window $[w_{pdk}]_k$ to the trajectory $[C_{tpi}]_{t=1}^{T}$. Here we have assumed that the same window is used by all vector components i of a given portion p of the speech parameters, but that different windows may be used for different portions. Extending the discussion in §3.2.2 slightly, the function $\overline{O}$ taking a speech parameter sequence C to the corresponding observation sequence O is an injective linear map. We refer to an observation sequence O as *realizable* if it is in the image of $\overline{O}$, that is if $O = \overline{O}(C)$ for some speech parameter sequence C, or equivalently if all its trajectory tuples are realizable.

The parameters of the statistical model $\mathbb{P}(O \,|\, l, \nu, \lambda)$ may be learned from data using expectation maximization and decision tree clustering, following the procedure described in §2.5.2, since $\mathbb{P}(O_t \,|\, \theta_t, \lambda)$ is of the form (2.94). The accumulators for the sufficient statistics in (2.100) and (2.101) are now

$$\tilde{b}_{pqid} = \sum_t \tilde{\gamma}_{tpq} O_{tpid} \tag{3.22}$$

$$\tilde{\tau}_{pqid} = -\tfrac{1}{2} \sum_t \tilde{\gamma}_{tpq} O_{tpid}^2 \tag{3.23}$$

$$\tilde{R}_{pq} = \sum_t \tilde{\gamma}_{tpq} \tag{3.24}$$

$$\text{where } \tilde{\gamma}_{tpq} = \sum_{\psi : \bar{q}_p(\psi) = q} \gamma_{t\psi} \tag{3.25}$$

The use of a separate partition for each portion p of the speech parameters is a minor extension of the model described in §2.5.2. Given the above accumulators, the updated parameters (b_{pqid}, τ_{pqid}) for the Gaussian distribution for each portion p, cluster q, vector component i and window d may be computed using (2.32) and (2.33). The standard practice in statistical parametric speech synthesis is to use a separate decision tree for each portion p and sublabel s (Zen et al., 2009). Conceptually this may be thought of as a tree where the root node splits s ways according to the sublabel. Accordingly the questions typically used only involve the label and not the sublabel. As mentioned in §3.2.1 the acoustic model typically does not use other aspects of the state $\psi = (m, j, s, d)$ such as the number of frames d remaining in the current sublabel. Expectation maximization and decision tree clustering can similarly be used to re-estimate the duration model parameters ν, though we omit the details since it is not the focus of this thesis; full details of expectation maximization for the Gaussian state transition model are given by Zen (2007).

3.2.4 Standard speech parameter generation

In this section we describe a speech parameter generation method which uses a simple but effective approach to make use of the information about the dynamics of trajectories encoded in the delta and delta-delta model parameters. We also describe an efficient algorithm for computing the trajectory generated by this method.

Speech parameter generation involves generating a speech parameter sequence $C = [C_{tpi}]_{t,p,i}$ given a state sequence $\theta = [\theta_t]_{t=1}^T$. For *standard speech parameter generation* (Tokuda et al., 2000, case 1) the speech parameter sequence C is chosen to be the one which maximizes the probability of the corresponding observation sequence O under the trained acoustic model $\mathbb{P}(O \,|\, \theta, \lambda)$. This is equivalent to finding the most likely realizable observation sequence O. It can thus be seen as a constrained form of maximum probability

speech parameter generation where we restrict the domain over which the maximization is conducted. Due to the way the acoustic model $\mathbb{P}(O \mid \theta, \lambda)$ factorizes with respect to portion p and vector component i, we may perform the above constrained maximization separately for each p and i. We therefore now focus on standard speech parameter generation for a given portion and vector component.

Given D windows, D corresponding Gaussian acoustic models with parameters $[b_{qd}, \tau_{qd}]_{q,d}$, and a clustering function $\overline{q}$, we wish to find the trajectory tuple $[o_{td}]_{t,d}$ which has the highest probability subject to the constraint that there is some trajectory $c \in \mathbb{R}^T$ for which $W_d c = [o_{td}]_{t=1}^T$ for all windows d. The distribution over the trajectory $[o_{td}]_{t=1}^T$ for window d is a multivariate Gaussian with b-value b_d^{win} and diagonal precision matrix P_d^{win} given by using (3.15) and (3.16) for window d. Thus the overall log probability we want to maximize with respect to c is

$$\sum_d \log \mathcal{N}_{\text{NP}}(W_d c; b_d^{\text{win}}, P_d^{\text{win}}) \stackrel{c}{=} \sum_d \left\{ -\tfrac{1}{2}(W_d c)^{\mathsf{T}} P_d^{\text{win}} (W_d c) + {b_d^{\text{win}}}^{\mathsf{T}} (W_d c) \right\} \tag{3.26}$$

$$= A(c) \tag{3.27}$$

$$\text{where } A(c) = -\tfrac{1}{2} c^{\mathsf{T}} P c + b^{\mathsf{T}} c \tag{3.28}$$

$$b = \sum_d {W_d}^{\mathsf{T}} b_d^{\text{win}} \tag{3.29}$$

$$P = \sum_d {W_d}^{\mathsf{T}} P_d^{\text{win}} W_d \tag{3.30}$$

where $\stackrel{c}{=}$ denotes equality up to a constant. Note that here we have reused b: it may refer either to a vector $b = [b_t]_{t=1}^T$ over time or to a collection $b = [b_{qd}]_{q,d}$ of model parameters. We hope that the intended meaning will be clear from context. The trajectory c generated by standard speech parameter generation is therefore $P^{-1}b$, since this maximizes the above quadratic function.

There exists an efficient algorithm to compute $P^{-1}b$. Since the matrices ${W_d}^{\mathsf{T}}$, P_d^{win} and W_d are all banded, their product is banded, so P is banded. The subdiagonal width of P, which is equal to the superdiagonal width since P is symmetric, is at most K, where $K = K^{\text{L}} + K^{\text{R}}$ is the sum of the left and right extents of the collection of windows. For example for the windows shown in Table 3.1 $K = 2$. If L is the conventional Cholesky factor of P, so that $P = LL^{\mathsf{T}}$, then $P^{-1}b = L^{-\mathsf{T}}L^{-1}b$. Thus the optimal trajectory $P^{-1}b$ may be computed in $O(TK^2)$ time and $O(TK)$ space by using a banded Cholesky decomposition followed by two banded triangular matrix solves (Tokuda et al., 2000). There are high quality implementations of the banded Cholesky decomposition available. For example LAPACK (Anderson et al., 1999) contains an implementation of the banded Cholesky decomposition and the subsequent matrix solves. We refer to the above Cholesky-based algorithm as the *standard speech parameter generation algorithm*.

There is an intuitive way to view the standard speech parameter generation method in terms of *soft constraints* which we will find helpful later. The Gaussian distribution specified by model parameters (b_{qd}, τ_{qd}) can be seen as encoding a soft constraint on the trajectory tuple value o_{td} that the model expects to see while in cluster q, i.e. for frames t where $\overline{q}(\theta_t) = q$. On average the model expects to see a value of b_{qd}/τ_{qd} for o_{td} while in cluster q, but also expects to see a certain range of variation controlled by τ_{qd}. In this view the standard speech parameter generation method trades off the conflicting soft constraints for the various windows d to produce a compromise trajectory. The constraints are conflicting because it is not possible to make each trajectory in the trajectory tuple precisely equal to its preferred value for every window d simultaneously. The parameter τ_{qd} controls how important the corresponding soft constraint is: the larger τ_{qd} the more harshly a potential trajectory tuple is penalized for deviating from the preferred value b_{qd}/τ_{qd}. The view of the standard speech parameter generation method as trading off competing soft constraints is fairly well known (see for example Figure 5 in (Zen et al., 2009)).

3.2.5 Time-recursive speech parameter generation algorithm

The above Cholesky-based algorithm is fast, but requires $O(T)$ time to compute the first frame since the Cholesky decomposition for the whole trajectory must first be computed. This means latency can potentially be high, and both latency and memory usage are not predictable at design time since utterances vary in length. In practice a time-recursive algorithm (Koishida et al., 2001) is often used in real-time synthesis systems and other applications that would otherwise use the standard Cholesky-based speech parameter generation algorithm and which require low latency (Koishida et al., 2001; Muramatsu et al., 2008; Han et al., 2012). This time-recursive algorithm is approximate and slower but has predictable latency, memory and CPU requirements.

3.2.6 Parameter generation considering global variance

There is a commonly used extension to the standard speech parameter generation method which dramatically improves the quality of synthesized speech. In particular it results in speech that is clearer and less "muffled" than the standard speech parameter generation method.

Define the *global variance (GV)* $\overline{v}(c)$ of a trajectory $c = [c_t]_{t=1}^{T}$ as the variance of the set $\{c_t : t \in \{1, \ldots, T\}\}$, that is

$$\overline{v}(c) = \frac{1}{T} \sum_t (c_t - \overline{m}(c))^2 \tag{3.31}$$

where

$$\overline{m}(c) = \frac{1}{T}\sum_t c_t \tag{3.32}$$

is the *global mean* of the trajectory. Trajectories produced by the standard speech parameter generation method are found to have global variance that is smaller than trajectories derived from natural speech. We saw in §3.2.4 that for a given portion p, vector component i and state sequence θ the standard speech parameter generation method generates the trajectory c by optimizing the quadratic function $A(c)$ defined in (3.28). *Speech parameter generation considering global variance* (Toda and Tokuda, 2007) generates each trajectory c by optimizing a modified utility function

$$G(c) = A(c) + \omega^{\mathrm{GV}} \log \mathcal{N}\left(\overline{v}(c); \mu^{\mathrm{GV}}, (\sigma^2)^{\mathrm{GV}}\right) \tag{3.33}$$

Since we have one trajectory for each portion p and vector component i, the extra parameters required for parameter generation considering global variance are

$$\left[\mu_{pi}^{\mathrm{GV}}, (\sigma^2)_{pi}^{\mathrm{GV}}\right]_{p,i} \tag{3.34}$$

Typically μ_{pi}^{GV} and $(\sigma^2)_{pi}^{\mathrm{GV}}$ are set to the empirical mean and variance of $\overline{v}([C_{tpi}]_t)$ over the training corpus. The trajectories generated by this approach have global variance closer to the average natural global variance μ^{GV} than trajectories generated by the standard speech parameter generation method. The *GV weight* ω^{GV} controls the trade-off between producing a likely trajectory and a trajectory with likely global variance. Typically $\omega^{\mathrm{GV}} = DT$, where D is the number of windows and T is the number of frames in the utterance (Toda and Tokuda, 2007). The conventional algorithm is to optimize $G(c)$ using a form of gradient ascent where the step direction is based on both the gradient and the diagonal of the Hessian matrix at the current point (Toda and Tokuda, 2007). The trajectory used to initialize gradient ascent is typically obtained by scaling the trajectory μ produced by standard speech parameter generation to have global variance μ^{GV}, that is

$$c_t^{\mathrm{VS}} = \sqrt{\frac{\mu^{\mathrm{GV}}}{\overline{v}(\mu)}}(\mu_t - \overline{m}(\mu)) + \overline{m}(\mu) \tag{3.35}$$

We refer to c^{VS} as the trajectory produced by *variance scaling*.

It has been noted that in practice the effect of using the conventional GV weight setting $\omega^{\mathrm{GV}} = DT$ with the standard HMM synthesis framework is to restrict the GV of the generated trajectories to be very close to the mean μ^{GV} of the GV pdf (King, 2011; Zen et al., 2010). We will observe this phenomenon in §6.5.

Speech parameter generation considering global variance does have some drawbacks. Due to the gradient ascent procedure it is slower than standard speech parameter generation, and it is not possible to use the low latency time-recursive parameter generation

algorithm. It is also known to sometimes introduce *artifacts* into the synthesized speech (Zen et al., 2006; Yamagishi et al., 2008; Zen et al., 2009; Toda, 2011; King, 2011). Here by an artifact we mean a short distortion in the audio, such as a click, pop or short high-pitched whine. Existing implementations of parameter generation considering global variance, such as that found in HTS, reduce artifacts by carefully tuning the convergence criterion used during gradient ascent, providing a form of *early stopping* (Yamagishi et al., 2008; HTS working group, 2012). We discuss ways to address all of these drawbacks in Chapter 6.

There is an alternative approach to alleviating muffledness which we now describe. *Post-filtering* for speech synthesis (Yoshimura et al., 2004, 2005; Ling et al., 2006) involves increasing the "dynamic range" of the filter power spectrum used during synthesis by raising it to a power $1 + \beta$, while making adjustments such that the overall energy and optionally the spectral tilt are not affected. When using mel cepstral coefficients to represent the filter power spectrum this corresponds to multiplying each coefficient by $1 + \beta$, then making an energy adjustment to the 0^{th} coefficient and optionally a spectral tilt adjustment to the 1^{st} coefficient. Post-filtering is conventionally applied to the output of the standard speech parameter generation method. Perceptually it is found that accentuating the peaks and valleys of the filter power spectrum in this way results in more natural and less muffled speech. It is generally agreed that parameter generation considering global variance results in more natural speech than post-filtering, though we are not aware of any literature directly investigating this, whereas post-filtering is simpler to implement.

3.2.7 Modelling fundamental frequency

The framework presented above cannot be used as is for modelling the fundamental frequency portion of the speech parameters due to the use of the special unvoiced value. We only discuss the issues surrounding fundamental frequency briefly, since it is not the focus of this thesis.

The conventional approach to modelling fundamental frequency is to use *multi-space distributions* (Zen et al., 2009). A multi-space distribution is simply a mixture distribution where the mixture components are defined over different dimensionalities of spaces. Note that this means which mixture component generated a given sample is observed. If the parameterized distribution for each mixture component is an exponential family and the parameterized distribution which is responsible for deciding which mixture component to use is an exponential family then the overall multi-space parameterized distribution is also an exponential family. The model typically used for each frame of the F_0 trajectory in statistical parametric speech synthesis is a Bernoulli mixture of a Gaussian of dimensionality 0 and a Gaussian of dimensionality 1, which is therefore an exponential family. The interaction between multi-space distributions and applying windows complicates both computing

trajectory tuples during training and speech parameter generation to some extent (HTS working group, 2012).

There are subtleties in how to deal with unvoiced frames during F_0 generation. It is not possible to generate the most likely F_0 trajectory, since there is no concept of a most likely value for a distribution defined over multiple spaces of differing dimensionalities. One possibility is to first find the most likely sequence of voiced-or-unvoiced decisions then find the most likely F_0 trajectory given these decisions. Here by "most likely" we really mean "maximum probability under the model used during training subject to constraints between static and dynamic quantities". For the first part of this process an approximate, heuristic scheme is sometimes used, and for the second part standard speech parameter generation can be used (HTS working group, 2012).

3.3 Making the standard framework probabilistic

The standard framework has two related conceptual problems. Firstly, in spite of the fact that the standard HMM synthesis framework uses probabilistic modelling techniques such as maximum likelihood and expectation maximization to do parameter estimation, and in spite of the fact that it is viewed an instance of *statistical* parametric speech synthesis, it does not in fact define a valid probabilistic model. Secondly the standard framework is inconsistent in its enforcement of the constraints between static and dynamic trajectories, ignoring these constraints during training but taking them into account at synthesis time. In this section we describe these two problems in detail, and discuss one approach to resolving them using a model known as the trajectory HMM (Zen et al., 2007b). We also touch on some of the potential advantages of having a valid probabilistic model as a precursor to the main discussion of this issue in §5.4.1. The issues surrounding the lack of consistency in the standard framework were explored in a conference paper (Shannon et al., 2011).

3.3.1 Inconsistency in the standard framework

The standard HMM synthesis framework described in §3.2 is inconsistent in its enforcement of the constraints between static and dynamic trajectories. The acoustic model used during training ignores the relationship between static and dynamic trajectories, while the methods used for speech parameter generation take this relationship into account. In this section we examine some perspectives on this inconsistency. We focus on the case of a single trajectory $c = [C_{tpi}]_{t=1}^{T}$ for some fixed portion p and vector component i. We make use of the concept of a realizable subspace defined in §3.2.2.

The acoustic model used during training in the standard framework is defined as a probabilistic model over the trajectory tuple o. It could be argued that it is a poor model

of o, since it assigns significant probability mass away from the realizable subspace, and does not assign as much probability mass as it should to the realizable subspace. Indeed strictly speaking the model assigns a probability of zero to the realizable subspace, which may be viewed as problematic since the realizable subspace is the set of everything that could possibly happen! Another way to express the fact that this model is a poor model of o is to note that it would be trivial to beat it in a competition where the goal was to achieve the highest TSLP score over o. For example any fixed Gaussian distribution defined over the realizable subspace has a TSLP over o of $+\infty$, and a fixed Gaussian distribution with probability mass concentrated in a "thin" slice around the realizable subspace can achieve arbitrarily large TSLP values.

Alternatively the acoustic model used during training in the standard framework may be viewed as an unnormalized model over the trajectory c. If we consider $\mathbb{P}(o \,|\, \theta, \lambda)$ as a function of o, then substitute $o = \overline{o}(c)$, we obtain a function

$$\text{“}\mathbb{P}\text{”}(c \,|\, \theta, \lambda) = \mathbb{P}(o = \overline{o}(c) \,|\, \theta, \lambda) \tag{3.36}$$

$$= \prod_d \mathcal{N}_{\mathrm{NP}}(W_d c; b_d^{\mathrm{win}}, P_d^{\mathrm{win}}) \tag{3.37}$$

which assigns a weight to each trajectory c, where b_d^{win} and P_d^{win} depend on θ and λ. Here we have used the suggestive notation "$\mathbb{P}$" to indicate a quantity we would like to treat as a probability but is in fact just an unnormalized weight function. The integral of this weight function with respect to c is not necessarily 1: in general if we take a Gaussian pdf and restrict to a subspace, the integral over the subspace (using some particular parameterization of the subspace) may be greater or less than 1.

The above considerations show that the acoustic model used during training in the standard framework may be viewed either as a probabilistic model over a quantity we do not directly observe and which places its probability mass on unrealizable possibilities, or as an unnormalized model over the quantity we (treat as if we) do observe.

The standard speech parameter generation algorithm optimizes a utility function, and so is not explicitly probabilistic, but it may be interpreted as maximum probability speech parameter generation in a probabilistic model. For example we may treat the quadratic function $A(c)$ optimized by the standard speech parameter generation algorithm as the log probability density function, up to a constant, of a Gaussian distribution over c. This Gaussian has natural parameters b and P from (3.29) and (3.30). The probabilistic model $\mathbb{P}(c \,|\, \theta, \lambda)$ so obtained is the same as that obtained by making (3.36) a valid probability distribution by adding the appropriate normalization constant. It is known as the *trajectory HMM acoustic model*, and is described more explicitly below. While the above is perhaps the most natural way to interpret standard speech parameter generation as maximum probability speech parameter generation in some probabilistic model, it is not the only

way. For example if we apply an overall scale factor $k > 0$ to both b and P then the optimum of the utility function $A(c)$ is in the same place, and so the standard speech parameter generation algorithm may also be interpreted as maximum probability speech parameter generation in the probabilistic model where the Gaussian distribution over c has natural parameters kb and kP. Indeed strictly speaking standard speech parameter generation may be regarded as maximum probability speech parameter generation in any probabilistic model $\mathbb{P}(c \,|\, \theta, \lambda)$ which has a maximum at $P^{-1}b$. Because the trajectory HMM acoustic model seems in some sense the most natural of these alternatives, we refer to it as the probabilistic model *effectively* used by standard speech parameter generation. Since standard speech parameter generation and parameter generation considering global variance are the two most commonly used speech parameter generation methods, and both are based on $A(c)$, we also refer to the trajectory HMM acoustic model as the probabilistic model effectively used at synthesis time by the standard framework.

The fact that the acoustic model used during training is defined over the wrong quantity, or alternatively is defined over the correct quantity but unnormalized, means that strictly speaking the probabilistic justification for the training process described in §3.2.3 does not apply, i.e. there is no guarantee that the probabilistically-motivated training procedure will do anything sensible from the point of view of how we use the model for parameter generation. However it is an interesting question to what extent the standard training procedure does in fact still do something useful for parameter generation. The practical consequences of the lack of normalization in the standard training procedure will be examined further in §5.4.1. As we will see it gets some things right, e.g. the mean trajectory it learns is reasonable, and some things wrong, e.g. the model is greatly over-confident, with much too small predictive variance.

3.3.2 Trajectory HMM

The *trajectory HMM* (Zen et al., 2007b) takes the probabilistic model effectively used at synthesis time by the standard framework and uses it at both training time and synthesis time. It is thus a consistent and normalized model, and has been shown to yield more natural synthesized speech than the standard framework (Zen et al., 2007b). In this section we outline how the trajectory HMM may be used for statistical parametric speech synthesis.

As discussed in §3.3.1, the *trajectory HMM acoustic model* is a trajectory-level acoustic model obtained by normalizing (3.36). For clarity we write out the model explicitly. Given a state sequence $\theta = [\theta_t]_{t=1}^T$ we say a trajectory $c = [c_t]_{t=1}^T$ is distributed according to the

trajectory HMM acoustic model with parameters $\lambda = ([b_{qd}, \tau_{qd}]_{q,d}, [w_{dk}]_{d,k}, \overline{q})$ if

$$c \,|\, \theta, \lambda \sim \mathcal{N}_{\mathrm{NP}}(b, P) \tag{3.38}$$

$$\text{where } b = \sum_d W_d{}^{\mathsf{T}} b_d^{\mathrm{win}} \tag{3.39}$$

$$P = \sum_d W_d{}^{\mathsf{T}} P_d^{\mathrm{win}} W_d \tag{3.40}$$

$$b_d^{\mathrm{win}} = \left[b_{\overline{q}(\theta_t)d}\right]_{t=1}^{T} \tag{3.41}$$

$$P_d^{\mathrm{win}} = \mathrm{diag}\left(\left[\tau_{\overline{q}(\theta_t)d}\right]_{t=1}^{T}\right) \tag{3.42}$$

$$(W_d)_{st} = w_{d(t-s)} \tag{3.43}$$

The model parameters λ consist of the b-value parameter b_{qd} and precision parameter τ_{qd} for each cluster $q \in \{1, \ldots, Q\}$ and window $d \in \{0, \ldots, D-1\}$, the window coefficients $[w_{dk}]_{d,k}$, and a function $\overline{q}$ specifying a partition of state space. Note that while the model parameters are the same as for the standard framework (for a given portion p and vector component i), they do not have the same meaning, e.g. τ_{qd} is not the inverse variance of o_{td} whenever $\theta_t = q$. We typically restrict the precision parameter τ_{qd} to be non-negative. Some authors further restrict the precision parameter to have a finite minimum and maximum possible value (Zhang, 2009). For simplicity we have assumed zero-input windows, but it is possible to use any type of window with the trajectory HMM acoustic model by replacing (3.43) with the appropriate definition of the window matrix W_d. The log pdf of the trajectory HMM acoustic model is, up to a constant, given by $A(c) = -\frac{1}{2}c^{\mathsf{T}}Pc + b^{\mathsf{T}}c$, and the log normalization constant which makes this a valid distribution is given by (2.30). Since the normalization constant depends ultimately on θ and λ, we often write it as $Z(\theta, \lambda)$. Due to the $\log\det P$ and $b^{\mathsf{T}}P^{-1}b$ terms in (2.30), the way that $\log Z(\theta, \lambda)$ depends on the state sequence θ and the parameters λ is complicated.

In case it is helpful to the reader, a *factor graph* (Bishop, 2006) which describes the trajectory HMM acoustic model is shown in Figure 3.9. In factor graphs such as Figure 3.9, each large circle represents a random variable and is shaded if the variables is observed or unshaded if it is latent. Each square represents a multiplicative factor in the density function describing the probability distribution over all random variables. Each factor is a function of some subset of the random variables as indicated by the edges emanating from it. A small black circle represents variables which are conditioned on. Conventionally a factor graph has no conditioned-on variables and any global normalization factor is not shown. We have shown Z in Figure 3.9 for clarity since there are conditioned-on variables.

The *trajectory HMM* is the combination of a Markovian state transition model $\mathbb{P}(\theta \,|\, l, \nu)$, for example the same Gaussian state transition model used by the standard HMM synthesis framework, with an acoustic model $\mathbb{P}(C \,|\, \theta, \lambda)$ obtained by using the trajectory

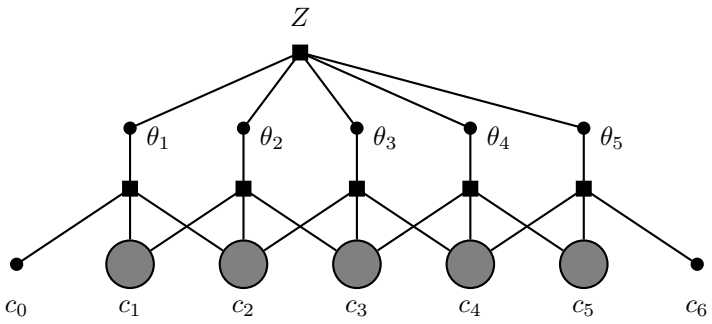


Figure 3.9: A factor graph which describes the trajectory HMM acoustic model with left extent $K^{\mathrm{L}} = 1$ and right extent $K^{\mathrm{R}} = 1$. Here the state sequence $\theta = \theta_{1:5}$ is conditioned on and the trajectory $c = c_{1:5}$ is observed. The values c_0 and c_6 are deterministic acoustic context, equal to zero when using zero-input windows. The factor Z is a normalization constant. Each factor also depends on the model parameters λ (not shown).

HMM acoustic model for $[C_{tpi}]_{t=1}^{T}$ for each portion p and vector component i. Thus the parameters of the acoustic model are the same as for the acoustic model used during training by the standard HMM synthesis framework, but it is defined over the speech parameter sequence C instead of the observation sequence O. Note that the despite the fact we use the term trajectory HMM for simplicity, it is not in fact an HMM, due to the non-factorized dependence of $\mathbb{P}(C \,|\, \theta, \lambda)$ on the state sequence θ.

Parameter estimation for the trajectory HMM is more complicated than for the standard framework for two reasons. Firstly the posterior distribution $\mathbb{P}(\theta \,|\, C, \nu, \lambda)$ over the state sequence θ does not factorize across time as a product of the form $\prod_t f_t(\theta_{t-1}, \theta_t)$ as $\mathbb{P}(\theta \,|\, O, \nu, \lambda)$ did for the standard framework. This is due to the normalization constant $Z(\theta, \lambda)$, which has a complicated non-factorized dependence on the state sequence. This means that the conventional forward-backward and Viterbi algorithms are not applicable to the trajectory HMM. In practice a plausible state sequence θ for each utterance in the training corpus is typically recovered using a delayed-decision Viterbi algorithm (Zen et al., 2007b) or using the conventional Viterbi algorithm with a standard framework system. Parameter estimation given a fixed state sequence is also more complicated for the trajectory HMM acoustic model than in the standard framework, and again this is because of the normalization constant $Z(\theta, \lambda)$, which depends on the parameters λ in a complicated way. An analytic solution exists for computing the maximum likelihood b-value parameters $[b_{qd}]_{q,d}$ given the precision parameters $[\tau_{qd}]_{q,d}$ (Zen et al., 2007b). This is usually framed in terms of computing the mean parameters $[b_{qd}/\tau_{qd}]_{q,d}$, which is an equivalent problem. The precision parameters $[\tau_{qd}]_{q,d}$ are typically optimized using a gradient ascent method such as *L-BFGS* or the *truncated Newton method* (Nocedal and Wright, 2006). The analytic solution for the

b-value parameters involves constructing and inverting a near-full $(QD) \times (QD)$ matrix, and so for models with many parameters gradient ascent is typically used for both sets of parameters (Zen et al., 2007b). Computing the gradient of the tricky terms $\log\det P$ and $b^\mathsf{T}P^{-1}b$ with respect to the model parameters may be done efficiently by exploiting the banded structure of P, allowing the overall gradient to be computed in $O(T)$ time.

We wish to highlight one aspect of the gradient computation which we believe may not be widely appreciated. Computing the gradient of the $\log\det P$ term involves computing the band of the inverse of the banded matrix P. This can be done by first computing the inverse and then extracting the band, but this is an expensive operation, and there are a number of $O(T)$ approaches. The approach we favour is using the Cholesky decomposition of P to find an equivalent composite linear Gaussian autoregressive distribution as described in §2.2.5, then using the simple recursions for the covariance of a composite linear Gaussian distribution presented by Bishop (2006, section 8.1.4). Hutchinson and de Hoog (1985, section 3) give explicit recursions for this approach, provide links to other approaches for solving this problem, and mention how this approach is related to a more general approach for computing what are sometimes called *sparse inverses* due to Takahashi. The band of the inverse of P can also be computed by formulating the problem as a Kalman smoothing problem. The required submatrices along the band of the covariance matrix are then computed as part of many smoothing algorithms, for example Rauch-Tung-Striebel smoothing (Rauch et al., 1965).

We have seen that the normalization constant for the trajectory HMM is the source of both of the difficulties involved in training. Having a difficult normalization constant is a general feature of *undirected graphical models* (Bishop, 2006), which define a probability distribution by globally normalizing a product of potential functions. The trajectory HMM is not an undirected graphical model, since the normalization is not global but rather conditional on θ, but the normalization constant is complicated for the same reason: we are normalizing a product of potential functions. In contrast, directed graphical models, which are locally normalized, interact better with parameter estimation methods like expectation maximization. We will see an example of a directed graphical model with simple and tractable parameter estimation in Chapter 4.

For speech parameter generation, the trajectory HMM can re-use the approaches described above for the standard framework. We can do maximum probability speech parameter generation for the trajectory HMM acoustic model by simply using standard speech parameter generation with the appropriate values of b and P. The most likely trajectory is the mean trajectory $\mu = P^{-1}b$. Speech parameter generation considering global variance can also be used with the trajectory HMM by using the appropriate values of b and P in the definition of $A(c)$ in (3.33).

There is a potential subtlety in applying parameter generation considering global vari-

ance to normalized models such as the trajectory HMM. This discussion also applies to the LGLAR HMM, which will be defined in Chapter 4. This turns out not to matter, but here we explain the potential problem and why it is not an issue in practice. For parameter generation considering global variance the GV weight ω^{GV} in (3.33) needs to be chosen, and prima facie the conventional setting $\omega^{\mathrm{GV}} = DT$ may not be appropriate for the trajectory HMM or LGLAR HMM. Indeed since the standard HMM synthesis framework underestimates predictive variance by around a factor of D (see §5.4.1), the "strength" or "dynamic range" of the first term in (3.33) is around D times smaller when using the trajectory HMM and LGLAR HMM than when using the standard HMM synthesis framework. It might therefore be thought necessary to reduce the dynamic range of the second term in (3.33) by using $\omega^{\mathrm{GV}} = T$ when using the trajectory HMM and LGLAR HMM. However, as mentioned in §3.2.6, in practice the effect of using the conventional setting $\omega^{\mathrm{GV}} = DT$ with the standard HMM synthesis framework is to restrict the GV of generated trajectories to be very close to the mean μ^{GV} of the GV pdf; that is, the second term already has much greater strength than the first term. Using the conventional setting $\omega^{\mathrm{GV}} = DT$ with the LGLAR HMM and trajectory HMM does further increase the relative strength of the second term, but the result is the same: generated trajectories have GV very close to μ^{GV}. Thus the conventional approach to setting ω^{GV} is appropriate for normalized models such as the trajectory HMM and LGLAR HMM.

It is sometimes helpful to consider the trajectory HMM acoustic model as a *product of experts* (Hinton, 2002; Zen et al., 2012). A product-of-experts model over a set $\mathcal{Y}$ is defined by a collection $[F_k]_{k=1}^K$ of *experts*, where each expert F_k is a parameterized positive function and the value $F_k(y; \lambda_k)$ of expert k at $y \in \mathcal{Y}$ depends on the parameters λ_k of that expert. The experts are combined to form a probabilistic model by taking their product and normalizing, that is

$$\mathbb{P}(y \,|\, \lambda) = \frac{1}{Z(\lambda)} \prod_k F_k(y; \lambda_k) \tag{3.44}$$

where $\lambda = [\lambda_k]_k$. Often F_k depends only on a low-dimensional projection of y, and thus can be seen as representing a soft constraint on some aspect of y. The concept of a product of experts is related to the concept of an exponential family but is more general, since taking $F_k(y; \lambda_k) = \lambda_k f_k(y)$ recovers an exponential family. The trajectory HMM acoustic model is a conditional product-of-experts model, where we normalize the product of a set of parameterized functions of the trajectory c and the state sequence θ, conditional on θ. Making up each distribution $\mathbb{P}(c \,|\, \theta, \lambda)$ we have DT experts, one for each time t and window d, and each expert is a log quadratic function of some low-dimensional projection of the trajectory c. Note that each factor in Figure 3.9 corresponds to a collection of D experts. The soft constraints these experts provide in the product-of-experts view of the trajectory HMM acoustic model are precisely the soft constraints discussed in §3.2.4 in the context of

standard speech parameter generation.

We mention in passing that the trajectory HMM is not the only valid probabilistic model that bears a similarity to the standard HMM synthesis framework. It is also possible to obtain a valid probabilistic model by normalizing at the level of the joint distribution $\mathbb{P}(C, \theta \,|\, l, \nu, \lambda)$, instead of normalizing at the level of the conditional distribution $\mathbb{P}(C \,|\, \theta, \lambda)$ as we did for the trajectory HMM. This model supports efficient alignment, for example using the Viterbi algorithm, but parameter estimation is even more complicated than for the trajectory HMM. For speech parameter generation given a fixed state sequence, the standard speech parameter generation algorithm can be used. This model also supports an expectation maximization-based parameter generation algorithm which iteratively maximizes the marginal $\mathbb{P}(C \,|\, \nu, \lambda)$ (Tokuda et al., 2000, case 3). A mild generalization of the jointly normalized model is a conditional exponential family, and this model was derived within a conditional maximum entropy framework and applied to speech recognition by van Horn (2002).

3.3.3 Trajectory HMM acoustic model as a conditional exponential family

The trajectory HMM acoustic model is a conditional exponential family. In this section we specify this conditional exponential family using the terminology of §2.2.3 and discuss some of the consequences of this perspective. This view provides a simple way to derive several facts about the trajectory HMM acoustic model which we believe are not widely appreciated: the conventional sufficient statistics used in the standard framework are also sufficient for the trajectory HMM acoustic model when combined with the state sequence; the log likelihood function is concave in a judiciously chosen parameterization and so has no non-global local maxima in any parameterization; the trajectory HMM acoustic model has a strong statistics-matching property; and the mean trajectory generated using a trajectory HMM acoustic model also has a statistics-matching property.

The trajectory HMM acoustic model is a conditional exponential family with natural parameters $[b_{qd}, \tau_{qd}]_{q,d}$. The feature function corresponding to the b-value parameter b_{qd} is

$$f_{qd}(\theta, c) = \sum_{t=1}^{T} \mathbb{I}\{\overline{q}(\theta_t) = q\}(W_d c)_t \tag{3.45}$$

The feature function corresponding to the precision parameter τ_{qd} is

$$g_{qd}(\theta, c) = -\tfrac{1}{2} \sum_{t=1}^{T} \mathbb{I}\{\overline{q}(\theta_t) = q\}(W_d c)_t^2 \tag{3.46}$$

These feature functions take the form of a window-specific and leaf-specific sum of trajectory values and sum of squared trajectory values. A few more quantities need to be specified to

properly define the conditional exponential family. The set in which c takes values ($\mathcal{Y}$ in §2.2.3) is the set of all finite-length real-valued sequences. The set in which θ takes values ($\mathcal{X}$ in §2.2.3) is the set of all finite-length Ψ-valued sequences, where Ψ is the state space. We can set $f_{qd}(\theta, c)$ and $g_{qd}(\theta, c)$ to $-\infty$ if θ and c are not the same length as a way to ensure that the model only assigns positive probability to trajectories of the same length as the state sequence. As mentioned above, the set of allowed model parameters (Ξ in §2.2.3) is typically restricted to those model parameters for which all the precision parameters are non-negative. This set is convex as required for a conditional exponential family.

We note in passing that forcing the precision parameters to be non-negative may be an unnecessarily severe restriction. It is certainly true that for a given corpus there are typically model parameters with some negative precision parameters which nevertheless give rise to a positive definite P for all utterances in the corpus, but we do not know whether it is possible to enlarge the set of allowed parameters while ensuring P is positive definite for all, or some useful proper subset of all, state sequences.

The fact that the trajectory HMM acoustic model is a conditional exponential family has two immediate consequences. Firstly the values (3.22), (3.23) and (3.24) (with O obtained by applying the windows to C in the usual way) are no longer sufficient statistics on their own, but are sufficient statistics together with the state sequence θ. This may conceivably be used to simplify code implementing trajectory HMM training. Exploiting this fact may also give a small computational speed-up, though the cost of the gradient evaluations needed for training is likely to be dominated by the cost of computing the Cholesky decomposition rather than the cost of computing the term involving the data. Secondly the log likelihood is concave in the parameters $[b_{qd}, \tau_{qd}]_{q,d}$. It is conceivable that this concavity may be useful in performing parameter estimation, for example by allowing the use of tools from convex optimization. Note that the concavity result depends on using the natural parameterization. The concavity result also implies that, regardless of the parameterization used, there are no local, non-global maxima in the log likelihood function for the trajectory HMM acoustic model, and so we do not have to worry about training getting stuck in such maxima.

The trajectory HMM acoustic model has a fairly strong statistics-matching property not shared by the standard framework. Suppose a trajectory HMM acoustic model is trained using maximum likelihood, and the estimated parameters lie in the interior of the space Ξ of allowed parameters. The general discussion of statistics-matching for conditional exponential families in §2.2.3 implies that the expected values of the statistics (3.22) and (3.23), given the training corpus state sequences, will be equal to the values of these statistics actually observed on the training corpus. Considering the whole training corpus as a single

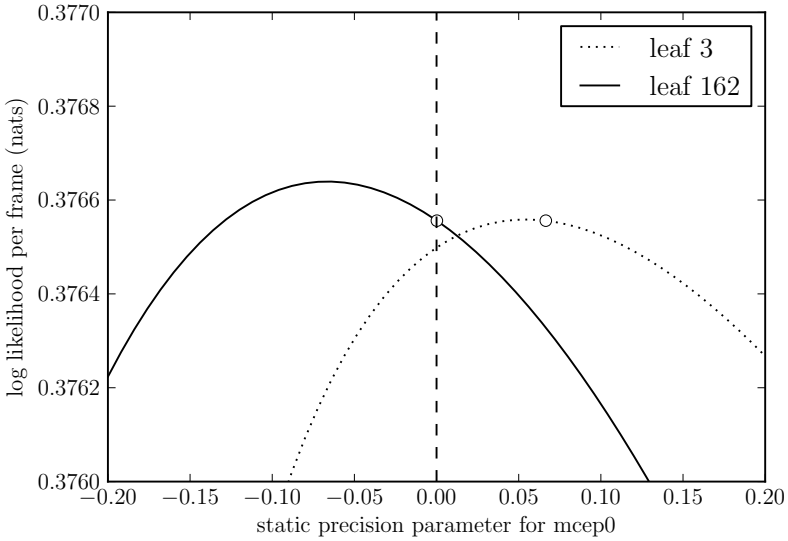


Figure 3.10: An example showing how the statistics-matching property can fail for certain leaves. The log likelihood is shown as a function of the static precision parameter τ_{q0} for two leaves for a trained trajectory HMM acoustic model. The maximum likelihood value subject to the constraint $\tau_{q0} \geq 0$ is indicated by a circle. The statistics-matching property (3.48) holds for leaf q if and only if the derivative with respect to τ_{q0} is zero. This is the case for leaf 3 but not for leaf 162.

long utterance with state sequence θ° and trajectory c°, we have

$$\mathbb{E} \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c)_t = \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c^\circ)_t \tag{3.47}$$

$$\mathbb{E} \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c)_t^2 = \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c^\circ)_t^2 \tag{3.48}$$

where the expectation is taken assuming the trajectory c is sampled from $\mathbb{P}(c \,|\, \theta = \theta^\circ, \lambda)$. Thus, for a trajectory HMM acoustic model trained using maximum likelihood, if we take the training corpus state sequence θ° and sample a corresponding trajectory c from the trained model, the expected values of the leaf-specific and window-specific first and second moments of c are equal to the corresponding moments of the actual trajectory c°. The above property only holds precisely on the training corpus but, assuming the training corpus state sequences and speech parameter sequences are reasonably representative of typical state sequences and speech parameter sequences, we might expect this property to hold approximately on the test corpus too. This viewpoint helps to make clear the type of statistical behaviour that is captured by the trajectory HMM acoustic model.

The statistics-matching property (3.48) may fail for certain leaves and windows. To simplify the discussion here we assume that the likelihood function has a local maximizer,

and so a global maximizer, though this is not always the case. Since the only constraints on the parameters are $\tau_{qd} \geq 0$ for all q and d, we know that at a local maximum the partial derivative of the likelihood function with respect to b_{qd} must be zero, and the partial derivative with respect to τ_{qd} must be zero if $\tau_{qd} > 0$ and at most zero if $\tau_{qd} = 0$. This together with (2.45) implies that at a local maximum (3.47) will always hold, but that the equality in (3.48) may sometimes become an inequality

$$\mathbb{E} \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c)_t^2 < \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c^\circ)_t^2 \tag{3.49}$$

if $\tau_{qd} = 0$. An example of such a failure is shown in Figure 3.10.

The statistics-matching property described above is expressed in terms of sampled trajectories, but it also tells us something about the mean trajectory. Since expectation is linear and $\mathbb{E}x^2 \geq (\mathbb{E}x)^2$ for any random variable x, we have

$$\sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d \mu)_t = \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c^\circ)_t \tag{3.50}$$

$$\sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d \mu)_t^2 \leq \sum_{t:\overline{q}(\theta_t^\circ)=q} (W_d c^\circ)_t^2 \tag{3.51}$$

where $\mu = \mathbb{E}c$ is the mean trajectory for state sequence θ°. These equations always hold at an exact local optimum in the likelihood function, even if $\tau_{qd} = 0$ for some leaves and windows. Thus the mean trajectory generated from the training corpus state sequence has the same leaf-specific and window-specific first moments as the training corpus trajectory, but typically has smaller second moments.

3.3.4 Sampling parameter generation

The use of a valid probabilistic model allows a form of speech parameter generation which was not possible for the unnormalized standard framework: given a trajectory-level acoustic model $\mathbb{P}(c \,|\, \theta, \lambda)$, we can generate a trajectory c by sampling from the distribution over c. We refer to this as *sampling parameter generation*. In this section we describe how this form of speech parameter generation may be implemented efficiently for models with banded precision matrix such as the trajectory HMM acoustic model. We also briefly discuss some of the advantages and disadvantages of sampling parameter generation compared to other forms of speech parameter generation.

Sampling parameter generation involves little extra work over standard parameter generation. We saw in §3.2.4 that it is possible to compute the mean trajectory $\mu = P^{-1}b$ of a Gaussian distribution with natural parameters b and P by computing $L^{-\mathsf{T}}L^{-1}b$ where L is the conventional Cholesky factor of P. To sample a trajectory from this distribution, we can first sample a T-dimensional vector z from $\mathcal{N}(0, I)$ and then compute $L^{-\mathsf{T}}(L^{-1}b + z)$.

Thus sampling a trajectory has the same computational complexity as computing the mean trajectory. The trajectory HMM acoustic model $\mathbb{P}(c \mid \theta, \lambda)$ is a Gaussian distribution with b-value b and banded precision matrix P, and so we can sample a trajectory efficiently using the above procedure. If we independently sample a trajectory $[C_{tpi}]_{t=1}^{T}$ for each portion p and vector component i, then we obtain a sampled speech parameter sequence C from the overall acoustic model $\mathbb{P}(C \mid \theta, \lambda)$ used by the trajectory HMM.

We note in passing that it is also possible to perform duration generation using sampling. For typical duration models we can sample a sequence d of durations by sampling the duration d_{js} of each (label, sublabel) pair (l_j, s) independently. If we sample from $\mathbb{P}(\theta \mid l, \nu)$ by sampling d in this way, convert this to a state sequence θ, then sample a speech parameter sequence C given this state sequence, we obtain a sample from the marginal distribution $\mathbb{P}(C \mid l, \nu, \lambda)$. Sampling interacts well, and extremely tractably, with this type of marginalization.

Doing generation by sampling is an attractive idea from a theoretical perspective. As mentioned in §2.1, maximum likelihood estimation assumes that the training corpus was generated by sampling. In the case of statistical parametric speech synthesis we assume the human speaker generated the speech parameter sequence C for each utterance by sampling from some true distribution $\mathbb{P}(C \mid l)$ given the label sequence l. Thus doing generation by sampling is more consistent with the assumption that our goal is to mimic the original human speaker than doing generation using the approaches presented previously. We will discuss some additional desirable properties of sampling parameter generation in §6.5.

Sampling parameter generation is also potentially a useful diagnostic tool. Listening to what the model expects to hear can shed light on the deficiencies of the model and suggest ways to improve it. In practice sampling trajectories from current normalized models such as the trajectory HMM results in poor synthesized speech with a characteristic warbling effect (Shannon et al., 2011). This points to a major deficiency in the trajectory HMM acoustic model, or rather in the speech parameter-level acoustic model obtained by using a separate trajectory HMM acoustic model for each portion and vector component, but we do not investigate this further in this thesis.

3.4 Evaluation for statistical speech synthesis

Speech synthesis systems can be evaluated in a variety of ways. A *subjective evaluation* assesses speech synthesis systems based on the preferences of human listeners, and these preferences are typically obtained by explicitly eliciting judgements about some aspect of the quality of the synthesized speech. An *objective evaluation* assesses speech synthesis systems using an automated metric. Subjective evaluations are typically the gold standard, but objective evaluations can be useful for making a large number of comparisons, building

systems, and investigating the strengths and weaknesses of different systems.

For the subjective evaluations in this thesis we use naturalness as judged by human opinion scores following the methodology of the Blizzard Challenge (Black and Tokuda, 2005). A naturalness opinion score is a judgement of how natural a given segment of audio is on a Likert scale from 1 (completely unnatural) to 5 (completely natural). Human judgements for the systems to be compared are elicited as part of a *listening test*. Each listener is presented with the audio for one utterance at a time, and the order of utterances is fixed but the system which a listener hears for a given utterance is based on the group that listener is assigned to when agreeing to take part in the listening test. The order of systems presented to the different listener groups is typically determined using a Latin square design (Bennett and Black, 2006). We used native English-speaking listeners (though not necessarily British) for all the subjective evaluations reported in this thesis.

In the remainder of this section we describe the two metrics we use for objective evaluation: test set log probability (TSLP), as defined in §2.1.4, and *mel cepstral distortion (MCD)*. The two metrics provide complementary views of a model. TSLP is a natural measure of how well a model predicts unseen frames, and achieving a high TSLP requires a model to have both accurate mean trajectories and accurate trajectory covariances. It also allows us to detect overfitting. MCD, which depends only on the mean trajectories, provides useful information about the accuracy of the mean trajectories independent of the trajectory covariances.

3.4.1 Test set log probability

Test set log probability (TSLP) is the log probability a trained model assigns to an unseen test corpus, as defined in §2.1.4. Since the statistical model $\mathbb{P}(C \,|\, l, \nu, \lambda)$ used in statistical parametric speech synthesis is a sequential model, we follow the notational convention described in §2.5 and assume the test corpus consists of a single utterance. The TSLP is given by

$$\log \mathbb{P}(C^* \,|\, l^*, \nu, \lambda) \tag{3.52}$$

where C^* is the test corpus speech parameter sequence, l^* is the test corpus label sequence, and ν and λ are collections of model parameters that have been estimated from the training corpus, typically using maximum likelihood estimation. We quote TSLP values in nats per frame.

3.4.2 Mel cepstral distortion

Mel cepstral distortion (MCD) (Kubichek, 1993) is a measure of the difference between a synthesized mel cepstral sequence and the corresponding natural mel cepstral sequence. We

use a form of MCD based on *dynamic time warping (DTW)*. Define

$$\mathrm{MCD}(C^{\mathrm{nat}}, C^{\mathrm{synth}}) = \frac{1}{T^{\mathrm{nat}}} \frac{10}{\log 10} \min_{\pi \in \Pi} \sum_{(s,t)\in\pi} \left(2 \sum_{i=1}^{I_1-1} \left(C^{\mathrm{nat}}_{t1i} - C^{\mathrm{synth}}_{t1i} \right)^2 \right)^{0.5} \tag{3.53}$$

where $C^{\mathrm{nat}} = [C^{\mathrm{nat}}_{tpi}]_{t,p,i}$ and $C^{\mathrm{synth}} = [C^{\mathrm{synth}}_{tpi}]_{t,p,i}$ are the natural and synthesized speech parameter sequences, $[C_{t1i}]_{i=0}^{I_1-1}$ is the spectral portion of the speech parameter sequence C at frame t, T^{nat} is the number of frames in the natural speech parameter sequence, $\pi \subset \mathbb{N}\times\mathbb{N}$ is a relation between frames in the natural and the synthesized speech parameter sequences, and Π is the set of admissible relations. A relation $\pi \subset \mathbb{N} \times \mathbb{N}$ is *admissible* if there is a sequence $[s_p, t_p]_{p=1}^{P}$ such that $(s_1, t_1) = (1, 1)$, $(s_P, t_P) = (T^{\mathrm{nat}}, T^{\mathrm{synth}})$, $(s_{p+1} - s_p, t_{p+1} - t_p)$ is either $(0, 1)$, $(1, 0)$ or $(1, 1)$ for $p = 1, \ldots, P - 1$, and the sets $\{(s_p, t_p) : p\}$ and π are equal. The minimum over admissible relations may be computed efficiently using dynamic programming. The unit of the right side of (3.53) is conventionally considered to be the *decibel (dB)*. Note that it is standard to omit 0^{th} cepstral component from the MCD computation as is done in (3.53) (Kubichek, 1993). Given a test corpus (l^*, C^*), we compute the MCD score for a given trained model by taking C^{nat} to be the test corpus speech parameter sequence C^* and taking C^{synth} to be the speech parameter sequence obtained by applying standard speech parameter generation to the test corpus label sequence l^*.

3.5 Experimental set-up used in this thesis

In this section we discuss aspects of experimental set-up that are shared by many of the experiments in the remainder of the thesis. The LGLAR HMM, which is used for some of the systems below, will be described in Chapter 4.

The systems were trained on the CMU ARCTIC corpus (Kominek and Black, 2003) for the single speaker "slt" (approximately 1 hour), with 50 held-out utterances.

The original waveforms had a sampling frequency of 16 kHz. The spectral portion of the speech parameters consisted of 40-dimensional mel cepstra (mcep) (Fukada et al., 1992) with frequency warping factor $\alpha = 0.42$, the fundamental frequency portion of the speech parameters consisted of $\log F_0$, and the aperiodicity portion of the speech parameters consisted of 5-band aperiodicity (Zen et al., 2007a). We used STRAIGHT vocoding (Kawahara et al., 1999). A frame shift of 5 ms was used, and F_0 was estimated using STRAIGHT (min 80 Hz, max 350 Hz).

All types of system used a 5-sublabel (by default) left-to-right topology for modelling at the phoneme level, with Gaussian explicit duration models for each sublabel used during both parameter estimation and synthesis (Zen et al., 2007c). We made a minor modification to HTS to ensure explicit duration distributions are properly normalized wherever they are

used, though the M step re-estimation equations were not modified. In the standard version of HTS these distributions are not fully normalized due to the fact a Gaussian pdf is being used for a random variable with range the positive integers.

For the autoregressive systems the spectral and aperiodicity portions of the speech parameters were modelled using the LGLAR HMM, with a depth of 3 and an MDL tuning factor of 0.3 by default. These values were chosen based on preliminary experiments and the effect of these choices is investigated in detail in §4.4. For the standard system these portions of the speech parameters were modelled using the standard HMM with the conventional three windows (HTS working group, 2012), a single Gaussian with diagonal covariance per sublabel, and an MDL tuning factor of 1.0. For all systems the F_0 portion of the speech parameters was modelled using standard multi-space distributions (Tokuda et al., 2002a) with the conventional three windows (HTS working group, 2012) and an MDL tuning factor of 1.0. This means that even the autoregressive systems suffer from some inconsistency between training and synthesis since the F_0 portion of the speech parameters is still modelled using the inconsistent standard approach. It is possible to model F_0 using the autoregressive HMM. Perhaps the most natural approach would be to use a continuous F_0 model such as that used by Yu (2011), though more complicated approaches would also fit naturally within the autoregressive framework. Investigation of these further departures from the standard approach is left for future work.

The standard and autoregressive systems were built using HTS 2.1 (HTS working group, 2012). The training regime was adapted from the HTS speaker-dependent training demo (HTS working group, 2012), with initialization of a monophone model based on initial phone-level alignments derived from a monophone speech recognition-style system followed by expectation maximization on the monophone model, decision tree clustering, expectation maximization on the resulting full-context model, another round of decision tree clustering, and further expectation maximization. For the standard systems, the band aperiodicity portion of the speech parameters was ignored during the E step of expectation maximization following standard practice of the HTS demo, whereas for the autoregressive systems all portions were taken into account during the E step.

The trajectory HMM system took the trained standard system as a starting point, and re-estimated the spectral leaf parameters based on a fixed alignment. The trajectory HMM training was kindly performed by Heiga Zen, then at Toshiba Research Europe.

By default the synthesized trajectories for all systems were produced using parameter generation considering global variance (Toda and Tokuda, 2007).

3.6 Summary of contributions

While this chapter is mainly an overview of previous work, there are some novel contributions. Most notable is the realization that the trajectory HMM acoustic model is a conditional exponential family, implying that the log likelihood function is concave in a particular parameterization and has no non-global local maxima in any parameterization, and that both sampled trajectories and the mean trajectory match certain leaf-specific statistics (§3.3.3).

Minor novel contributions of this chapter include: the realization that the discrete Gaussian could be used for duration modelling and could re-use the sufficient statistics used currently (§3.1.4); an appreciation that maximum probability speech parameter generation is not even defined in the case of multi-space distributions (§3.2.7); the realization that it is possible to use algorithms which compute the band of the inverse of a banded matrix in order to compute the gradient of the log likelihood function for the trajectory HMM acoustic model in linear time (§3.3.2); an explanation of why the GV weight used during parameter generation considering global variance does not need to be adjusted for normalized models such as the trajectory HMM and LGLAR HMM (§3.3.2); the realization that the jointly normalized model considered by van Horn (2002) supports EM-based parameter generation (§3.3.2); and an appreciation that sampling speech parameter generation is a potentially useful diagnostic tool (§3.3.4).

Chapter 4

Autoregressive HMMs

We saw in Chapter 3 that the standard HMM synthesis framework is able to capture aspects of the dynamic information present in speech parameter sequences in a simple and tractable way. However this approach is inconsistent, and directly removing this inconsistency using the trajectory HMM complicates parameter estimation. In this chapter we will see how an extension of the conventional HMM known as the *autoregressive HMM* provides an alternative way to capture aspects of the dynamics of speech parameter sequences.

The layout of this chapter is as follows. We first describe the autoregressive HMM as a probabilistic model. We then discuss how to use a specific form of autoregressive HMM, the *linear Gaussian linear autoregressive HMM (LGLAR HMM)*, to model speech parameters. The LGLAR HMM uses the linear Gaussian linear regression model described in §2.2.4 as a component in a larger sequential model. We show that the LGLAR HMM: supports efficient parameter estimation using expectation maximization and decision tree clustering; supports existing high quality speech parameter generation methods such as parameter generation considering global variance; and supports a simple and exact time-recursive form of speech parameter generation that is not available for the standard HMM synthesis framework or the trajectory HMM and which may be used for low latency parameter generation. We also discuss a potential pathology with the LGLAR HMM whereby generated trajectories can diverge far outside the range of values which might be considered reasonable. The LGLAR HMM, like the standard HMM synthesis framework, has certain parameters such as MDL tuning factor that are set by the experimenter, and we investigate how to set the values of these *model structure parameters* experimentally. Finally we evaluate the LGLAR HMM for statistical parametric speech synthesis by comparing it to the standard HMM synthesis framework and the trajectory HMM in subjective and objective evaluations. We find that the LGLAR HMM is capable of producing speech that is as natural as that of the standard HMM synthesis framework with its conventional settings, but not as natural as the trajectory HMM.

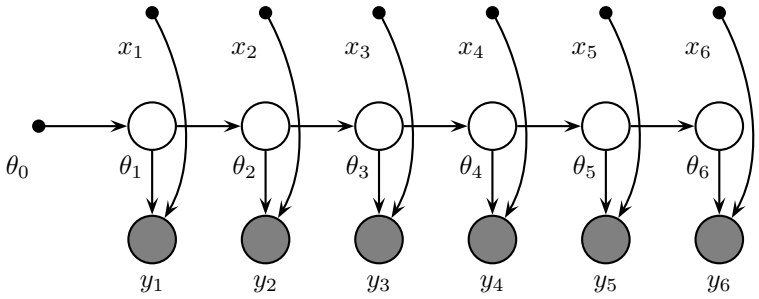


Figure 4.1: A directed graphical model for an input output HMM where the observed variables, but not the state variables, depend on the input. Here $x = x_{1:6}$ is the input sequence, $\theta = \theta_{1:6}$ is the state sequence, and $y = y_{1:6}$ is the sequence of observations. The value θ_0 is a deterministic initial state. Nodes also depend on the model parameters ν and λ (not shown). The model is a conditional model of the hidden θ_t values and the observed y_t values given the x_t values.

4.1 Autoregressive probabilistic modelling

In this section we describe the input output HMM and autoregressive HMM, which are both generalizations of the conventional HMM. The input output HMM serves as a useful conceptual intermediary between the conventional HMM and the autoregressive HMM since it has similarities to both models. This section reviews the basic properties of the autoregressive HMM as a probabilistic model. Its application to speech will be described in §4.3.

4.1.1 Input output HMMs

There is a slight generalization of the HMM where the sequence of observations depends on an *input sequence*. Consider a probabilistic model $\mathbb{P}(y, \theta \,|\, x, \nu, \lambda)$ over an observation sequence $y = [y_t]_{t=1}^T$ given an input sequence $x = [x_t]_{t=1}^T$ with a latent state sequence $\theta = [\theta_t]_{t=1}^T$ as for the HMM, where

$$\mathbb{P}(y, \theta \,|\, x, \nu, \lambda) = \prod_{t=1}^{T} \mathbb{P}(y_t \,|\, \theta_t, x_t, \lambda) \mathbb{P}(\theta_t \,|\, \theta_{t-1}, \nu) \tag{4.1}$$

As for the HMM we assume there is some deterministic *initial state* θ_0. Dependence on the input sequence x is what distinguishes the input output HMM from the conventional HMM given by (2.89). The graphical model (4.1) is also shown in Figure 4.1. The class of probabilistic models given by (4.1), as well as the more general class of models where the state also depends on the input, is known as the *input output HMM* (Bengio and Frasconi, 1995; Bishop, 2006). The Viterbi and forward-backward algorithms may be applied to the

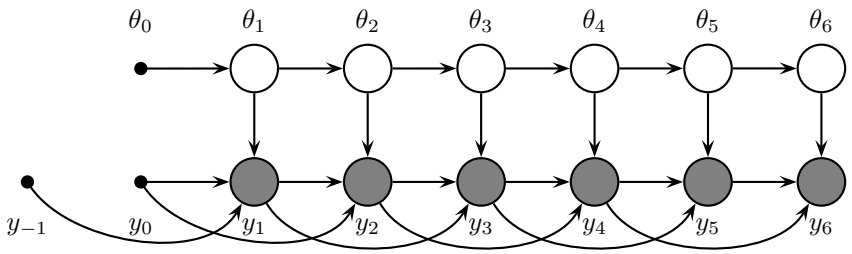


Figure 4.2: A directed graphical model for an autoregressive HMM of depth 2. Here $\theta = \theta_{1:6}$ is the state sequence and $y = y_{1:6}$ is the sequence of observations. The value θ_0 is a deterministic initial state, and the values y_{-1} and y_0 constitute a deterministic initial context for the observation sequence. Nodes also depend on the model parameters ν and λ (not shown). We consider θ_t hidden and y_t observed.

input output HMM since $\mathbb{P}(y, \theta \,|\, x, \nu, \lambda)$ factorizes with respect to the state sequence θ in the same way as for the conventional HMM.

A particular form of input output HMM is obtained by taking the distribution $\mathbb{P}(y_t \,|\, \theta_t, x_t, \lambda)$ to be a linear regression model (described in §2.2.4) with parameters depending on the cluster index $q = \overline{q}(\theta_t)$ of the current state θ_t, where $\overline{q}$ is a clustering function. Since the linear regression model is a conditionally additive exponential family it has sufficient statistics. This means that the above form of input output HMM supports tractable expectation maximization and decision tree clustering using the same approach used for the conventional HMM in §2.5.2 but with accumulators appropriate for the linear regression model instead of those appropriate for an exponential family. This model was perhaps first described by Goldfeld and Quandt (1973), with the first derivation of expectation maximization for parameter estimation given by Lindgren (1976, 1978).

4.1.2 Autoregressive HMMs

The *autoregressive HMM* (Wellekens, 1987; Bilmes, 2004; Bishop, 2006) extends the conventional HMM to explicitly take into account correlations over time in the observation sequence. It does this by using previous values of the observation sequence $y_{t-K:t-1}$ as the input x_t for an input output HMM. Thus the autoregressive HMM is a probabilistic model

$$\mathbb{P}(y, \theta \,|\, \nu, \lambda) = \prod_{t=1}^{T} \mathbb{P}(y_t \,|\, y_{t-K:t-1}, \theta_t, \lambda) \mathbb{P}(\theta_t \,|\, \theta_{t-1}, \nu) \tag{4.2}$$

where the non-negative integer K is referred to as the *order* or *depth* of the model, and we assume there is some deterministic *initial observation context* $y_{-(K-1):0}$. The graphical model (4.2) in the case $K = 2$ is also shown in Figure 4.2. If each factor $\mathbb{P}(y_t \,|\, y_{t-K:t-1}, \theta_t, \lambda)$ is linear Gaussian in y then we refer to the model as a *linear Gaussian autoregressive HMM*. Note that the conventional HMM is an autoregressive HMM with depth 0.

By using the observed values in the recent past as context for a regression model predicting the current observed value, the autoregressive HMM provides a simple way to turn a sequence modelling problem into a finite-dimensional regression problem. In principle it is very flexible since almost any regression model can be used for $\mathbb{P}(y_t \,|\, y_{t-K:t-1}, \theta_t, \lambda)$, though parameter estimation may be more complicated if this regression model does not have finite-dimensional sufficient statistics.

As in the case of the input output HMM, the Viterbi and forward-backward algorithms may be applied to the autoregressive HMM, since $\mathbb{P}(y, \theta \,|\, \nu, \lambda)$ factorizes with respect to the state sequence θ in the same way as for the conventional HMM. The existence of this factorization follows from the fact that the extra dependencies in the autoregressive HMM directed graphical model are on observed, not latent, variables.

In the case where we take the distribution $\mathbb{P}(y_t \,|\, \theta_t, x_t, \lambda)$ to be a linear regression model with parameters depending on the cluster index $q = \overline{q}(\theta_t)$, the autoregressive HMM has sufficient statistics and allows efficient parameter estimation using expectation maximization and decision tree clustering in precisely the same way as for the input output HMM. This model appears to have first been described explicitly in a technical report by Lindgren (1976), though Goldfeld and Quandt (1973) did consider a restricted form of autoregression. It was subsequently described to the automatic speech recognition community by Wellekens (1987) and Brown (1987), and to the econometrics community in an influential paper by Hamilton (1989). Lindgren (1976), Wellekens (1987) and Brown (1987) describe the use of expectation maximization for parameter estimation in this model.

The fixed initial context $y_{-(K-1):0}$, which is shared across all utterances, may in principle be learned from data using maximum likelihood. However for the experimental systems described in this thesis we set each value in the initial context either to zero or to the average of the first observed value in all utterances in the training corpus. Specifying this initial context is necessary to define $\mathbb{P}(y_t \,|\, y_{t-K:t-1}, \theta_t, \lambda)$ for $t \leq K$.

We note in passing that the restriction to finite depth K is not by itself essential for tractable inference and learning. For example it would be easy, and require very little extra computation, to do parameter estimation in a model where one of the basis functions used by the linear regression model was the average of all past observations in the current utterance, or the variance of these past values.

4.2 Autoregressive linear filters

In this section we describe some of the theory of time-invariant autoregressive linear filters. This theory will be related to the autoregressive HMM in §4.3.3 and §4.3.4.

A *time-invariant causal autoregressive linear filter* is a linear function which takes a real-valued input signal $[x_t]_{t=1}^T$ and produces a real-valued output signal $[y_t]_{t=1}^T$ according

to the difference equation

$$y_t = \sum_{k=1}^{K} a_k y_{t-k} + x_t \tag{4.3}$$

where $a = [a_k]_{k=1}^K$ is the set of autoregressive filter coefficients defining the filter, and it is assumed that the initial context $y_{-(K-1):0}$ is fixed. The assumption of initial or final conditions is needed to ensure that (4.3) has a unique solution, and our choice to use initial conditions means that the filter is causal (Oppenheim and Schafer, 1975).

If the input signal x is identically zero then it possible to analytically solve the difference equation (4.3). The *characteristic polynomial* for the autoregressive filter coefficients a is defined as

$$z^K - \sum_{k=1}^{K} a_k z^{K-k} \tag{4.4}$$

The characteristic polynomial has K complex roots $[r_k]_{k=1}^K$, where non-real roots occur in complex-conjugate pairs. Let R denote the magnitude of the largest-magnitude root, i.e. $R = \max_k |r_k|$. We say the autoregressive coefficients a are *stable* if $R < 1$, i.e. if all the roots of the characteristic polynomial have magnitude less than 1, and *unstable* otherwise. If the K complex roots are all distinct then the general solution of $y_t = \sum_{k=1}^K a_k y_{t-k}$ is given by

$$y_t = \sum_{k=1}^{K} I_k r_k^t \tag{4.5}$$

where r_k^t is the t^{th} power of r_k, and the complex-valued vector $I = [I_k]_{k=1}^K$ depends on the initial context $y_{-(K-1):0}$. The output y is thus a linear combination of exponential-times-sinusoids: for a real root r_k the contribution to the output y is proportional to $[r_k^t]_t$, and for a complex-conjugate pair (r_k, r_l) of roots the contribution to the output y is proportional to $[|r_k|^t \sin(\omega_k t + \varphi_k)]_t$ for some $\omega_k, \varphi_k \in \mathbb{R}$. If $|r_k| < 1$ then the corresponding exponential-times-sinusoid term converges to 0 as $t \to \infty$. If $|r_k| > 1$ then the corresponding exponential-times-sinusoid term diverges, i.e. fails to converge, as $t \to \infty$, since the exponential part of the term tends to ∞ while the sinusoid part remains bounded. We refer loosely to this type of divergence as *exponential divergence*. In this case $I_k r_k^t$ also diverges unless the initial conditions are such that I_k happens to be exactly 0. Thus the stability of the autoregressive coefficients a determines the typical behaviour for large t: if a is stable, i.e. $|r_k| < 1$ for all k, then it can be shown that $y_t \to 0$ as $t \to \infty$; if a is unstable then typically y_t diverges as $t \to \infty$. The limit $t \to \infty$ is never approached in practice since T is finite, but exponentially diverging trajectories may still lead to inappropriate trajectory values after a finite amount of time.

If the input x is constant instead of zero, then again y_t typically diverges exponentially if the autoregressive coefficients are unstable and converges to a finite value if the autore-

gressive coefficients are stable. If the input x is white noise, then again y_t typically diverges exponentially if the autoregressive coefficients are unstable, and it converges to a limiting distribution if the autoregressive coefficients are stable.

It is sometimes helpful to consider the *time constant* $\tau = 1/\log R$ associated with given autoregressive coefficients a. The time constant is measured in number of frames, and is negative for stable autoregressive coefficients. Since the behaviour of y_t for large t is dominated by terms involving roots of magnitude R, and $R^t = \exp(t/\tau)$, the time constant determines how many frames it typically takes for y_t to converge by a given ratio if a is stable or to diverge by a given ratio if a is unstable, in the limit of large t.

4.3 Autoregressive models for speech

In this section we describe how the autoregressive HMM may be used to model speech parameter sequences. The layout of this section is as follows. We first describe the *linear Gaussian linear autoregressive acoustic model (LGLAR acoustic model)*, which is a trajectory-level acoustic model based on the linear Gaussian linear regression model described in §2.2.4. We then describe an autoregressive HMM based on the LGLAR acoustic model, which we term the *linear Gaussian linear autoregressive HMM (LGLAR HMM)*, which can be used to model speech parameter sequences. We then describe speech parameter generation for the LGLAR acoustic model, and discuss a potential pathology with the LGLAR acoustic model which can in some cases lead to divergent trajectories. Finally we summarize previous work using the LGLAR HMM and related models for speech processing.

The only autoregressive HMM we consider in detail in this thesis is the LGLAR HMM. However we would like to note that, as discussed above, the autoregressive HMM approach is very flexible, and there are many possibilities for applying more general forms of autoregressive HMM to modelling speech. For example, instead of modelling $c_t \,|\, c_{t-K:t-1}, \theta_t$ using a linear Gaussian linear regression model as we do below, we could imagine using a Gaussian process, a hierarchical mixture of experts, or a neural net where the outputs are the parameters of a probabilistic model.

4.3.1 Linear Gaussian linear autoregressive acoustic model

In general we refer to a trajectory-level acoustic model $\mathbb{P}(c \,|\, \theta, \lambda)$ which satisfies $\mathbb{P}(c \,|\, \theta, \lambda) = \prod_t \mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda)$ as an *autoregressive acoustic model*. If $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda)$ is linear Gaussian in c then we refer to it as a *linear Gaussian autoregressive acoustic model*. The *linear Gaussian linear autoregressive acoustic model (LGLAR acoustic model)* is a simple linear Gaussian autoregressive acoustic model based on the linear Gaussian linear regression

model described in §2.2.4. The LGLAR acoustic model assumes

$$c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda \sim \mathcal{N}\left(\sum_{k=1}^{K} a_{qk} c_{t-k} + a_{q(K+1)}, \sigma_q^2\right), \quad t \in \{1, \ldots, T\} \tag{4.6}$$

$$\text{where } q = \overline{q}(\theta_t) \tag{4.7}$$

$$\lambda = \left(\left[[a_{qk}]_{k=1}^{K+1}, \sigma_q^2\right]_{q=1}^{Q}, \overline{q}\right) \tag{4.8}$$

where $\overline{q}$ is a clustering function, and $c_{-(K-1):0}$ is some deterministic *initial acoustic context*. The $[a_{qk}]_{k=1}^{K}$ are referred to as the *autoregressive coefficients*, and $a_{q(K+1)}$ is referred to as the *autoregressive bias*. If a trajectory $c = [c_t]_{t=1}^{T}$ given a state sequence $\theta = [\theta_t]_{t=1}^{T}$ is distributed according to the LGLAR acoustic model we write

$$c \,|\, \theta, \left[[a_{qk}]_{k=1}^{K+1}, \sigma_q^2\right]_{q=1}^{Q}, \overline{q} \sim \text{LGLAR}\left(\theta; \left[[a_{qk}]_{k=1}^{K+1}, \sigma_q^2\right]_{q=1}^{Q}, \overline{q}\right) \tag{4.9}$$

The name is chosen because it is a linear Gaussian autoregressive acoustic model with a linear regression model-style parameterization. In "linear Gaussian linear autoregressive", the first "linear" refers to the mean value in (4.6) being linear in the trajectory c, while the second "linear" refers to it being linear in the autoregressive coefficients a.

4.3.2 Linear Gaussian linear autoregressive HMM

If the LGLAR acoustic model is used for each portion p and vector component i of the speech parameters, and we use the Gaussian state transition model, then we obtain a complete statistical model $\mathbb{P}(C \,|\, l, \nu, \lambda)$ for the speech parameter sequence given the label sequence:

$$\theta \,|\, l, \nu, \overline{q}^{\text{DUR}} \sim \text{GSTM}\left(l; \left[\mu_q^{\text{DUR}}, (\sigma^2)_q^{\text{DUR}}\right]_{q=1}^{Q^{\text{DUR}}}, \overline{q}^{\text{DUR}}\right) \tag{4.10}$$

$$[C_{tpi}]_{t=1}^{T} \,|\, \theta, \lambda \sim \text{LGLAR}\left(\theta; \left[[a_{pqik}]_{k=1}^{K_p+1}, \sigma_{pqi}^2\right]_{q=1}^{Q_p}, \overline{q}_p\right), \quad \begin{cases} p \in \{1,2,3\} \\ i \in \{0, \ldots, I_p - 1\} \end{cases} \tag{4.11}$$

$$\text{where } \nu = \left(\left[\mu_q^{\text{DUR}}, (\sigma^2)_q^{\text{DUR}}\right]_q, \overline{q}^{\text{DUR}}\right) \tag{4.12}$$

$$\lambda = \left[\left[[a_{pqik}]_{k=1}^{K_p+1}, \sigma_{pqi}^2\right]_{q,i}, \overline{q}_p\right]_p \tag{4.13}$$

This model is a linear Gaussian autoregressive HMM, and we refer to it as the *linear Gaussian linear autoregressive HMM (LGLAR HMM)*.

The LGLAR HMM is an autoregressive HMM with linear regressive distributions $\mathbb{P}(C_t \,|\, C_{t-K:t-1}, \theta_t, \lambda)$, and so supports efficient parameter estimation using expectation maximization and decision tree clustering as described above. In general expectation maximization and decision tree clustering are conceptually and implementationally quite similar for the

LGLAR HMM and the standard framework, but we mention two differences in decision tree clustering which are sometimes important. Firstly, when setting the clustering threshold ξ using the MDL criterion, the default MDL tuning factor $\rho = 1$ is found empirically not to be appropriate for the LGLAR HMM. In preliminary experiments we plotted the difference between the log probability on the training corpus and the test corpus for an LGLAR HMM system as the number of leaves was varied. If the penalized maximum likelihood objective function considered in MDL theory was a good approximation then this might be expected to give a straight line with slope $\frac{1}{2}K \log \tilde{R}_{\text{root}}$. However the actual plot was approximately a straight line with slope 0.3 times this, suggesting the use of $\rho \approx 0.3$. We will see experimentally that this is indeed roughly the MDL tuning factor which gives the highest TSLP value, and that using $\rho = 1$ gives decision trees which are much too small. However we are not sufficiently familiar with MDL theory to be able to present this discrepancy as anything other than an empirical finding. Secondly, for the Gaussian exponential family without variance flooring, the maximum likelihood parameters together with the occupancy can be used to recover the sufficient statistics, as we can see from (2.32) and (2.33). This is not possible for the linear Gaussian linear regression model with $K > 0$, as can be seen by considering the dimensionality of the sufficient statistics (2.52), (2.53), (2.54) and (2.55). This is a general feature of a conditionally additive exponential family with non-trivial interaction terms in the log normalization constant (2.47). This means that the sufficient statistics themselves, and not re-estimated parameters together with occupancies, must be used as the input to decision tree clustering for the LGLAR HMM. This is a relevant consideration when implementing decision tree clustering for the LGLAR HMM in existing software such as HTS.

We have assumed the trajectories for different portions and vector components are conditionally independent given the state sequence, so that C_{tpi} explicitly depends only on the recent past observations $C_{(t-K:t-1)pi}$ in the same portion p and vector component i. We refer to this as the *diagonal-covariance* form of LGLAR HMM, and it is this form that we will mainly consider in this thesis. However it causes no problem for inference or learning, other than increasing the risk of overfitting, to allow C_{tpi} to explicitly depend on the recent past observations in other portions and vector components or even on the present observations $C_{tp(0:i-1)}$ up to the given vector component. Allowing C_{tpi} to depend on $C_{(t-K:t-1)p(0:I-1)}$ and on $C_{tp(0:i-1)}$ is equivalent to using the full-covariance linear regression model in §2.2.4 to predict the vector $C_{tp(0:I-1)}$ from previous vectors $C_{(t-K:t-1)p(0:I-1)}$. We refer to this as the *full-covariance* form of LGLAR HMM. It is also possible to use the Gaussian acoustic model described in §3.2.1 for some portions p and vector components i, and the LGLAR acoustic model for other portions and vector components, and efficient inference and learning is possible using the same approaches we have described so far.

4.3.3 Speech parameter generation

In this section we study the form of distribution over trajectories used by the LGLAR acoustic model in more detail. This view will be useful theoretically when we compare the LGLAR HMM to existing models, and shows how existing speech parameter generation algorithms such as parameter generation considering global variance may be applied to the LGLAR HMM. We also describe how the LGLAR acoustic model supports efficient and low latency parameter generation algorithms not available for the standard HMM synthesis framework and the trajectory HMM.

Given a state sequence $\theta = [\theta_t]_{t=1}^T$, the LGLAR acoustic model defines a distribution $\mathbb{P}(c \,|\, \theta, \lambda)$ over the trajectory $c = [c_t]_{t=1}^T$. Since each of the component distributions $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda)$ is linear Gaussian, the overall distribution $\mathbb{P}(c \,|\, \theta, \lambda)$ has the form of a composite linear Gaussian autoregressive distribution as defined in §2.2.5. In this case (2.67) becomes

$$c_t = \sum_{k=1}^{K} a_{qk} c_{t-k} + a_{q(K+1)} + \sigma_q z_t \tag{4.14}$$

where $q = \overline{q}(\theta_t)$, i.e. the parameters of the linear Gaussian distribution at frame t are the parameters of the linear regression model for the corresponding leaf $q = \overline{q}(\theta_t)$. Using the results in §2.2.5, we can therefore see that $c \,|\, \theta, \lambda \sim \mathcal{N}_{\mathrm{NP}}(b, P)$ for $b = L^{\mathsf{T}}\xi$ and $P = L^{\mathsf{T}}L$, where the t^{th} component of the vector ξ and the t^{th} row of the banded lower triangular matrix L are given by the parameters, in the $\tilde{a}$ parameterization, of the linear regression model for the leaf $q = \overline{q}(\theta_t)$. Thus for the LGLAR HMM, as for the standard HMM synthesis framework and the trajectory HMM, b and P are easy to compute and P is banded. This means that many existing speech parameter generation algorithms, including standard parameter generation and parameter generation considering global variance, can be used directly with the LGLAR HMM simply by passing the appropriate b and P to these algorithms. As explained in §3.3.2, the conventional GV weight setting $\omega^{\mathrm{GV}} = DT$ is appropriate for the LGLAR HMM.

The LGLAR HMM also supports an efficient and low latency form of speech parameter generation not available for the standard HMM synthesis framework or the trajectory HMM. Writing $\mu = \mathbb{E}c$ for the mean trajectory, we can see from (4.14) that for the LGLAR acoustic model we have

$$\mu_t = \sum_{k=1}^{K} a_{qk} \mu_{t-k} + a_{q(K+1)} \tag{4.15}$$

where $q = \overline{q}(\theta_t)$. Thus the mean trajectory may be computed by a one-pass recursion forwards in time, applying (4.15) at each frame $t = 1, 2, \ldots, T$ in turn. We refer to this as the *autoregressive speech parameter generation algorithm*. This algorithm has time complexity $O(TK)$, compared to $O(TK^2)$ for the standard Cholesky-based parameter generation al-

gorithm described in §3.2.4. It is also low latency, since the first frame c_1 may be computed immediately, without having to wait for the result of any trajectory-level computation. The practical advantages of this exact, low latency speech parameter generation algorithm compared to the standard speech parameter generation algorithm are discussed in §5.4.3. A similar one-pass recursion forwards in time can be used to sample a trajectory from $\mathbb{P}(c \,|\, \theta, \lambda)$. Specifically we use (4.14) at each frame $t = 1, 2, \ldots, T$ in turn, sampling $z_t \sim \mathcal{N}(0, 1)$ as we go.

There is a close connection between the LGLAR acoustic model and time-varying autoregressive linear filters. Given the definition of a time-invariant autoregressive linear filter in §4.2, one way to obtain a time-varying filter is to allow the filter coefficients in (4.3) to vary with time, that is

$$y_t = \sum_{k=1}^{K} a_{tk} y_{t-k} + x_t \tag{4.16}$$

where the autoregressive filter coefficients at time t are $[a_{tk}]_{k=1}^{K}$. From (4.15) we can see that the mean trajectory is obtained by passing a signal $[a_{\overline{q}(\theta_t)(K+1)}]_{t=1}^{T}$ through a time-varying autoregressive linear filter of the form (4.16) with autoregressive filter coefficients $[a_{\overline{q}(\theta_t)k}]_{k=1}^{K}$ at time t. Similarly from (4.14) we can see that a sampled trajectory is obtained by passing a signal $[a_{q(K+1)} + \sigma_q z_t]_{t=1}^{T}$, where q is $\overline{q}(\theta_t)$, through the same filter.

The above discussion applies to any linear Gaussian autoregressive HMM, including the LGLAR HMM; in the case of an autoregressive HMM that is not linear Gaussian, there are a few choices for speech parameter generation. It is still possible to use the algorithm described above to exactly sample a trajectory, and this is efficient as long as the individual component distributions $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda)$ support efficient sampling. In general it is not possible to compute the mean trajectory or the most likely trajectory, which may be distinct in the non-Gaussian case, analytically. However we can often obtain a reasonable trajectory by using the same forwards recursion over time, at each frame picking the mean or mode of $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda)$ given the part of the trajectory generated so far.

4.3.4 Divergent trajectories

The LGLAR acoustic model suffers from a potential pathology, whereby the mean trajectory or sampled trajectories can diverge far outside the range of values which might be considered reasonable. In this section we describe this potential pathology, outline what could be done in principle to guarantee divergent trajectories do not occur, and explain why in practice we do not make any adjustments to avoid divergent trajectories.

The notion of stability for time-invariant autoregressive linear filters described in §4.2 sheds some light on how divergent trajectories can occur for the LGLAR acoustic model.

As explained in §4.3.3, the mean trajectory or a sampled trajectory can be viewed as an input signal passed through a time-varying autoregressive linear filter. Since the notion of stability was defined for time-invariant filters and involved the large-time limit, it is not directly applicable to the time-varying case where the autoregressive filter coefficients are not constant. Nevertheless the above analysis indicates what type of behaviour we might expect to see if the autoregressive coefficients $[a_{pqik}]_{k=1}^{K_p}$ for a given portion p, cluster index q and vector component i are unstable. In this case, if the duration of a given segment of frames with cluster index q is substantially greater than the time constant associated with the autoregressive coefficients, then we would expect the mean trajectory or a sampled trajectory to typically diverge substantially over the course of the segment, possibly to the point where the trajectory value at the end of the segment is outside the range of values which are plausible for that portion and vector component. Since the divergence is exponential with time, the extent of the divergence can depend strongly on the exact duration of the segment. The extent of divergence for a given segment also depends on the initial trajectory values at the start of the segment. It should be noted that trajectories generated by a learned autoregressive model can diverge far outside the range of values occurring in the training data, even with plenty of training data and when the prediction is over similar timescales to the training data. This phenomenon will be explored in §5.4.8.

One way to ensure divergent trajectories are unlikely to occur is to ensure that the autoregressive coefficients for all portions, vector components and leaves are stable. For example, in principle (2.56) could be replaced with an equation that estimated the maximum likelihood solution given the constraint that the re-estimated coefficients must be stable. This would agree with the unconstrained maximum likelihood solution in cases where it is stable, and would give a solution right on the limit of stability in cases where the unconstrained solution is unstable. Alternatively Quillen (2012) has suggested two heuristic schemes to ensure the re-estimated autoregressive coefficients are stable.

However divergent trajectories of the type described above seem to be rare in practice. We have only observed them a few times, for some particularly poorly trained systems which used little training data or poor initial alignments. For example, badly divergent trajectories do not occur in the training or test sets for any of the autoregressive systems which will be explored in §4.4, even though some of those systems have a very large number of parameters that are not robustly estimated. Because we have not encountered serious problems due to instability, we make no effort to ensure estimated coefficients are stable in our experiments.

4.3.5 Related work

In this section we briefly summarize previous work using LGLAR HMM-like models for speech processing. For clarity we refer to the non-autoregressive model used during training by the standard HMM synthesis framework, which is defined over the observation sequence, as the *dynamic-augmented Gaussian HMM*, and the corresponding model defined over the speech parameter sequence as the *static-only Gaussian HMM*.

Previous work using the LGLAR HMM to model speech parameters has mainly focused on speech recognition. The possibility of using the LGLAR HMM for this purpose was perhaps first recognized by Wellekens (1987) and Brown (1987), who both described the full-covariance LGLAR HMM in the depth 1 case and gave the equations for parameter estimation using expectation maximization. Experimental results have been reported by a number of authors (Brown, 1987; Kenny et al., 1990; Woodland, 1992; Maxwell and Woodland, 1993; Chin and Woodland, 2002). However in many cases the LGLAR HMM was used to model the observation sequence rather than the speech parameter sequence. This approach suffers from the same lack of consistency as the conventional dynamic-augmented Gaussian HMM. Kenny et al. (1990), using the full-covariance LGLAR HMM on a small scale isolated word recognition task, and Chin and Woodland (2002), using the diagonal-covariance LGLAR HMM on a large vocabulary continuous speech recognition task, both found that the consistent form of LGLAR HMM had a better word error rate than an analogous static-only Gaussian HMM, but a worse error rate than an analogous dynamic-augmented Gaussian HMM. It has also been found that choosing the input to the linear regression to be the frame at offset -3 or -4 or -5 gave a much better word error rate than the frame at offset -1 (which corresponds to the depth 1 case in our treatment) (Kenny et al., 1990; Maxwell and Woodland, 1993). As in the case of the dynamic-augmented Gaussian HMM, it is possible to improve recognition performance by converting the LGLAR HMM generative model into a discriminative model (Brown, 1987; Woodland, 1992; Chin and Woodland, 2002). The qualities of a probabilistic model that are desirable for speech synthesis are very different to those that are desirable for speech recognition, so we do not necessarily expect the above observations to generalize to our use case. A key difference is that for speech recognition a more accurate acoustic model is only really of interest if it improves discrimination of phonemes or words, whereas for speech synthesis a more accurate acoustic model is inherently of interest.

The LGLAR HMM can also potentially be used to model the waveform directly (Brown, 1987). In this case it is common to make two slight tweaks: allowing the state to only transition once every k waveform samples; and resetting the acoustic context to zero every k samples. This model is sometimes known as the *hidden filter HMM* (Poritz, 1982; Juang and Rabiner, 1985; Ephraim et al., 1989), and it can be seen as generalizing the linear

predictive analysis sometimes used to extract speech parameters, integrating it into the statistical model. A similar model was used for speech synthesis by Donovan (1996, chapter 5), though in this case it was assumed that the state sequence was observed.

The LGLAR HMM has only recently been used to model speech parameters for speech synthesis (Shannon and Byrne, 2009a; Quillen, 2012). Quillen investigated the problem of stability of autoregressive coefficients for speech synthesis using the LGLAR HMM, and in addition found that using alignments derived from a speech recognition system to estimate LGLAR HMM parameters led to an improvement in an objective metric compared with expectation maximization from a flat start (Quillen, 2012).

4.4 Experiments on setting model structure parameters

In this section we experimentally investigate the values of model structure parameters such as depth, number of sublabels and MDL tuning factor that are appropriate for modelling speech using the LGLAR HMM. The customary values used for the standard HMM synthesis framework may not be optimal for the LGLAR HMM, and some parameters such as depth have no direct analogue in the standard framework. Investigating these choices involves evaluating an extensive set of systems, and so we chose to measure objective performance only. These results have been reported previously (Shannon et al., 2013).

For each of the model structure parameters investigated, we used a 5 sublabel, depth 3 autoregressive system with an MDL tuning factor of 0.3 (system A) and a 5 sublabel, depth 2 autoregressive system with an MDL tuning factor of 0.18 (system AM) as starting points, and varied the model structure parameter under investigation. The other details of the experimental set-up are as described in §3.5.

Ideally for each number of sublabels and depth considered we would choose the optimal MDL tuning factor. However just choosing the MDL tuning factor which achieves the best score on the test set would involve substantial re-use of the test set, and conducting a full 3-dimensional search with a held-out validation set or using cross validation would be computationally intensive. Preliminary experiments suggested that using an MDL tuning factor of 0.3 gives good test set log probability for a wide variety of choices of the depth and the number of sublabels, and using an MDL tuning factor of 0.18 gives good mel cepstral distortion. Therefore we only considered MDL tuning factors of 0.3 and 0.18 for all depths, except depth 0 where we used an MDL tuning factor of 1.0. In the depth 0 case the autoregressive HMM is just a conventional HMM, and 1.0 is generally considered an appropriate choice of MDL tuning factor for the conventional HMM, and experiments confirmed that using an MDL tuning factor of 1.0 gave better TSLP and MCD than using an MDL tuning factor of 0.3 or 0.18. We also report the test set log probability and MCD of the monophone system, since this is not sensitive to the choice of MDL tuning factor.

4.4.1 Depth

We trained autoregressive systems of various depths. The depth was varied for the spectral portion of the speech parameters only. We found that local maxima during training were a problem, with the *training set* log likelihood for depth 5 monophone models lower than for depth 4 monophone models in spite of the fact depth 4 models are a special case of depth 5 models. We therefore decided to use a more careful training procedure: after performing several iterations of EM at depth 0, we increased the depth to 1 and performed several more iterations of EM, then to 2 and performed several more iterations of EM, etc, until the desired depth was reached.

The results are shown in Figure 4.3. The best test set log probability is at depth 3 with system A, and depths 3, 4 and 5 all have good TSLP. We can see that increasing the depth gives decent improvements in TSLP up to depth 3, and thereafter results in minor degradation. The sudden drop in TSLP for depth 6 system AM is due to overfitting; the training set log likelihood for the depth 6 system is greater than for the depth 5 system. The best MCD is at depth 1 with system AM, and depths 1, 2 and 3 all have good MCD. We can see that increasing the depth gives a decent improvement in MCD up to depth 1, and thereafter results in minor degradation. We therefore suggest 2 or 3 is the most appropriate choice of depth for the LGLAR HMM.

The gradual training procedure used here was not used in the other experiments. Without gradual training, depth 3 system A still has the best TSLP, but depth 2 system AM now has the best MCD and is $-0.10\,\mathrm{dB}$ better than depth 1. For the depth 3 system A and depth 2 system AM used elsewhere the difference made by gradual training was minimal (TSLP within $0.04\,\mathrm{nats}$ and MCD within $0.02\,\mathrm{dB}$).

4.4.2 Number of sublabels

We trained autoregressive systems with various numbers of sublabels. The results are shown in Figure 4.4. We can see a clear peak in TSLP at the conventional value of 5 sublabels for system A, and 5 and 6 sublabels both have good TSLP. The best MCD is at 5 sublabels with system AM, and 5, 6 and 7 sublabels all have good MCD. The convention of using 5 sublabels inherited from the standard framework is thus appropriate for the LGLAR HMM.

We noticed an effect where the *training set* log likelihood was lower for the monophone models with 6 to 8 sublabels than with 5 sublabels. This might at first seem surprising since the 5 sublabel model is almost a special case of the 8 sublabel model. However the 5 sublabel model is not quite a special case of the 8 sublabel model due to the *minimum duration restriction*: each instance of a sublabel is restricted to have duration at least one frame. Thus for the 8 sublabel model each phoneme is restricted to have a minimum duration of 8 frames whereas for the 5 sublabel model each phoneme is restricted to have a minimum

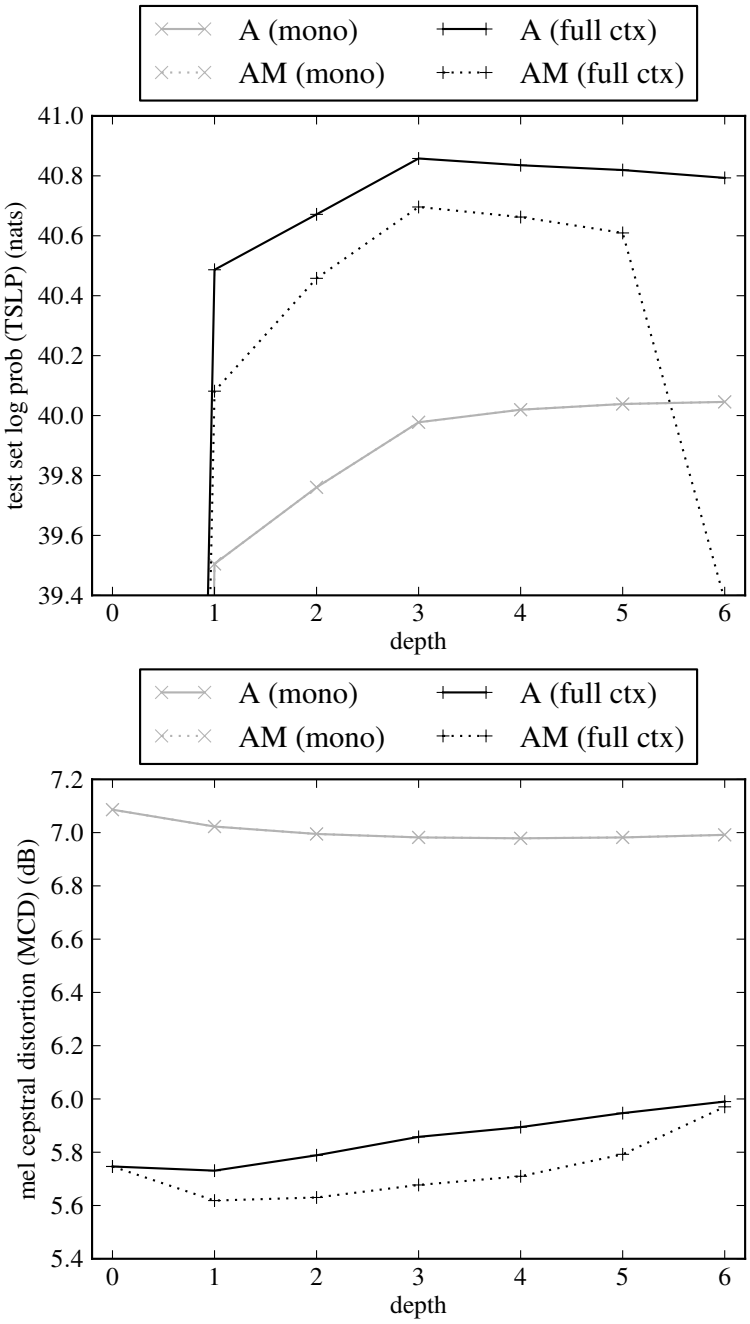


Figure 4.3: How depth affects (top) test set log probability and (bottom) mel cepstral distortion. All systems used 5 sublabels. The full context system A used an MDL tuning factor of 0.3 for all depths except 0 where it used an MDL tuning factor of 1.0. The full context system AM used an MDL tuning factor of 0.18 for all depths except 0 where it used an MDL tuning factor of 1.0. The monophone system A and monophone system AM are identical and so have identical results in this figure.

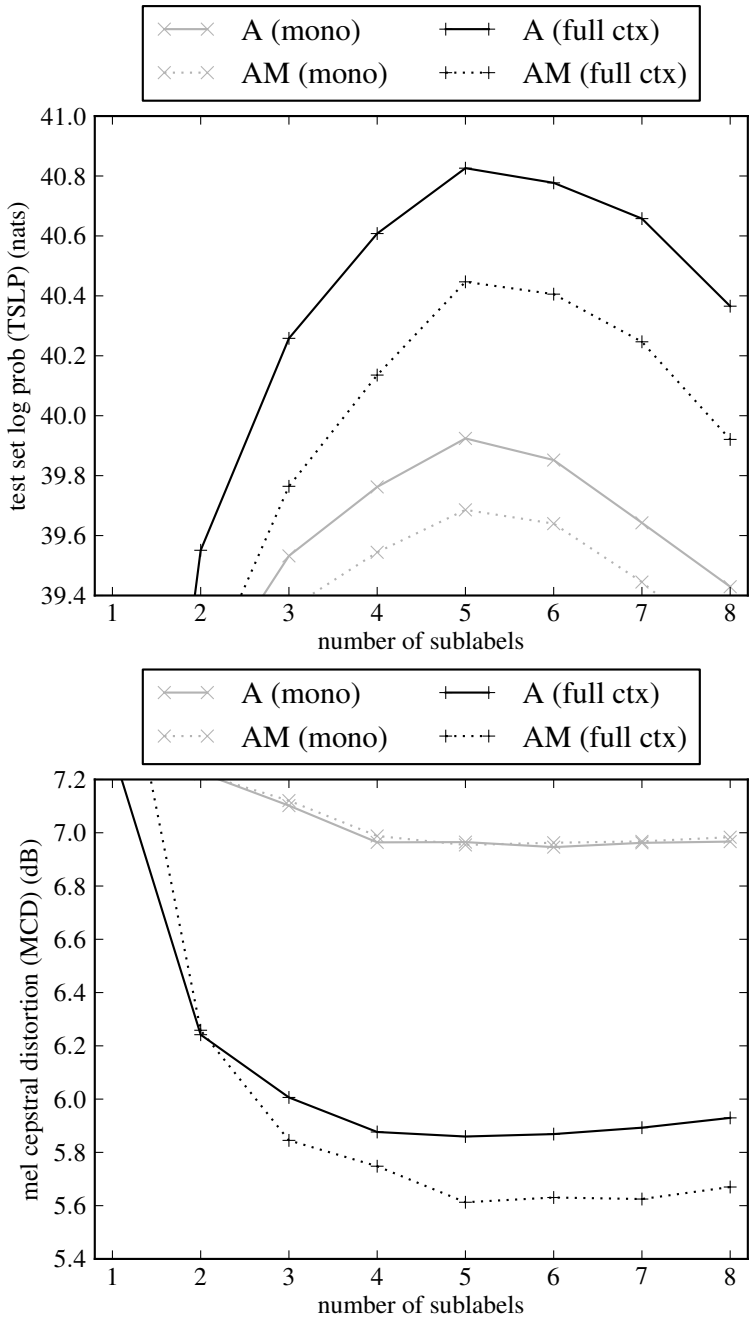


Figure 4.4: How number of sublabels affects (top) test set log probability and (bottom) mel cepstral distortion. System A had depth 3 and system AM had depth 2. The full context system A used an MDL tuning factor of 0.3 and the full context system AM used an MDL tuning factor of 0.18.

duration of 5 frames. Preliminary investigations suggested that the decrease in training set log likelihood was mainly caused by the minimum duration restriction rather than being a local maximum effect as in the case of depth. This warrants further investigation.

4.4.3 MDL tuning factor

We trained autoregressive systems with various MDL tuning factors used during decision tree clustering. The MDL tuning factor was varied for the spectral portion of the speech parameters only. The results are shown in Figure 4.5. For the TSLP of system A we can see that there is no clear peak, with MDL tuning factors between 0.20 and 0.35, corresponding to a total of roughly 600 to 1600 mcep leaves, having good TSLP. For the MCD of system AM we see a narrower range of good MDL tuning factors, with values between 0.17 and 0.20, corresponding to a total of roughly 2000 to 3500 mcep leaves, having good MCD. We mention in passing that the number of leaves increases very rapidly as we lower the MDL tuning factor from 0.18: for example system AM has around 1000 mcep leaves at an MDL tuning factor of 0.3, 2900 leaves at 0.18, 6400 leaves at 0.15, and 16 000 leaves at 0.13. This means that the degradation in TSLP and MCD in Figure 4.5 for small MDL tuning factors would not appear as sudden on a plot with number of leaves instead of MDL tuning factor as the abscissa. Based on the above results we suggest a value between 0.18 and 0.3 is the most appropriate choice of MDL tuning factor for the LGLAR HMM.

It is interesting to note that the number of leaves that is best for MCD is greater than the number of leaves that is best for TSLP. In other words a bit of overfitting is a good thing from the point of view of MCD. This is presumably because the increase in modelling flexibility obtained by using more leaves has more positive consequences on the mean trajectory than the fact that we cannot robustly estimate the parameters of the additional leaves has negative consequences. We will see in §4.6.4 that this effect also occurs when looking at MCD for the standard HMM synthesis framework and the trajectory HMM. We have previously observed a somewhat similar relationship between human naturalness judgements and TSLP (Shannon and Byrne, 2010). When we evaluated the mean opinion score of LGLAR HMM systems with varying numbers of leaves, we found that: using the number of leaves which gave the best TSLP also gave approximately the best naturalness; slight underfitting due to using fewer leaves resulted in a large degradation in naturalness; slight overfitting due to using more leaves resulted in no degradation or possibly a small improvement in naturalness; and fairly severe overfitting resulted in very little degradation in naturalness. It would be interesting to see whether the relationships between TSLP and MCD and TSLP and naturalness described above persist when an estimation method is used that has fewer issues with robustness than maximum likelihood. Alternative estimation methods for use during decision tree clustering were discussed briefly in §2.3.

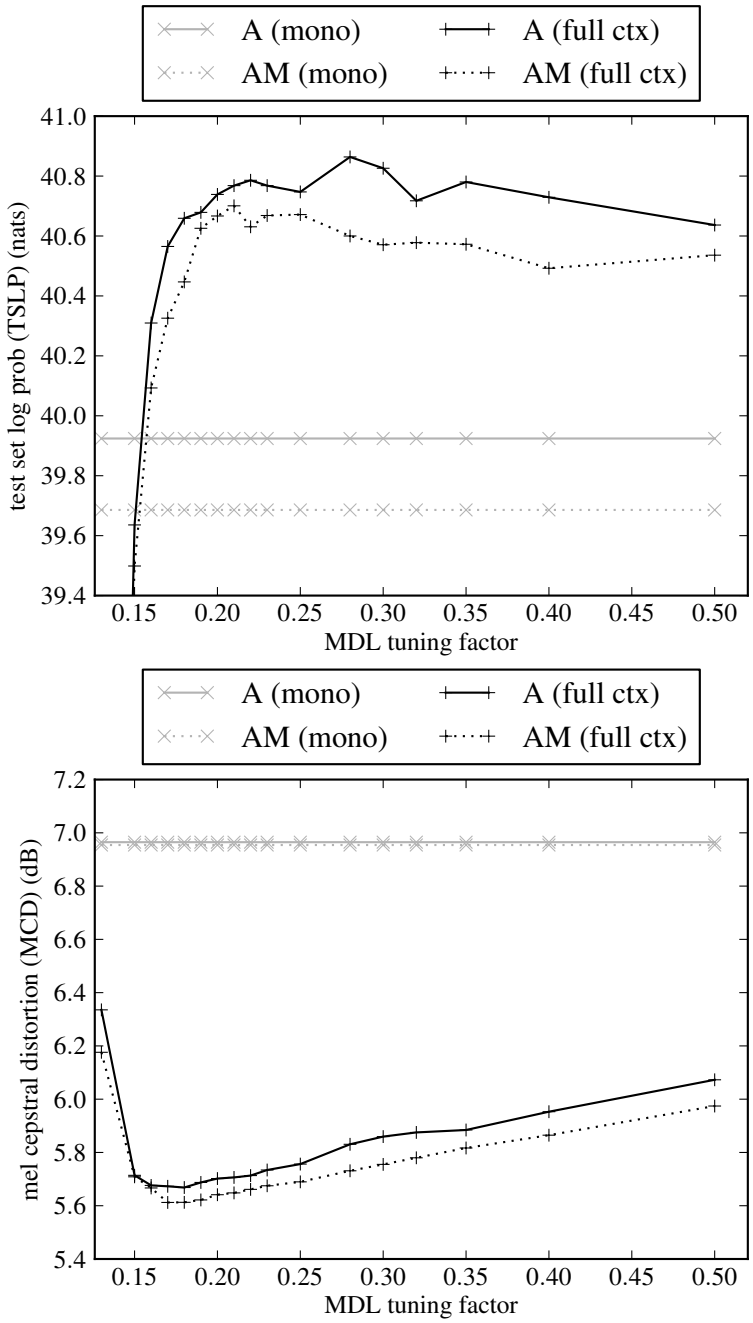


Figure 4.5: How MDL tuning factor affects (top) test set log probability and (bottom) mel cepstral distortion. All systems used 5 sublabels. System A had depth 3 and system AM had depth 2.

4.5 Median alignments

In some of the experiments in this chapter and later chapters we make use of a novel type of alignment which we refer to as a *median alignment*. In this section we describe what median alignments are, how to compute them, and some of their useful properties.

We will first describe label-level median alignments, and then explain how this generalizes to other levels. Consider an utterance with label sequence $l = [l_j]_{j=1}^{J}$ and speech parameter sequence C. Typical statistical models assume that, for each frame t, precisely one label j_t in the label sequence is "current", and so the statistical model defines a posterior marginal distribution $\mathbb{P}(j_t \,|\, l, C, \nu, \lambda)$ over the set of label indices $\{1, \ldots, J\}$. Since the set of label indices possesses a natural total order, the median $\hat{j}_t$ of this marginal distribution is well-defined. The sequence $[\hat{j}_t]_{t=1}^{T}$ of median label indices over time may be converted to a label index-level alignment by grouping contiguous identical label indices together. This may then be converted to a label-level alignment by looking up the label for each label index, and we refer to the alignment obtained in this way as the (label-level) *median alignment*. Computationally, median alignments for an autoregressive HMM and for the model used during training for the standard HMM synthesis framework are easily obtained using the forward-backward algorithm, since the marginal posterior distribution over the label index for each frame may be computed by summing state occupancies.

The above definition can be applied to any ordered sequence that is part of our model as long as precisely one element of the sequence is "current" for each frame t. In particular we can define the sublabel-level median alignment, since each label sequence l defines a sequence of (label, sublabel) pairs as described in §3.1.3. Under the standard approach to encoding the duration model for each sublabel mentioned in §3.1.5 the state is of the form $\psi = (m, j, s, d)$, and the number of frames d remaining in the current sublabel provides a natural ordering for the states, so we may even talk about a state-level median alignment.

Median alignments have a number of nice theoretical properties. Firstly they are well-behaved under a certain type of marginalization. If we take the state-level median alignment and convert it to a sublabel-level alignment, we get the sublabel-level median alignment. Similarly if we take the sublabel-level median alignment and convert it to a label-level alignment, we get the label-level median alignment. Compare this to the situation for Viterbi alignments: taking the most likely state sequence and converting it to a sequence of labels over time does not give the most likely sequence of labels over time. We describe this as a nice theoretical property "under marginalization" since the posterior distribution over the sublabel-level alignment, for example, is obtained by marginalizing over the duration part of the state-level alignment.

Median alignments have a second nice theoretical property that is closely related to the fact they are well-behaved under marginalization, namely that they are indifferent to the

representation used for deeper levels of the model. The X-level median alignment depends only on the probability distribution over the X-level alignment given by the model, and not on the details of how this probability distribution is encoded. For example the sublabel-level median alignment depends only on the probability distribution over the sublabel-level alignment, and not on details of how the duration model for each sublabel is encoded. In contrast the sublabel-level alignment obtained from the Viterbi state sequence does depend on the details of how the duration model for each sublabel is encoded. For example if the duration for a given sublabel was modelled as a geometric distribution, then this may be represented either as a single state with a self-transition or using the explicit (m, j, s, d) representation, and the sublabel-level alignment obtained from the Viterbi state sequence will generally be different for these two representations.

Since median alignments are defined in terms of framewise marginal probabilities, it might be thought that they would have pathologies when viewed as sequences of (label, sublabel) pairs over time. For example, perhaps the ordering of (label, sublabel) pairs in the median alignment could be non-monotonic with respect to the ordering in the original (label, sublabel) sequence, or (label, sublabel) pairs that appear in the original (label, sublabel) sequence might not appear, or equivalently might have duration zero, in the median alignment. These pathologies would not necessarily be a problem for our purposes if they did occur. However for the form of models we use here, the median alignments are in fact guaranteed to correspond to valid (label, sublabel) sequences over time, in the sense that their prior probability is non-zero. In particular they are always monotonic, and satisfy the constraint that each sublabel must have duration at least 1 frame.

Finally we note in passing that randomly sampled alignments share many of the desirable properties of median alignments: they are also well-behaved under marginalization, they are also indifferent to the representation used for deeper levels of the model, and by definition they are valid alignments with non-pathological posterior probability. Computationally it is easy to efficiently sample a state sequence from the posterior distribution $\mathbb{P}(\theta \mid l, C, \nu, \lambda)$ using the values computed by the backward algorithm mentioned in §2.5.1. The main disadvantage of sampled alignments is that they are not deterministic, though this may or may not be considered a problem depending on the application.

4.6 Experimental comparison to existing models

To evaluate the LGLAR HMM for statistical parametric speech synthesis we compared a baseline standard HMM synthesis framework system, a trajectory HMM system and two LGLAR HMM systems using the subjective and objective metrics described in §3.4. We also compared a wider range of systems using objective metrics only. Many of these results were previously presented in a journal paper (Shannon et al., 2013).

4.6.1 Systems

The systems under comparison are shown in Table 4.1. Systems S and T are the standard HMM synthesis framework system and the trajectory HMM system described in §3.5. System A is a "conventional" autoregressive system (5 sublabels, depth 3, MDL tuning factor of 0.3) which has structure parameters which give good TSLP. System AM is an autoregressive system with structure parameters tuned to have good MCD (5 sublabels, depth 2, MDL tuning factor of 0.18). System SB is system S with a uniform variance boost (see below for details). The other details of the experimental set-up are as described in §3.5.

4.6.2 Evaluation methodology

The subjective evaluation was conducted with systems N, S, T, A and AM following Blizzard Challenge-style methodology described in §3.4. The listening test consisted of 10 sections of 5 utterances each. For all sections listeners were asked to rate the naturalness of each utterance on a scale of 1 to 5 inclusive. Prompts were the 50 held-out utterances in a fixed order. The listening test was presented via an interactive website over two weeks. System SB was not included since it had not been conceived of at the time of the listening test, but the generated trajectories are extremely similar to those of system S, and so it is anticipated that the naturalness ratings would also be very similar. The reason for the similarity is explained in Chapter 6: conventional GV generation is for all intents and purposes equal to fixed-spread GV generation, and fixed-spread GV generation produces the same trajectories whether variance-boosted or not.

For the objective evaluation we computed test set log probability and mel cepstral distortion on the 50 held-out utterances. For this experiment we computed the *approximate mcep-only TSLP* $\log \mathbb{P}([C_{t1i}]_{t,i} \,|\, \theta^*, \lambda)$ where $[C_{t1i}]_{t,i}$ is the spectral portion of the speech parameter sequence and θ^* is the median alignment computed using all portions of the speech parameter sequence C. Median alignments from system A were used to evaluate system A, system AM to evaluate system AM, and system S to evaluate systems S, SB and T. A fixed state sequence was used because the true test set log probability, which is obtained by marginalizing $\mathbb{P}(C, \theta \,|\, l, \nu, \lambda)$ over θ, is difficult to compute for the trajectory HMM and the standard HMM synthesis framework. Note that for the standard HMM synthesis framework the test set log probabilities we compute are for the model effectively used during synthesis, i.e. the trajectory HMM with the same parameters, not the model used during training.

system	description
N	natural speech
S	standard HMM synthesis framework
SB	system S with 3× variance boost
T	trajectory HMM
A	LGLAR HMM (standard structure parameters)
AM	LGLAR HMM (modified structure parameters)

Table 4.1: Systems used to compare the LGLAR HMM to existing models.

system	mcep leaves	opinion score		MCD DTW (dB)	using median alignments	
		mean	median		from sys	mcep TSLP (nats)
N	-	4.7	5	-	-	-
S	812	2.4	2	5.58	S	29.25
SB	812	-	-	5.58	S	46.86
T	812	2.6	3	5.46	S	47.58
A	771	2.1	2	5.86	A	47.88
AM	2879	2.4	2	5.61	AM	47.58

Table 4.2: Comparison of the LGLAR HMM to the standard HMM synthesis framework and trajectory HMM using subjective and objective evaluations.

	N	S	T	A	AM
N		■	■	■	■
S	■		□	■	□
T	■	□		■	□
A	■	■	■		■
AM	■	□	□	■	

Table 4.3: Pairwise comparisons of significant differences between naturalness using Bonferroni-corrected Mann-Whitney U tests (■ indicates a significant difference at 1%).

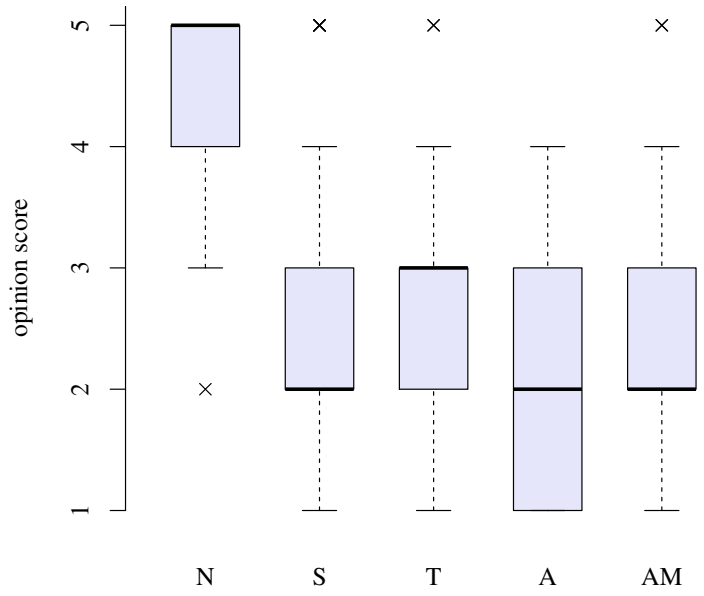


Figure 4.6: Box plot showing results of the subjective evaluation.

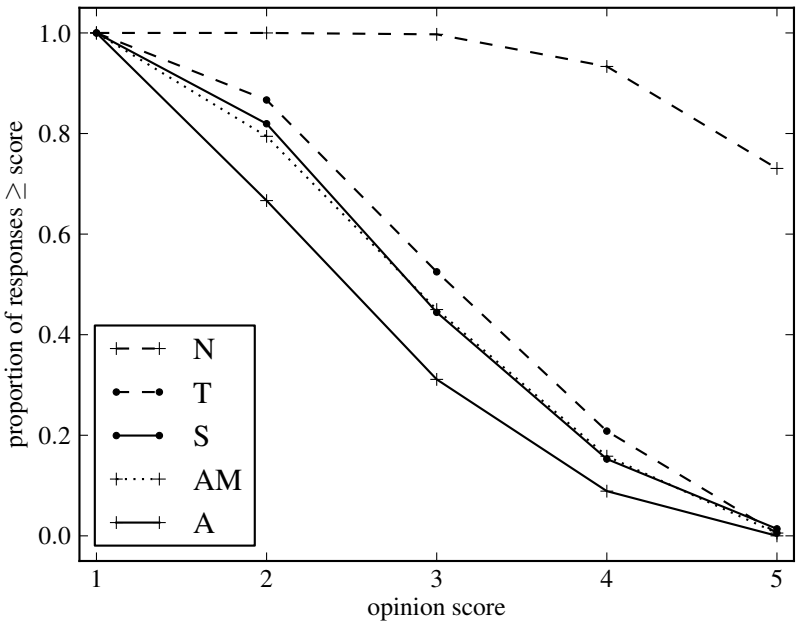


Figure 4.7: Complementary cumulative plot showing results of the subjective evaluation in more detail. For an opinion score s, the ordinate gives the proportion of participant responses that were s or greater. For any given opinion score larger ordinate values are better.

4.6.3 Results of comparison to existing models

The listening test was completed by 36 native English speakers. Table 4.2 shows a summary of the results. Figure 4.6 is an opinion score box plot (Clark et al., 2007), and a matrix of statistically significant differences between the various systems is shown in Table 4.3. Figure 4.7 shows a *complementary cumulative plot* of these results, which displays more information than the box plot. We can see that the modified autoregressive system (AM) has extremely similar performance to the standard HMM synthesis framework (S). The trajectory system (T) does slightly better than these two systems, and the default autoregressive system (A) does noticeably worse than these two systems. The statistical significance test has S, T and AM possibly identical in performance with A statistically different from the other three. The results also show that MCD was more useful than TSLP as a surrogate for human judgement when selecting autoregressive model structure parameters.

The objective results are presented in Table 4.2. To provide some intuitive calibration of the TSLP and MCD scales, the reader may be interested to know that using full-context instead of monophone models results in a typical improvement of roughly $+0.4$ nats to $+0.6$ nats for approximate mcep-only TSLP, and roughly -1.1 dB to -1.3 dB for MCD.

We can see that the trajectory system (T) and the modified autoregressive system (AM) are comparable in terms of test set log probability. The default autoregressive system (A) does notably better than any of the other systems. The standard system (S) has extremely low test set log probability, due to the fact it systematically underestimates predictive variance, as will be explained in §5.4.1. We also computed the test set log probability of the standard system with a multiplier of 3 applied to the covariance of each trajectory. As will be discussed in §5.4.1, the value of 3 is close to optimal for all mcep components. This results in a system (SB) that no longer systematically underestimates predictive variance and has a much greater test set log probability. However there is still a large gap between the variance-boosted standard system (SB) and the normalized models. These results suggest that the LGLAR HMM performs favourably compared to existing models as a probabilistic model of speech.

The MCD results are qualitatively similar to the subjective listening test results. The modified autoregressive system (AM) and the standard HMM synthesis framework (S) have very similar MCD, with the trajectory HMM system (T) very slightly better and the default autoregressive system (A) noticeably worse. Thus the trajectory HMM appears to provide the best model of the mean trajectory. These results suggest that the LGLAR HMM inherently provides a slightly poorer model of the mean trajectory than the standard HMM synthesis framework, but that MCD performance on the level of the standard approach can be obtained with the LGLAR HMM by using more leaves (system AM).

It should be noted that the MCD results appear to depend strongly on the precise form

of MCD used. In preliminary experiments with forms of MCD using a fixed alignment rather than dynamic time warping, we found that for some methods of computing the alignment systems S and AM were similar in MCD score, but for other methods of computing the alignment system S was noticeably better than system AM.

4.6.4 Improved training regimes

Since conducting the subjective evaluation above we have noticed a number of ways to improve the training regimes, at least in the sense of improving the objective scores. Here we briefly discuss these improvements and give an updated table of results with objective scores for the new training regimes.

For autoregressive systems we noticed that the procedure used at the monophone stage of training, before the first autoregressive decision tree clustering is performed, can have a reasonably large influence on final system performance. Firstly we found that, when a good initial alignment is used, performing expectation maximization at the monophone stage often hurts the final system performance. This is not too surprising given the limitations of autoregressive HMMs with monophone labels that will be mentioned in §5.1. Secondly we found that using median alignments derived from the full-context standard system S as the initial alignments was typically better than using median alignments derived from the monophone stage of system S, and better than the monophone initialization procedure used for systems A and AM. For the remaining autoregressive systems described in this chapter we therefore decided to use the median alignments from system S as the initial alignments, and began decision tree clustering directly from these alignments rather than performing expectation maximization at the monophone stage of training. This initialization procedure is used below for system A2, which has the same model structure parameters as system A (5 sublabels, depth 3, MDL tuning factor 0.3), and system AM2, which has the same model structure parameters as system AM (5 sublabels, depth 2, MDL tuning factor 0.18).

Secondly the experimental comparison presented above is not completely fair since for the LGLAR HMM we chose the MDL tuning factor ρ to give good TSLP (system A) or MCD (system AM) whereas for the for the standard HMM synthesis framework we used the fixed conventional value $\rho = 1.0$. This was done because the standard HMM synthesis framework is well established, whereas the LGLAR HMM is a new model requiring investigation of an appropriate MDL tuning factor. However we have since found that using a smaller value for the MDL tuning factor also improves the performance of the standard HMM synthesis framework, with a maximum in approximate mcep TSLP at around $\rho = 0.6$ when using a variance boost of 3, and a minimum in MCD at around $\rho = 0.35$. We refer to the $\rho = 0.6$ and $\rho = 0.35$ systems as S2 and SM2 respectively, and the corresponding variance-boosted systems as SB2 and SMB2.

system	MDL tuning factor ρ	mcep leaves	MCD DTW (dB)	using median alignments	
				using sys	mcep TSLP (nats)
SB2	0.60	1693	5.44	S2	46.94
T2	1.00	812	5.46	S	48.21
A2	0.30	964	5.74	A2	48.02
SMB2	0.35	4866	5.36	SM2	46.72
TM2	0.60	1693	5.39	S2	48.14
AM2	0.18	3410	5.52	AM2	47.59

Table 4.4: Comparison of the LGLAR HMM to the standard HMM synthesis framework and the trajectory HMM using recommended training regimes for all systems. The MDL tuning factor ρ controls the size of the decision trees estimated during training. The systems for the top three lines have structure parameters such as ρ chosen to give good TSLP, while the systems for the bottom three lines have structure parameters chosen to give good MCD.

Finally, we implemented our own trajectory HMM trainer, and found that it resulted in a system with substantially higher TSLP than system T. Our procedure used median alignments from system S as the fixed alignment during trajectory HMM training whereas system T used Viterbi alignments from system S. Differences in the convergence threshold or gradient ascent algorithm used may also have been responsible for the difference in performance. We considered MDL tuning factors of 1.0, 0.6 and 0.35 for the standard system used to obtain the trees and alignments used for trajectory HMM training, and found that $\rho = 1.0$ (system T2 below) gave the best approximate mcep TSLP while $\rho = 0.6$ (system TM2 below) gave the best MCD. Ideally more values of the MDL tuning factor would be considered. It should be noted that we used no variance flooring or regularization in our trajectory HMM implementation. Extreme values of the delta and delta-delta precision parameters were observed for some leaves and some vector components for the $\rho = 0.60$ and $\rho = 0.35$ systems, and so using regularization might improve the test set performance of these systems. No extreme precision parameter values were observed for the $\rho = 1.0$ system.

The results with all the above improvements are shown in Table 4.4. We can see that the newer systems are all at least as good as the systems in Table 4.2 according to both metrics, and substantially better by some metrics. In terms of TSLP, the normalized models still do much better than the unnormalized standard HMM synthesis framework system after a variance boost (SB2). However the trajectory HMM system (T2) now has better TSLP than the autoregressive system (A2) by a sizeable margin (0.19 nats), despite the fact that for evaluating system A2 we use median alignments from itself whereas for evaluating system T we use median alignments from system S. This result is reassuring since in our limited experience the trajectory HMM seems like the better probabilistic model of speech.

In terms of MCD, the autoregressive system here (AM2) does almost as well as the previous trajectory HMM system (T), which shows the improvement brought by better training. However the gap between the LGLAR HMM system (AM2) and the other two models is now wider, with the standard system (SM2) having a better MCD by a reasonable margin (0.16 dB). Perhaps surprisingly the best system in Table 4.4 in terms of MCD is a standard system rather than a trajectory system, though the difference is small (0.03 dB). The $\rho = 0.35$ trajectory HMM system (not shown), which we would have expected to have the best MCD score of any of the models considered, has an MCD of 5.43 dB, which is worse than the $\rho = 0.35$ standard system (5.36 dB). This could conceivably be an inherent weakness of the trajectory HMM parameterization, but we think it is most likely due to overfitting. This could be investigated by using regularization during trajectory training. Note that both system SMB2 and system TM2 obtain a sizeable improvement in MCD compared to the previous systems presented in Table 4.2.

It is interesting to note that a modest amount of overfitting appears to improve MCD scores: for all three models above the number of leaves that is best for MCD is greater than the number of leaves that is best for TSLP. We observed this effect for the LGLAR HMM in §4.4.3, and the above results show that the same is true for the standard HMM synthesis framework and the trajectory HMM.

In terms of both TSLP and MCD, the best number of leaves for the trajectory HMM is lower than the best number of leaves for the standard framework. Zhang (2009) observed a similar effect while using the trajectory HMM for articulatory inversion and using root-mean-squared error as a metric. These results seem to suggest that the trajectory HMM is more susceptible to overfitting than the standard framework. However care should be taken before reaching this conclusion, since our standard system was trained using variance flooring whereas the trajectory system was not, and Zhang used a fixed, manually chosen value for a variance floor and variance ceiling. Zhang also used early stopping based on a held out validation set as a simple form of regularization. It would be interesting to see if this pattern of results persisted with a more fine grained regularization scheme. It should also be noted that both we and Zhang observe that the trajectory HMM performance with fewer leaves rivals or beats the standard framework performance with more leaves.

We conclude from the MCD results that the LGLAR HMM is capable of modelling the mean trajectory better than the standard HMM synthesis framework with default structure parameters, and from the TSLP results that the LGLAR HMM is much better as a probabilistic model of speech. However if we choose the MDL tuning factor for the standard framework judiciously then the standard framework models the mean trajectory better than the LGLAR HMM. The trajectory HMM provides the best probabilistic model of speech and amongst the best model of the mean trajectory, despite the fact it uses a fixed alignment for training.

4.7 Summary of contributions

The major novel theoretical contributions of this chapter are: the realization that the LGLAR HMM constitutes a consistent and tractable alternative to existing models used for statistical parametric speech synthesis (§4.3); the realization that it supports existing effective speech parameter generation methods (§4.3.3); and the realization that there is a simple and exact time-recursive speech parameter generation algorithm for the LGLAR HMM with low latency benefits (§4.3.3). The major novel experimental contributions of this chapter are: a detailed objective evaluation of how to set the model structure parameters of the LGLAR HMM for statistical parametric speech synthesis (§4.4); and an objective and subjective evaluation of the performance of the LGLAR HMM compared to existing models for statistical parametric speech synthesis, showing that the autoregressive framework presented here is capable of producing speech that is as natural as that of the standard HMM synthesis framework with its conventional settings, but not as natural as the trajectory HMM (§4.6).

Minor novel contributions of this chapter include: an appreciation that the tractability of expectation maximization and decision tree clustering for the LGLAR HMM stems from the fact the linear regression model at its heart is a conditionally additive exponential family (§4.1.1 and §4.1.2); the realization that the autoregressive HMM is a flexible framework which supports the use of complicated regression models (§4.1.2); the concept of a median alignment and recognition of its nice theoretical properties (§4.5); the idea of using a complementary cumulative plot to summarize the results of a opinion score listening test (§4.6.3); and the discovery that a bit of overfitting improves MCD scores for both the LGLAR HMM and existing models (§4.4.3 and §4.6.4).

Chapter 5

Connections between new and existing models

In this chapter we examine a variety of theoretical connections between the standard HMM synthesis framework, the trajectory HMM and the LGLAR HMM. As we will see, there are both strong similarities and important differences between the three models.

The layout of this chapter is as follows. We first describe a weakness in the autoregressive HMM, namely that future states cannot influence the present observation. We illustrate the effects of this *future state blindness* on a simple synthetic data set in order to help get a feel for the challenges faced by the autoregressive HMM when modelling speech parameter sequences. We then present two novel ways to view the three models under consideration within a unified framework. As well as hopefully being theoretically interesting, these unified views allow us a certain amount of insight into the strengths and weaknesses of the three models. Finally we summarize the similarities and differences between the three models from a theoretical point of view and examine some of the differences in more detail using a combination of theoretical arguments and experiments.

5.1 Future state blindness

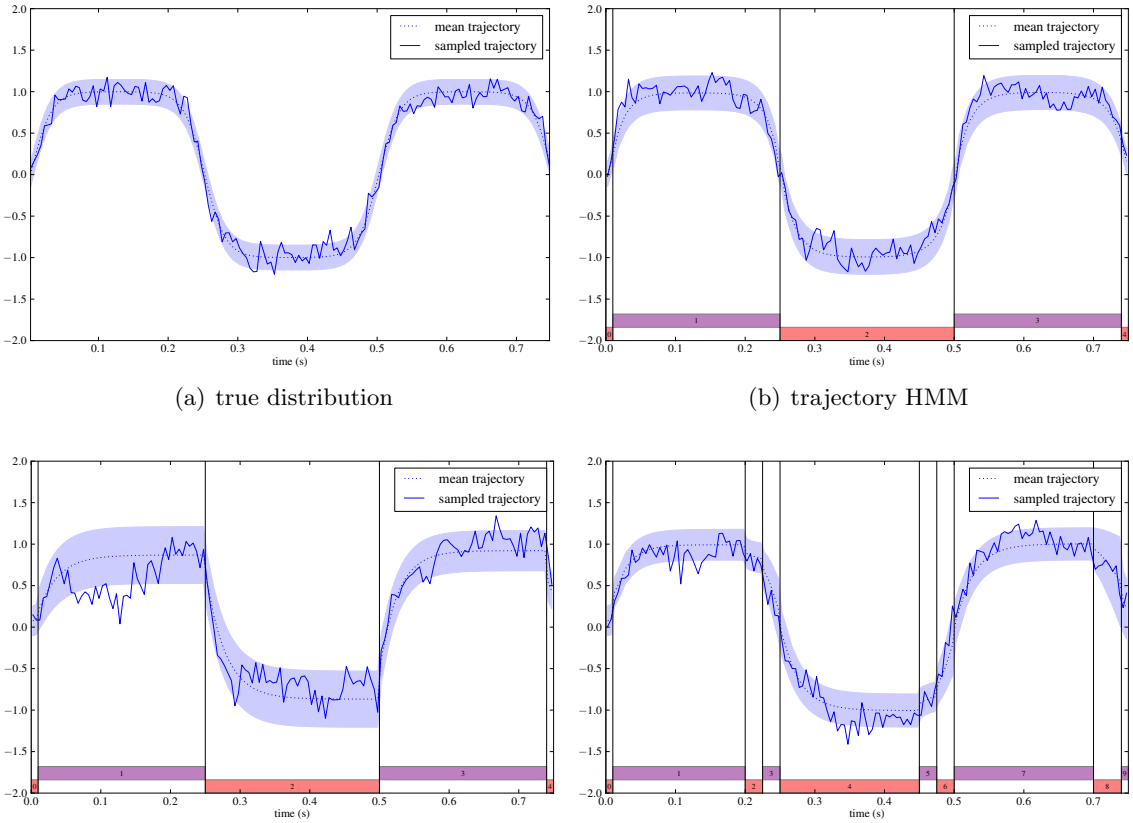
Autoregressive acoustic models and autoregressive HMMs suffer from an obvious weakness, which we refer to as future state blindness. In this section we illustrate this weakness using a simple synthetic data set. We also discuss its implications for autoregressive speech synthesis and how its worst effects might be alleviated or mitigated. The possibility that we might be able to mitigate the worst effects of future state blindness was part of our original motivation for considering autoregressive models for parametric speech synthesis.

Future state blindness concerns the effect of future states on the present speech parameters. Consider an autoregressive trajectory-level acoustic model $\mathbb{P}(c \,|\, \theta, \lambda)$, for example

the LGLAR acoustic model. The marginal distribution $\mathbb{P}(c_t \,|\, \theta, \lambda)$ over the trajectory value c_t at frame t given the state sequence θ does not depend on future states $\theta_{t+1:T}$, i.e. $\mathbb{P}(c_t \,|\, \theta, \lambda) = \mathbb{P}(c_t \,|\, \theta_{1:t}, \lambda)$. We refer to this phenomenon as *future state blindness*. In contrast $\mathbb{P}(c_t \,|\, \theta, \lambda)$ typically does depend, at least slightly, on previous states $\theta_{1:t-1}$ going arbitrarily far back in time. For comparison, the marginal $\mathbb{P}(c_t \,|\, \theta, \lambda)$ for the trajectory HMM acoustic model typically depends on all elements of the state sequence, and so the trajectory HMM acoustic model does not suffer from future state blindness. Future state blindness can be seen as an inability of autoregressive acoustic models to accurately represent certain conditional distributions: if, for the "true" conditional distribution $\mathbb{P}_{\text{true}}(c \,|\, \theta)$, the marginal distribution $\mathbb{P}_{\text{true}}(c_t \,|\, \theta)$ has a strong dependence on θ_{t+1}, say, then an autoregressive acoustic model will not be able to accurately represent $\mathbb{P}_{\text{true}}(c \,|\, \theta)$. In the context of statistical speech synthesis, future state blindness means that autoregressive acoustic models are incapable of modelling any dependence of the current speech parameters on future phonemic and linguistic events that are not already encoded in the current state θ_t. Here by future phonemic and linguistic events we mean items such as the next phoneme and the ToBI end tone of the next phrase. We have considered trajectory-level acoustic models above for simplicity, but the above discussion also applies to speech parameter-level acoustic models, and indeed to full statistical models such as the autoregressive HMM.

To illustrate the phenomenon of future state blindness we generated a synthetic data set by adding white noise to an underlying signal and trained trajectory HMM and autoregressive HMM systems. The underlying signal was a sum of logistic sigmoids and was 0.5 s long at a frame rate of 200 frames per second, and the white noise had standard deviation 0.1. This signal, with smooth transitions between steady values, is the sort of signal that the mean trajectory of the trajectory HMM is very good at capturing, making this a good test case for investigating the weaknesses of the autoregressive HMM. We generated a training corpus consisting of 100 instances of the noisy signal, and a test corpus of 50 instances of the noisy signal. We then trained a trajectory HMM system with standard windows and two autoregressive HMM systems of depth 3 on this data. In all cases we used fixed manually-chosen alignments.

Before discussing the results of this experiment, it is helpful to establish some terminology that will be used throughout this chapter. A trained trajectory-level acoustic model $\mathbb{P}(c \,|\, \theta, \lambda)$ specifies a distribution over the trajectory c for each state sequence θ. We refer to this distribution as the *predictive distribution*, refer to its covariance matrix Σ as the *predictive covariance*, refer to the diagonal of Σ as the *predictive variance*, and refer to the square root of the predictive variance at time t as the *predictive standard deviation* at time t. This terminology avoids potential confusion between the predictive variance, which is the variance of an observed quantity, and variance parameters in the model such as (the reciprocal of) τ_{qd} for the trajectory HMM acoustic model and σ_q^2 for the LGLAR acoustic



(a) true distribution

(b) trajectory HMM

(c) autoregressive HMM

(d) autoregressive HMM with additional labels

Figure 5.1: Illustration of the effect of the autoregressive HMM's future blindness, and how it can be mitigated to an extent by adding extra labels. Each panel shows the mean trajectory of a distribution, a shaded area ±1.5 predictive standard deviations around the mean trajectory, and a trajectory sampled from the distribution.

model, which have only an indirect effect on the variance of an observed quantity.

The fixed alignments, the true distribution and the learned distributions are shown in Figure 5.1. In each panel the mean trajectory is shown together with a region ±1.5 predictive standard deviations around the mean trajectory. The test set log probabilities for the four models were: 0.88 nats per frame for the true distribution, 0.77 nats for the trajectory HMM, 0.58 nats for the autoregressive HMM, and 0.77 nats for the autoregressive HMM with additional labels. We can see from Figure 5.1(b) that even when the alignment only has a few labels, the trajectory HMM is able to capture the true distribution reasonably accurately. We can see from Figure 5.1(c) that the autoregressive HMM with the same alignment struggles to capture the true distribution, with reasonable transitions into the steady values but abrupt transitions out of them. This is reflected in the lower test set log probability of this system compared to the trajectory HMM. The abrupt transitions out

of the steady values are due to future state blindness: at 0.22 s the model has no way of knowing that there is about to be a state transition, for example, and even if it did, it still would not know which trajectory value to aim for. In contrast, for the trajectory HMM the trajectory value at time t can be seen as the result of a kind of negotiation between values that are good for the previous few states and values that are good for the next few states, and so it captures this smooth transition well. However we can see from Figure 5.1(d) that by adding extra labels in the regions where this future state blindness effect is most severe, namely just before transitions, the autoregressive HMM is able to capture the true distribution much better: the learned mean trajectory is closer to the true mean trajectory with smoother transitions between steady values, the predictive variance is smaller and closer to the true predictive variance, and the test set log probability suggests that this model fits the data as well as the trajectory HMM.

Note that none of the models is capable of capturing the true distribution exactly, except in the limit of using as many labels as there are frames. All the models introduce spurious correlations over time, and some of the aspects of the true distribution which they cannot capture they explain as extra noise, which manifests itself as a larger predictive standard deviation than the true distribution. The imperfectness of the models is also reflected in the fact that their test set log probabilities are smaller than that of the true distribution.

There a number of ways the autoregressive HMM used in the experiments in Chapter 4 might already be partially compensating for future state blindness. The use of full-context labels, and in particular quinphones, means that the model does already have some information about future phonemic and linguistic events. If the autoregressive HMM is taking advantage of this extra phonemic context to partially overcome this weakness we might expect that the decision trees learned for the autoregressive HMM would use more questions about future phonemes, and fewer questions about past phonemes, than the trees for the standard HMM synthesis framework. In §5.4.6 we will see that this is indeed the case. For a similar reason we might expect future state blindness to be a more serious problem when using monophone labels than when using full-context labels, though we do not investigate this here. In the toy example above we saw that adding extra labels can sometimes help. In the case of the models used for speech this would correspond to adding extra sublabels. However we saw in §4.4.2 that adding extra sublabels does not improve the performance of the LGLAR HMM. This may be partly because using more sublabels increases the number of parameters, which will increase the amount of overfitting. Additionally there is a minimum duration of 1 frame for each instance of a sublabel, and around half of sublabel instances are already at this minimum value[1], and so adding extra sublabels may cause bad alignments by forcing two or three neighbouring sublabels to together take more frames than

[1]For median alignments from both the standard HMM synthesis framework system S and the autoregressive system A below, the proportion of sublabel instances lasting 1 frame is 45%.

the acoustics warrant. It is therefore possible that adding extra sublabels would improve performance if the minimum duration constraint were removed to allow sublabels to sometimes have a duration of 0 frames. Finally we might expect that expectation maximization with full-context autoregressive models would shift label and sublabel boundaries earlier in time, to effectively allow the model to see into the future a bit further than it otherwise could. For example it seems likely that if the autoregressive system in Figure 5.1(c) was allowed to choose its own alignment then it would shift the boundaries further left and that this would result in a higher TSLP. However in §5.4.6 we will see that for real systems this only happens to a small extent. This suggests that either expectation maximization finds a local maximum that it is difficult to move away from, or that for substantial shifts of the label and sublabel boundaries the positive consequences of knowing more about the future are outweighed by the negative consequences of knowing less about the present. From the above discussion we can conclude that the autoregressive HMM used in the experiments in Chapter 4 does already compensate for future state blindness in a limited way.

It might also be possible to mitigate the effects of future state blindness by extending the model used while staying within the autoregressive HMM framework. For example we could add questions about the number of frames remaining in the current sublabel to the decision tree clustering process. This is a simple example of the general idea of adding information about future phonemic and linguistic events to the present state as a way to make the mitigate the effect of future state blindness. We will see in §5.3.2 that in a certain sense the ability to look at future states is all that distinguishes the trajectory HMM and the LGLAR HMM. We could also consider using a more advanced regression model than the decision tree-based linear Gaussian linear regression model used by the LGLAR HMM. As noted in Chapter 4 the flexibility in the autoregressive HMM framework allows a variety of regression models to be used. The decision tree-based approach provides a relatively coarse way to access the information about future phonemic and linguistic events encoded in the current state. A more advanced or differently parameterized regression model might conceivably make better use of this information, and so partially mitigate the effects of future state blindness.

Finally we mention a phenomenon observed in conditional sequential models defined over discrete spaces which is closely related to future state blindness. A (linear chain) *conditional random field (CRF)* (Lafferty et al., 2001) is a conditional model $\mathbb{P}(l \,|\, o, \lambda)$ over a sequence l of labels given a sequence o of observations of the same length T. Typically the set of possible labels is discrete and the set of possible observations may be discrete or continuous. A conditional random field builds up the overall conditional pdf by taking the product of a collection of experts, each of which is local to some subsequence of the overall (observation, label) sequence, and then normalizing this product conditional on the observation sequence. It is thus closely related to the trajectory HMM acoustic model, and

indeed the trajectory HMM acoustic model may be viewed as a CRF with continuous labels and discrete observations: the CRF label is the trajectory value and the CRF observation is the state. Conditional random fields have been compared to *maximum entropy Markov models (MEMMs)* (McCallum et al., 2000), which are also models over a sequence of labels given a sequence of observations of the same length, but which are autoregressive and locally normalized. They are thus analogous to an autoregressive HMM acoustic model. MEMMs suffer from an analogue of future state blindness often known as the *label bias problem* (Lafferty et al., 2001): for an MEMM $\mathbb{P}(l \,|\, o, \lambda)$, the marginal distribution $\mathbb{P}(l_t \,|\, o, \lambda)$ over the label at index t given all observations does not depend on future observations $o_{t+1:T}$. If the true marginal distribution $\mathbb{P}_{\text{true}}(l_t \,|\, o)$ does depend strongly on these future observations, then it is not possible to accurately represent the overall true distribution $\mathbb{P}_{\text{true}}(l \,|\, o)$ with an MEMM. Conditional random fields have been found to perform better than maximum entropy Markov models on some tasks (Lafferty et al., 2001; Seigel et al., 2013).

5.2 A mild generalization of the trajectory HMM

In this section we introduce a very slight generalization of the trajectory HMM acoustic model presented in §3.3.2. Considering this new model will simplify the discussion in several places in the remainder of this chapter.

We first recap certain aspects of the trajectory HMM acoustic model defined in §3.3.2. To fully define the trajectory HMM acoustic model it is necessary to decide how to cope with end effects, as discussed in detail in Appendix B. One approach, which we favour for its simplicity, is to condition on the before-the-start trajectory values $c_{1-K:0}$ and after-the-end trajectory values $c_{T+1:T+K}$ having fixed, known values. This is referred to as solution 3 in Appendix B, or solution 2 if the conditioned-on values are zero.

The new model may be defined as follows. Consider a trajectory HMM acoustic model which is defined over a trajectory $c_{1-K:T+K}$ given a corresponding state sequence $\theta_{1-K:T+K}$. Suppose the window left extent K^{L} and the window right extent K^{R}, as defined in §3.2.2, are both non-negative, and that K above is equal to $K^{\text{L}}+K^{\text{R}}$. If we condition on $c_{1-K:0}$ and $c_{T+1:T+K}$ having known values, for example zero, then we obtain a trajectory-level acoustic model over $c_{1:T}$ which depends on a portion $\theta_{1-K^{\text{R}}:T+K^{\text{L}}}$ of the original state sequence; this is our new model. A factor graph which describes the new model is shown in Figure 5.2.

The new model is almost identical to the trajectory HMM acoustic model described in §3.3.2 using solution 2 or solution 3 to cope with end effects, but is slightly more general, since we now have dependence on additional states just before θ_1 and just after θ_T. These correspond to additional experts in the product-of-experts view of the trajectory HMM acoustic model. The model defined in §3.3.2 can be recovered from the new model by

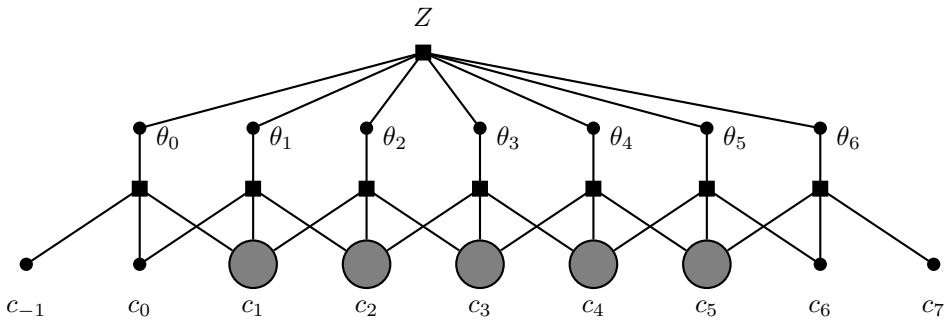


Figure 5.2: A factor graph which describes a mild generalization of the trajectory HMM acoustic model with left extent $K^{\mathrm{L}} = 1$ and right extent $K^{\mathrm{R}} = 1$. Here the state sequence $\theta = \theta_{0:6}$ is conditioned on and the trajectory $c = c_{1:5}$ is observed. The values c_{-1}, c_0, c_6 and c_7 are deterministic acoustic context, equal to zero when using zero-input windows. The factor Z is a normalization constant. Each factor also depends on the model parameters λ (not shown).

setting the additional states to some new value in a new leaf q which has $b_{qd} = \tau_{qd} = 0$ for all windows d. It could be argued that the model obtained in this way is conceptually and mathematically more natural than the original model, and thus that the dependence on states before θ_1 and after θ_T is in a sense "missing" from the original definition in §3.3.2.

One benefit of the new model is that it helps to clarify the relationship between the state sequence and time. Throughout the thesis we have associated a given state in the state sequence, say θ_t, with a given frame t. However the timing of a given state is not directly observed; it affects something observable only through its interaction with the trajectory. Thus which frame we associate a given state in the state sequence with is essentially arbitrary. This can be expressed by a kind of symmetry the model should obey: if we shift the state sequence one frame earlier and shift all the window coefficients one frame later then we should end up with the same model, modulo end effects. The new model has the advantage that it makes this symmetry precise, and it holds even taking end effects into account (as long as the original and resulting window left and right extents are both non-negative). This also means that, by shifting the state sequence appropriately, we can assume without loss of generality that the window right extent K^{R} is zero. This has the advantage of removing some superficial differences between the trajectory HMM acoustic model and the LGLAR acoustic model, simplifying the presentation of various theoretical arguments below such as those in §5.3.2.

The above discussion can be extended to the trajectory HMM described in §3.3.2, as opposed to just the trajectory HMM acoustic model, as long as each portion p and vector component i uses windows with the same left and right extents. In converting a trajectory HMM of the form in §3.3.2 to a trajectory HMM of the new form, we can if desired use

the duration model to specify that the duration of the new state at the start of the state sequence lasts precisely a certain number of frames, and similarly for the (distinct) new state at the end of the state sequence. The symmetry above also still holds, and means that for a trajectory HMM of the new form the only possible difference between a trained model with shifted windows and a trained model with the original windows is in the choice of initial alignments and initial model parameters, and even these can be chosen in such a way that the trained models are equivalent.

5.3 Two unified views of the trajectory HMM and autoregressive HMM

The trajectory HMM acoustic model presented in §3.3.2 and the LGLAR acoustic model presented in §4.3.1 appear to be very different models at first glance. In this section we present two ways of viewing the two models within a common framework. As we will see in §5.4, these theoretical viewpoints highlight both the similarities and the differences between the two models, and provide new perspectives when comparing them.

5.3.1 Decomposition into local contributions

In this section we show that, for both the trajectory HMM acoustic model and the LGLAR acoustic model, the b-value and precision matrix may be decomposed as the sum of overlapping local contributions, where the difference between the two models is in the form of the local contributions. As we saw in §3.3.1, the trajectory HMM acoustic model is also the model effectively used at synthesis time by the standard framework, so this unified view also applies to the synthesis-time part of the standard framework. This unified view was first presented in a technical report (Shannon and Byrne, 2009b).

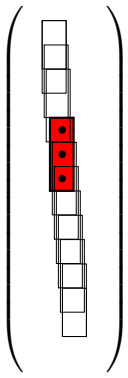
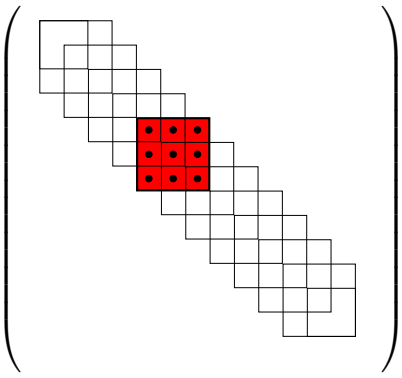
Given a state sequence $\theta = [\theta_t]_{t=1}^T$, the trajectory HMM acoustic model in §3.3.2 and the LGLAR acoustic model in §4.3.1 both specify a Gaussian distribution over the trajectory $c = [c_t]_{t=1}^T$. We claim that the natural parameters $b(\theta;\lambda)$ and $P(\theta;\lambda)$ for both models can be written as

$$b_s(\theta;\lambda) = \sum_u b^{\mathrm{LC}}_{(s-u)}\left(\overline{q}(\theta_u);\lambda\right), \quad s \in \{1,\ldots,T\} \tag{5.1}$$

$$P_{st}(\theta;\lambda) = \sum_u P^{\mathrm{LC}}_{(s-u)(t-u)}\left(\overline{q}(\theta_u);\lambda\right), \quad s,t \in \{1,\ldots,T\} \tag{5.2}$$

where λ is the collection of model parameters which includes a clustering function $\overline{q}$ as defined in §2.3. We refer to $b^{\mathrm{LC}}(q;\lambda)$ and $P^{\mathrm{LC}}(q;\lambda)$ as *local contributions*. Thus for both models the overall b-value is the sum of overlapping local contributions, where each local contribution depends only on the state θ_u at some time u, and similarly for the precision

matrix. The decompositions into local contributions given by (5.1) and (5.2) can be shown schematically as follows:

$b =$  $P =$  (5.3)

The elements of the precision matrix and b-value affected by one particular local contribution, that is by the local contribution for one particular element θ_u of the state sequence, are indicated by small black circles with a dark or red background. The outlines of the local contributions for other elements of the state sequence are shown as thin black rectangles. The b-value rectangles have a small horizontal offset to enable them to be seen more clearly.

For the trajectory HMM acoustic model in §3.3.2 with parameters $\lambda = ([b_{qd}, \tau_{qd}]_{q,d}, [w_{dk}]_{d,k}, \overline{q})$ the local contributions are

$$b_k^{\mathrm{LC}}(q;\lambda) = \sum_{d=0}^{D-1} b_{qd} w_{dk}, \quad k \in \{-K^{\mathrm{L}}, \ldots, K^{\mathrm{R}}\} \tag{5.4}$$

$$P_{kl}^{\mathrm{LC}}(q;\lambda) = \sum_{d=0}^{D-1} \tau_{qd} w_{dk} w_{dl}, \quad k,l \in \{-K^{\mathrm{L}}, \ldots, K^{\mathrm{R}}\} \tag{5.5}$$

where K^{L} and K^{R} are the left and right extents of the collection of windows. We can verify (5.1), (5.2), (5.4) and (5.5) for the trajectory HMM acoustic model by writing out (3.39) and (3.40) componentwise. Setting $K = K^{\mathrm{L}} + K^{\mathrm{R}}$, $b^{\mathrm{LC}}(q;\lambda)$ is a $(K+1)$-dimensional vector and $P^{\mathrm{LC}}(q;\lambda)$ is a $(K+1)\times(K+1)$ matrix. As described in §3.2.2 the window coefficients used for the first few and last few frames, corresponding to the first few and last few rows of the window matrix W, are adjusted. This means that none of the local contributions above extend before the first frame 1 or after the last frame T. Note that $P^{\mathrm{LC}}(q;\lambda)$ is a leaf-specific linear combination of a set of fixed matrices, where each fixed matrix is the outer product of a vector of window coefficients with itself. Thus $P^{\mathrm{LC}}(q;\lambda)$ typically has rank D.

In the paragraph above we assumed the form of trajectory HMM acoustic model in §3.3.2. For the mild generalization presented in §5.2, there are additional local contributions corresponding to the additional states. For example in the $K^{\mathrm{L}} = K^{\mathrm{R}} = 1$ case, both models have contributions from frame 1 to frame 3 and from frame 1 to frame 2, but only the model in §5.2 has a contribution from frame 1 to frame 1.

For the LGLAR acoustic model with parameters $\lambda = ([[a_{qk}]_{k=1}^{K+1}, \sigma_q^2]_{q=1}^{Q}, \overline{q})$ the local contributions are

$$b_k^{\mathrm{LC}}(q) = -\tilde{a}_{q(K+1)}\tilde{a}_{q(-k)}, \quad k \in \{-K, \ldots, 0\} \tag{5.6}$$

$$P_{kl}^{\mathrm{LC}}(q) = \tilde{a}_{q(-k)}\tilde{a}_{q(-l)}, \quad k, l \in \{-K, \ldots, 0\} \tag{5.7}$$

where K is the depth, and the $\tilde{a}_{qk}$ values are using the $\tilde{a}$ parameterization described in §2.2.4 for the linear regression model for leaf q. We can verify (5.1), (5.2), (5.6) and (5.7) for the LGLAR acoustic model by writing out $b = L^{\mathsf{T}}\xi$ and $P = L^{\mathsf{T}}L$ componentwise and using the definitions of L and ξ given in §2.2.5 and §4.3.3. For a depth K model, $b^{\mathrm{LC}}(q;\lambda)$ is a $(K+1)$-dimensional vector and $P^{\mathrm{LC}}(q;\lambda)$ is a $(K+1) \times (K+1)$ matrix. Some of the local contributions extend before the first frame, and one way to cope with this is to simply truncate $b^{\mathrm{LC}}(q;\lambda)$ and $P^{\mathrm{LC}}(q;\lambda)$, corresponding to assuming the initial acoustic context is equal to zero. Note that $P^{\mathrm{LC}}(q;\lambda)$ is the outer product of a leaf-specific vector with itself. Thus $P^{\mathrm{LC}}(q;\lambda)$ has rank 1.

Given the viewpoint described above, a natural generalization of both the trajectory HMM acoustic model and the LGLAR acoustic model would be to allow $P^{\mathrm{LC}}(q;\lambda)$ to be the sum of a small collection of outer products of leaf-specific vectors, or even an arbitrary positive semi-definite matrix. We do not evaluate this generalization in this thesis.

The view in terms of local contributions presented here is closely related to the view of the trajectory HMM as a product-of-experts model described in §3.3.2. Any product-of-experts model where the experts have Gaussian dependence on the trajectory values has an overall distribution over trajectories whose precision matrix and b-value are built up additively from contributions. If each expert depends only on a single element θ_t of the state sequence, and is local in the sense that it depends only on trajectory values $c_{t-A:t+B}$ for some fixed non-negative integers A and B, then the precision matrix is banded and built-up from local contributions in the same way as above.

5.3.2 Viewing the trajectory HMM acoustic model autoregressively

In this section we describe a novel way to view both the trajectory HMM acoustic model and the LGLAR acoustic model as instances of a certain class of generalized autoregressive acoustic model. After establishing terminology, we state and prove our main result, then show how this allows the trajectory HMM to be written as a directed graphical model. For clarity of presentation we state our results for the trajectory HMM acoustic model with right extent $K^{\mathrm{R}} = 0$. As discussed in §5.2 this involves no fundamental loss of generality, though allowing general right extent would necessitate minor changes to the definitions and results below.

We first establish some terminology. In §4.3.1 we defined an autoregressive acoustic model as a trajectory-level acoustic model $\mathbb{P}(c \,|\, \theta, \lambda)$ which satisfies

$$\mathbb{P}(c \,|\, \theta, \lambda) = \prod_{t=1}^{T} \mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda) \tag{5.8}$$

for some fixed initial acoustic context $c_{-(K-1):0}$ and non-negative integer K. We define a *generalized autoregressive acoustic model* as a trajectory-level acoustic model which satisfies

$$\mathbb{P}(c \,|\, \theta, \lambda) = \prod_{t=1}^{T} \mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_{t:T+K}, \lambda) \tag{5.9}$$

The difference between these two definitions is in how much of the state sequence θ is "available" to the model when predicting c_t from $c_{t-K:t-1}$. Note that it is possible to view a generalized autoregressive acoustic model as an autoregressive acoustic model defined on an expanded state space: defining the *generalized autoregressive state* $\tilde{\theta}_t = \theta_{t:T+K}$, the generalized autoregressive acoustic model with state sequence θ is an autoregressive acoustic model with state sequence $\tilde{\theta}$.

The main result of this section is that the trajectory HMM acoustic model in §5.2 with right extent 0 is a generalized autoregressive acoustic model where $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_{t:T+K}, \lambda)$ is linear Gaussian in c. Note that the distribution $\mathbb{P}(c \,|\, \theta, \lambda)$ over c for a given θ and λ is Gaussian with banded precision matrix, and so, following the discussion in §2.2.5, it can be written as a composite linear Gaussian autoregressive distribution. Thus

$$\mathbb{P}(c \,|\, \theta, \lambda) = \prod_{t=1}^{T} \mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta, \lambda) \tag{5.10}$$

where $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta, \lambda)$ is linear Gaussian in c. The remaining part of the claim is that the parameters of the linear Gaussian distribution $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta, \lambda)$ at time t do not depend on $\theta_{1:t-1}$. This can be proved in a number of ways. Firstly, the states $\theta_{1:t-1}$ only affect the block $P_{(1:t-1)(1:t-1)}$ of the precision matrix P. For the alternative Cholesky decomposition $P = L^{\mathsf{T}}L$, changing matrix entries in the block $P_{(1:t-1)(1:t-1)}$ of the original matrix only affects matrix entries in the block $L_{(1:t-1)(1:t-1)}$ of the Cholesky factor. This may be verified by writing $P = L^{\mathsf{T}}L$ as a block matrix equation. Thus the autoregressive coefficients and conditional standard deviation at time t, which are given by the t^{th} row of L, do not depend on $\theta_{1:t-1}$. Similarly, by noting that $\theta_{1:t-1}$ only affects the block $b_{1:t-1}$ of the b-value vector and writing $b = L^{\mathsf{T}}\xi$ as a block matrix equation, it may be verified that the autoregressive bias at time t does not depend on $\theta_{1:t-1}$. This establishes the desired result. A second way to see the claimed result uses the language of *factor graphs* (Bishop, 2006, section 8.4.3). The trajectory HMM acoustic model in §5.2 with right extent 0 is of

the form

$$\mathbb{P}(c \,|\, \theta, \lambda) = \frac{1}{Z(\theta, \lambda)} \prod_{t=1}^{T+K} w(c_{t-K:t}, \theta_t, \lambda) \tag{5.11}$$

for some function $w : \mathbb{R}^K \times \Psi \times \Xi \to \mathbb{R}$, where Ψ is the state space and Ξ is the space of allowed parameters. This is essentially just the product-of-experts view of the trajectory HMM acoustic model that was presented in §3.3.2, but grouping the DT experts into T experts. The *Markov blanket* of $c_{t:T}$ in the factor graph specified by (5.11) is $(c_{t-K:t-1}, \theta_{t:T+K}, \lambda)$, so $\mathbb{P}(c_{t:T} \,|\, c_{1:t-1}, \theta, \lambda) = \mathbb{P}(c_{t:T} \,|\, c_{t-K:t-1}, \theta_{t:T+K}, \lambda)$. Thus by marginalizing over $c_{t+1:T}$ we have $\mathbb{P}(c_t \,|\, c_{1:t-1}, \theta, \lambda) = \mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_{t:T+K}, \lambda)$ as desired. Finally it is possible to see the claimed result by explicitly manipulating (5.11) using elementary rules of probability. Full details of this approach have previously been given in a technical report (Shannon and Byrne, 2012). The explicit approach shows that

$$\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_{t:T+K}, \lambda) = \frac{w(c_{t-K:t}, \theta_t, \lambda)\beta(c_{t-K+1:t}, \theta_{t+1:T+K}, \lambda)}{\beta(c_{t-K:t-1}, \theta_{t:T+K}, \lambda)} \tag{5.12}$$

where β is defined by

$$\beta(c_{t-K:t-1}, \theta_{t:T+K}, \lambda) = \int \prod_{u=t}^{T+K} w(c_{u-K:u}, \theta_u, \lambda) \, \mathrm{d}c_{t:T} \tag{5.13}$$

and can be computed recursively using

$$\beta(c_{t-K:t-1}, \theta_{t:T+K}, \lambda) = \int w(c_{t-K:t}, \theta_t, \lambda)\beta(c_{t-K+1:t}, \theta_{t+1:T+K}, \lambda) \, \mathrm{d}c_t \tag{5.14}$$

with the initial value given by

$$\beta(c_{T-K+1:T}, \theta_{T+1:T+K}, \lambda) = \prod_{u=T+1}^{T+K} w(c_{u-K:u}, \theta_u, \lambda) \tag{5.15}$$

For concreteness we associate $\beta(c_{t-K:t-1}, \theta_{t:T+K}, \lambda)$ with frame $t-1$. By a slight abuse of notation we sometimes write $\beta_{t-1}(c_{t-K:t-1})$ instead of $\beta(c_{t-K:t-1}, \theta_{t:T+K}, \lambda)$ when the state sequence θ and model parameters λ are clear from context. The log quadratic function β_t can be thought of as encoding the collective preferences of future states regarding the value of the trajectory near frame t. Thus in the autoregressive view of the trajectory HMM acoustic model, the β_t values pass back information about the preferences of future states, and it is using this information to inform the autoregressive distribution at the current time that allows the trajectory HMM acoustic model to avoid suffering from future state blindness. The fact that the claimed result can be viewed either in terms of a Cholesky decomposition or in terms of a β recursion is not all that surprising: the β recursion presented above is akin to Kalman filtering backwards in time, and the connection between Kalman filtering and the Cholesky decomposition is well known (Eubank and Wang, 2002).

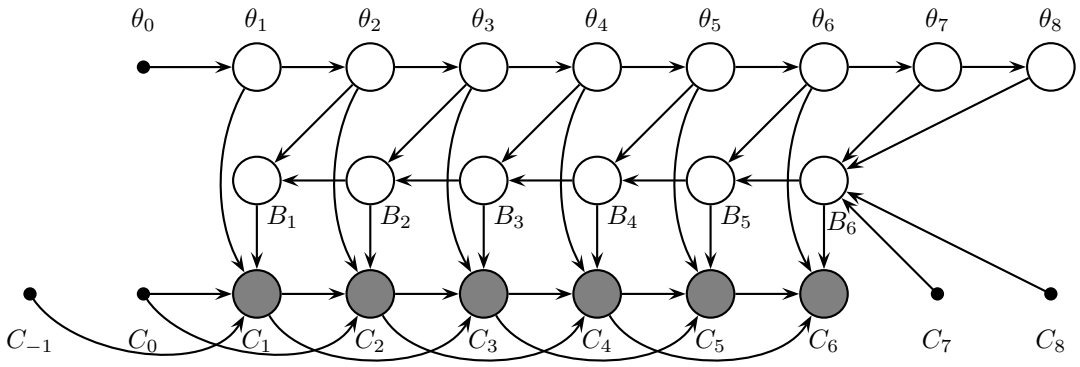


Figure 5.3: A directed graphical model equivalent to the trajectory HMM with window left extent $K^{\mathrm{L}} = 2$ and window right extent $K^{\mathrm{R}} = 0$, so $K = K^{\mathrm{L}} + K^{\mathrm{R}} = 2$. Here $\theta = [\theta_t]_{t=1}^{8}$ is the state sequence and $C = [C_t]_{t=1}^{6}$ is the speech parameter sequence. For each frame t, portion p and vector component i, B_{tpi} consists of the parameters of a log quadratic function $\mathbb{R}^2 \to \mathbb{R}$. Each B_t node is deterministic given its parents. The value θ_0 is a deterministic initial state. The initial acoustic context $C_{-1:0}$ and the final acoustic context $C_{7:8}$ are fixed. The nodes $\theta_{1:8}$ also depend on the label sequence l and the model parameters ν, and the nodes $B_{1:6}$ and $C_{1:6}$ also depend on the model parameters λ (not shown).

The introduction of the β values allows a new view of the trajectory HMM as a *directed* graphical model. To see this, first note that the β values can be computed recursively, with β_{t-1} computed from $(\beta_t, \theta_t, \lambda)$ using (5.14). The autoregressive parameters to use at frame t can be computed from $(\beta_t, \theta_t, \lambda)$ using (5.12). This means that the trajectory HMM acoustic model can be viewed as a directed graphical model where: β_{t-1} is a deterministic node with parents θ_t and β_t; the node c_t has parents θ_t, β_t and $c_{t-K:t-1}$; and the state sequence θ is conditioned on. In the case of a trajectory HMM, which consists of a Markovian state transition model together with a trajectory HMM acoustic model for each portion p and vector component i of the speech parameter sequence, we have a corresponding log quadratic β function for each frame t, portion p and vector component i, and we denote this log quadratic function, or equivalently its parameters, by B_{tpi}. This allows us to write the trajectory HMM as the directed graphical model shown in Figure 5.3. Note that we are free to choose the arrows between the state sequence nodes to be directed forwards or backwards; the procedure in §3.1.5 used to construct the state transition model from the duration model can be applied in either case. In contrast it is an essential feature of the model that the arrows between B nodes point in a different direction to the arrows between C nodes. The graphical model provides a simple way to see how the trajectory HMM effectively passes information about future states backwards through time.

The LGLAR acoustic model can of course also be written as a generalized autoregressive acoustic model. In this case the β values that get passed back are degenerate: they have a

zero vector for the b-value and a zero matrix for the precision. Thus no information about future elements of the state sequence is passed backwards to the present time, though it should be noted that the state θ_t does already encode some information about future phonemic events due to the use of full-context labels and the fact that the state includes the number of frames remaining in the current sublabel.

The above view of the trajectory HMM acoustic model as a generalized autoregressive acoustic model may be used to interpret any model built up from local contributions as a generalized autoregressive acoustic model, so in a sense the result presented in this section is more general than the local contribution result presented in §5.3.1. However the two results provide different insights into the relationship between the trajectory HMM acoustic model and the LGLAR acoustic model, and so we view them as complementary.

5.4 Investigation of theoretical and experimental differences

In this section we examine some of the differences between the standard framework, the trajectory HMM and the LGLAR HMM. Many of the subsections start by summarizing the theoretical differences between the three models, then investigating certain aspects of this difference further using a combination of theoretical arguments and experiments.

5.4.1 Consistency

The autoregressive HMM and trajectory HMM are both consistent in the sense that they use the same normalized model during training and synthesis. As discussed in §3.3.1 the standard HMM synthesis framework is inconsistent, with an unnormalized model effectively used during training. The lack of normalization in the standard model used during parameter estimation means that the probabilistic justification for conventional training procedures in terms of maximizing the likelihood strictly speaking does not apply. However it is an interesting question to what extent this lack of normalization has practical consequences. In this section we investigate this question, showing that the standard HMM synthesis framework greatly underestimates predictive variance. While the trajectory HMM and autoregressive HMM differ in many ways, the fact they are both consistent means they do not systematically underestimate predictive variance. Some of the work presented in this section was previously presented in a conference paper (Shannon et al., 2011).

A representation of the distribution over trajectories for the three models, for part of one test corpus utterance, is shown in Figure 5.4. For each system we plot the mean trajectory along with ± 1.5 predictive standard deviations, as we did for a synthetic data set in Figure 5.1. This gives some insight into the distribution over trajectories encoded

by the various models by showing the range of values each model expects to see at each time. We can see that the standard approach underestimates the variance: the natural trajectory is often outside the range predicted by the model, and we see a few events that are so many standard deviations from the mean that they should only happen *extremely* rarely according to the model. The normalized models have much larger variances, and this looks more reasonable: the natural trajectory is a long way outside the predicted range much less often.

The extremely low test set log probability of standard systems is mainly due to their predictive variance being too small. System SB2 in Table 4.4 artificially boosts the predictive variance by multiplying both the precision matrix P and b-value b by a factor of $\frac{1}{3}$, which is equivalent to multiplying the trajectory covariance by a factor of 3 while leaving the mean trajectory unaltered. This increases the approximate mcep-only test set log probability from 29.12 nats to 46.94 nats. This is strong evidence that the standard approach systematically underestimates predictive variance as Figure 5.4 suggests and as we claimed in §4.6.3.

In fact it is analytically tractable to compute the *optimal* variance boost, meaning the variance boost which gives the highest log probability on a given corpus. For each vector component of the speech parameters the predictive distribution is a Gaussian with mean trajectory μ and precision matrix P depending on the state sequence θ and model parameters λ. A system with variance boost $1/k$ has mean trajectory μ and precision matrix kP, and so for a trajectory c has log pdf

$$\log \mathbb{P}(c \,|\, \theta, \lambda) = -\tfrac{T}{2} \log 2\pi + \tfrac{1}{2} \log \det(kP) - \tfrac{1}{2}(c-\mu)^{\mathsf{T}}(kP)(c-\mu) \tag{5.16}$$

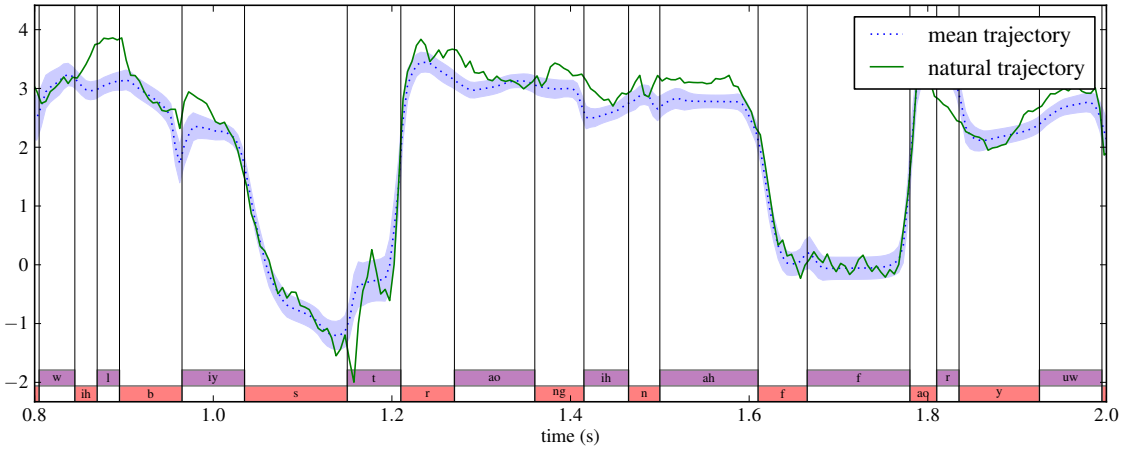
$$= -\tfrac{T}{2} \log 2\pi + \tfrac{1}{2} \log \det P + \tfrac{T}{2} \log k - \tfrac{1}{2} k \Big((c-\mu)^{\mathsf{T}} P (c-\mu) \Big) \tag{5.17}$$

Thus the optimal variance boost is given by

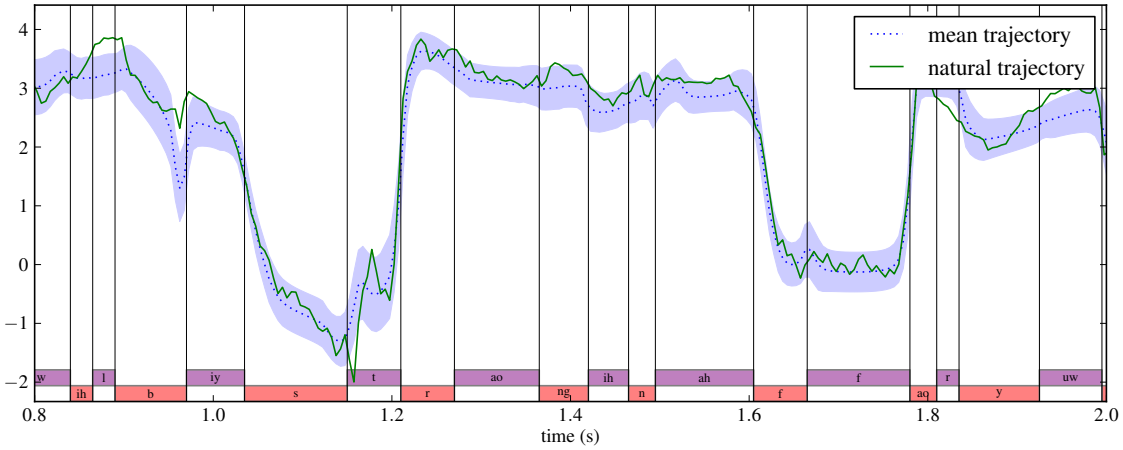
$$\frac{1}{\hat{k}} = \frac{1}{T}(c-\mu)^{\mathsf{T}} P (c-\mu) \tag{5.18}$$

The optimal variance boosts for the standard system S2 on the training corpus assuming fixed median alignments are shown in Figure 5.5. We can see that the optimal variance boost for all cepstral components is around 3. Using the precise optimal variance boost rather than the simple heuristic value of 3 as we did for system SB2 makes very little difference, increasing the median-alignment mcep-only TSLP from 46.94 nats to 46.95 nats.

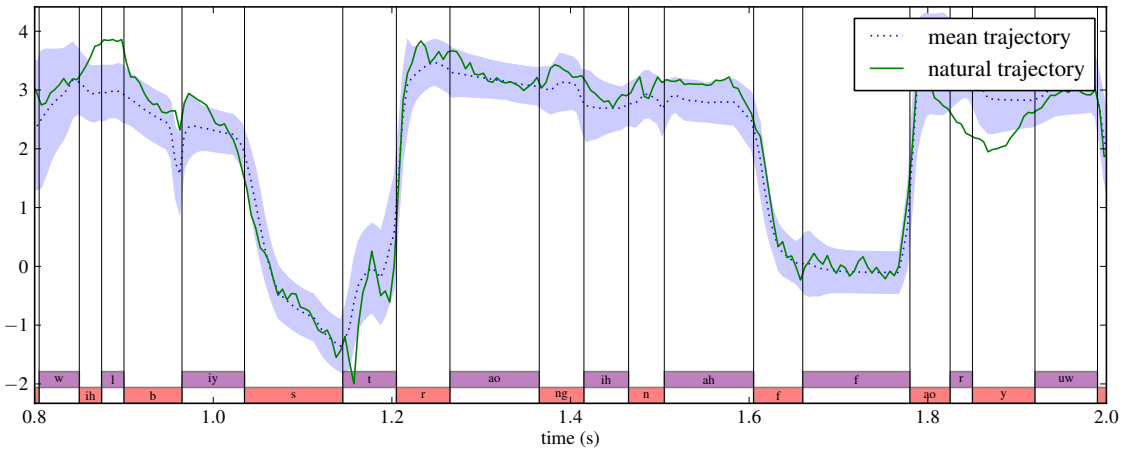
For a standard system with 4 windows that we trained, the optimal variance boost was around 4 for all cepstral components, and we hypothesize that the optimal variance boost for a standard system with n windows is often roughly n. Note that in the simple case where we take D copies of a trained probabilistic model $\mathbb{P}(c \,|\, \theta, \lambda)$ which is Gaussian in c and combine them in a product-of-experts framework, the optimal variance boost is



(a) standard HMM synthesis framework (system S2)



(b) trajectory HMM (system T2)



(c) autoregressive HMM (system A2)

Figure 5.4: Visualization of the distribution over trajectories for a test utterance for the 1$^{\text{st}}$ mel cepstral coefficient for each of the three models, together with the natural trajectory actually generated by the speaker. A median alignment was used for each system to allow visual comparison with the natural trajectory.

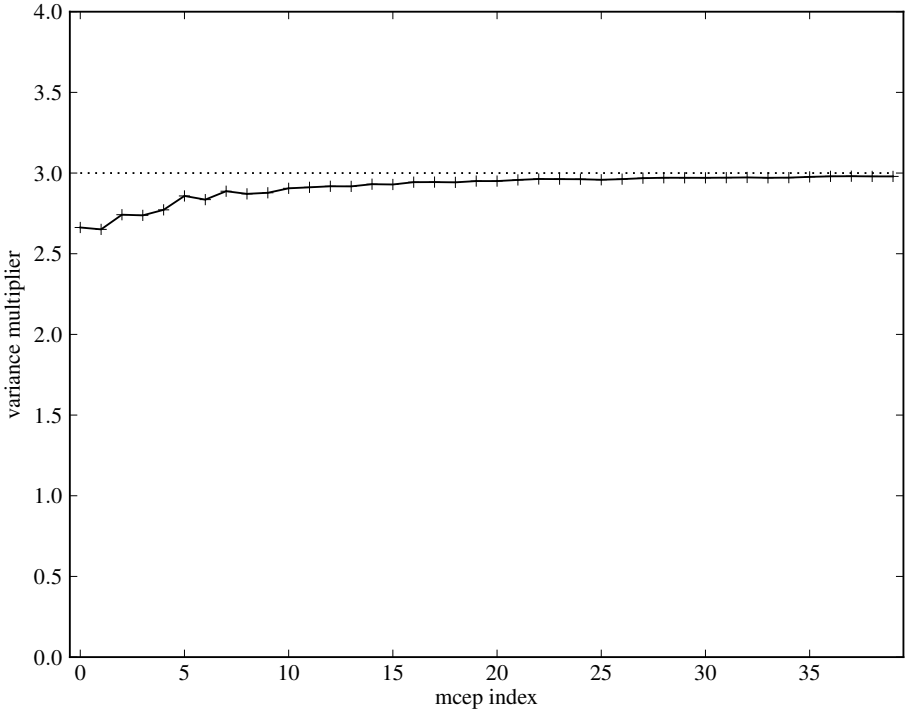


Figure 5.5: The optimal variance boost for the standard system S2 for each component of the spectral portion of the speech parameters. The values are optimal in the sense of giving the highest log probability on the training corpus assuming a fixed alignment. The dotted line is at a value of 3, which is an upper bound on the optimal variance boost.

precisely D. In fact it is possible to prove that if a standard system is trained using a fixed alignment then the optimal variance boost computed on the training set will always be less than or equal to the number of windows. We show this in Appendix A.

Note that while the simple variance boost presented above fixes the largest pathology in the predictive covariance of the standard HMM synthesis framework, the inconsistent training also has other, subtler deleterious effects, as we can see from the fact that the approximate mcep-only TSLP of system SB2 is still a long way behind that of system T2 and system A2. In contrast the inconsistent training used by the standard framework appears to have a more modest impact on the accuracy of the mean trajectory, as evidenced by the relatively small difference in MCD between the standard and trajectory systems.

5.4.2 Computational efficiency of parameter estimation

Parameter estimation for the autoregressive HMM and the standard HMM synthesis framework is more computationally efficient than for the trajectory HMM, and in two distinct ways. Firstly, for the simplest form of training assuming a fixed state sequence, the autoregressive HMM and standard HMM synthesis framework have simple closed form solutions for the maximum likelihood parameters. Part of the tractability of the maximum likelihood solution for these models comes from the fact that the log likelihood function can be decomposed as a sum of terms where each term depends only on the parameters for one leaf, and so the overall maximization problem is separable in the sense that it can be solved by solving a simpler maximization problem for each leaf. Having a log likelihood function which decomposes in this way is a common property of directed graphical models. In contrast the trajectory HMM does not have a closed form solution for the variance parameters, and requires a gradient ascent scheme to optimize these (Zen et al., 2007b). The mean parameters do have a closed form solution, but it is not separable over the parameters for different states and involves solving a potentially large and dense system of linear equations. Secondly, for the autoregressive HMM and standard HMM synthesis framework the distribution $\mathbb{P}(C, \theta \,|\, l, \nu, \lambda)$ factorizes over time with respect to the state sequence θ, which allows the Viterbi and forward-backward algorithms to be used. In contrast the trajectory HMM must resort to an approximate delayed decision Viterbi decoder for alignment. The above two points, together with the existence of sufficient statistics for the Gaussian distribution and the linear regression model, mean that the standard HMM synthesis framework and the LGLAR HMM both support efficient re-estimation using expectation maximization and efficient decision tree clustering whereas the trajectory HMM does not.

It should be noted that the LGLAR HMM can be less efficient during training than the standard HMM synthesis framework if very large depths are used. Accumulation requires $O(K^2)$ memory and the M step of re-estimation requires $O(K^3)$ time where K is the depth, in contrast to the acoustic model used during training by the standard HMM synthesis framework, which requires $O(D)$ memory and time where D is the number of windows. However for typical depths of $K = 2$ or $K = 3$ this effect is not substantial.

Given that one of the advantages of the autoregressive HMM over the trajectory HMM is that it supports expectation maximization and decision tree clustering, it is interesting to ask how much benefit the final trained autoregressive system derives from using these parameter estimation methods. Indirectly this also tells us something about how much the trajectory HMM system might be losing by reusing the trees and alignments from the standard system. To investigate this question experimentally, we trained autoregressive systems which did not take full advantage of the training procedures available for the autoregressive HMM. The systems are shown in Table 5.1. System AF used the same

system	update trees	update alignments	description
AF	×	×	LGLAR HMM with fixed trees and alignments
AFC	✓	×	AF followed by decision tree clustering
AE	×	✓	AF followed by expectation maximization
A2	✓	✓	AF followed by multiple rounds of clustering and EM

Table 5.1: Systems used to investigate the benefit gained from using the efficient training procedures available for the autoregressive HMM.

system	mcep leaves	MCD DTW (dB)	using median alignments	
			using sys	mcep TSLP (nats)
AF	812	5.74	S	47.84
			AF	47.97
AFC	960	5.73	S	47.83
			AFC	48.01
AE	812	5.75	AE	47.98
A2	964	5.74	A2	48.02

Table 5.2: An evaluation of the benefit gained from using the efficient training procedures available for the autoregressive HMM.

approach used to train the trajectory HMM systems T and T2 in §4.6, namely using the trees from system S and fixed median alignments from system S. System AFC used fixed standard alignments but did decision tree clustering, whereas system AE used the tree from system S but updated the parameters with EM using system AF as an initialization. System A2 was described in §4.6.4 and uses several rounds of autoregressive clustering and EM. It can be viewed as system AFC followed by four iterations of EM, clustering, and four more iterations of EM. The experimental set-up otherwise was the same as used in §4.6, with the model structure parameters chosen to give good TSLP: 5 sublabels, depth 3 autoregressive modelling for the spectral and aperiodicity portions of the speech parameters, and an MDL tuning factor of 0.3 used during autoregressive clustering. The results are shown in Table 5.2. We can see that:

- The system trained using fixed standard alignments and trees (AF) does very well, with the same MCD as the fully autoregressive system (A2) and only slightly worse TSLP (47.97 nats instead of 48.02 nats).
- Expectation maximization on its own (system AE) makes almost no difference by either of the metrics, while decision tree clustering on its own (system AFC) achieves a very small gain in TSLP (48.01 nats instead of 47.97 nats).

- Performing multiple iterations of clustering and expectation maximization (system A2) makes almost no difference by either metric compared to just doing autoregressive clustering (system AFC).

We can thus conclude that, when a good initial alignment and good initial trees from a standard system are used, autoregressive decision tree clustering and expectation maximization have only a small effect. While it is not possible to directly extrapolate from the autoregressive HMM case to the trajectory HMM case, this suggests that, for good initial alignments and trees, the trajectory HMM systems may not lose too much from not being able to do expectation maximization and decision tree clustering.

Table 5.2 also allows us to investigate the effect of doing re-alignment during evaluation. For the systems which use a fixed alignment from system S during training (system AF and system AFC), Table 5.2 shows both the approximate mcep TSLP obtaining using this fixed alignment and the approximate mcep TSLP obtained using median alignments from the system being evaluated. We can see that using median alignments from the system being evaluated gives a sizeable improvement: 0.13 nats for system AF and 0.18 nats for system AFC. Again, while it is not possible not directly extrapolate from the autoregressive HMM case to the trajectory HMM case, this suggests that the trajectory HMM systems T2 and TM2 in Table 4.4 might have had slightly higher mcep TSLP values if we had been able to use median alignments from those systems rather than from system S during evaluation. If so, the improvement in true TSLP given by the trajectory HMM system T2 over the autoregressive HMM system A2 is probably slightly greater than Table 4.4 would suggest.

5.4.3 Low latency synthesis

In this section we compare the standard framework, the trajectory HMM and the LGLAR HMM in terms of their ability to do low latency speech parameter generation. This is of interest in many applications. For example for embedded devices with a limited CPU it may be useful to start emitting the synthesized audio for the first part of an utterance before the audio for the later parts of the utterance has been computed, and low latency parameter generation provides one component of such a low latency speech synthesis system.

We first consider implementations of standard speech parameter generation. The standard speech parameter generation algorithm described in §3.2.4 computes the mean trajectory exactly, but requires $O(T)$ time to compute the first frame. This means latency can potentially be high, and both latency and memory usage are not predictable at design time since utterances vary in length. As mentioned in §3.2.5 there is a low latency time-recursive parameter generation algorithm for the standard HMM synthesis framework and the trajectory HMM, but it is slower than the standard Cholesky-based algorithm, and only computes the mean trajectory approximately. In contrast the autoregressive speech

parameter generation algorithm described in §4.3.3 requires only $O(1)$ time to compute the first frame, computes the mean trajectory exactly, and has predictably small memory and CPU requirements. Thus the LGLAR HMM supports an exact and low latency generation algorithm whereas for the standard framework and the trajectory HMM we must compromise on one of these aspects.

Conventionally post-filtering is used as the approach to reduce mumfledness when low latency speech parameter generation is desired, since the conventional gradient ascent-based algorithm for parameter generation considering global variance is inherently high latency. Post-filtering can be used with any of the approaches described in the previous paragraph. We will see in Chapter 6 that it is in fact possible to do low latency GV-like parameter generation for the standard framework and trajectory HMM, though we do not investigate this possibility experimentally in this thesis.

5.4.4 The form of local contributions

We saw in §5.3.1 that the b-value and precision matrix for both the trajectory HMM acoustic model and the LGLAR acoustic model can be decomposed into overlapping local contributions, where the two models differ only in the form of these local contributions. Specifically we can identify four ways in which the form of the collection of local contributions differs between the two models:

- *Rank of precision contributions.* For the trajectory HMM acoustic model, $P^{\mathrm{LC}}(q;\lambda)$ has rank at most D and typically has rank equal to D, whereas for the LGLAR acoustic model $P^{\mathrm{LC}}(q;\lambda)$ has rank 1. If we write $\mathcal{P}_r$ to denote the set of $(K+1)\times(K+1)$ real positive semi-definite matrices of rank at most r, then we have that $P^{\mathrm{LC}}(q;\lambda)$ is in $\mathcal{P}_D$ for the trajectory HMM acoustic model and is in $\mathcal{P}_1$ for the LGLAR acoustic model.

- *Flexibility of precision contributions given rank.* We say a matrix P in $\mathcal{P}_r$ is *attainable* for a given model if there is some choice of parameters for leaf q such that $P^{\mathrm{LC}}(q;\lambda)$ is equal to P. For the trajectory HMM acoustic model only a convex proper subset of $\mathcal{P}_D$ is attainable, whereas for the LGLAR acoustic model any element of $\mathcal{P}_1$ is attainable.

- *Relationship between b-value and precision contributions.* For the LGLAR acoustic model there is a constraint between the local contributions to the b-value and precision: $b^{\mathrm{LC}}(q;\lambda)$ determines $P^{\mathrm{LC}}(q;\lambda)$ up to a constant and $P^{\mathrm{LC}}(q;\lambda)$ determines $b^{\mathrm{LC}}(q;\lambda)$ up to a constant. For the trajectory HMM acoustic model the subcontribution for each window satisfies the same constraint, but overall there is no such constraint.

- *Location of final contribution.* For the LGLAR acoustic model the final contribution is the one that extends from time $T-K$ to time T. For the trajectory HMM acoustic model in §5.2 there are K additional contributions, one from time $T-K+1$ to time T, one from time $T-K+2$ to time T, etc.

The first three differences are in the parameterization used for each contribution. The final difference is in the layout of the collection of contributions over time with respect to the trajectory. The right extent K^{R}, meaning the number of elements of the trajectory after frame t which are directly affected by the state at frame t, is also a difference between the two models, since the LGLAR acoustic model has $K^{\mathrm{R}} = 0$ and the trajectory HMM acoustic model typically has $K^{\mathrm{R}} > 0$, but as discussed in §5.2 this is essentially only a superficial difference and so we do not include it above.

5.4.5 Contextual information in the generalized autoregressive state

We saw in §5.3.2 that the trajectory HMM acoustic model and the LGLAR acoustic model are both generalized autoregressive acoustic models, or equivalently are autoregressive acoustic models using an expanded state which we termed the generalized autoregressive state. In this section we compare the trajectory HMM acoustic model and the LGLAR acoustic model from the point of view of how they use the contextual information contained in this generalized autoregressive state. For clarity of presentation we again restrict to the case where the trajectory HMM acoustic model has right extent 0.

To recap, as we saw in §5.3.2, the generalized autoregressive state $\tilde{\theta}_t$ consists of both the conventional current state θ_t and the conventional future states $\theta_{t+1:T+K}$. As we saw in §3.1.5, the conventional state $\psi = (m, j, s, d)$ consists of a label m, index j of the current label in the label sequence, sublabel s and the number of frames d remaining in the current sublabel. As we saw in §3.1.1, the full-context label m may be further decomposed into a quinphone, which specifies the current, previous two and next two phonemes, and which we refer to as the phonemic context for brevity, and additional, mostly non-phonemic, context which we refer to as the linguistic context.

The information about phonemic and linguistic context contained in the generalized autoregressive state $\tilde{\theta}_t = \theta_{t:T+K}$ may thus be categorized as follows:

- the current phoneme and sublabel, previous two phonemes, next two phonemes, and current linguistic context
- the number of frames remaining in the current sublabel
- the durations of all sublabels in the next two phonemes, and of all sublabels remaining in the current label

- the identity of, and duration of all sublabels in, phonemes after the next two
- linguistic context for the next phoneme and all subsequent phonemes

Clearly some of these items are likely to be more important than others for modelling speech well. For example we would guess that the penultimate item has a relatively minor effect, and the last item probably contains little relevant information that is not already present in the current linguistic context. In this generalized autoregressive view, the trajectory HMM acoustic model and the LGLAR acoustic model can thus be seen to differ in the information about future phonemic and linguistic events that is available to them at a given time. The first item in the list is information that is already able to be used by the LGLAR acoustic model during decision tree clustering, whereas the remaining items consist of information that may conceivably be used by the trajectory HMM acoustic model but which is not available to the LGLAR acoustic model with conventional decision tree questions. Note that, as we saw in §5.3.2, the trajectory HMM acoustic model does not have "direct access" to the additional information in the generalized autoregressive state; it only has access to a β value encoding the collective preference of future states regarding the current (and nearby) trajectory values. Nevertheless this β value depends on, and contains some information about, all future states.

The two models also differ in the way they use the information that both have access to at the present time, because the two models are parameterized differently. For example the next two phonemes at a given time are known to both models. For the LGLAR HMM acoustic model the next two phonemes only affect the choice of leaf, and the autoregressive parameters for the current time are read directly from that leaf. In contrast, for the trajectory HMM acoustic model the next two phonemes also affect the autoregressive parameters for the current time via their indirect effect on the current β value, which depends on the parameters for the next few labels and sublabels and these are stored across multiple leaves.

Thus in terms of the generalized autoregressive viewpoint the trajectory HMM acoustic model and the LGLAR acoustic model differ in two ways: the information about the future which may in principle affect the autoregressive distribution at the current time; and the way that the current autoregressive distribution is parameterized.

5.4.6 Compensating for future state blindness

In §5.1 we discussed ways in which the autoregressive systems used in Chapter 4 might already be mitigating the effects of future state blindness to some extent. Specifically we hypothesized: that *right context*, meaning the future phonemes that are included in a full-context label, might be more important for the autoregressive HMM than for the standard HMM synthesis framework, and left context less important; and that the autore-

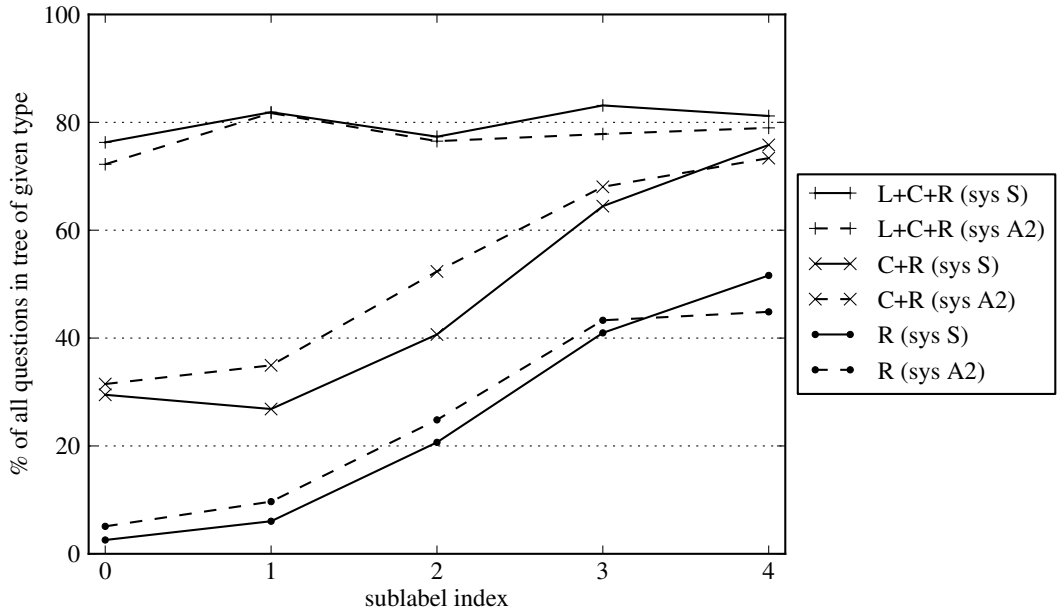


Figure 5.6: Graph showing the prevalence of decision tree questions about various parts of the phonemic context for a standard system and an autoregressive system. The ordinate is the percentage of all questions in the decision tree for a given sublabel that are about a certain part of the phonemic context. In the legend, L refers to the previous and previous-previous phoneme, C refers to the current phoneme, and R refers to the next and next-next phoneme. For example looking at the line labelled "C+R (sys S)", we can see that for system S roughly 40% of the questions in the decision tree for sublabel 2 were about the current, next, or next-next phoneme.

gressive HMM might move label or sublabel boundaries earlier in time during expectation maximization, effectively allowing the present state to contain more information about future phonemic events. In this section we describe the results of some simple experiments designed to investigate these hypotheses.

As a crude test of our hypothesis about the importance of right and left context, we examined the prevalence of questions about various parts of the phonemic context in the learned decision tree for system S and system A2. We decided to compare systems S and A2 rather than systems S2 and A2 so that the two systems under comparison would use similar numbers of leaves overall. Since a separate decision tree is grown for each sublabel we decided to split our analysis by sublabel. For each sublabel and each system, we computed the percentage of questions in the learned decision tree which were about "left" phonemic context (part L) consisting of the previous and previous-previous phoneme, the central phonemic context (part C) consisting of the current phoneme, the "right" phonemic context

(part R) consisting of the next and next-next phoneme, and the linguistic context (part O) consisting of everything else. For each sublabel and each system we thus have a kind of "distribution" over parts (L, C, R, O) where the value for a given part is the probability that a randomly selected question in the tree would be about that part. Our hypothesis is that in going from system S to system A2 the probability mass tends to shift away from L and towards R. Since it is hard to examine such a shift in probability mass by looking at the raw distribution values, we decided to look at the corresponding cumulative distributions, i.e. the distribution value for part R, the sum of the values for parts R and C, and the sum of the values for parts R, C and L. If the values of the cumulative distribution for a given sublabel are uniformly greater for system A2 than for system S, then we can conclude that system A2 tends to ask questions about "further right" phonemic context more of the time. The results are shown in Figure 5.6. From the lines labelled "L+C+R" we can see that the percentage of questions about any phonemic context, i.e. not about O, is roughly constant across sublabels and systems. This means little probability mass shifts between the phonemic and linguistic parts, which simplifies analysis of the results. From the lines labelled "R" and "C+R" we can see that for most sublabels, the probability mass does indeed shift right in going from system S to system A2. This supports the hypothesis that further right context is more important for the autoregressive HMM than for the standard HMM synthesis framework. Curiously for the last sublabel probability mass appears to shift left. We can also see that for both systems the probability mass shifts further and further right with increasing sublabel, as we might expect based on co-articulation considerations: it seems reasonable that future phonemes would have a stronger influence on the trajectory value towards the end of the realization of the current phoneme than towards the beginning.

To investigate whether the autoregressive HMM moves label or sublabel boundaries earlier in time during expectation maximization, we looked at the placement of the boundaries between labels, and between sublabels, for system S and system A2. Because alignments from system S were used to initialize the training of system A2, comparing these two systems should allow us to see any systematic label boundary shifts introduced by the autoregressive training procedure. We treated the utterance-final boundary as a boundary, but not the initial boundary. We computed median alignments for each utterance for system S and system A2, and computed the change or shift in each boundary in going from system S to system A2. We then computed the average shift over all boundaries in the test set. The median shift was 0 frames at both the label and sublabel level. The mean shift was -0.20 frames at the label level and -0.19 frames at the sublabel level. Note that a negative number corresponds to a shift earlier in time as predicted. Thus we may conclude that if there is a systematic effect it is in the direction predicted, but the magnitude of the systematic effect appears to be small at less than 1 frame. This suggests that either expectation maximization finds a local maximum that it is difficult to move away from, or

that for substantial shifts of the label and sublabel boundaries the positive consequences of knowing more about the future are outweighed by the negative consequences of knowing less about the present.

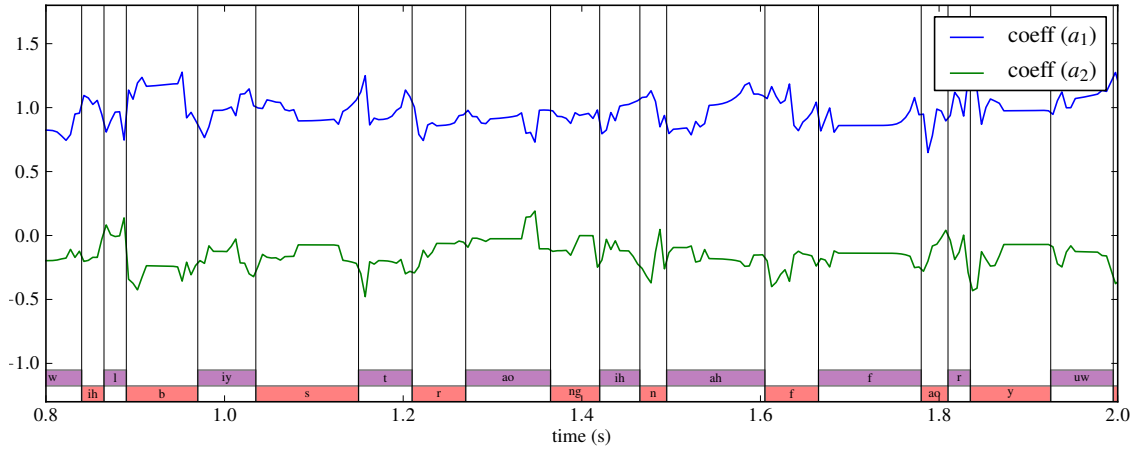
5.4.7 Visualization of the generalized autoregressive viewpoint

In this section we illustrate graphically the view of the trajectory HMM acoustic model as a generalized autoregressive acoustic model that was presented in §5.3.2, with the intention of making the somewhat abstract view presented there more concrete. We also view the LGLAR acoustic model in the same way for comparison.

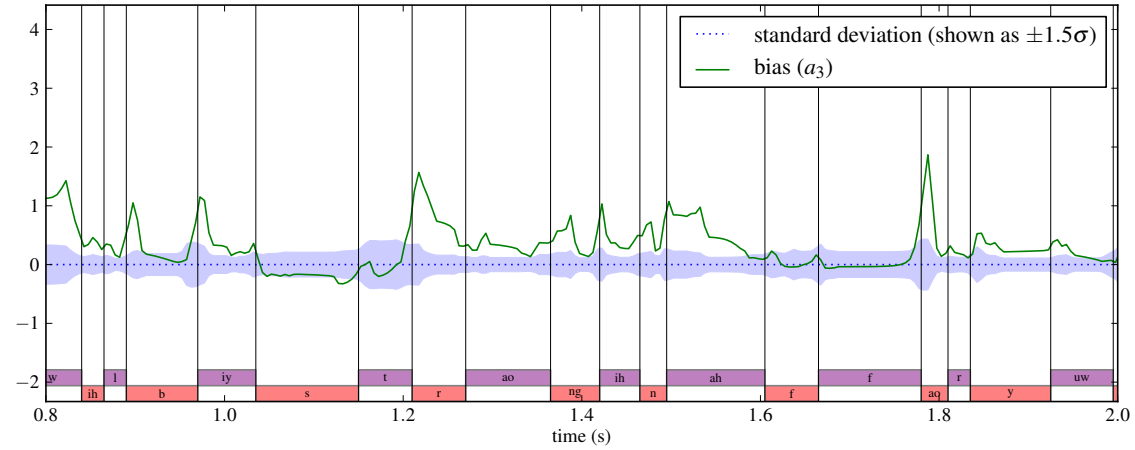
In §5.3.2 we saw that the trajectory HMM acoustic model can be viewed as a generalized autoregressive acoustic model. The autoregressive parameters for a given frame consist of autoregressive coefficients, an autoregressive bias and a conditional standard deviation. The result in §5.3.2 shows that the autoregressive parameters at frame t depend not only on the state θ_t as they do for the LGLAR acoustic model but also on future states $\theta_{t+1:T+K}$ and, in the case of window right extent $K^{\mathrm{R}} = 1$, the previous state θ_{t-1}.

The top two panels of Figure 5.7 show the autoregressive parameters for the trajectory HMM system T2, given a fixed median alignment, for a particular utterance and vector component. The top two panels of Figure 5.8 show the equivalent figure for the LGLAR HMM system A2, again given a fixed median alignment. Median alignments are used to allow comparison between the two figures. We can see that the autoregressive parameters derived from system T2 vary frame-by-frame due to the dependence on future states, whereas the autoregressive parameters derived from system A2 are piecewise constant since they are constant for the duration of each sublabel.

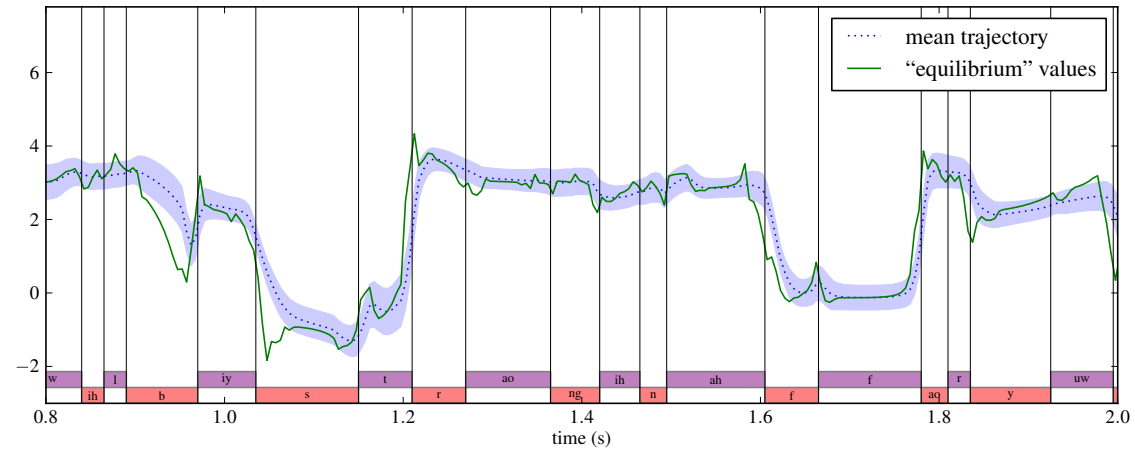
The two composite linear Gaussian autoregressive distributions may also be viewed in terms of their equilibrium or target values. Suppose we have a composite linear Gaussian autoregressive distribution with constant and stable autoregressive parameters where the initial conditions are specified at some time. This stochastic process has some *equilibrium distribution* which it settles down to after a while irrespective of the initial conditions, and the mean trajectory settles down to a constant value, which we refer to as the *equilibrium value*. As in §4.2, the difference between the mean trajectory and the equilibrium value is a sum of exponentially decaying sinusoids, referred to as *transients*, with amplitudes that depend on the initial conditions. The equilibrium value can be seen as a kind of "target" value that the process is heading towards, but which it takes some finite time to get close to. The two composite linear Gaussian autoregressive distributions above do not have constant autoregressive parameters, but we can still compute the equilibrium value for the autoregressive parameters for each frame. The bottom panels in Figure 5.7 and Figure 5.8 show the trajectories of these equilibrium values. These can again be interpreted as "target"



(a) autoregressive coefficients

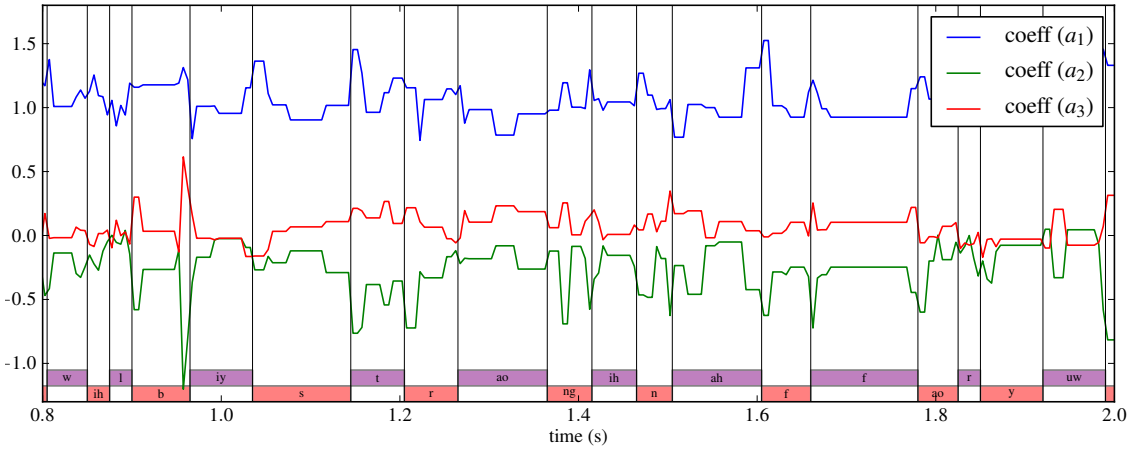


(b) autoregressive bias and conditional standard deviation

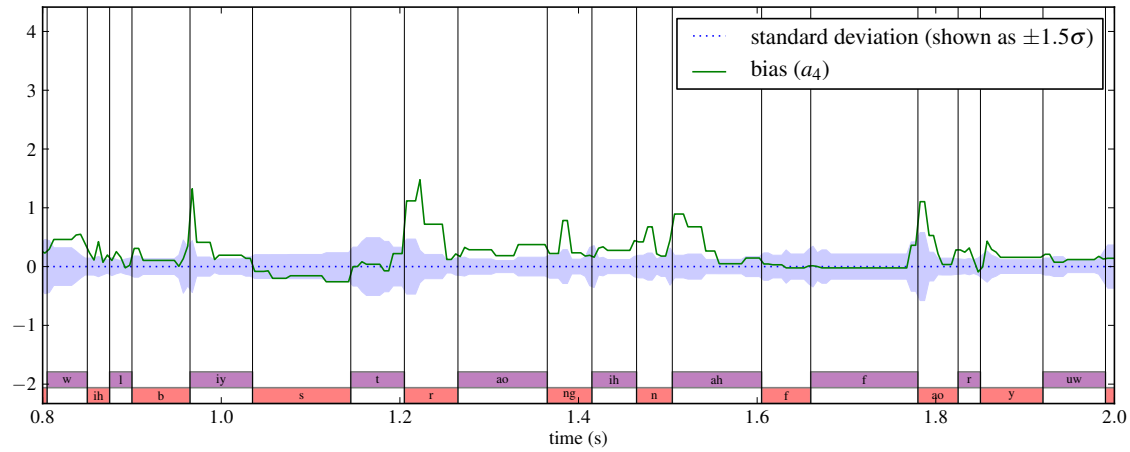


(c) distribution including autoregressive equilibrium values

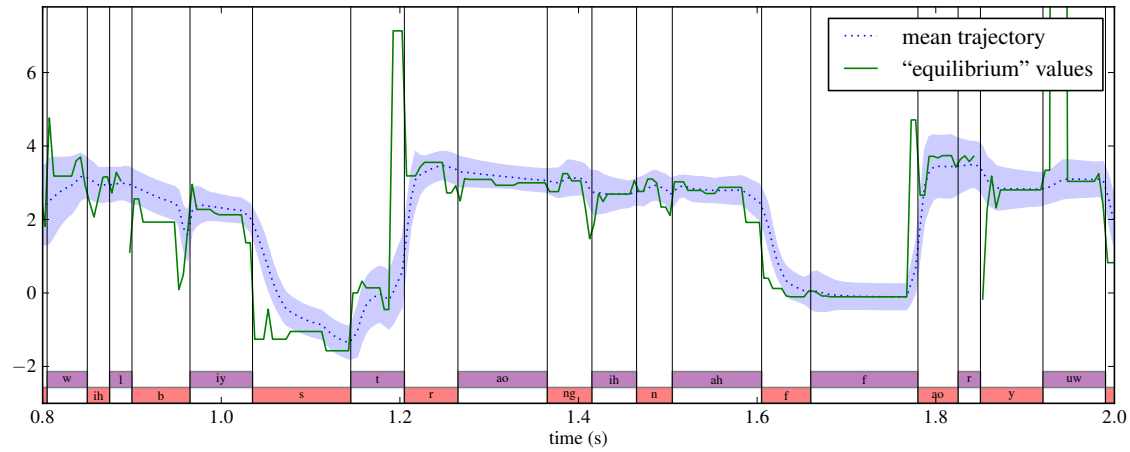
Figure 5.7: Visualization of a trajectory HMM acoustic model as a generalized autoregressive acoustic model (of depth $K = 2$, for system T2, mcep 1). The top two panels show the autoregressive parameters for each frame. The bottom panel shows the equilibrium value of each frame's autoregressive parameters, compared to the distribution over trajectories.



(a) autoregressive coefficients



(b) autoregressive bias and conditional standard deviation



(c) distribution including autoregressive equilibrium values

Figure 5.8: Visualization of an LGLAR acoustic model as a generalized autoregressive acoustic model (of depth $K = 3$, for system A2, for mcep 1). Unstable autoregressive parameters (around 0.9 s and 1.85 s) show as gaps in the line of equilibrium values.

values: the process at time t "heads towards" the equilibrium value at time t. We can see that for both models, changes in these target values precede corresponding changes in the trajectory, for example at the transition into the `s` phoneme just after 1.0 s. The equilibrium values for the two models are broadly similar, though the LGLAR acoustic model values tend to be more extreme. We find this view in terms of equilibrium or target values somewhat insightful, but care should be taken since the analysis in terms of the limiting behaviour says nothing about transients, and in the case of time-varying autoregressive parameters it seems likely that transients may have a large influence.

In Figure 5.8 we can also see the effect of unstable autoregressive parameters, which occur around 0.9 s and 1.85 s. Unstable autoregressive parameters have no equilibrium value, and are shown as gaps in the figure. If the autoregressive parameters at time t are unstable then the process at time t "heads away from" some fixed value. However this divergence is gradual and is controlled by the learned parameters, so it may have a minimal effect, or possibly even a beneficial effect, on the distribution over trajectories.

5.4.8 Stability and divergent trajectories

As discussed in §4.3.4, the LGLAR HMM suffers from a potential pathology whereby the mean trajectory or sampled trajectories can diverge far outside the range of values which might be considered reasonable, though divergent trajectories appear to be rare in practice. In this section we discuss further the issue of instability in autoregressive coefficients, providing a simple example showing that instability can arise even in seemingly non-pathological situations, and discussing why unstable trajectories are not observed more frequently in practice. We suspect that this pathology is either impossible or much less likely to occur for the standard HMM synthesis framework and trajectory HMM with conventional windows. It is certainly the case that, when system T2 is viewed autoregressively using the approach presented in §5.3.2 and using median alignments, the autoregressive coefficients are stable for all frames in the test corpus for all vector components.

Divergent trajectories are rare for the LGLAR HMM despite the fact that unstable autoregressive coefficient vectors are relatively common. For example, for system A2 8.0% of test corpus frames for mcep 0 are generated using unstable autoregressive coefficients, and 2.1% for mcep 1.

In the past we have argued (Shannon et al., 2013) that divergent trajectories are unlikely to occur with standard parameter generation methods and duration models for well trained systems, since typical durations of a given label and sublabel during parameter generation will be similar to the durations of that label and sublabel during training, and it seemed unlikely to us that grossly divergent distributions over trajectories of fixed length would be learned given sufficient training data from a true distribution which did not display

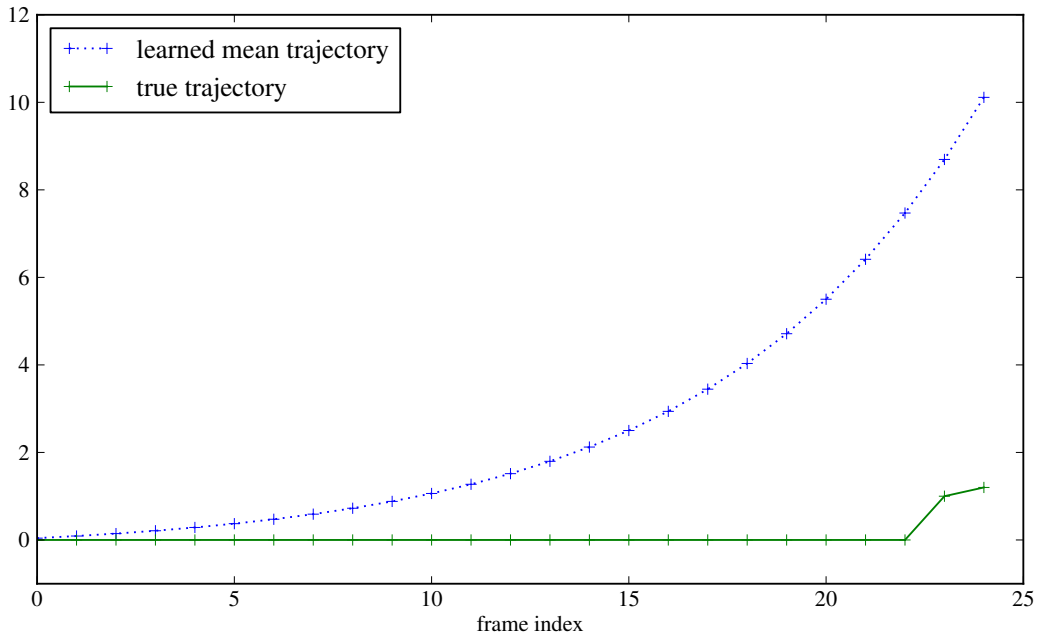


Figure 5.9: An artificial example demonstrating that a generated trajectory from a learned autoregressive model can diverge far outside the range of values occurring in the data the model was trained on, even when the generated trajectory has the same length as the training data trajectories and even when unlimited training data is available.

these divergent trajectories. However this assumption is incorrect: for some quite non-pathological true distributions, even given an arbitrary amount of training data, the learned mean trajectory diverges far outside the range of values occurring in the training data. We show this by providing a simple example of such a true distribution. Consider an LGLAR acoustic model of depth $K = 1$, where θ is always of length $T = 25$, the state θ_t always takes some fixed value, and the initial acoustic context c_0 is 0. Thus the only model parameters are an autoregressive coefficient, an autoregressive bias and a conditional standard deviation. Suppose the true distribution has no noise, so the mean trajectory is the entire distribution, and that the true mean trajectory μ^{true} consists of $T-2$ zeros followed by 1 followed by 1.2. Note that the learned autoregressive coefficient and bias are the same no matter how much training data from the true distribution is used. The learned autoregressive parameters give rise to a learned mean trajectory μ shown in Figure 5.9. We can see that the learned mean trajectory diverges massively outside the values occurring in the true trajectory. This bad behaviour can be exacerbated by increasing the number of zeros at the start of the true trajectory. Indeed if we take any true distribution which causes the learned autoregressive parameters to be unstable, and modify the true distribution by deterministically prepending zeros to any sampled trajectory, then the learned autoregressive parameters will be the

same, but the learned mean trajectory will have more time to diverge. In the limit of prepending enough zeros, we will always get divergence in any situation where the original learned autoregressive coefficients were unstable. This provides another way to see that there are true distributions which lead to badly divergent learned mean trajectories.

In fact we observed the above effect, where learned mean trajectories are divergent even for well-trained models with no durational mismatch, when training the systems for the illustration of future state blindness in §5.1. When the standard deviation of the noise of the natural distribution is low, the learned mean trajectory in Figure 5.1(c) diverges before the end of label 1.

Given that learned mean trajectories can be divergent even for well-trained models with no durational mismatch, a natural question is why this phenomenon does not occur more often for typical autoregressive HMM models of speech. The future state blindness example shows that more noise can sometimes help to prevent divergent trajectories occurring. This is presumably because the noise is acting as a type of regularization, making the learned autoregressive coefficients less extreme. Indeed if we were to add white Gaussian noise to every trajectory in the training corpus, the expected effect on the accumulators (2.52), (2.53), (2.54) and (2.55) would be to add a constant to $\tilde{u}$ and to the non-bias elements of the diagonal of $\tilde{S}$, which is similar to the effect of using a simple regularizer. Natural mel cepstral trajectories computed using standard analysis procedures look very "noisy", as can be seen for example in Figure 5.4, and it is therefore possible that one reason divergent trajectories do not occur more often in practice is that this noisiness is having a regularizing effect. As an aside, it is also possible that this noisiness has a regularizing effect during parameter estimation in the case of the standard framework. Another contributing factor may be the fact that segments with unstable coefficients are typically quite short. For system A2 using median alignments on the test corpus, runs of unstable frames have a median length of 3 frames for cepstral coefficient 0, or 4 frames for cepstral coefficient 1.

Unstable autoregressive coefficients are not always undesirable. To emphasize this, we point out that the learned autoregressive parameters for labels 3 and 6 in Figure 5.1(d) are unstable, yet the distribution over trajectories seems reasonable. It is even possible that allowing unstable coefficients leads to a better learned distribution over trajectories, since it makes the probabilistic model slightly more flexible. Indeed for labels 3 and 6 it seems entirely appropriate that unstable autoregressive coefficients would provide the best approximation of the true mean trajectory. As discussed in §4.3.4, a useful quantity for assessing whether unstable autoregressive coefficients are likely to lead to badly divergent trajectories is the time constant. In Figure 5.1(d) the duration of each of the unstable segments, which is 5 frames, is substantially less than the time constant of the autoregressive coefficients for these segments, which is 13.3 frames for label 3 and 13.7 frames for label 6.

5.5 Summary of contributions

The major novel theoretical contributions of this chapter are two unifying views of the trajectory HMM acoustic model and the LGLAR acoustic model, one viewing the b-value and precision matrix as being built-up additively from local contributions (§5.3.1), and the other viewing both models as generalized autoregressive models (§5.3.2). The second view also leads to a novel view of the trajectory HMM as a directed graphical model. The most substantial experimental contribution of this chapter is the fact that the standard framework systematically underestimates the variance in the distribution over trajectories, by roughly a factor of 3 for a system with 3 windows (§5.4.1).

Minor novel contributions of this chapter include: identification of future state blindness as a weakness of the autoregressive HMM and an illustration of its effect for a simple synthetic data set (§5.1); the realization that the trajectory HMM acoustic model can be described as a conditional random field, and the associated relationship between future state blindness and the label bias problem (§5.1); an appreciation that the window right extent is essentially arbitrary (§5.2); a proof that the factor by which the standard framework systematically underestimates the variance is at most the number of windows (§5.4.1 and Appendix A); the discovery that the autoregressive HMM tries to compensate for future state blindness by asking more questions about further right phonemic context compared to the standard framework (§5.4.6); an example showing that learned autoregressive distributions with constant coefficients can be very divergent even if learned on a lot of data drawn from a non-pathological true distribution (§5.4.8); and the idea that noise in the cepstra might act as a regularizer during parameter estimation for all three models (§5.4.8).

Chapter 6

Investigation of spread-based speech parameter generation

We mentioned in §3.2.6 that parameter generation considering global variance often dramatically improves the quality of synthesized speech compared to standard speech parameter generation. We also saw that it has certain drawbacks: it is slower than standard speech parameter generation and sometimes introduces artifacts. In this chapter we investigate parameter generation considering global variance in detail and address these drawbacks.

The layout of this chapter is as follows. We first describe related previous work. We then describe a slight generalization of parameter generation considering global variance which we call *spread-based generation*. In the terminology established in this section, parameter generation by maximizing a utility function such as (3.33) is referred to as *utility GV generation* and is a special case of *GV generation*. We then show that GV generation is amenable to an analysis using Lagrange multipliers. This analysis is ultimately made possible by the fact that global variance and the Gaussian log probability are both quadratic functions of the trajectory. For the standard framework and the trajectory HMM, this analysis shows that GV generation is equivalent to standard generation on a modified model. It also motivates a simple approximation, *fixed-multiplier GMSD generation*, which provides GV-like generation at standard generation speeds. *Global mean squared deviation (GMSD)* is a metric similar to GV which will be defined below.

We then examine the relationship between normalized models and GV generation. It is sometimes thought that GV generation is compensating for a defect in the probabilistic model rather than a defect in the standard parameter generation method, but we show experimentally that normalized models such as the trajectory HMM and LGLAR HMM model GV very well. For the trajectory HMM we show that modelling GV reasonably well is a direct consequence of the feature functions used to define it.

Next we focus on the issue of artifacts. As mentioned in §3.2.6, the conventional way to

prevent artifacts is to use early stopping during gradient ascent on the GV utility function, but this is not an option for generation methods based on the analytic results presented in the first part of the chapter since gradient ascent is not used. We therefore look for an alternative way to prevent artifacts. We start by examining why GV generation introduces artifacts. We present a simple synthetic example demonstrating a pathology in GV generation and how this can lead to *excursions* in the generated trajectory. We then show that excursions sometimes occur when using GV generation with a trained statistical parametric speech synthesis system. We argue that these excursions are the cepstral-level phenomenon responsible for the perceptual-level phenomenon of artifacts. We then discuss practical ways to reduce excursions. Here the analytic results again prove useful, providing a simple way to detect which frames are "at risk" of excursions for the standard framework and the trajectory HMM, and suggesting a way to reduce their occurrence which we refer to as *local static parameter adjustment (LSPA)*. In an objective and subjective evaluation we find that LSPA is very effective at reducing excursions and artifacts without degrading naturalness.

Finally we bring the above threads together. In a subjective evaluation we find that *fixed-multiplier LSPA GMSD generation*, which combines the use of fixed-multiplier GMSD generation for speed and LSPA to reduce artifacts, gives as good naturalness as conventional parameter generation considering global variance. This makes it a very attractive parameter generation algorithm when fast, high-quality generation is desired. A particularly nice aspect of fixed-multiplier LSPA GMSD generation is that it can be implemented as a simple adjustment to the model, with the standard generation algorithm, or its time recursive variant, used at synthesis time. We end with a discussion of the questions raised and conclusions that can be drawn from our investigation of spread-based generation. We argue that excursions and artifacts are due to an inherent weakness of GV generation, exacerbated by a deficiency in the probabilistic model, specifically in the decision tree.

6.1 Related work

Previous attempts to make GV generation faster at synthesis time have mainly focused on incorporating aspects of the GV utility function into training (Wu et al., 2008; Toda and Young, 2009; Zen et al., 2012). However these approaches result in a significant increase in the complexity of training, unlike our approach. A very different approach to obtaining fast GV-like generation is simply to use the variance-scaled trajectory (3.35), which is the trajectory usually used as the initialization for parameter generation considering global variance (Silén et al., 2012). This approach has the disadvantage that it does not take into account which frames are more likely to have trajectory values which deviate substantially from the global mean, but has the advantage of speed and simplicity. In subjective exper-

iments variance scaling was preferred to post-filtering and was within measurement error of parameter generation considering global variance (Silén et al., 2012). An experimental comparison of variance scaling and our proposed method would be informative but is not undertaken in this thesis.

There has been very little previous work investigating why artifacts occur. It has been suggested previously that one of the causes of artifacts is the fact that, as discussed in §3.2.6, utility GV generation effectively uses μ^{GV} as the target GV for all utterances. This corpus average GV may be too large for certain utterances, particularly short ones, taking the acoustic model "out of its comfort zone" and leading to artifacts (Toda, 2011; King, 2011). However our experimental results suggest that this is a relatively minor source of artifacts for our experimental systems.

Previous work attempting to reduce artifacts includes using full-covariance models and a full-covariance GV distribution (Zen et al., 2006), carefully tuning the stopping criterion used during gradient ascent (Yamagishi et al., 2008), and using "GV-constrained" trajectory HMM training (Toda and Young, 2009). The use of context-dependent GV distributions (Yamagishi et al., 2008) may also help to reduce artifacts by selecting a more appropriate amount of GV for each utterance, though the pathology identified in §6.6 and the experimental results in §6.8.2 suggest that selecting an appropriate amount of GV is not sufficient to prevent artifacts occurring. We take a very different approach to reducing artifacts in this thesis, informed by our understanding of why they occur.

6.2 Spread-based generation

In this section we introduce a class of speech parameter generation methods which we refer to as *spread-based generation*. Parameter generation considering global variance is a form of spread-based generation. We first define spread functions and spread-based generation, then specify some commonly used spread functions and specific forms of spread-based generation. These forms of spread-based generation will be used in our mathematical analysis of parameter generation considering global variance and our experimental investigations.

To define the concept of a spread function, consider a quadratic function $\overline{s} : \mathbb{R}^T \to \mathbb{R}$ for $T \in \mathbb{N}$. Any such $\overline{s}$ is of the form $\overline{s}(c) = \frac{1}{T}(c^{\mathsf{T}} W c - 2c^{\mathsf{T}} w + w_0)$ for some constant $w_0 \in \mathbb{R}$, vector $w \in \mathbb{R}^T$ and matrix $W \in \mathbb{R}^{T \times T}$. If W is positive semi-definite then we refer to $\overline{s}$ as a *spread function* and refer to W as the *spread matrix*. The result of applying a spread function to a trajectory $c \in \mathbb{R}^T$ is referred to as the *spread* of the trajectory. The spread of a trajectory is a scalar quantity characterizing some aspect of how spread out it is.

We saw in §3.2.6 that parameter generation considering global variance optimizes a utility function G given by (3.33). The utility $G(c)$ of a trajectory c depends on both its log probability $A(c)$ and its global variance $\overline{v}(c)$. We can generalize this slightly by defining

a utility function

$$G(c) = A(c) + B(\overline{s}(c)) \tag{6.1}$$

where $\overline{s}$ is a spread function and the *utility-of-spread function* $B : \mathbb{R} \to \mathbb{R}$ is an arbitrary function. We refer to parameter generation by maximizing (6.1) as *spread-based generation*. This is a generalization of parameter generation considering global variance since using global variance as the spread function and using a quadratic utility-of-spread function $B(v) = \omega^{\mathrm{GV}} \log \mathcal{N}\left(v; \mu^{\mathrm{GV}}, (\sigma^2)^{\mathrm{GV}}\right)$ recovers parameter generation considering global variance. Note that as always the log probability function A, which is defined as $A(c) = -\frac{1}{2}c^{\mathsf{T}}Pc + b^{\mathsf{T}}c$, depends implicitly on the natural parameters b and P of the distribution over trajectories specified by the model. For spread-based generation we allow the utility-of-spread function B to depend implicitly on the state sequence and on the model parameters. For example for parameter generation considering global variance with the conventional setting of ω^{GV}, B depends on ω^{GV} which in turn depends on the length of the state sequence. Similarly we allow the spread function $\overline{s}$ to depend implicitly on the state sequence and on the model parameters, though for most of the spread functions considered below $\overline{s}$ only depends implicitly on the length of the state sequence.

In the remainder of this section we consider various spread functions and forms of spread-based generation. For later reference, Table 6.1 summarizes the parameter generation methods described below, as well as some methods described in subsequent sections, for the case of the GV spread function.

6.2.1 Standard generation

Standard generation is a form of spread-based generation for any spread function, since it can be obtained by setting $B(s) = 0$ for $s = \overline{s}(\mu)$ and $B(s) = -\infty$ otherwise, where μ is the mean trajectory.

6.2.2 GMSD and GV generation

We consider two spread functions in detail in this thesis. Firstly the *global mean squared deviation (GMSD)* around a given value $k \in \mathbb{R}$ is defined as

$$\overline{s}_k(c) = \frac{1}{T}\sum_t (c_t - k)^2 \tag{6.2}$$

$$= \frac{1}{T}(c - k\mathbb{1})^{\mathsf{T}}(c - k\mathbb{1}) \tag{6.3}$$

where c is a trajectory of length T and $\mathbb{1}$ is a vector of ones of the same length. For any $k \in \mathbb{R}$, $\overline{s}_k$ is a spread function with spread matrix equal to the identity matrix I. We refer to spread-based generation using GMSD as the spread function as *GMSD generation*.

name	description
standard generation	maximize A
fixed-multiplier GV generation	$c^* = (P - \lambda J)^{-1} b$ for fixed $\lambda \in \mathbb{R}$
fixed-spread GV generation	maximize A subject to $\overline{v}(c^*) = v$ for fixed $v \geq 0$
expected-spread GV generation	maximize A subject to $\overline{v}(c^*) = \mathbb{E}\overline{v}(c)$ where $c \sim \mathcal{N}_{\text{NP}}(b, P)$
exact utility GV generation	globally maximize G
early-stopped utility GV generation	use gradient ascent with early stopping to approximately maximize G

Table 6.1: A summary of various forms of GV generation considered in this chapter. Here c^* is the generated trajectory, A is a function which returns the log probability of a trajectory (up to a constant), G is the GV utility function, $\overline{v}$ is the function which returns the global variance of a trajectory, b and P are the b-value and precision matrix of the Gaussian distribution over trajectories, and J is a matrix defined in (6.6). See §6.2 for detailed descriptions. Early-stopped utility GV generation is included for completeness but strictly speaking is not a GV generation method by our definition. The other methods are all GV generation methods and differ only in the amount of GV they choose to use for a given state sequence.

Typically we set k to the mean value of c_t over all frames of the training corpus. Secondly the GV function $\overline{v}$ defined by (3.31) is a spread function, since (3.32) implies

$$\overline{m}(c) = \frac{1}{T} \sum_t c_t = \frac{1}{T} \mathbb{1}^\mathsf{T} c \tag{6.4}$$

so

$$\overline{v}(c) = \frac{1}{T} \sum_t (c_t - \overline{m}(c))^2 = \frac{1}{T} \sum_t c_t^2 - (\overline{m}(c))^2 = \frac{1}{T} c^\mathsf{T} J c \tag{6.5}$$

where

$$J = I - \frac{1}{T} \mathbb{1}\mathbb{1}^\mathsf{T} \tag{6.6}$$

is the spread matrix. We refer to spread-based generation using GV as the spread function as *GV generation*. The main reason that we consider GMSD generation in addition to GV generation is that GMSD generation is slightly easier to analyze while being similar conceptually and giving very similar results.

6.2.3 Fixed-spread and expected-spread generation

We use the term *fixed-spread generation* to mean maximizing A subject to the constraint that the generated trajectory has fixed spread s. By setting the utility-of-spread function B in (6.1) to the function which has value 0 at s and value $-\infty$ elsewhere we see that this is a form of spread-based generation. Unless stated otherwise we assume that, for a given portion p and vector component i of the speech parameters, s is chosen to be the mean spread of training corpus trajectories. In the GV case this is equal to the mean μ^{GV} of the GV pdf used by conventional parameter generation considering global variance.

It is also possible to incorporate the model's own beliefs about what the spread should be. Normalized probabilistic models such as the trajectory HMM acoustic model and the LGLAR acoustic model define a probability distribution over the trajectory c, and so also over the spread $\overline{s}(c)$. The implied probability distribution over the spread specifies the model's belief about how likely the trajectory for a given state sequence is to have a given spread. For a spread function with spread matrix W the expected spread can be computed as

$$\mathbb{E}\overline{s}(c) = \frac{1}{T}\mathrm{tr}\big(WP^{-1}\big) + \overline{s}(\mu) \tag{6.7}$$

where $c \sim \mathcal{N}_{\mathrm{NP}}(b, P)$ defines the distribution used for the expectation and $\mu = P^{-1}b$ is the mean trajectory. We refer to maximizing A subject to the constraint that the generated trajectory has spread equal to the expected spread as *expected-spread generation*. Note that the expected spread is always at least as big as the spread of the mean trajectory, that is

$$\mathbb{E}\overline{s}(c) \geq \overline{s}(\mu) \tag{6.8}$$

since $\frac{1}{T}\mathrm{tr}\big(WP^{-1}\big) \geq 0$ since P is positive definite and W is positive semi-definite. This means that, compared to standard speech parameter generation, the effect of expected-spread generation will be to boost the spread of the generated trajectory.

We might worry about whether fixed-spread generation is well-defined, i.e. if we denote the set of trajectories with the given fixed spread by C and the function A restricted to this set by $A|_C$, we might wonder whether there always exists a unique maximizer of $A|_C$. It is fairly easy to show that at least one maximizer exists. By working in an eigenbasis of P it can be shown that $A(c) \to -\infty$ as $c \to \infty$. Combined with the fact that C is closed this implies that $A|_C$ attains its supremum somewhere. It is harder to show that the maximizer is typically unique, but this falls out of the mathematical analysis presented below, and is shown for fixed-spread GMSD generation in §6.4.1 and for fixed-spread GV generation in §6.4.2.

6.2.4 Utility spread-based generation

If the utility-of-spread function B is differentiable then we refer to parameter generation by maximizing (6.1) as *exact utility spread-based generation*. In fact for most of this chapter we assume stronger conditions on B: that it is twice continuously differentiable, strictly concave, and bounded above. Here by bounded above we mean that there is some $K \in \mathbb{R}$ such that $B(s) \leq K$ for all $s \geq 0$, which since B is assumed concave is equivalent to saying $B(s) \not\to \infty$ as $s \to \infty$. The utility-of-spread function used by parameter generation considering global variance has these three properties since it is quadratic with a positive leading coefficient.

We refer to parameter generation using gradient ascent with early stopping to approximately maximize G as *early-stopped utility spread-based generation*. Early-stopped utility GV generation is thus another name for conventional speech parameter generation considering global variance. Strictly speaking early-stopped utility GV generation is not a form of GV generation since it does not maximize a utility function of the form (6.1) where $\overline{s}$ is the GV spread function.

We might worry about whether exact utility spread-based generation is well-defined, i.e. whether there always exists a unique maximizer of G. Since $A(c) \to -\infty$ as $c \to \infty$ and B is bounded above, we have $G(c) \to -\infty$ as $c \to \infty$. This implies that G attains its supremum somewhere. It is harder to show that the maximizer is typically unique, but we see this in the mathematical analysis presented in §6.4.3.

We also might worry that early-stopped utility spread-based generation would approach a local rather than a global maximum. This might be a problem since selecting a suboptimal local maximum might result in a low probability trajectory, which might in turn result in degraded speech. It would also mean the generated trajectory might depend strongly on the initialization used for gradient ascent, which may be undesirable. We discuss this issue in a bit more detail in §6.4.5.

6.2.5 Corpus-level spread-based generation

So far in this section we have assumed that spread-based generation operates at the level of the utterance. It is also possible to consider spread-based generation at the corpus level. Here we briefly describe this alternative approach.

Given a state sequence θ, spread-based generation produces the trajectory which maximizes the utility function (6.1). If we were instead given a collection $\Theta = [\Theta_r]_r$ of state sequences, we could produce a collection $C = [C_r]_r$ of trajectories by considering Θ as a single long state sequence, finding the single long trajectory which maximizes (6.1), then considering this single long trajectory as a collection of trajectories. We refer to this as *corpus-level spread-based generation*. Any form of spread-based generation thus has a corres-

ponding form of corpus-level spread-based generation. For example fixed-spread generation produces the most likely trajectory subject to a constraint on the spread of this trajectory, while corpus-level fixed-spread generation produces the most likely collection of trajectories subject to a constraint on the overall spread of this collection of trajectories.

In some ways corpus-level spread-based generation is more intuitively appealing than spread-based generation, since we would not expect the spread for every utterance to be precisely equal to its expected value. However it also seems slightly undesirable for the trajectory generated for one state sequence to depend on the corpus of state sequences that utterance happens to be part of. We further discuss how spread-based generation and corpus-level spread-based generation compare in §6.4.6 and §6.8.3.

6.3 Method of Lagrange multipliers

The *method of Lagrange multipliers* (Whittle, 1971) provides a general approach to solving constrained optimization problems. It does this by converting a constrained optimization problem into a family of unconstrained optimization problems in such a way that the solutions of the unconstrained problems tell us something about the solution of the original constrained problem. This approach is particularly useful if the unconstrained problems can be solved analytically, as they can be for our application. In this section we present a brief but self-contained introduction to the aspects of the general theory of Lagrange multipliers that we will use in this chapter. It should be noted that the *sufficiency*-based approach which we use here differs from the *necessity*-based approach which seems to be more commonly presented in introductions to the method of Lagrange multipliers (see for example Bishop (2006)). The relationship between these two approaches is discussed in the last paragraph.

Suppose we wish to maximize an *objective function* $f : X \to \mathbb{R}$ subject to the constraints $\overline{g}_m(x) = g_m$ for $m = 1, \ldots, M$, where X is an arbitrary set, $\overline{g}_m : X \to \mathbb{R}$, and $g_m \in \mathbb{R}$. Equivalently we can view the constraints as being specified by $\overline{g}(x) = g$ where $\overline{g} : X \to \mathbb{R}^M$ and $g \in \mathbb{R}^M$. The method of Lagrange multipliers defines a function

$$L(x; \lambda) = f(x) - \sum_{m=1}^{M} \lambda_m \overline{g}_m(x) \tag{6.9}$$

for $x \in X$ and $\lambda \in \mathbb{R}^M$. Here L is referred to as the *Lagrangian* and $\lambda_m \in \mathbb{R}$ is referred to as a *Lagrange multiplier*.

There is a simple, indeed essentially trivial, result known as the *Lagrangian sufficiency theorem* (Theorem 1 in (Everett III, 1963), Theorem 3.1 in (Whittle, 1971), Theorem 5 in (Courcoubetis and Weber, 2003)) which forms the basis of our use of the method of Lagrange multipliers. It states that if $x^* \in X$ maximizes $x \mapsto L(x; \lambda)$ for some $\lambda \in \mathbb{R}^M$ and

x^* satisfies the constraints, then x^* is a solution to the original constrained maximization of f. We can see the Lagrangian sufficiency theorem as follows. We are given $x^* \in X$, $g \in \mathbb{R}^M$ and $\lambda \in \mathbb{R}^M$, and have that $\overline{g}(x^*) = g$ and $L(x^*; \lambda) \geq L(x; \lambda)$ for any $x \in X$. If $x \in X$ satisfies the constraints, i.e. if $\overline{g}(x) = g$, then $L(x; \lambda) = f(x) - \sum_m \lambda_m g_m$. Now we certainly have $L(x^*; \lambda) \geq L(x; \lambda)$ for any $x \in X$ which satisfies the constraints, since this inequality holds for any $x \in X$ at all. Therefore we have $f(x^*) - \sum_m \lambda_m g_m \geq f(x) - \sum_m \lambda_m g_m$ for any $x \in X$ which satisfies the constraints, and so $f(x^*) \geq f(x)$ for any $x \in X$ which satisfies the constraints. This establishes the desired result. A minor extension of this result will also be useful. If $x^* \in X$ is the unique maximizer of $x \mapsto L(x; \lambda)$ for some $\lambda \in \mathbb{R}^M$ and x^* satisfies the constraints, then x^* is the unique solution to the original constrained maximization of f. This can be seen by replacing $\geq$ with $>$ above.

The Lagrangian sufficiency theorem can sometimes be used to obtain a complete solution to a constrained maximization problem. Typically this proceeds along lines such as the following. Let Λ be the set of $\lambda \in \mathbb{R}^M$ such that $x \mapsto L(x; \lambda)$ has a unique maximizer, and let $\overline{x}^* : \Lambda \to X$ denote the function which returns this maximizer. Now for any $\lambda \in \Lambda$, we know by the Lagrangian sufficiency theorem that $\overline{x}^*(\lambda)$ is the unique solution of the original constrained maximization of f for some choice of $g \in \mathbb{R}^M$, namely for $g = \overline{g}(\overline{x}^*(\lambda))$. Thus we can search over values of $\lambda \in \Lambda$ to try to find one for which $\overline{x}^*(\lambda)$ satisfies the constraints for our chosen g. Such a λ is not guaranteed to exist, but if it does then we know that $\overline{x}^*(\lambda)$ is the unique solution of the original constrained maximization problem. Often the function $\overline{x}^*$ can be found analytically, and the search over λ can be performed taking advantage of the form of $\overline{x}^*$.

As mentioned above, there is another approach to using the Lagrangian L to learn something about solutions of the original constrained maximization problem. This necessity-based approach comes up with conditions on L that any constrained maximum of f must satisfy; that is, it establishes *necessary* conditions for a value $x \in X$ to be a solution to the original problem. Typically for this approach it is assumed that X is subset of $\mathbb{R}^N$ for some N and f and g are differentiable, and a typical result in this approach is that, under certain regularity conditions on X, f and g, any constrained maximum x^* of f satisfies $\frac{\partial L}{\partial x}(x^*; \lambda) = 0$ for some $\lambda \in \mathbb{R}^M$. Thus at best it provides a finite list of candidate solutions, and if the original constrained maximization problem has a maximizer, then this list is guaranteed to contain it. However if f, restricted to the set of $x \in X$ which satisfy the constraints, does not attain its supremum anywhere, then the candidate solutions may all be far from optimal. In contrast the sufficiency-based approach, in cases where it works, ensures that we have the optimal solution. A fairly comprehensive introduction to the method of Lagrange multipliers is given by Whittle (1971), and this shows how the above two approaches fit within the general theory.

6.4 Analysis of spread-based generation

In this section we present a theoretical analysis of spread-based generation using the method of Lagrange multipliers. This analysis shows that the trajectory produced by spread-based generation is always of a particular parametric form, and leads naturally to a new algorithm for doing spread-based generation. The enabling fact that makes the method of Lagrange multipliers so fruitful in this case is the similarity in form of the log probability function A and the spread function $\overline{s}$. These are both quadratic functions of the trajectory, and we will see that this allows the family of unconstrained optimization problems introduced by the method of Lagrange multipliers to be solved analytically, which will prove useful both conceptually and computationally.

We start by presenting an analysis of fixed-spread generation. Fixed-spread generation is a natural fit for the method of Lagrange multipliers since computing the generated trajectory is a constrained optimization problem. We then show that this analysis is also useful in the case of general spread-based generation. We only consider GMSD and GV spread functions below, but much of the analysis holds with minor alterations for general spread functions. The major results in this section have been previously described in a technical report (Shannon and Byrne, 2013b) and a conference paper (Shannon and Byrne, 2013a).

6.4.1 Analysis of fixed-spread GMSD generation

In this section we consider the problem of finding the trajectory produced by fixed-spread GMSD generation. Given $s > 0$ and a trajectory value $k \in \mathbb{R}$ around which to measure the GMSD, we wish to find the trajectory $c \in \mathbb{R}^T$ which maximizes $A(c) = -\frac{1}{2}c^\mathsf{T}Pc + b^\mathsf{T}c$ subject to the constraint $\overline{s}_k(c) = s$. We first use the approach described in §6.3 to solve this constrained optimization problem in the case $k = 0$, then consider the case of general k, then discuss algorithmic and computational considerations.

In the case $k = 0$ we have $T\overline{s}_0(c) = c^\mathsf{T}c$. This follows from (6.2). For simplicity of notation we consider the constraint to be $-\frac{1}{2}T\overline{s}_0(c) = -\frac{1}{2}Ts$ instead of $\overline{s}_0(c) = s$. The Lagrangian is

$$L(c;\lambda) = A(c) + \tfrac{1}{2}\lambda T\overline{s}_0(c) \tag{6.10}$$

$$= -\tfrac{1}{2}c^\mathsf{T}Pc + b^\mathsf{T}c + \tfrac{1}{2}\lambda c^\mathsf{T}c \tag{6.11}$$

$$= -\tfrac{1}{2}c^\mathsf{T}(P - \lambda I)c + b^\mathsf{T}c \tag{6.12}$$

Let λ_I be the minimum eigenvalue of a matrix P. If $\lambda < \lambda_\mathrm{I}$ then $P - \lambda I$ is positive definite, and so $x \mapsto L(x;\lambda)$ has a unique global maximizer given by

$$\overline{c}^*(\lambda) = (P - \lambda I)^{-1}b \tag{6.13}$$

By the Lagrangian sufficiency theorem this means that $\overline{c}^*(\lambda)$ is the optimal trajectory for some s, namely for $s = \overline{s}_0(\overline{c}^*(\lambda))$. For notational convenience we define a function $\overline{s}^*(\lambda) = \overline{s}_0(\overline{c}^*(\lambda))$ with domain $(-\infty, \lambda_{\mathrm{I}})$ which computes the GMSD of the optimal trajectory for a given λ. All that remains is to investigate the range of values of s for which we can obtain a provably optimal solution using the above approach, i.e. to compute the image of the function $\overline{s}^*$. For $\lambda < \lambda_{\mathrm{I}}$ we have

$$T\overline{s}^*(\lambda) = T\overline{s}_0\left(\overline{c}^*(\lambda)\right) \tag{6.14}$$

$$= \left(\overline{c}^*(\lambda)\right)^{\mathsf{T}}\overline{c}^*(\lambda) \tag{6.15}$$

$$= b^{\mathsf{T}}(P - \lambda I)^{-2}b \tag{6.16}$$

Differentiating we obtain

$$T(\overline{s}^*)'(\lambda) = 2b^{\mathsf{T}}(P - \lambda I)^{-3}b \tag{6.17}$$

Since $P - \lambda I$ is positive definite, $x^{\mathsf{T}}(P - \lambda I)x \geq 0$ for any x and in particular for $x = (P - \lambda I)^{-2}b$, and so we have $(\overline{s}^*)'(\lambda) \geq 0$, i.e. $\overline{s}^*$ is increasing. If $b \neq 0$ then $(\overline{s}^*)'(\lambda) > 0$, so $\overline{s}^*$ is strictly increasing. By working in a basis consisting of eigenvectors of P, the following results can also be shown (Shannon and Byrne, 2013b): $\overline{s}^*(\lambda) \to 0$ as $\lambda \to -\infty$; and $\overline{s}^*(\lambda) \to \infty$ as $\lambda \to \lambda_{\mathrm{I}}$ as long as

$$b \notin \mathrm{im}(P - \lambda_{\mathrm{I}} I) \tag{6.18}$$

where $\mathrm{im}(A)$ is the image of a matrix A. Suppose (6.18) holds. Then we have that the image of the function $\overline{s}^*$ is $(0, \infty)$. We already saw that $\overline{s}^*$ is injective since it is strictly increasing, so it has a well-defined inverse $\overline{\lambda} : (0, \infty) \to (-\infty, \lambda_{\mathrm{I}})$. Thus for any $s > 0$ we can find a λ, namely $\lambda = \overline{\lambda}(s)$, such that $\overline{c}^*(\lambda)$ is the unique global maximizer of A amongst trajectories with GMSD s. Note that in a sense it is unlikely for (6.18) not to hold, since $\mathrm{im}(P - \lambda_{\mathrm{I}} I)$ is a proper subspace of $\mathbb{R}^T$, and it seems unlikely that a general b would happen to lie precisely in this subspace. It turns out that even if $b \in \mathrm{im}(P - \lambda_{\mathrm{I}} I)$, an analysis similar to the above can be performed, with the result that for small values of s the unique optimal trajectory with GMSD s is of the form (6.13), and for large values of s there are multiple optimal trajectories with GMSD s, corresponding to the multiple solutions of $(P - \lambda_{\mathrm{I}} I)c = b$ with GMSD s (Shannon and Byrne, 2013b).

We can extend the above analysis, which examines the case $k = 0$, to the case of general k by considering the transformation $c \mapsto c - k\mathbb{1}$, $P \mapsto P$, $b \mapsto b - kP\mathbb{1}$. In the case of general k, (6.13) becomes

$$\overline{c}^*(\lambda) = (P - \lambda I)^{-1}(b - k\lambda\mathbb{1}) \tag{6.19}$$

For the sake of ease of comparison with methods defined below we rewrite this as

$$\overline{c}^*(\lambda) = \left(P - \lambda I\right)^{-1}\left(b - \overline{\nu}(\lambda)\mathbb{1}\right) \tag{6.20}$$

where

$$\overline{\nu}(\lambda) = k\lambda \tag{6.21}$$

The condition (6.18) becomes

$$b - k\mathbb{1} \notin \mathrm{im}(P - \lambda_{\mathrm{I}} I) \tag{6.22}$$

As before, we define $\overline{s}^*(\lambda) = \overline{s}_k(\overline{c}^*(\lambda))$. If (6.22) holds then $\overline{s}^*$ is a strictly increasing function with range $(0, \infty)$ and inverse $\overline{\lambda}$, and as before the unique solution of the original constrained maximization problem is of the form (6.19) for $\lambda = \overline{\lambda}(s)$.

We now consider how to turn the above discussion into an algorithm for fixed-spread GMSD generation. For a given $\lambda < \lambda_{\mathrm{I}}$, the optimal trajectory (6.19) can be computed by the same Cholesky-based algorithm used to compute the most likely trajectory in §3.2.4. This also allows $\overline{s}^*(\lambda)$ to be evaluated. Given $s > 0$, the appropriate value $\overline{\lambda}(s)$ of λ can be computed using a one-dimensional search algorithm such as simple bisection or Brent's method to find the solution of the equation $\overline{s}^*(\lambda) = s$. This parameter generation algorithm therefore involves computing one banded Cholesky decomposition and two banded triangular matrix solves for each value $\lambda \in \mathbb{R}$ at which $\overline{s}^*(\lambda)$ is evaluated.

We might be concerned that the one-dimensional search in the above algorithm might sometimes return solutions to $\overline{s}^*(\lambda) = s$ with $\lambda > \lambda_{\mathrm{I}}$. This would be a problem if it occurred since these solutions correspond to stationary points of the constrained function $A|_C$ that are not global maxima. However if $\lambda \geq \lambda_{\mathrm{I}}$ then the Cholesky decomposition of $P - \lambda I$ fails since this matrix is not positive definite. Thus we can easily ensure $\lambda < \lambda_{\mathrm{I}}$ by simply making the routine which computes $\overline{s}^*(\lambda)$ return ∞ if the Cholesky decomposition fails. Alternatively we may explicitly compute the smallest eigenvalue of P and restrict the one-dimensional search over λ to values smaller than this.

6.4.2 Analysis of fixed-spread GV generation

In this section we consider the problem of finding the trajectory produced by fixed-spread GV generation. Given $v > 0$, we wish to find the trajectory $c \in \mathbb{R}^T$ which maximizes $A(c)$ subject to the constraint $\overline{v}(c) = v$.

A similar approach can be used here as was used in §6.4.1. Considering the constraint to be $-\frac{1}{2}T\overline{v}(c) = -\frac{1}{2}Tv$ we obtain the Lagrangian

$$L(c;\lambda) = A(c) + \tfrac{1}{2}\lambda T\overline{v}(c) \tag{6.23}$$

$$= -\tfrac{1}{2}c^{\mathsf{T}}(P - \lambda J)c + b^{\mathsf{T}}c \tag{6.24}$$

Let λ_{J} be the smallest λ for which $P - \lambda J$ fails to be positive definite. If $\lambda < \lambda_{\mathrm{J}}$ then $P - \lambda J$ is positive definite, and so $x \mapsto L(x;\lambda)$ has a unique global maximizer given by

$$\overline{c}^*(\lambda) = (P - \lambda J)^{-1} b \tag{6.25}$$

By the Lagrangian sufficiency theorem this means that $\overline{c}^*(\lambda)$ is the optimal trajectory for some v, namely for $v = \overline{v}(\overline{c}^*(\lambda))$. For notational convenience we define a function $\overline{v}^*(\lambda) = \overline{v}(\overline{c}^*(\lambda))$ with domain $(-\infty, \lambda_{\mathrm{J}})$ which computes the GV of the optimal trajectory for a given λ. For $\lambda < \lambda_{\mathrm{J}}$ we have

$$T\overline{v}^*(\lambda) = (\overline{c}^*(\lambda))^{\mathsf{T}} J \overline{c}^*(\lambda) \tag{6.26}$$

$$= b^{\mathsf{T}}(P - \lambda J)^{-1} J (P - \lambda J)^{-1} b \tag{6.27}$$

Differentiating we obtain

$$T(\overline{v}^*)'(\lambda) = 2b^{\mathsf{T}}(P - \lambda J)^{-1} J (P - \lambda J)^{-1} J (P - \lambda J)^{-1} b \tag{6.28}$$

Since $P - \lambda J$ is positive definite, $x^{\mathsf{T}}(P - \lambda J)x \geq 0$ for any x and in particular for $x = (P - \lambda J)^{-1} J (P - \lambda J)^{-1} b$, and so we have $(\overline{v}^*)'(\lambda) \geq 0$, i.e. $\overline{v}^*$ is increasing. The following results can also be shown (Shannon and Byrne, 2013b): as long as the mean trajectory $\mu = P^{-1}b$ is not constant, i.e. is not equal to $k\mathbb{1}$ for some k, then $\overline{v}^*$ is strictly increasing; $\overline{v}^*(\lambda) \to 0$ as $\lambda \to -\infty$; and $\overline{v}^*(\lambda) \to \infty$ as $\lambda \to \lambda_{\mathrm{J}}$ as long as

$$b \notin \mathrm{im}(P - \lambda_{\mathrm{J}} J) \tag{6.29}$$

Suppose (6.29) holds. Then we have that the image of the function $\overline{v}^*$ is $(0, \infty)$, and so it has a well-defined inverse $\overline{\lambda} : (0, \infty) \to (-\infty, \lambda_{\mathrm{J}})$. Thus for any $v > 0$ we can find a λ, namely $\lambda = \overline{\lambda}(v)$, such that $\overline{c}^*(\lambda)$ is the unique global maximizer of A amongst trajectories with global variance v. As before it is in a sense unlikely for (6.29) not to hold, but even in this case an analysis similar to the one above can be performed, with the result that for small values of v the unique optimal trajectory with GV v is of the form (6.25), and for large values of v there are multiple optimal trajectories with GV v, corresponding to the multiple solutions of $(P - \lambda_{\mathrm{J}} J)c = b$ with GV v (Shannon and Byrne, 2013b).

There is an alternative expression for $\overline{c}^*(\lambda)$ that involves only banded matrix operations and so is more efficient to compute. The matrix J is full, but it is only a rank-one update away from the identity matrix, so we can use the matrix inversion lemma to write

$$(P - \lambda J)^{-1} = \left(P - \lambda I + \tfrac{\lambda}{T}\mathbb{1}\mathbb{1}^{\mathsf{T}}\right)^{-1} \tag{6.30}$$

$$= (P - \lambda I)^{-1} - \frac{\lambda (P - \lambda I)^{-1}\mathbb{1}\mathbb{1}^{\mathsf{T}}(P - \lambda I)^{-1}}{T + \lambda \mathbb{1}^{\mathsf{T}}(P - \lambda I)^{-1}\mathbb{1}} \tag{6.31}$$

as long as $P - \lambda I$ is invertible. This implies that the generated trajectory $\overline{c}^*(\lambda)$ is of the form (6.20) with

$$\overline{\nu}(\lambda) = \frac{\lambda b^{\mathsf{T}}(P - \lambda I)^{-1}\mathbb{1}}{T + \lambda \mathbb{1}^{\mathsf{T}}(P - \lambda I)^{-1}\mathbb{1}} \tag{6.32}$$

as long as $P - \lambda I$ is invertible. We have $P - \lambda I$ invertible for all $\lambda < \lambda_{\mathrm{J}}$ except $\lambda = \lambda_{\mathrm{I}}$ (Shannon and Byrne, 2013b). This alternative form also helps to highlight the similarity

between GMSD and GV generation: fixed-spread GMSD generation and fixed-spread GV generation both satisfy (6.20), but differ in the function $\overline{\nu}$ used.

We now consider how to turn the above discussion into an algorithm for fixed-spread GV generation. Since $P - \lambda I$ is banded, quantities of the form $(P - \lambda I)^{-1}x$ for various x can be computed efficiently using a banded LU decomposition. This allows $\overline{\nu}(\lambda)$, and so $\overline{c}^*(\lambda)$, to be computed in $O(T)$ time. By structuring the computation sensibly it is possible to compute $\overline{c}^*(\lambda)$ using one banded LU decomposition and four banded triangular matrix solves. Using the LU decomposition rather than Cholesky decomposition is necessary since $(P - \lambda I)$ is not positive definite for $\lambda_{\mathrm{I}} < \lambda < \lambda_{\mathrm{J}}$. There are high quality implementations of the banded LU decomposition available. For example LAPACK (Anderson et al., 1999) contains an implementation of the banded LU decomposition and the subsequent matrix solves required for computing $(P - \lambda I)^{-1}x$. The above discussion shows how $\overline{c}^*(\lambda)$, and so $\overline{v}^*(\lambda)$, can be evaluated. A one-dimensional search can then be used to find the solution of the equation $\overline{v}^*(\lambda) = v$ as in the GMSD case.

Care should be taken to ensure that the one-dimensional search in the above algorithm only returns solutions to $\overline{v}^*(\lambda) = v$ with $\lambda < \lambda_{\mathrm{J}}$, since solutions with $\lambda > \lambda_{\mathrm{J}}$ correspond to stationary points of the constrained function $A|_C$ that are not global maxima. One way to do this is to explicitly compute λ_{J} ahead of time. This can be done by using the result that λ_{J} is the unique solution of $T + \lambda \mathbb{1}^{\mathsf{T}}(P - \lambda I)^{-1}\mathbb{1} = 0$ between the first two eigenvalues of P (Shannon and Byrne, 2013b).

A typical example of the form of the function $\overline{v}^*$ is shown in Figure 6.1. This shows visually the points discussed earlier in this section: $\overline{v}^*$ is a strictly increasing function; $\overline{v}^*(\lambda) \to 0$ as $\lambda \to -\infty$; and $\overline{v}^*(\lambda) \to \infty$ as $\lambda \to \lambda_{\mathrm{J}}$.

We can also examine the relationship between the log probability of the optimal trajectory and v. Define $A^*(v) = A(\overline{c}^*(\overline{\lambda}(v)))$, so that $A^*(v)$ gives the maximum value of $A(c)$ amongst trajectories c with global variance v. We can gain various insights into the form of A^* assuming (6.29) holds. By considering A^* as a function of λ instead of v it can be verified that $(A^*)'(v) = -\frac{T}{2}\overline{\lambda}(v)$; indeed this sort of relationship is a standard feature of the method of Lagrange multipliers (Whittle, 1971). This provides a concrete interpretation of the Lagrange multiplier λ as (proportional to) the slope of the graph of $A^*(v)$ against v. The unique maximizer of the function A^* is of course the global variance $\overline{v}(\mu)$ of the mean trajectory, since the mean trajectory has the highest log probability of any trajectory. Differentiating and using the inverse function theorem we obtain

$$(A^*)''(v) = -\frac{T}{2}\frac{1}{(\overline{v}^*)'\left(\overline{\lambda}(v)\right)} \tag{6.33}$$

Since $(\overline{v}^*)'(\lambda) > 0$ this shows that $(A^*)''(v) < 0$, so A^* is strictly concave. It is also possible to show that $A^*(v) \to -\infty$ as $v \to \infty$ (Shannon and Byrne, 2013b). A typical example of the form of the function A^* is shown in Figure 6.2. We can see that A^* is indeed concave,

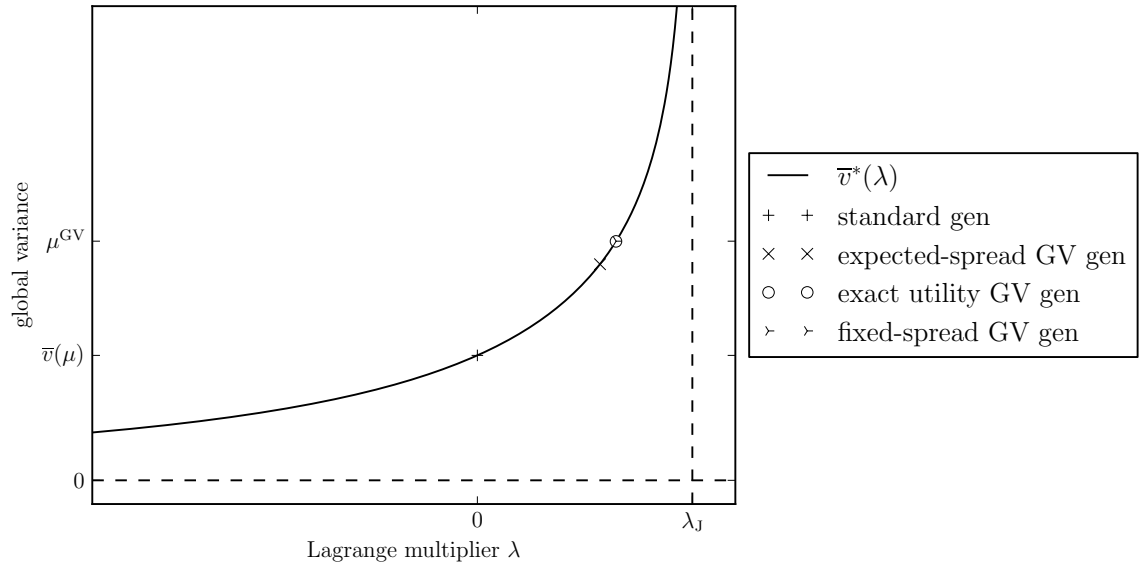


Figure 6.1: A typical example of the form of the function $\overline{v}^*$. For a Lagrange multiplier value $\lambda \in \mathbb{R}$, $\overline{v}^*(\lambda)$ gives the global variance of the optimal trajectory for that λ. Here $\overline{v}(\mu)$ is the global variance of the mean trajectory, μ^{GV} is the mean of the global variance pdf, and λ_{J} is the critical value of λ at which $P - \lambda J$ first fails to be positive definite. Points on the curve chosen by various different speech parameter generation methods are also shown.

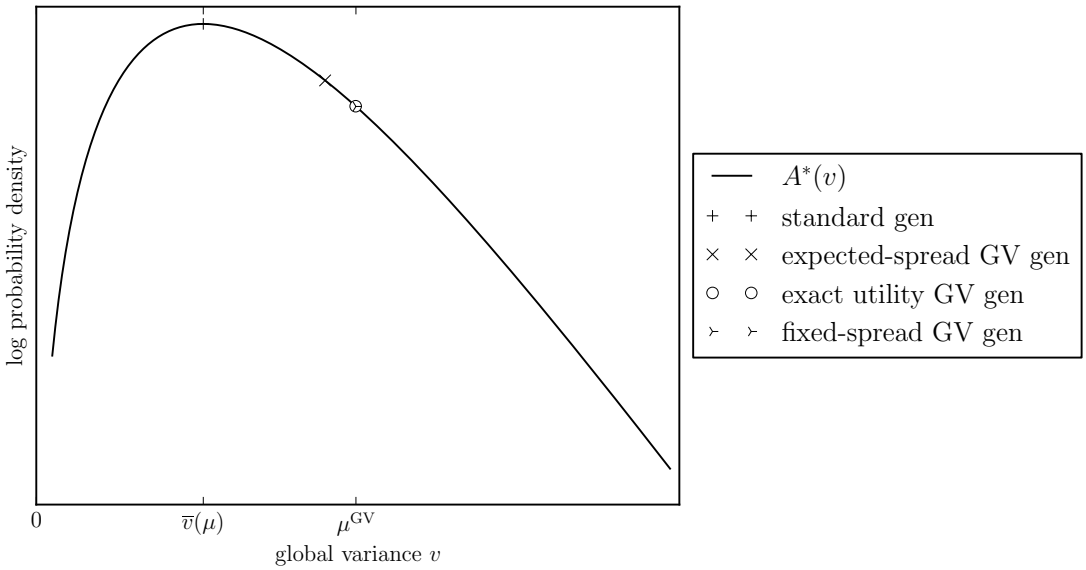


Figure 6.2: A typical example of the form of the function A^*. For a global variance value $v \geq 0$, $A^*(v)$ gives the log probability (up to a constant) of the optimal trajectory with global variance v. Here $\overline{v}(\mu)$ is the global variance of the mean trajectory and μ^{GV} is the mean of the global variance pdf. Points on the curve chosen by various different speech parameter generation methods are also shown.

with its maximum at $v = \overline{v}(\mu)$ where the slope is 0, corresponding to $\lambda = 0$, as expected.

6.4.3 Analysis of GV generation

In §6.4.2 we saw that the trajectory produced by fixed-spread GV generation is typically of the form (6.25) for some λ. Perhaps surprisingly this is also true for general GV generation. In this section we prove this fact, and use it as the basis of a novel algorithm for performing exact utility GV generation. For concreteness we focus on GV generation in this section, but essentially the same results apply to GMSD generation.

Firstly we show that, if (6.29) holds, any global maximizer of the utility function G is of the form (6.25). Suppose $c \in \mathbb{R}^T$ is a maximizer of the utility function G and has global variance $v > 0$. From §6.4.2 we know that there is some $\lambda < \lambda_J$ such that $\overline{v}(\overline{c}^*(\lambda)) = v$. But $\overline{c}^*(\lambda)$ is the unique maximizer of A, and so G, amongst trajectories with global variance v. Thus $c = \overline{c}^*(\lambda)$, and so c is of the form (6.25).

The above result also implies that there is typically a unique maximizer of G in the case of exact utility GV generation. If we define $G^*(v) = A^*(v) + B(v)$, then $G^*(v)$ gives the maximum value of $G(c)$ amongst trajectories c with global variance $v \geq 0$. Since G^* is the sum of two strictly concave functions it is also strictly concave, and so has at most one maximizer. Since $A^*(v) \to -\infty$ as $v \to \infty$ and B is bounded above, we have $G^*(v) \to -\infty$ as $v \to \infty$, and this implies that G^* has a maximizer, and so by strict concavity G^* has a unique maximizer. Since there is a unique optimal trajectory with global variance v, it follows that there is a unique maximizer of the utility function G.

A typical example of the form of the function G^* is shown in Figure 6.3. The utility-of-spread function B used here is the conventional one used by parameter generation considering global variance. This demonstrates visually the result proved in the previous paragraph that G^* is concave. The figure also provides an example of the phenomenon that, for the conventional choice of B, A has almost no influence on the global variance selected by exact utility GV generation. In the figure we plotted $200A^*$ rather than A^* itself, since otherwise A^* would have appeared indistinguishable from a horizontal line. This means the global variance selected by maximizing G^* is almost exactly equal to the value μ^{GV} preferred by B. We commented on this phenomenon in §3.2.6, and we will observe it experimentally in more detail in §6.5.2. This phenomenon is also responsible for the fact that exact utility GV generation and fixed-spread GV generation produce almost the same result in Figure 6.1 and Figure 6.2.

The fact that the maximizer of G is of the form (6.25) leads naturally to a new algorithm for doing exact utility GV generation. From this result we know that

$$\max_{c \in \mathbb{R}^T} G(c) = \max_{\lambda < \lambda_J} G(\overline{c}^*(\lambda)) \tag{6.34}$$

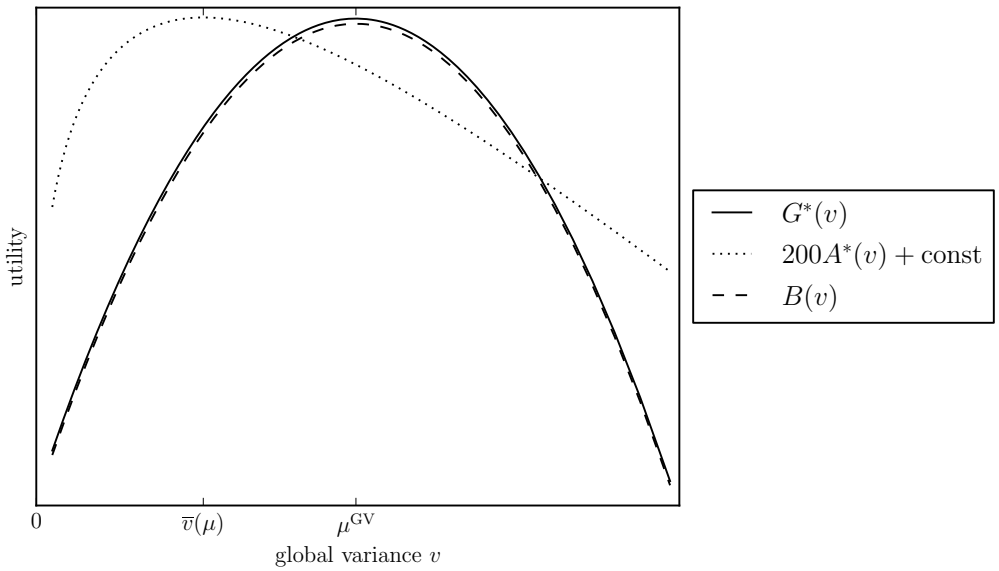


Figure 6.3: A typical example of the form of the function G^*. For a given global variance value $v \geq 0$, $G^*(v)$ gives the utility of the optimal trajectory with global variance v. Here $\overline{v}(\mu)$ is the global variance of the mean trajectory and μ^{GV} is the mean of the global variance pdf. Also shown are the two functions A^* and B which sum to give G^*, where A^* is the contribution from the log probability and B is the utility-of-spread function. The log probability contribution A^* is given a large scale factor to allow its variation to be seen.

where there exists a unique maximizer for each of the maximum operations. This means that rather than doing a T-dimensional numerical optimization of G we can instead do a 1-dimensional numerical optimization of $\lambda \mapsto G(\overline{c}^*(\lambda))$, with $\overline{c}^*(\lambda)$ for each potential $\lambda < \lambda_{\mathrm{J}}$ computed analytically using the LU decomposition-based algorithm described in §6.4.2. We refer to this as the *partially analytic GV generation algorithm*. This algorithm may be faster to get within a certain tolerance of the exact optimum of G than the conventional T-dimensional gradient ascent procedure is, but we do not investigate its speed here since in §6.4.6 we come up with an even faster algorithm that appears to give just as good naturalness.

6.4.4 View of GMSD and GV generation as modifying static parameters

For the standard HMM synthesis framework and the trajectory HMM it is possible to interpret GMSD and GV generation as a change to the model rather than a change to the parameter generation method. In this section we describe this interpretation; we discuss some of its implications in subsequent sections.

We first recap what static model parameters are and how they contribute to the natural parameters b and P of the distribution over trajectories. From (3.29) and (3.30) we know

that b and P consist of a sum of terms, one for each window. For the static window the vector b_0^{win} of b-value parameters over time contributes additively to b and the vector τ_0^{win} of precision parameters over time contributes additively to the diagonal of P. The vectors b_0^{win} and τ_0^{win} of static parameters over time are determined from the static model parameters $[b_{q0}, \tau_{q0}]_q$ based on the sequence $[\overline{q}(\theta_t)]_{t=1}^T$ of leaf indices.

In §6.4 we saw that the trajectory produced by GMSD and GV generation is of the form $(P - \lambda I)^{-1}(b - \nu \mathbb{1})$ for some $\lambda, \nu \in \mathbb{R}$. This is of the same form as the expression $P^{-1}b$ for the standard generation trajectory, but with $b - \nu\mathbb{1}$ instead of b and $P - \lambda I$ instead of P. This means that we can interpret the GMSD and GV generation methods as standard generation on a model with modified static parameters: subtracting λ from each static precision model parameter τ_{q0} and ν from each static b-value model parameter b_{q0} has the effect of subtracting λ from the vector τ_0^{win} and ν from the vector b_0^{win}, and doing standard generation after this modification gives the same result as the original GMSD or GV generation method.

There remain two important differences between standard generation on the modified model and the original GMSD or GV generation. Firstly, except for the fixed-multiplier GMSD generation method described in §6.4.6, the values of λ and ν vary from utterance to utterance, so the model modification posited above would have to be done for each utterance to make the correspondence exact. Secondly, while for the original model we typically enforce $\tau_{q0} > 0$ during training, for the modified model τ_{q0} may be negative. The consequences of this will be considered in §6.7.1.

6.4.5 Analysis of local maxima for utility GV generation

We mentioned in §6.2.4 that the GV utility function might have multiple local optima, potentially leading early-stopped utility GV generation to approach a suboptimal trajectory. By extending the analysis presented above, it is possible to conduct an analysis of the local maxima as well as the global maxima of G. For concreteness we focus on GV generation in this section, but essentially the same results apply to GMSD generation.

For GV generation we maximize the utility function (6.1), and we have seen that the global maximum of the utility function is of the form (6.25) for some $\lambda < \lambda_{\text{J}}$. By extending the analysis in §6.4.2 it can be shown that for utility GV generation any local, non-global maximum of the utility function is of the form (6.25) for some $\lambda_{\text{J}} < \lambda < \lambda_{\text{L}}$, where λ_{L} is the first stationary point of $\overline{v}^*(\lambda)$ greater than λ_{J} (Shannon and Byrne, 2013b). A given $\lambda_{\text{J}} < \lambda < \lambda_{\text{L}}$ corresponds to a local maximum of G if and only if it is a local maximizer of the function $\lambda \mapsto G(\overline{c}^*(\lambda))$. This makes it possible to evaluate how many local maxima a given utility function G has by simply counting them on a one-dimensional plot. Furthermore it can be shown that if the utility-of-spread function B is quadratic then G has at most one

global, non-local maximum (Shannon and Byrne, 2013b).

Performing the above analysis for the standard system S described in §4.6 shows that for all cepstral components and all training and test corpus utterances the utility function has no local, non-global maxima. Thus early-stopped utility GV generation is guaranteed to approach the global maximum rather than a suboptimal local maximum. For the trajectory system T roughly 0.2% of (utterance, cepstral component) pairs have a local, non-global maximum on the training and test corpora, and for the autoregressive system A roughly 5% of (utterance, cepstral component) pairs have a local, non-global maximum on the test corpus.

6.4.6 Fixed-multiplier generation

In this section we detail a form of GMSD generation which shares many of the advantages of the other forms of GMSD and GV generation presented above while being as computationally efficient as standard generation. We have seen that trajectories generated by GMSD and GV generation are of the form (6.20), where λ is an utterance-specific Lagrange multiplier and the precise value of λ for a given utterance depends on the form of GMSD or GV generation used. A natural question is whether quality degrades substantially if we instead use a fixed value of λ for all utterances. We refer to parameter generation which uses (6.20) with a fixed value of λ as *fixed-multiplier generation*. Note that the trajectory produced by fixed-multiplier generation is optimal (maximizes the log probability) given its spread. Fixed-multiplier generation is a form of spread-based generation since it can be obtained by setting $B(s) = 0$ for $s = \overline{s}(\overline{c}^*(\lambda))$ and $B(s) = -\infty$ elsewhere.

Using fixed-multiplier GMSD generation with the standard HMM synthesis framework or the trajectory HMM allows very fast generation, since computation of the modified static model parameters can be performed off-line and then the standard speech parameter generation algorithm used at synthesis time. This also makes it possible to use the time-recursive speech parameter generation algorithm for low latency generation.

For fixed-multiplier GV generation the value of $\nu = \overline{\nu}(\lambda)$ given by (6.32) or (6.48) varies from utterance to utterance and depends on the whole of b and P, so the tricks presented in the previous paragraph for the GMSD case are not possible. We could attempt to remedy this by fixing ν as well as λ, but we would soon realize that this is just GMSD generation with k set to ν/λ. For simplicity we therefore tend to use GMSD as the spread function rather than GV when using fixed-multiplier generation.

We propose three methods to train the fixed value of λ used for each portion and vector component of the speech parameters. Let $\Theta^\circ = [\Theta_r]_{r \in R^\circ}$ denote the training corpus state sequences and $C^\circ = [C_r]_{r \in R^\circ}$ denote the training corpus trajectories for the specified portion and vector component. We will find it helpful to make the dependence of (6.19) on

the state sequence explicit by writing $\overline{c}^*(\lambda;\theta)$ to denote the trajectory generated for a given multiplier λ and state sequence θ. Firstly for each utterance r we can find the multiplier Λ_r such that the generated trajectory $\overline{c}^*(\Lambda_r;\Theta_r)$ has GMSD equal to that of the corresponding natural trajectory C_r. We can then set λ to be the median, or slightly more generally some percentile, of the set $\{\Lambda_r : r \in R^\circ\}$. We refer to this as the *percentile method* of choosing the multiplier for fixed-multiplier GMSD generation. Secondly we can choose to use the value of λ which minimizes the *mean squared error (MSE)*

$$\frac{1}{|R^\circ|}\sum_{r\in R^\circ}\left(\overline{s}_k\left(\overline{c}^*(\lambda;\Theta_r)\right)-\overline{s}_k(C_r)\right)^2 \tag{6.35}$$

in the GMSD of the generated trajectories over the training corpus. We refer to this as the *MSE GMSD method* of choosing the multiplier for fixed-multiplier GMSD generation. Thirdly we can choose the value of λ using the procedure described for expected-spread GMSD generation in §6.2.3, but working at the level of the whole training corpus instead of a single utterance. That is, we can choose to use the value of λ for which the GMSD of the generated whole-training-corpus trajectory is equal to its expected value. We refer to this as the *corpus-level expected-spread method* of choosing the multiplier for fixed-multiplier GMSD generation.

So far we have presented fixed-multiplier generation as a rough approximation to other forms of spread-based generation, but it can also be viewed as a good approximation to other forms of *corpus-level* spread-based generation. Corpus-level expected-spread GMSD generation and fixed-multiplier GMSD generation using the corpus-level expected-spread method of choosing the multiplier yield the same collection of trajectories when applied to the collection Θ° of training corpus state sequences. The difference between the two generation methods is that for a new collection $\Theta = [\Theta_r]_r$ of state sequences, fixed-multiplier GMSD generation re-uses the fixed value of λ learned on the training corpus, whereas corpus-level expected-spread GMSD generation uses a new value of λ specific to Θ. Fixed-multiplier GMSD generation with the multiplier chosen using the corpus-level expected-spread method can therefore be seen as an approximation to corpus-level expected-spread GMSD generation, where Θ° is used as a proxy for Θ when deciding what value of λ to use. It seems reasonable to expect that if Θ° and Θ are both large and representative collections of state sequences, then the value of λ selected for these two collections will be very similar, and so the trajectories generated by the two methods will be very similar.

6.5 Normalized models and GV generation

Standard generation produces trajectories that have less global variance than natural trajectories. However this does not necessarily imply a deficiency in the probabilistic model,

since (6.8) shows that even if the probabilistic model was perfect we would expect the mean trajectory to have too little global variance. In this section we examine how well normalized models such as the trajectory HMM and autoregressive HMM model the global variance of trajectories. We approach this question from both a theoretical and empirical standpoint. We find that the trajectory HMM and autoregressive HMM both model the global variance fairly well. This suggests that GV generation is perhaps best thought of as compensating for a weakness in standard generation rather than a weakness in the probabilistic model.

6.5.1 Spread-matching property for the trajectory HMM

In §3.3.3 we described a statistics-matching property of the trajectory HMM acoustic model. Here we show that this implies a spread-matching property.

Suppose a trajectory HMM acoustic model is trained using maximum likelihood, and the estimated parameters lie in the interior of the space Ξ of allowed parameters. From (3.47) and (3.48) applied to the static window we have

$$\mathbb{E} \sum_{t:\overline{q}(\theta_t^\circ)=q} c_t = \sum_{t:\overline{q}(\theta_t^\circ)=q} c_t^\circ \tag{6.36}$$

$$\mathbb{E} \sum_{t:\overline{q}(\theta_t^\circ)=q} c_t^2 = \sum_{t:\overline{q}(\theta_t^\circ)=q} (c_t^\circ)^2 \tag{6.37}$$

where as before θ° is the whole-training-corpus state sequence, c° is the whole-training-corpus trajectory, and the expectation is taken assuming c is sampled from $\mathbb{P}(c \,|\, \theta = \theta^\circ, \lambda)$. The above equations imply that

$$\mathbb{E} \sum_{t:\overline{q}(\theta_t^\circ)=q} (c_t - k)^2 = \sum_{t:\overline{q}(\theta_t^\circ)=q} (c_t^\circ - k)^2 \tag{6.38}$$

for any $k \in \mathbb{R}$, i.e. the expected leaf-specific GMSD under the trained model is equal to the GMSD observed for that leaf on the training corpus. This in turn implies that

$$\mathbb{E} \sum_t (c_t - k)^2 = \sum_t (c_t^\circ - k)^2 \tag{6.39}$$

Thus a trained trajectory HMM system matches the corpus-level GMSD on the training corpus.

As discussed in §3.3.3, the statistics-matching property may fail for certain leaves q with $\tau_{q0} = 0$, and in practice we have found this does often occur for a few leaves. This implies that (6.39) only holds approximately, since (6.37) only holds for most leaves. Thus a trained trajectory HMM system approximately matches the corpus-level GMSD. Since there is little difference between the corpus-level GV and the corpus-level GMSD measured around the training corpus global mean, we would expect a similar statement to hold for GV with a further approximation.

method	description
N	using the natural trajectory
M	standard generation
H	HTS implementation of early-stopped utility GV generation, i.e. conventional speech parameter generation considering global variance
U	exact utility GV generation
E1	expected-spread GV generation
E2	expected-spread GMSD generation
S	sampling generation

Table 6.2: Generation methods used in the evaluation of how well trajectory HMM and autoregressive HMM model global variance.

The property described above does not guarantee everything we might like it to: it applies to the training corpus state sequences and trajectories not the test corpus; it applies at the corpus level so it does not guarantee the spread for every utterance will be modelled well; and it constrains only the expected value of the implied distribution over spread, not the other moments. Nevertheless it suggests that a trained trajectory HMM system is unlikely to systematically underestimate or overestimate the amount of GV for a collection of utterances. Indeed it shows that a trained trajectory HMM system approximately satisfies the much stronger property of on average matching the amount of spread for each leaf.

6.5.2 Spread-matching evaluation

In this section we present a simple evaluation of how well the trajectory HMM and autoregressive HMM model the GV for the unseen utterances in the test corpus. For each of a range of cepstral components and for each of the generation methods in Table 6.2, we computed the global variance of the generated trajectory for the 50 test corpus utterances. We then summarized each set of 50 values by a box-with-whiskers in a box plot. This allows the implied distribution over GV for different generation methods to be compared. The results for the trajectory HMM system T are shown in Figure 6.4 and the results for the autoregressive system A are shown in Figure 6.5. For the trajectory HMM we can see that:

- The mean trajectory (M) has too little GV, as expected. This is most pronounced for the higher cepstral components.
- Exact utility GV generation (U) and early-stopped utility GV generation (H) both do a fairly good job of matching the average amount of GV. Note that the GV of every generated trajectory has almost exactly the same value. This phenomenon was previously mentioned in §3.2.6 and §6.4.3.

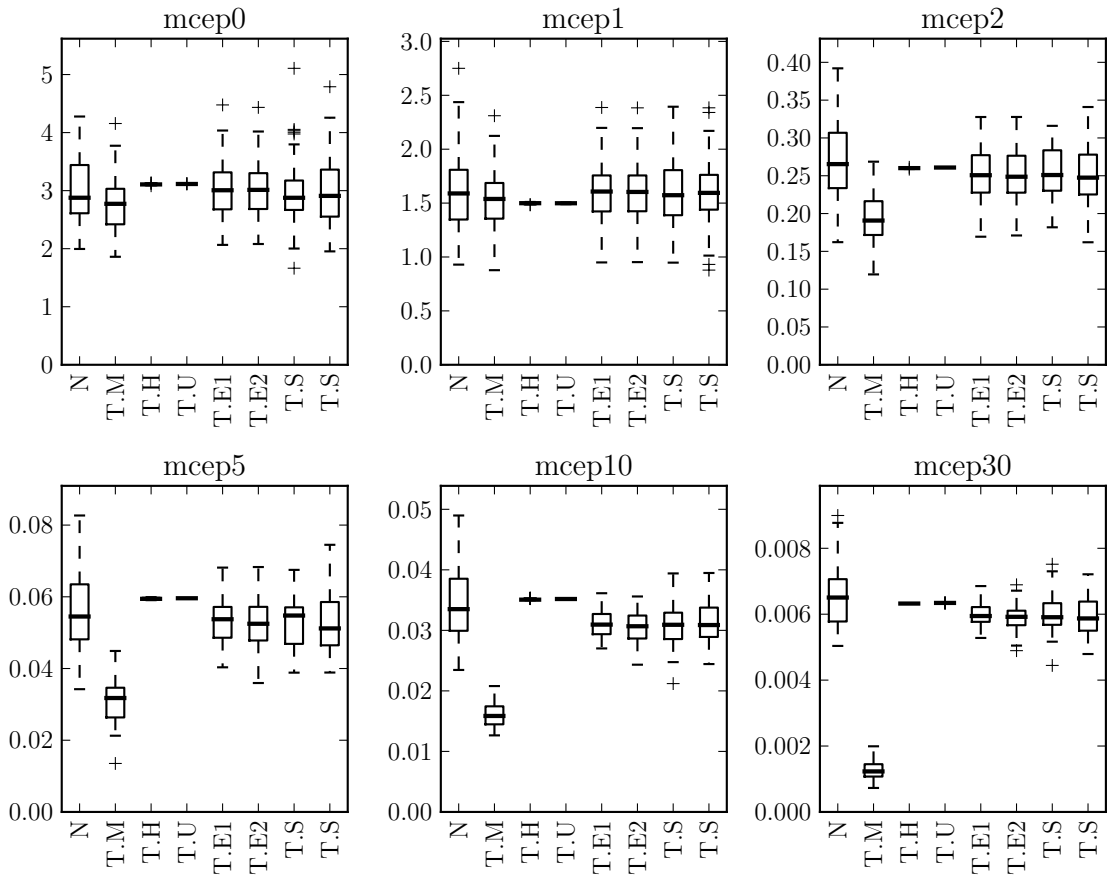


Figure 6.4: Box plots showing the amount of GV in trajectories generated by various generation methods for the trajectory HMM system T. Each box-with-whiskers is a summary of 50 GV values, one for each test corpus utterance. Each panel shows one cepstral component. Since sampling generation (T.S) is a random method it was performed twice.

- Expected-spread GV generation (E1) does a fairly good job of matching the average amount of GV. This implies that the trajectory HMM does not systematically underestimate or overestimate GV, as expected based on the discussion in §6.5.1. There is no reason that expected-spread GV generation should necessarily match the variance in GV, though for many cepstral components it seems to.

- Comparing method E2 to E1 shows that the typical amount of GV produced by expected-spread GMSD generation is similar to the expected GV.

- Sampling generation (S twice) does a fairly good job of matching both the average amount of GV and the variance in the amount of GV. This shows that the trajectory HMM models GV fairly well.

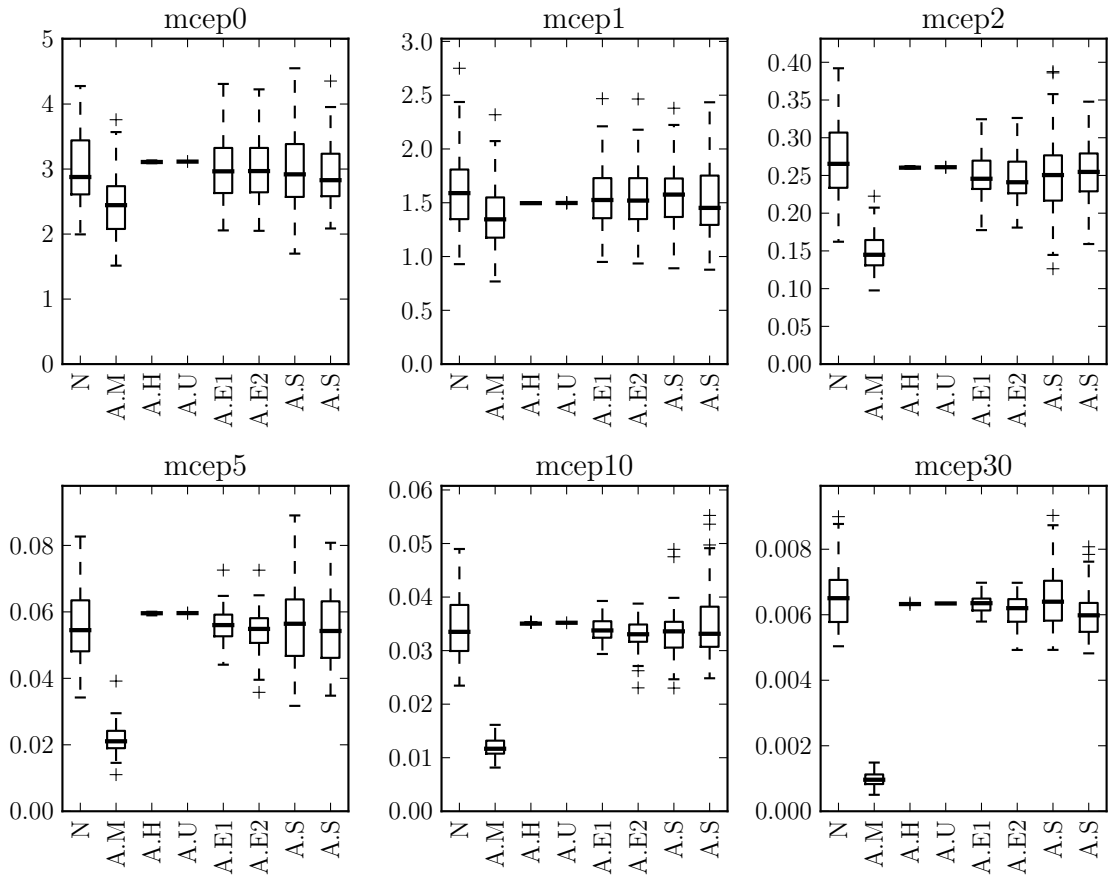


Figure 6.5: Box plots showing the amount of GV in trajectories generated by various generation methods for the autoregressive system A. Each box-with-whiskers is a summary of 50 GV values, one for each test corpus utterance. Each panel shows one cepstral component. Since sampling generation (A.S) is a random method it was performed twice.

The results for the autoregressive HMM are qualitatively very similar. The autoregressive HMM mean trajectories seem to have slightly less GV than the trajectory HMM mean trajectories, but we were not able to discern any other salient differences. We can therefore conclude that normalized models such as the trajectory HMM and the autoregressive HMM model GV fairly well.

For comparison the equivalent results for the standard framework are shown in Figure 6.6. Again every trajectory produced by exact utility GV generation and early-stopped utility GV generation has almost exactly the same value. The expected GV is now too small in many cases. This is unsurprising, since the lack of predictive variance in standard systems means that we would expect the first term in (6.7) to be smaller than it should be.

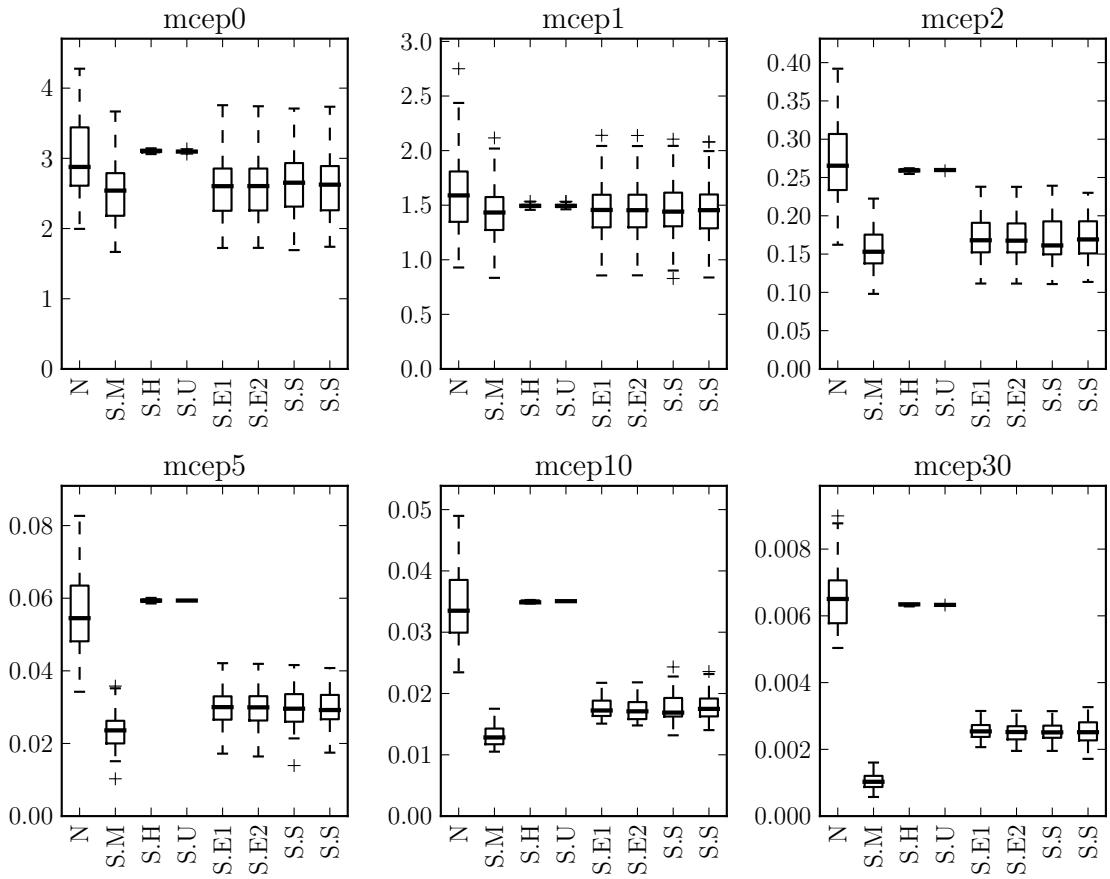


Figure 6.6: Box plots showing the amount of GV in trajectories generated by various generation methods for the standard system S. Each box-with-whiskers is a summary of 50 GV values, one for each test corpus utterance. Each panel shows one cepstral component. Since sampling generation (S.S) is a random method it was performed twice.

Note that if the trajectory HMM had a poor implied distribution over GV then, as discussed in §2.2.3, it would be conceptually easy to fix this by adding powers of GV to the list of feature functions defining the conditional exponential family. For example by adding GV and squared GV we can ensure that the first and second moments of the implied distribution over GV are correct. The simple evaluation above suggests that this extended model may not provide a much better probabilistic model of speech since the trajectory HMM already has a reasonable implied distribution over GV. Parameter estimation for the extended model is very complicated since the squared GV term makes the integral (2.35) intractable, but remarkably Zen et al. (2012) found that they were able to approximately train this model by combining contrastive divergence and Markov chain Monte Carlo methods. Unfortunately they did not compare the trajectory HMM and the extended model. It

should be noted that in their experiments they used a context-dependent model of log GV rather than the context-independent model of GV considered here.

6.6 A pathology in GMSD and GV generation

Prima facie expected-spread GMSD generation may seem like an extremely sensible generation method: we first compute a reasonable GMSD value and then compute the most likely trajectory with this GMSD. However GMSD and GV generation suffer from a pathology: for certain seemingly reasonable distributions over trajectories, the generated trajectory has large *excursions* where the trajectory value for a given frame is far outside the range of values that are "reasonable" for that frame according to the distribution over trajectories. We think this pathology is somewhat surprising, and so in this section we present a simple synthetic example to illustrate it. The pathology is not specific to expected-spread generation and would also occur for many other forms of GMSD and GV generation.

Consider a random signal, or equivalently a distribution over trajectories, that consists of amplitude-modulated white Gaussian noise plus a mean signal, where the mean signal and the envelope of the amplitude modulation are both smooth slowly-varying signals of small amplitude. A distribution over trajectories of this form with $\sigma_t \approx 1$ is shown in Figure 6.7. Despite the lack of any obvious pathologies in this distribution, we can see that GMSD and GV generation both produce a trajectory with a large excursion.

As a first step towards understanding why this behaviour occurs, consider the case where the mean trajectory is identically zero, since in this case the trajectory generated by expected-spread GMSD generation can be found analytically. Let σ_t denote the standard deviation at frame t, and assume that the sequence $\sigma = [\sigma_t]_{t=1}^T$ has a unique maximum at t^*. Suppose we measure GMSD around 0 and the expected spread is s. Then the optimal trajectory c^* consists of a positive or negative spike of height $\sqrt{Ts}$ at t^* and is zero everywhere else. To see this, suppose c is a trajectory with $\overline{s}(c) = s$. Then

$$\frac{1}{\sigma_t^2} \geq \frac{1}{\sigma_{t^*}^2} \quad \text{for all } t \tag{6.40}$$

$$\text{so } \frac{c_t^2}{\sigma_t^2} \geq \frac{c_t^2}{\sigma_{t^*}^2} \quad \text{for all } t \tag{6.41}$$

$$\text{so } \sum_t \frac{c_t^2}{\sigma_t^2} \geq \frac{1}{\sigma_{t^*}^2} \sum_t c_t^2 = \frac{1}{\sigma_{t^*}^2} Ts \tag{6.42}$$

$$\text{so } -\tfrac{1}{2} \sum_t \frac{c_t^2}{\sigma_t^2} \leq -\tfrac{1}{2} \frac{c_{t^*}^2}{\sigma_{t^*}^2} \tag{6.43}$$

$$\text{so } A(c) \leq A(c^*) \tag{6.44}$$

with equality if and only if $c_t = 0$ for all $t \neq t^*$. For reasonably long utterances the spike

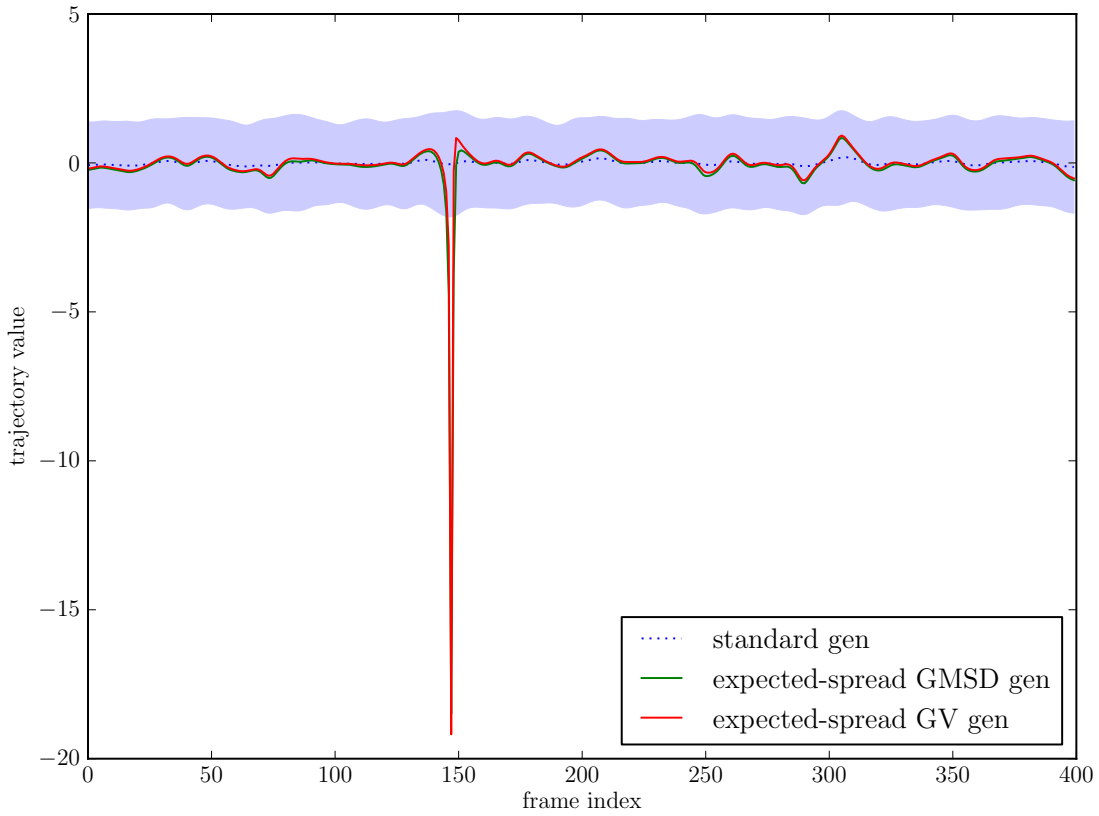


Figure 6.7: A simple synthetic example showing how GMSD and GV generation can introduce large excursions. The covariance matrix of the distribution over trajectories is diagonal. The shaded region shows the mean trajectory ±1.5 standard deviations. GMSD generation is performed with $k = 0$. The spike occurs where it does because of almost imperceptible changes in the distribution there, such as a slight increase in standard deviation.

can be quite large: if $\sigma_t \approx 1$ as in Figure 6.7 then the spike at t^* has magnitude $\sqrt{T}$, which for $T = 400$ is equal to 20, i.e. the spike is 20 standard deviations from the mean.

The slight variation in the mean trajectory in Figure 6.7 mellows the extreme analytic result just presented, but the same qualitative behaviour can be seen. In general we might expect similar behaviour whenever the amplitude of the mean trajectory is small relative to the standard deviation and there are only short-term correlations over time. When an excursion does occur, we would expect it to more often be in a region of high variance than in a region of low variance, since large deviations from the mean trajectory are penalized less severely, in terms of log probability, in high variance regions.

It is quite easy to formulate a statistic which shows how extreme the generated trajectory is. Consider the *maximum absolute z-value* $\max_t |z_t|$ where the *z-value* $z_t = (c_t - \mu_t)/\sigma_t$.

We refer to $z = [z_t]_{t=1}^T$ as the *z-value trajectory*. Note that $z_t \sim \mathcal{N}(0,1)$. For $T = 400$ the expected value of the maximum absolute z-value statistic is roughly 3.2, and the probability that its value is greater than 6 is very roughly $2 \cdot 10^{-6}$. However for the trajectory produced by expected-spread GMSD generation the value of this statistic is roughly 20. A value this large is tremendously unlikely. The relationship between the maximum absolute z-value and expected-spread GMSD generation is thus similar to the relationship between the GMSD and standard generation: the trajectory produced by standard generation has the maximum probability but an extremely unlikely GMSD value; similarly the trajectory produced by expected-spread GMSD generation is the most likely trajectory with given GMSD but has an extremely unlikely maximum absolute z-value.

6.7 Artifacts

As mentioned in §3.2.6, GV generation sometimes introduces artifacts. In this section we investigate the issue of artifacts in some detail. We will argue that excursions of the kind described in §6.6 are the cepstral-level phenomenon responsible for the perceptual-level phenomenon of artifacts. We therefore start by discussing excursions in detail, and discuss how and in what circumstances they are likely to lead to audible artifacts. It can be argued that operationally the important aspect of whether excursions lead to artifacts is whether reducing excursions also reduces artifacts. However formulating a simple and efficient generation method which reduces excursions while maintaining a reasonable amount of GV is non-trivial. For the standard HMM synthesis framework and the trajectory HMM, we show that the mathematical analysis presented in §6.4 provides a neat interpretation of how expected-spread GMSD and GV generation can introduce excursions, and suggests a simple way to reduce their occurrence. The result is a new efficient generation method designed to reduce excursions while generating trajectories with an appropriate amount of spread. In §6.8 we will see that it does indeed reduce excursions and that this leads to a reduction in artifacts.

In §6.6 we saw that GMSD and GV generation can sometimes introduce large excursions in the generated trajectory. For certain systems, utterances and cepstral components, excursions are indeed observed, and an example is shown in Figure 6.8. We chose to plot mcep 32 because it had the largest excursion for this utterance. Excursions are most common in the higher cepstral components, presumably because the higher cepstral components are quite noisy, with the amplitude of the mean trajectory being small relative to the standard deviation, and so they are susceptible to the effect described in §6.6.

For the diagonal-covariance case considered in §6.6, we found that z-values were a useful concept when discussing excursions, and this is also true in the non-diagonal-covariance case. For a distribution over trajectories $\mathcal{N}(\mu, \Sigma)$ and a trajectory c, we define the z-value

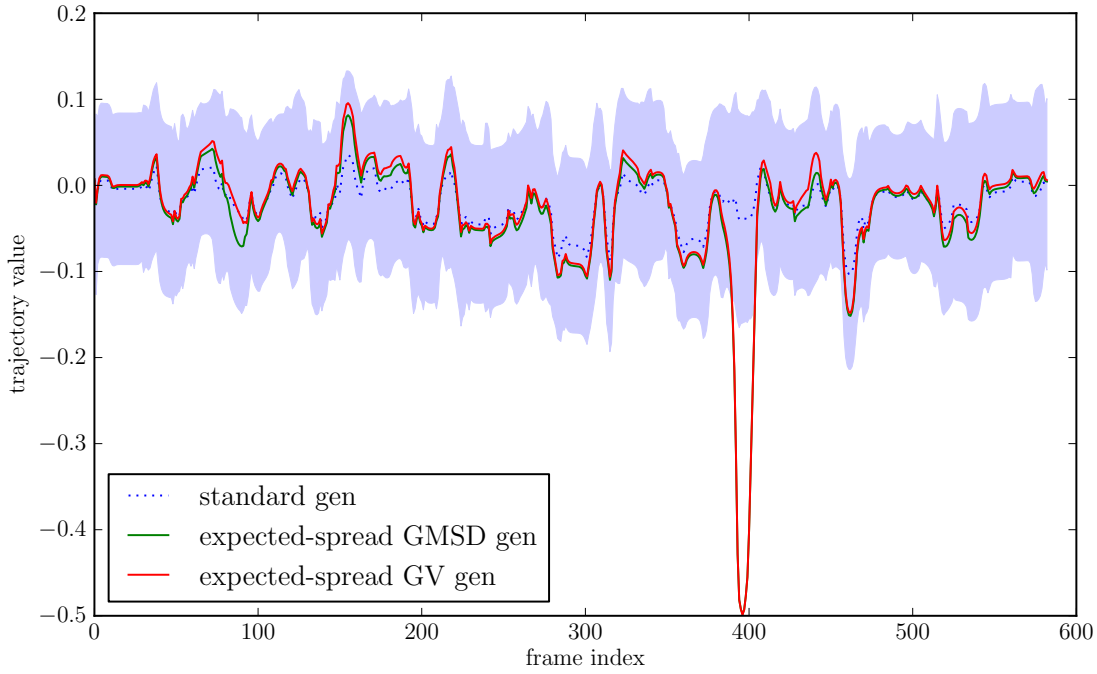


Figure 6.8: An example of an excursion introduced by GMSD and GV generation for mcep 32 for a test corpus utterance for trajectory HMM system T. In conjunction with similar excursions in other cepstral components, the excursion around frame 400 gives rise to an audible artifact.

at frame t by $z_t = (c_t - \mu_t)/\sqrt{\Sigma_{tt}}$. The function which takes a trajectory c and computes the z-value trajectory $z = [z_t]_{t=1}^T$ is a statistic. If $c \sim \mathcal{N}(\mu, \Sigma)$ then each z_t has marginal distribution $\mathcal{N}(0, 1)$, though the z_t values at different frames are not typically independent. If $|z_t|$ is large for a generated trajectory c then this indicates that the trajectory value c_t is extreme, in the sense that the probability under the model that c_t would be so far from μ_t is small. Examining absolute z-values thus tells us where excursions are happening and provides a fairly reasonable way to compare how "severe" two given excursions are. It also allows us to easily compare excursions in different cepstral components since the absolute z-values are directly comparable. As before the statistic $\max_t |z_t|$ gives an indication of whether an excursion occurs anywhere in a given generated trajectory.

Excursions in different cepstral components often occur simultaneously. The top panel of Figure 6.9 shows the absolute z-value for expected-spread GMSD generation for all frames and cepstral components for the same utterance as in Figure 6.8. For comparison the bottom panel shows the absolute z-values for the method designed to reduce excursions which will be described in §6.7.2. It is striking that many cepstral components simultaneously have a fairly large excursion just before frame 400. Anecdotally the occurrence of these "stripes"

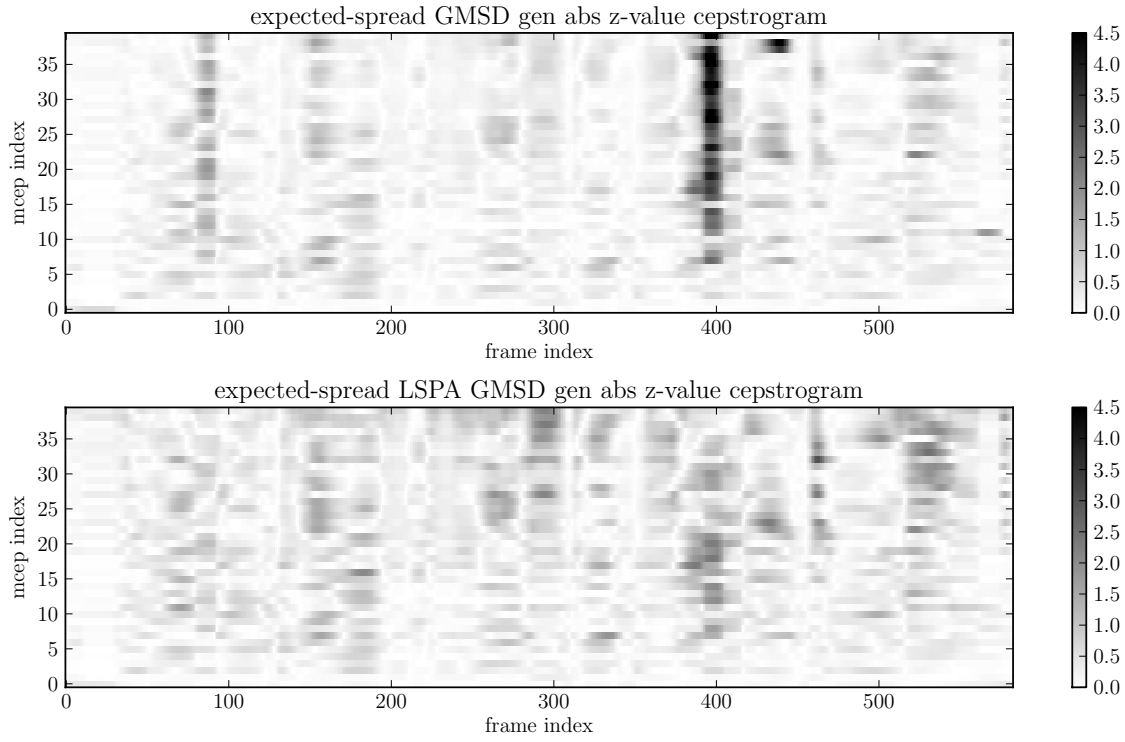


Figure 6.9: Cepstrograms showing absolute z-value trajectories for all cepstral components for two generation methods for one utterance for trajectory HMM system T. The "stripe" of large absolute z-values around frame 400 in the top panel corresponds to an audible artifact.

seems to be strongly correlated with the occurrence of audible artifacts. While even large stripes do not always give rise to a substantial audible artifact, audible artifacts appear to almost always arise from a stripe of some sort. This seems reasonable since an excursion in one cepstral component may not affect the log spectrogram as noticeably as simultaneous excursions in many components. The top panel of Figure 6.10 shows the log spectrogram for expected-spread GMSD generation for the same utterance, along with two other generation methods for comparison. We can see that there is indeed a visible anomaly around 4.7 kHz. Unsurprisingly this results in a clearly audible high-pitched whine artifact just before frame 400 in the synthesized audio.

If excursions are indeed responsible for artifacts then it should be possible to reduce the occurrence of artifacts by using generation methods which reduce the occurrence of excursions. In principle it is possible to define a generation method which does not suffer from excursions by simply restricting the value of $\max_t |z_t|$ for the generated trajectory, say to be at most 3. This would involve computing, for a given spread $s > 0$, the most likely trajectory c such that $\overline{s}(c) = s$ and such that $\mu_t - 3\sqrt{\Sigma_{tt}} \leq c_t \leq \mu_t + 3\sqrt{\Sigma_{tt}}$. It seems likely

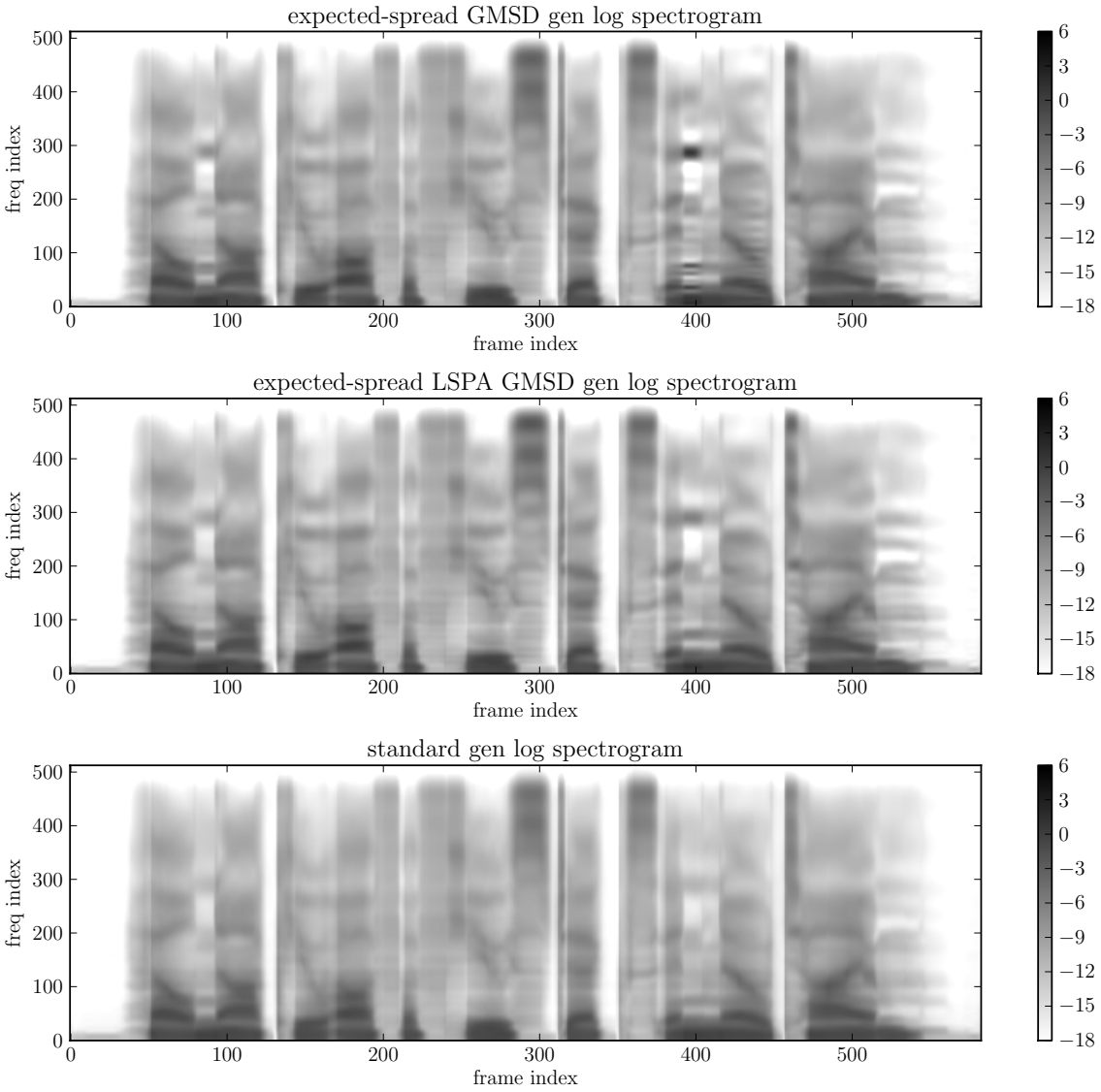


Figure 6.10: Log power spectrograms for three generation methods for one utterance for trajectory HMM system T. The frequency index is linear in frequency with 513 corresponding to 8 kHz.

that a reasonably efficient algorithm could be found to solve this constrained optimization problem, which has a quadratic objective function, a quadratic constraint and $2T$ linear constraints. However it seems unlikely this would be as fast as the algorithm in §6.4.1 for expected-spread GMSD generation, which has the same objective function and only the quadratic constraint. It is therefore desirable to find a more tractable generation method which reduces excursions.

In the remainder of this section we describe a simple way to predict which frames are "at risk" of excursions for the standard HMM synthesis framework and the trajectory HMM, and show how this can be used as the basis of a simple generation method which reduces excursions. The results in both sections are made possible by the mathematical analysis presented in §6.4. Some of the content below was previously presented in a conference paper (Shannon and Byrne, 2013a).

6.7.1 Negative modified static precision parameters

The analytic results presented in §6.4.4 show that for the standard framework and the trajectory HMM it is possible to interpret GMSD and GV generation as standard generation on a modified model. This involves subtracting utterance-specific values λ and ν from the static model parameters τ_{q0} and b_{q0} respectively. However whereas for the standard case we have $\tau_{q0} > 0$, the modified parameter $\tau_{q0} - \lambda$ may be negative. In this section we explain how such negative modified static precision parameters can lead to excursions in the generated trajectory. This provides a simple way of detecting which frames are "at risk" of excursions.

In §3.2.4 we presented a view of standard generation in terms of soft constraints. In this view, for a frame t with leaf index $\overline{q}(\theta_t) = q$, τ_{q0} and b_{q0} together place a soft constraint on the trajectory value c_t. The preferred trajectory value for this constraint is b_{q0}/τ_{q0}, and a large τ_{q0} means that trajectories deviating from this preferred value are harshly penalized, while a small positive τ_{q0} means that deviating trajectories are only mildly penalized. In this view $\tau_{q0} < 0$ corresponds to the counterintuitive soft constraint that the trajectory value c_t should *deviate* from the value b_{q0}/τ_{q0} as much as possible. The more negative τ_{q0}, the more a potential trajectory is penalized for not deviating substantially from this value. This means that, for frames where $\tau_{q0} < 0$, the static part of the model is pushing the trajectory away from finite values and towards $\pm\infty$. The non-static parts of the model still have an influence, so the trajectory does not actually reach $\pm\infty$, but the influence of the static part can lead to excursions.

The correspondence between negative modified static precision parameters and the occurrence of excursions is not exact. If the modified static precision parameter is not too negative, the segment is sufficiently short, or the influence of the non-static parts of the

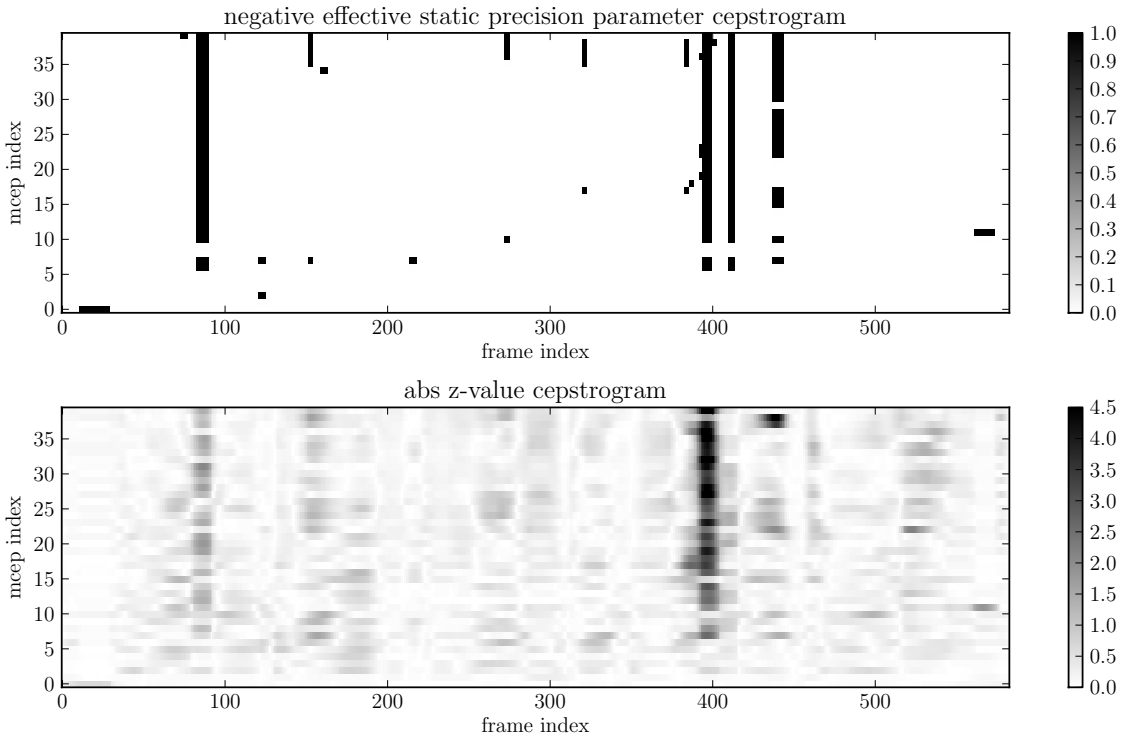


Figure 6.11: Cepstrograms showing the correspondence between frames with a negative modified static precision parameter (dark in top panel) and frames with excursions (dark in bottom panel) for expected-spread GMSD generation.

model is sufficiently strong, then there may be no excursion. Similarly not all excursions have negative modified static precision parameter. For the synthetic example shown in Figure 6.7, all the modified static precision parameters are positive. However the correspondence is often fairly close. In Figure 6.11 we show the correspondence for the utterance considered previously, and we can see that cepstral components and frames with substantial excursions almost always have a negative modified static precision parameter.

6.7.2 Local static parameter adjustment

We have seen that the modified static precision parameter effectively used by GMSD and GV generation can be negative, and that this may lead to excursions. In this section we describe a simple but effective fix for preventing excursions by soft-thresholding the modified static precision parameter to ensure it is non-negative. Note that because this approach involves adjusting the static parameters it is only applicable to the standard HMM synthesis framework and trajectory HMM, and not to the LGLAR HMM which has no concept of a static parameter.

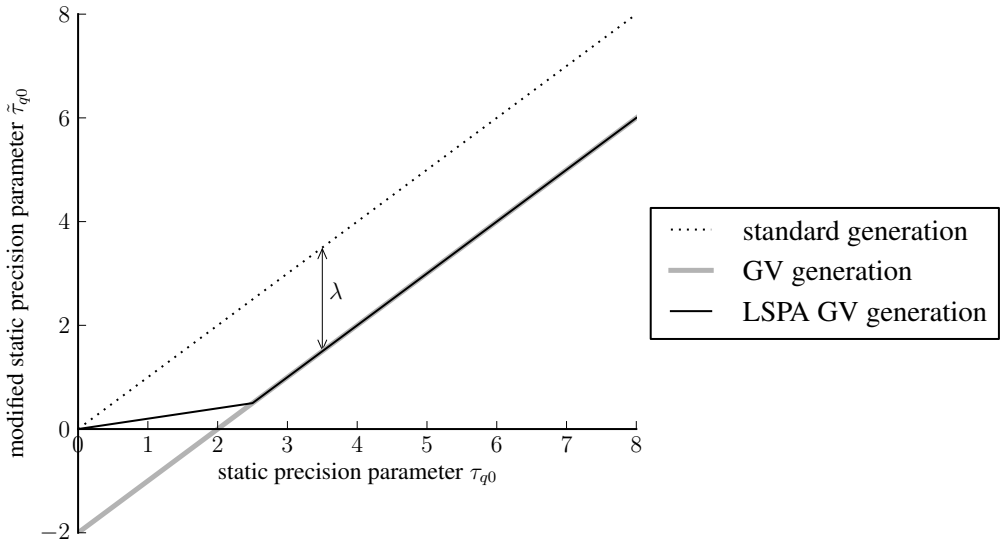


Figure 6.12: Conventional GMSD and GV generation effectively subtract a constant λ from the static precision parameter τ_{q0} for each leaf q. Local static parameter adjustment (LSPA) alters this to ensure the new value is never negative. Here $\xi = 0.2$ and $\lambda = 2$ in (6.45).

We have seen that GMSD and GV generation effectively use a modified static precision parameter $\tilde{\tau}_{q0}(\lambda)$ set to $\tau_{q0} - \lambda$, which may be negative. For *local static parameter adjustment (LSPA)* we instead set

$$\tilde{\tau}_{q0}(\lambda) = \max(\tau_{q0} - \lambda, \xi\tau_{q0}) \tag{6.45}$$

where λ may be any real value and $0 \leq \xi < 1$. We refer to a leaf q with $\tilde{\tau}_{q0}(\lambda) \neq \tau_{q0} - \lambda$ as having been *adjusted*. The form of this function is shown in Figure 6.12. LSPA may also be viewed in terms of an *adjustment weight*

$$w_q(\lambda) = \begin{cases} \min\left(1, \frac{1}{\lambda}(1-\xi)\tau_{q0}\right) & \text{if } \lambda > 0 \\ 1 & \text{if } \lambda \leq 0 \end{cases} \tag{6.46}$$

This definition is chosen so that $\tilde{\tau}_{q0}(\lambda) = \tau_{q0} - \lambda w_q(\lambda)$. Note that $0 \leq w_q(\lambda) \leq 1$, with $w_q(\lambda) = 1$ for leaves which are not adjusted. LSPA is local in time in the sense that it only makes an adjustment to the static precision parameter for frames where $\tau_{q0} - \lambda$ is negative or close to negative.

We generalize (6.20) to the LSPA case by setting

$$\overline{c}^*(\lambda) = (P - \lambda \operatorname{diag}(w(\lambda)))^{-1}(b - \overline{\nu}(\lambda)w(\lambda)) \tag{6.47}$$

where $w(\lambda)$ is the vector of adjustment weights over time, i.e. the t^{th} entry of $w(\lambda)$ is given by $w_q(\lambda)$ for $q = \overline{q}(\theta_t)$. We recover (6.20) in the case $w(\lambda) = \mathbb{1}$. For LSPA GMSD

generation we set $\overline{\nu}(\lambda) = k\lambda$ as before. For LSPA GV generation we set

$$\overline{\nu}(\lambda) = \frac{\lambda b^{\mathsf{T}}(P - \lambda \operatorname{diag}(w(\lambda)))^{-1} w(\lambda)}{\mathbb{1}^{\mathsf{T}} w(\lambda) + \lambda w(\lambda)^{\mathsf{T}}(P - \lambda \operatorname{diag}(w(\lambda)))^{-1} w(\lambda)} \tag{6.48}$$

We recover (6.32) in the case $w(\lambda) = \mathbb{1}$. There are several choices of $\overline{\nu}(\lambda)$ which satisfy the criteria of incorporating $w(\lambda)$ and of recovering (6.32) in the case $w(\lambda) = \mathbb{1}$ and we do not claim (6.48) is the only or necessarily the best choice. It can be verified that for both LSPA GMSD generation and LSPA GV generation there is a point after which increasing λ further makes no difference to the generated trajectory. Specifically if we define

$$\lambda_{\max} = (1 - \xi) \max_q \tau_{q0} \tag{6.49}$$

then for $\lambda \geq \lambda_{\max}$ all leaves are adjusted and the generated trajectory does not depend on λ.

The parameter ξ controls the amount of adjustment done: if $\xi = 0$ then the modified static precision parameter is hard-thresholded at zero, and so only leaves which would otherwise have a negative modified static precision parameter are adjusted; if $\xi = 1$ then for all $\lambda > 0$ all leaves are adjusted back to their original, unmodified values, and so the generated trajectory is just the mean trajectory. We set $\xi = 0.2$ based on small-scale preliminary experiments.

LSPA can be used with many different types of spread-based generation. It can be incorporated into fixed-spread, expected-spread and fixed-multiplier GMSD and GV generation. In the case of expected-spread generation the expectation is taken before any adjustment is made to the static parameters. For fixed-spread and expected-spread generation it may occasionally not be possible to attain the desired level of spread even for large λ, in which case we set λ to $\lambda_{\max}$ to attain a large spread. Fixed-multiplier LSPA GMSD generation, just like fixed-multiplier GMSD generation, allows very fast generation, since the modification of the model parameters can be performed off-line and the standard parameter generation algorithm, or its low latency time-recursive variant, used at synthesis time. LSPA can also be incorporated into exact utility GMSD and GV generation by restricting the maximization of G to trajectories of the form (6.47) for some $\lambda \in \mathbb{R}$.

We illustrate the effect of LSPA in Figure 6.13. The bottom two panels demonstrate visually how expected-spread GV generation effectively subtracts a constant λ from all static precision parameters and a constant ν from all static b-value parameters. There are four regions of frames where the modified static precision parameter is negative (middle panel), including the region just before frame 400 where an excursion (top panel) occurs. LSPA alters the modified static precision parameter in regions where it is negative or close to negative (middle panel), and this eliminates the excursion (top panel). It should be noted that for this figure the value of λ selected by expected-spread GV generation was

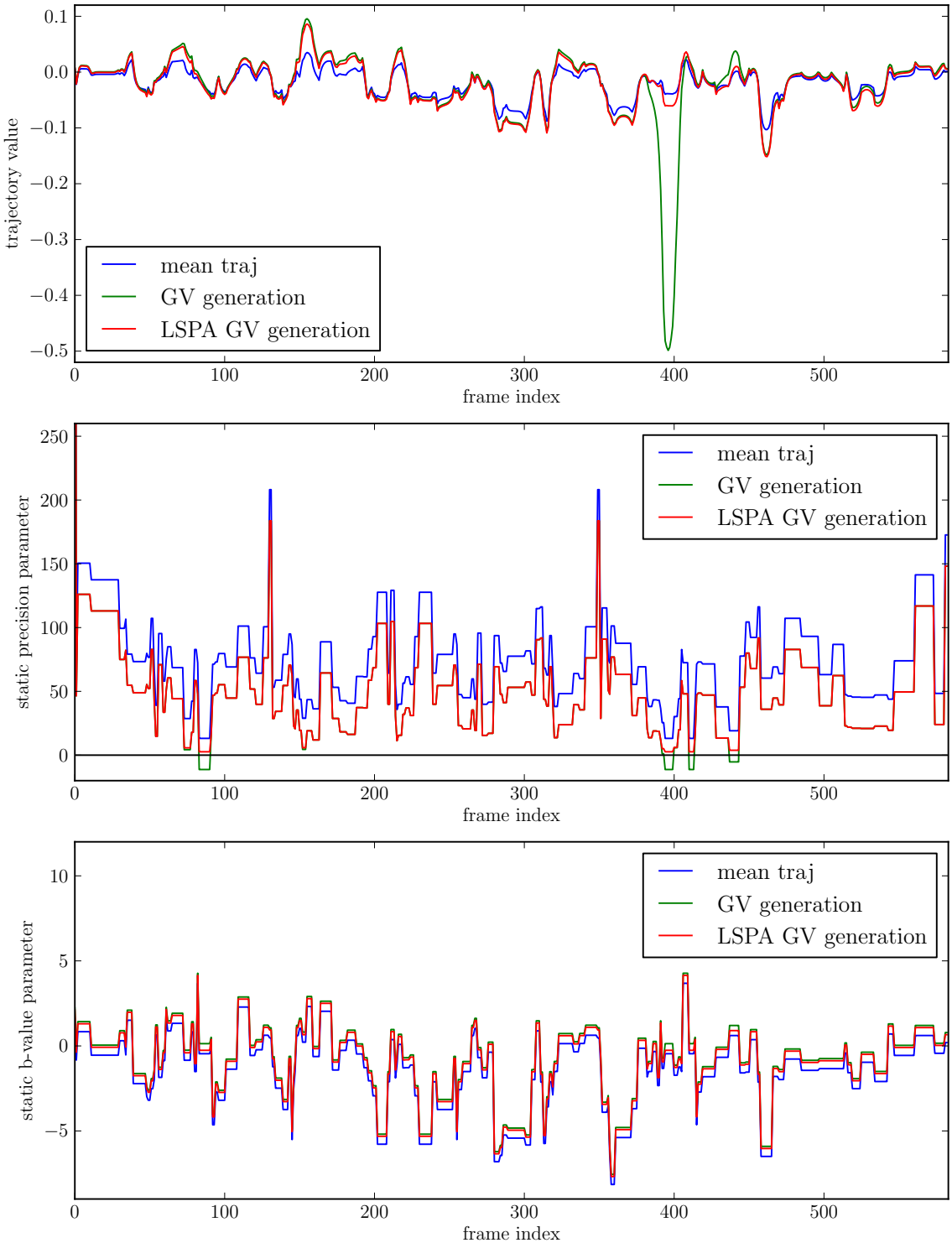


Figure 6.13: An example of the effect of LSPA, for a single utterance for mcep 32 for trajectory HMM system T. The top panel shows the trajectories produced by standard generation (mean traj), expected-spread GV generation and a form of LSPA GV generation. The middle panel shows the corresponding sequences of modified static precision parameters. The bottom panel shows the corresponding sequences of modified static b-value parameters.

re-used for LSPA GV generation. This more clearly illustrates the effect of LSPA. However since LSPA eliminates the excursion, this results in a trajectory with too small GV, and in practice expected-spread LSPA GV generation would be used instead.

Strictly speaking the above LSPA GMSD generation methods are not forms of GMSD generation since they do not maximize a utility function of the form (6.1) where $\overline{s}$ is a GMSD spread function. Similarly the above LSPA GV generation methods are not forms of GV generation. However fixed-multiplier LSPA GMSD generation is closely related to a form of corpus-level spread-based generation with a different spread function. Given a fixed multiplier λ and a trajectory value $k \in \mathbb{R}$, consider a spread function $\overline{s}_k(c) = \sum_t w_t(c_t - k)^2$ where the weight w_t is determined by the leaf $q = \overline{q}(\theta_t)$ at time t according to (6.46). Since $w_t = 1$ for frames where no adjustment is made, this spread function is similar to the GMSD spread function, but deweights frames with a small static precision parameter. It can be verified that fixed-multiplier LSPA GMSD generation is equivalent to corpus-level fixed-multiplier spread-based generation with the above spread function. This equivalence seems interesting but perhaps slightly contrived, especially since it only works for a fixed value of λ, and we are not sure that it is an illuminating way of thinking about LSPA generation. It is also interesting to note that generation where we match the corpus-level GMSD with a restriction on the maximum amount of GMSD any individual leaf can have results in a trajectory of the form (6.47) for some weight function. This can be verified using the method of Lagrange multipliers, though we omit the details here.

If excursions due to negative modified static precision parameters are indeed a source of artifacts as hypothesized in §6.7.1, then we might hope that LSPA would reduce artifacts while maintaining the other advantages of GMSD and GV generation. We will see experimentally that this is indeed the case.

6.8 Experiments

We performed a number of experiments. Firstly we used a range of generation methods with the trajectory HMM system T described in §4.6 to evaluate the effectiveness of LSPA at reducing excursions and artifacts. We assessed its ability to reduce excursions using an objective evaluation based on the maximum absolute z-value statistic. We assessed its ability to reduce artifacts without degrading naturalness using a subjective evaluation where listeners judged naturalness and the prevalence of artifacts. Secondly we used a range of generation methods with the standard system S to evaluate the extent to which fixing the value of the Lagrange multiplier λ causes a degradation in naturalness. For this we used a subjective evaluation where listeners judged naturalness. Both subjective evaluations were conducted as part of a single listening test. Many of these results were previously presented in a conference paper (Shannon and Byrne, 2013a). In addition, we performed timing

experiments to get an idea of how much slower the conventional GV generation algorithm is than the standard generation algorithm, and we performed a simplistic listening test designed to investigate the extent to which the trajectory HMM is more susceptible to artifacts than the standard framework.

6.8.1 Listening test

A listening test following the Blizzard Challenge-style methodology described in §3.4 was conducted over several weeks. It was completed by 24 native English speakers. The listening test consisted of three parts containing 48 utterances each.

In the first part of the listening test, the listeners evaluated the naturalness of the methods in §6.8.2 on a scale of 1 to 5. The second part was similar but used the methods in §6.8.3. In the third part the listeners heard the methods in §6.8.2 and for each utterance were asked to judge whether it contained an artifact, described as "a short distortion in the audio, e.g. a blip, a click, a pop, or a short high-pitched whine, but NOT a short pause in the incorrect position". They were told that they should expect roughly 1 in 10 utterances to contain an artifact, and were presented with two examples, generated by method T.U described below, of utterances containing an artifact. The third part was conducted after the first two so that perceived artifacts would not influence naturalness judgements.

6.8.2 LSPA evaluation

A number of different generation methods were used with the trajectory HMM system T in order to investigate the effectiveness of LSPA for reducing excursions and artifacts. These methods are summarized in Table 6.3. Exact utility GV generation for method T.U was implemented using the partially analytic GV generation algorithm, but we checked that running conventional HTS gradient ascent for millions of iterations gave nearly indistinguishable trajectories. Method T.E allows us to investigate the previously suggested hypothesis, mentioned in §6.7, that artifacts are partly caused by μ^{GV} being an inappropriate GV value for some utterances. In contrast to T.U, T.E allows the model to decide how much global variance it expects for each utterance. The spread-matching property of the trajectory HMM is useful here, since it ensures that any decrease in the number of artifacts for method T.E compared to method T.U is not due to method T.E systematically using less GV. Sampling generation was not used for the subjective evaluation since it is known to give very poor naturalness (Shannon et al., 2011). We used the trajectory HMM system T for assessing LSPA since exact utility GV generation with this model produced more artifacts than with the standard system S, providing a more robust test of LSPA's effectiveness at reducing artifacts.

method	description
N	natural speech
T.H	HTS implementation of early-stopped utility GV generation, i.e. conventional speech parameter generation considering global variance
T.U	exact utility GV generation
T.E	expected-spread GV generation
T.L1	expected-spread LSPA GV generation
T.L2	expected-spread LSPA GMSD generation
T.S	sampling generation (objective evaluation only)

Table 6.3: Generation methods used for the LSPA evaluation. All synthetic methods use the trajectory HMM system T.

To assess the ability of LSPA to reduce excursions, we computed the maximum absolute z-value on unseen test corpus utterances and summarized these values in a box plot. This is similar to the approach used above for the spread-matching evaluation. We used median alignments since otherwise it would not have been possible to compute the z-value of the natural trajectory. We considered only the higher cepstral components since this is where excursions are most prevalent.

The results of the objective evaluation are shown in Figure 6.14. We can see that:

- As expected, exact utility GV generation (T.U) and expected-spread GV generation (T.E) sometimes introduce excursions, particularly for the highest cepstral components.

- Early stopping (T.H) and LSPA (T.L1 and T.L2) are both effective at reducing the occurrence of excursions. In fact they are both overly conservative, and generate trajectories with fewer excursions than natural or sampled trajectories.

We can also see (N versus T.S) that natural trajectories tend to have slightly more excursions than expected by the model. This is of course an indication of a (slight) deficiency in the model. We defined excursions as trajectory values that are very unlikely under the model's probability distribution, but it is also possible to define them as trajectory values that are very unlikely under the "true" distribution, and by definition the natural trajectories do not have many excursions in the second sense. These results show that LSPA is very effective, and indeed is overly conservative, at reducing the occurrence of excursions.

The results of the subjective evaluation are shown in Table 6.4, Figure 6.15 and Figure 6.16. In a Mann-Whitney U test none of the synthetic methods had significantly different naturalness to any other. We can see that:

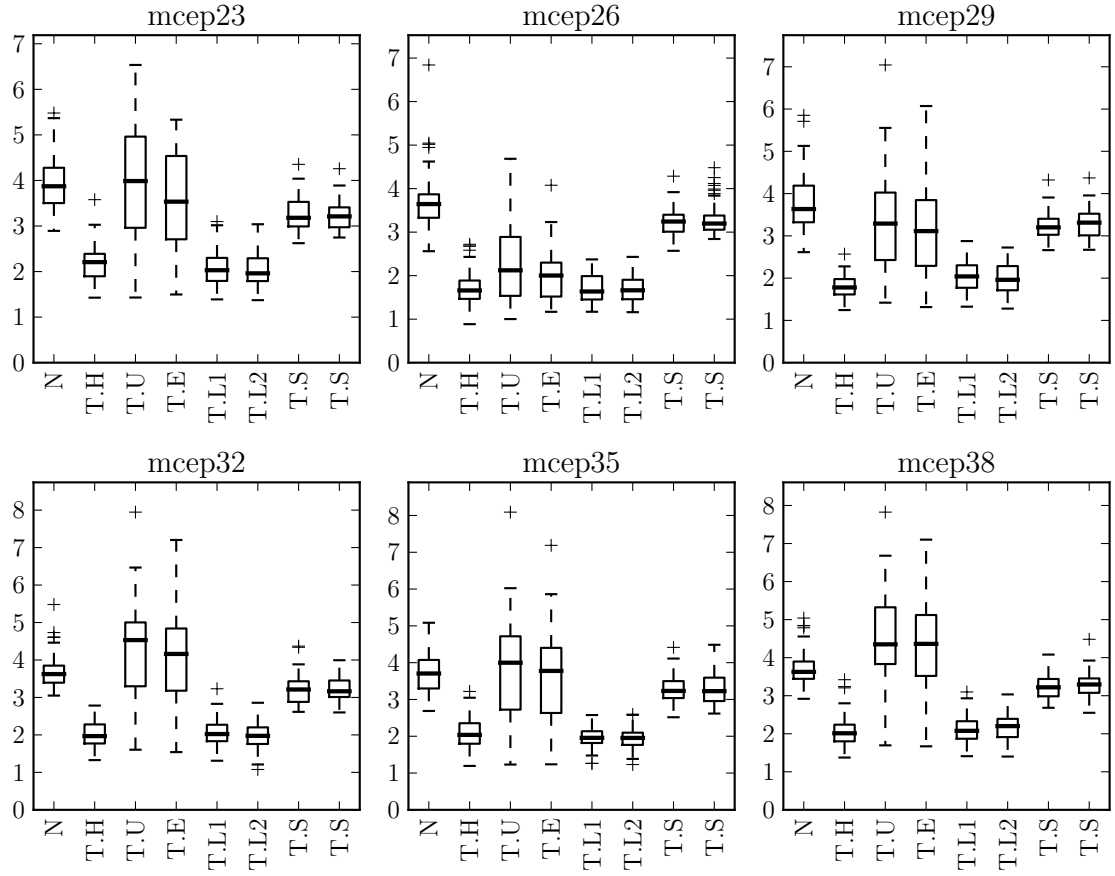


Figure 6.14: Box plots showing the maximum absolute z-value for trajectories generated by various generation methods for the trajectory HMM system T using median alignments. The maximum absolute z-value statistic is used to detect excursions. Each box-with-whiskers is a summary of 50 statistic values, one for each test corpus utterance. Each panel shows one cepstral component. Since sampling generation (T.S) is a random method it was performed twice.

method	mean OS	median OS	artifacts
N	4.6	5	5%
T.H	2.6	3	15%
T.U	2.5	3	66%
T.E	2.7	3	60%
T.L1	2.7	3	18%
T.L2	2.6	3	17%

Table 6.4: Results of the subjective LSPA evaluation. OS stands for opinion score and the artifacts column lists the proportion of utterances judged to contain an artifact.

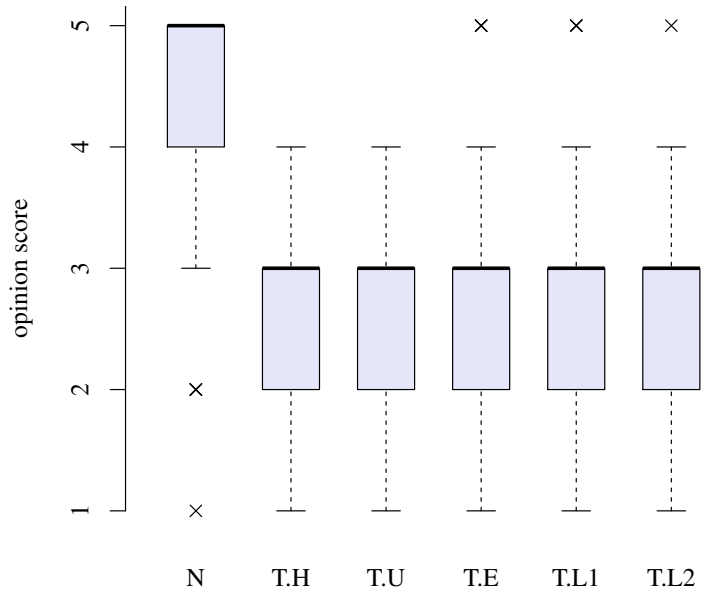


Figure 6.15: Box plot showing results for the subjective LSPA evaluation.

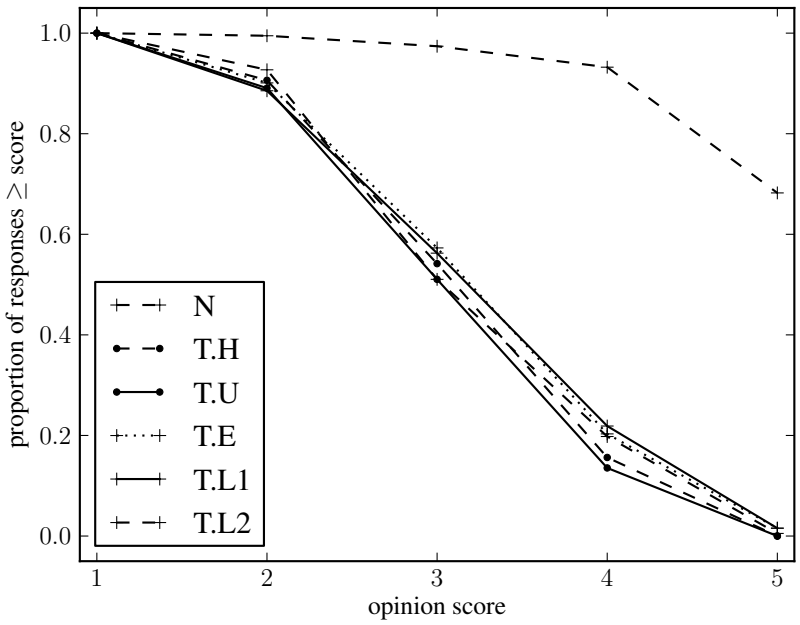


Figure 6.16: Complementary cumulative plot showing results for the subjective LSPA evaluation. For an opinion score s, the ordinate gives the proportion of participant responses that were s or greater. For any given opinion score larger ordinate values are better.

- Exactly optimizing the global variance utility function introduces many artifacts (T.U).
- Early stopping is very effective at reducing artifacts (T.U versus T.H).
- Few of the artifacts introduced by exact utility GV generation are due to the probabilistic model being asked to generate trajectories with global variance values it views as unreasonable (T.U versus T.E).
- Most artifacts occur for frames with small or negative modified static precision parameters (T.E versus T.L1). This can be seen by noting that LSPA does not adjust the modified static precision parameter when this is large and positive and that T.L1 suffers from far fewer artifacts than T.E.
- LSPA is almost as effective as early stopping at preventing artifacts and equally as natural (T.H versus T.L1 and T.L2).
- The relationship between excursions and artifacts appears to be fairly direct for the methods we consider. The methods T.U and T.E, which often introduce excursions for some of the higher cepstral components in Figure 6.14, suffer from many artifacts. The methods T.H, T.L1 and T.L2, which prevent excursions, suffer from far fewer artifacts.

We perceived almost no "GV-like" artifacts for methods T.H, T.L1 or T.L2, so listeners may be identifying artifacts not due to GV generation. In retrospect it would have been beneficial to include standard generation amongst the methods investigated in order to allow the rate of artifacts not due to GV generation to be established. Perhaps surprisingly the presence of artifacts had very little effect on naturalness judgements, with T.U and T.E rated as natural as T.H, T.L1 and T.L2 despite having many more artifacts. These results show that LSPA is very effective at reducing the occurrence of artifacts. They also provide some evidence that excursions are the cepstral-level phenomenon responsible for the perceptual-level phenomenon of artifacts, as the example in §6.7 suggested.

6.8.3 Fixed-multiplier LSPA evaluation

A number of different generation methods were used with the standard framework system S in order to investigate the effectiveness of fixed-multiplier LSPA GMSD generation. These methods are summarized in Table 6.5. We did not investigate the corpus-level expected-spread method of choosing the multiplier for fixed-multiplier LSPA GMSD generation here because we were not aware of this possibility at the time the experiments were conducted.

method	description
N	natural speech
S.M	standard generation
S.H	HTS implementation of early-stopped utility GV generation, i.e. conventional speech parameter generation considering global variance
S.FL1	fixed-multiplier LSPA GMSD generation with λ set using percentile method at 50%
S.FL2	fixed-multiplier LSPA GMSD generation with λ set using percentile method at 85%
S.FL3	fixed-multiplier LSPA GMSD generation with λ set using MSE GMSD method

Table 6.5: Generation methods for the fixed-multiplier LSPA evaluation. All synthetic methods use the standard system S.

method	mean OS	median OS
N	4.8	5
S.M	2.0	2
S.H	2.5	2
S.FL1	2.5	3
S.FL2	2.5	2
S.FL3	2.5	2

Table 6.6: Results of the subjective fixed-multiplier LSPA evaluation. OS stands for opinion score.

Preliminary listening by the authors suggested no "GV-like" artifacts were present for any of the above methods so only naturalness was formally evaluated.

The results are shown in Table 6.6, Figure 6.17 and Figure 6.18. In a Mann-Whitney U test, S.M was significantly different to all other methods, but none of the spread-based methods (S.H, S.FL1, S.FL2, S.FL3) was significantly different to any other. We can see that:

- As expected, conventional early-stopped utility GV generation gives substantial gains in naturalness over standard generation (S.M versus S.H).
- Fixed-multiplier LSPA GMSD generation appears to give at least as good naturalness as conventional early-stopped utility GV generation (S.H versus S.FL1/2/3).
- The three methods for choosing the fixed value of λ give very similar results (S.FL1, S.FL2 and S.FL3). This seems to suggest that the naturalness of trajectories generated using fixed-multiplier LSPA GMSD generation does not depend too strongly

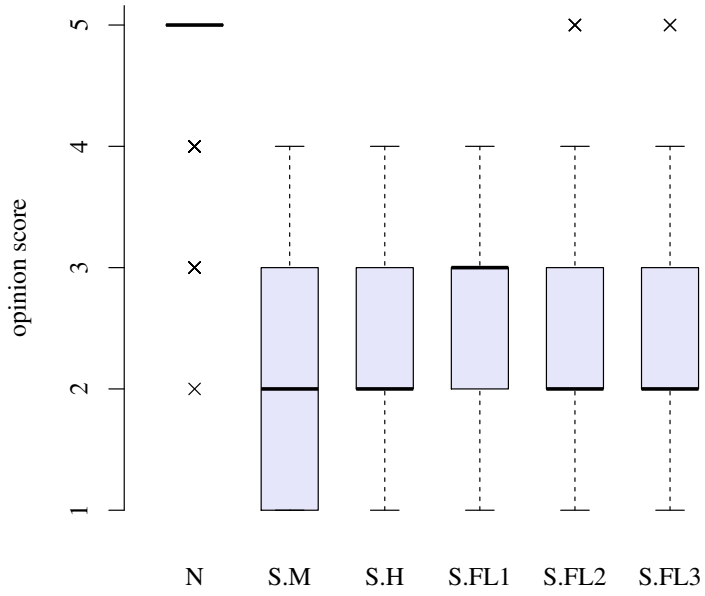


Figure 6.17: Box plot showing results for the subjective fixed-multiplier LSPA evaluation.

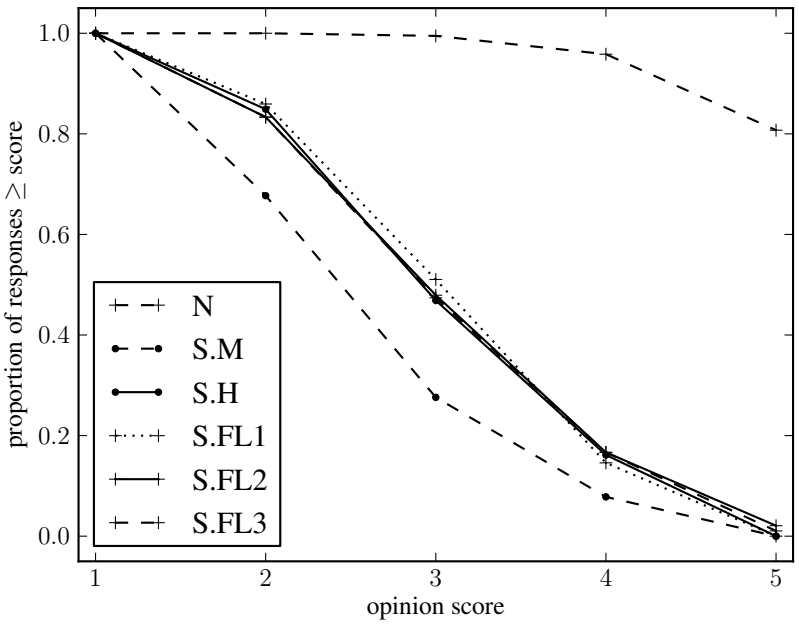


Figure 6.18: Complementary cumulative plot showing results for the subjective fixed-multiplier LSPA evaluation. For an opinion score s, the ordinate gives the proportion of participant responses that were s or greater. For any given opinion score larger ordinate values are better.

on the details of how the fixed multipliers are chosen, and we would conjecture that using the corpus-level expected-spread method to choose the fixed multipliers would also give similar naturalness.

These results show that fixing the multipliers, which has substantial computational advantages, leads to no degradation in performance. This makes fixed-multiplier LSPA GMSD generation an attractive parameter generation algorithm, since it is as fast as standard generation while improving naturalness as effectively as conventional parameter generation considering global variance.

6.8.4 Speed of generation methods

In this section we attempt to quantify precisely how much slower the conventional gradient ascent-based algorithm for GV generation is than the standard speech parameter generation algorithm. We measure the time taken to compute the generated trajectory starting from the b-value vector b and banded precision matrix P since this is where the two approaches differ, though it should be noted that this step is only one small part of the overall synthesis pipeline. Since fixed-multiplier LSPA GMSD generation can be implemented using the standard generation algorithm at synthesis time, our results also apply to this method.

An experimental investigation is particularly valuable because the GV algorithm is iterative, and so its complexity depends when it typically terminates. The standard algorithm has time complexity $O(TK^2)$, where T is the number of frames and K is the upper and lower bandwidth of P, whereas the GV algorithm has time complexity $O(TK^2)$ for initialization and $O(TKN)$ for gradient ascent, where N is the number of iterations performed.

Our experimental procedure was as follows. We used the profiler included in gperftools 2.2.1 (Google, 2014), which samples the current call stack once every 10 ms. By looking at the number of such samples taken within the relevant computational routines, it is possible to measure the speed of each algorithm. We evaluated the implementations in HTS 2.1, which is designed mainly for research use, and hts_engine 1.01 (HTS working group, 2014), which is designed to be faster and more lightweight. We used the standard system S to generate speech parameters for the label sequences in the training and test corpora, producing 679 717 frames. Whereas HTS has a carefully tuned stopped criterion for GV generation, hts_engine uses a fixed number of iterations, and we chose 13 since this is similar to the average number used by HTS. We ran the measurements on a system with a 3.2 GHz Intel dual-core CPU with 4 GB RAM. For all timings we took the median of 5 runs of the measurement procedure.

The speed comparison results are shown in Table 6.7. In all cases we can see that conventional GV generation is substantially slower than standard generation. We can also

software	time (s)		GV iterations per traj	speed factor
	standard gen	GV gen		
HTS	1.63	24.67	12.9	15.1×
hts_engine	0.92	7.63	13	8.3×

Table 6.7: Results of a speed comparison between the standard generation algorithm and the conventional GV generation algorithm. The time measured is the total time spent in the computational routines which take (b, P) and return the generated trajectory.

see that hts_engine is faster than HTS, though note that hts_engine omits band aperiodicity parameters and so generates 41 trajectories per utterance compared to 46 for HTS.

6.8.5 Susceptibility of trajectory HMM systems to artifacts

We mentioned above that artifacts seemed to be more prevalent with the trajectory HMM than with the standard framework. To quantify this difference we performed a very rudimentary one-listener listening test comparing S.U to T.U. For each of the test corpus utterances, the audio for the two systems was presented in a randomized order, and we judged whether each presentation contained an artifact or not. Afterwards the responses were collated. We judged that 1 of the 50 utterances for S.U and 31 of the 50 utterances for T.U contained an artifact. Thus the difference is quite large. We are not sure why the trajectory HMM is more susceptible to artifacts than the standard framework. The fact that the standard framework systematically underestimates predictive variance is not responsible, since it is easy to show that variance boosting makes no difference to the trajectories produced by fixed-spread GV generation, and Figure 6.6 and Figure 6.4 show that exact utility GV generation produces essentially the same trajectories as fixed-spread GV generation.

6.9 Discussion

In this section we discuss some remaining questions about spread-based generation and summarize our views on why GMSD and GV generation sometimes introduce artifacts.

6.9.1 Why does early stopping reduce excursions?

We saw experimentally that early stopping is very effective at reducing excursions and artifacts, but so far we have not considered why. In this section we outline a simplistic analysis of the effect of early stopping by working in an eigenbasis of the precision matrix. This sheds some light on why early stopping reduces excursions.

Consider a distribution over trajectories with natural parameters b and P. Instead of working in the conventional *frame basis* of $\mathbb{R}^T$ where the basis vectors are the discrete delta spikes at each frame t, we can instead work in the *eigenbasis* where the basis vectors are eigenvectors. We sort the eigenvalues $[\tilde{p}_i]_{i=1}^T$ in increasing order and refer to i as the *eigenindex*. For a trajectory c we denote the corresponding vector in the eigenbasis by $\tilde{c}$. The precision matrix is diagonal in the eigenbasis, so sampling generation samples each $\tilde{c}_i$ independently. The GMSD computed in the eigenbasis is the same as the GMSD computed in the frame basis, so it does not matter which basis we do GMSD generation in. As would be expected, excursions also occur for expected-spread GMSD generation in the eigenbasis, and these are almost always in the smallest two or three eigenindices. The excursions in the eigenbasis correspond to the excursions in the frame basis since the eigenvectors for the smallest eigenindices are typically smooth "bumps" localized to a particular segment of frames.

Early-stopped utility GV generation works by doing gradient ascent on the utility function $G(c) = A(c) + B(\overline{v}(c))$. This involves taking a series of discrete steps in $\mathbb{R}^T$, where the direction of each step is proportional to the gradient of G at the current point. It can be shown that the gradient of G at c is of the form $-(P - \lambda I)c + (b - \nu\mathbb{1})$ where $\lambda, \nu \in \mathbb{R}$ depend on c. We make two approximations to simplify analysis. Firstly we consider gradient ascent where the path in $\mathbb{R}^T$ is continuous instead of discrete. If we parameterize the path by α then $\frac{\mathrm{d}\overline{c}}{\mathrm{d}\alpha}(\alpha)$ is equal to the gradient of G at $\overline{c}(\alpha)$. Secondly we assume that $\lambda, \nu \in \mathbb{R}$ above are constant in α. If we choose the values of λ and ν appropriately then the final trajectory $\overline{c}(\infty)$ is the same as in the non-approximate case, but the path taken to this final trajectory will be different in general. With these two approximations the function $\overline{c}$ can be computed analytically by solving

$$\frac{\mathrm{d}\overline{c}}{\mathrm{d}\alpha}(\alpha) = -(P - \lambda I)\overline{c}(\alpha) + (b - \nu\mathbb{1}) \tag{6.50}$$

This involves a matrix exponential, and working in the eigenbasis this turns into a scalar exponential for each eigenindex, with

$$\tilde{\overline{c}}_i(\alpha) = \tilde{K}_i e^{-(\tilde{p}_i - \lambda)\alpha} + \frac{\tilde{b}_i - \nu\tilde{\mathbb{1}}_i}{\tilde{p}_i - \lambda} \tag{6.51}$$

for some constant $\tilde{K}_i \in \mathbb{R}$ depending on the initial conditions. The second term on the right side is the optimal value which would be chosen by exact utility GMSD generation. Thus the coefficient of eigenindex i decays exponentially from its initial value towards the optimal value, and the time constant for this decay is $1/(\tilde{p}_i - \lambda)$. The decay is therefore slowest for the smallest eigenindices, where the optimal value sometimes constitutes an excursion. By carefully tuning the stopping criterion it is therefore possible to avoid potential excursions caused by getting too close to the optimal value in the lowest eigenindices while gaining the benefit of increased spread from getting close to the optimal value for the other eigenindices.

6.9.2 Why do excursions occur in stripes?

We mentioned in §6.7 that excursions in different cepstral components often occur simultaneously. This seems surprising since the statistical model assumes that the trajectories for different cepstral components are conditionally independent and speech parameter generation operates separately on each component. In this section we describe a possible explanation for this phenomenon.

Since parameter estimation and speech parameter estimation operate largely separately for each cepstral component, a stripe of excursions at a given time suggests that there is some property of the state at that time, or rather the leaf at that time, that results in many of the cepstral components for that state being susceptible to excursions. As a crude way to investigate this we performed a simple experiment using trajectory HMM system T. Let Z_{t1i} be the z-value corresponding to the speech parameter value C_{t1i} for frame t and cepstral component i. For each utterance in the test corpus, we computed the frame

$$\arg\max_t \left(\frac{1}{40} \sum_{i=0}^{39} Z_{t1i}^2 \right)^{\frac{1}{2}} \tag{6.52}$$

for which the root-mean-square z-value was greatest, as a proxy for the position of the most substantial stripe, and recorded the leaf index for this frame. We thus obtained a list of 50 leaf indices which may be susceptible to excursions. Just two "bad leaves" made up around 40% of this list, and these leaves had an average root-mean-square z-value of more than 5. Just four bad leaves made up around 60% of the list. Spot checks for a few utterances suggested that these leaves did indeed often lead to stripes. We mention in passing that for around 60% of the utterances the current phoneme at the maximum was `n`, so this nasal appears to be far more susceptible to excursions than other phonemes, though we are not sure why this is the case.

There were a few exceptional aspects that these four "bad leaves" had in common. A summary of where these four leaves ranked amongst all leaves according to several metrics is shown in Table 6.8. We can see that all four leaves had very high occupancy. A leaf having a high occupancy means that, if it is susceptible to excursions, then these excursions will tend to occur in many utterances. The high occupancy of these leaves also shows that robust estimation was not an issue. The four bad leaves all had very large variance for some cepstral components, for example mcep 32, and large variance across many of the higher cepstral components. We commented in §6.6 that we might expect regions with large variance to be more susceptible to excursions. The combination of a very large occupancy and a large variance across many cepstral components was rare in general; there was only one other leaf that had both as extreme an occupancy percentile and as extreme an average variance percentile as any of these four leaves. The four bad leaves also all had very small

description	leaf 1	leaf 2	leaf 3	leaf 4
occupancy percentile	98%	96%	97%	95%
variance percentile, mcep 32	98%	99%	96%	97%
variance percentile, mean for mcep 25 to 34	93%	87%	88%	93%
τ_{q0} percentile, mcep 32	98%	98%	96%	93%
τ_{q0} percentile, mean for mcep 25 to 34	97%	93%	95%	88%
"no" proportion percentile	99%	86%	96%	76%

Table 6.8: Various percentiles computed for four "bad leaves" which we identified as being susceptible to excursions for system T. The percentile value shows where the given leaf ranks amongst all leaves, e.g. 98% of leaves had an occupancy smaller than leaf 1. The results are computed on the training corpus. Here variance means the variance of trajectory values for frames assigned to a given leaf, τ_{q0} refers to the static precision parameter for a given leaf, and "no proportion" is the proportion of "no" answers encountered on the path from the root node to a given leaf node in the decision tree.

static precision parameter for mcep 32, and small static precision parameter across many of the higher cepstral components. We saw in §6.7.1 and §6.8.2 that excursions tend to occur where the static precision parameter is small. The above discussion suggests that the reason excursions are often observed in many cepstral components simultaneously for these four leaves is that the leaves are both commonly occurring and have a large variance, or a small static precision parameter, across many cepstral components.

Why do certain leaves have both a large occupancy and a large variance across many cepstral components? Is it an inherent feature of the data, or something to do with the estimation procedure? Examining the list of question answers for the four leaves, we noticed that they contained a lot of "no" answers. This is quantified in the last row of Table 6.8. It has previously been suggested that, because of the style of question typically used for decision tree clustering, the states which are clustered together for such a leaf may be heterogeneous, sharing few phonemic and linguistic properties other than *not* belonging to certain relatively small sets.[1] This heterogeneity may be responsible for the four leaves having large variances. Since decision tree clustering does not treat each cepstral component separately, but rather chooses the question at a given node based on all components at once, this would explain why these leaves have large variances across many components.

[1] Personal communication, Simon King, 2014.

6.10 Fixed-multiplier LSPA GMSD generation recipe

For convenience we now specify the steps involved in implementing fixed-multiplier LSPA GMSD generation starting from a trained standard system or trajectory HMM system. This is done separately for each cepstral component. We choose the multiplier λ using the corpus-level expected-spread method. We assume that we have a function f which implements standard generation. The input to f is a collection $[b_{qd}]_{q,d}$ of b-value model parameters for each leaf q and window d, a collection $[\tau_{qd}]_{q,d}$ of precision model parameters, and a state sequence $\theta = [\theta_t]_{t=1}^T$. The output of f is the mean trajectory $\mu = [\mu_t]_{t=1}^T$. The function also needs to know the windows used and the mapping $\overline{q}$ from states to leaves, but we leave these implicit in our notation. The steps involved are as follows:

- Implement a function g which takes a value $\lambda \in \mathbb{R}$ and a static precision parameter τ_{q0}, or a vector of such values, and returns the value $g(\lambda, \tau_{q0}) = \max(\lambda, (1-\xi)\tau_{q0})$, or a vector of such values. Here $\xi = 0.2$.

- Implement a function h which takes values $k, \lambda \in \mathbb{R}$ and a collection $[b_{qd}, \tau_{qd}]_{q,d}$ of model parameters and returns a new collection of model parameters with altered static b-value parameters $\tilde{b}_{q0} = b_{q0} - kg(\lambda, \tau_{q0})$ and altered static precision parameters $\tilde{\tau}_{q0} = \tau_{q0} - g(\lambda, \tau_{q0})$.

- Implement a function j which takes a value of $\lambda \in \mathbb{R}$ and computes the overall GMSD of the generated trajectories for the whole training corpus. Computing $j(\lambda)$ involves first altering the original model parameters using h, then generating trajectories on the whole training corpus for this altered model using f, then computing the overall GMSD of the generated trajectories.

- Compute the average trajectory value $k \in \mathbb{R}$ over the whole training corpus. This is the value around which we will measure GMSD.

- Compute the overall GMSD $s > 0$ of all the training corpus trajectories (as if the training corpus trajectories were one long trajectory).

- Use Brent search or simply bisection to find the value $\lambda \in \mathbb{R}$ for which $j(\lambda)$ is equal to s.

- Use this value of λ to alter the original model parameters using h.

- Use the altered model instead of the original one during standard generation. This provides GV generation-like naturalness without artifacts but using the fast and simple standard generation algorithm.

6.11 Summary of contributions

The major novel contributions of this chapter are: a theoretical analysis of GV generation based on Lagrange multipliers showing that, for the standard framework and trajectory HMM, GV generation is equivalent to standard generation on a modified model (§6.4); the discovery that, both theoretically and in practice, normalized models such as the trajectory HMM and autoregressive HMM model GV very well (§6.5); the hypothesis that excursions are the cepstral-level phenomenon responsible for audible artifacts (§6.7) and experimental evidence supporting this hypothesis (§6.8.2); and a new generation method (fixed-multiplier LSPA GMSD generation) which is as fast as standard generation but improves naturalness as much as conventional GV generation, without introducing artifacts (§6.7.2 and §6.8.3).

Minor novel contributions of this chapter include: the concept of spread-based generation (§6.2); the idea of using GMSD generation as an easy-to-analyze alternative to GV generation (§6.2.2); the idea of using the expected GV under the model as a way of selecting an appropriate GV value for generation (§6.2.3); the discovery that the conventional GV utility function has a unique global maximum (§6.4.3) and at most one local, non-global maximum (§6.4.5); the idea of matching GV at the corpus level instead of the utterance level (§6.2.5); an efficient algorithm for expected-spread GV generation (§6.4.2); a partially analytic algorithm for exact utility GV generation, involving a one-dimensional numerical optimization instead of the T-dimensional numerical optimization typically used (§6.4.3); the concept of fixed-multiplier generation, its relationship to corpus-level generation, and its computational advantages (§6.4.6); the realization that the trajectory HMM matches GMSD, and in fact leaf-specific GMSD, on the training corpus (§6.5.1); the realization that GV generation suffers from a pathology (§6.6); the realization that, for the standard framework and trajectory HMM, negative modified static precision parameters can result in frames that are "at risk" of excursions (§6.7.1); experimental verification that most artifacts occur for frames with small or negative modified static precision parameters (§6.8.2); a GV-like generation method (LSPA GV generation) which improves naturalness and reduces artifacts like conventional GV generation does, but which is more amenable to being used with the partially analytic generation algorithms (§6.7.2 and §6.8.2); the notion of a z-value as a convenient way of measuring excursions (§6.7); the discovery that excursions in different cepstral components often occur simultaneously (§6.7) and a plausible explanation as to why this occurs (§6.9.2); the discovery that, for normalized models, sampling generation results in appropriate levels of spread without introducing excursions (§6.5.2 and §6.8.2); experimental investigation of a previously suggested hypothesis on why artifacts occur (§6.8.2); and a partial explanation of why early stopping reduces excursions (§6.9.1).

Chapter 7

Conclusion

In this thesis we have investigated various aspects of the challenge of specifying and using probabilistic acoustic models for parametric speech synthesis. We have proposed the LGLAR HMM as a consistent yet highly tractable alternative to the standard HMM synthesis framework and the trajectory HMM. We have proposed fixed-multiplier LSPA GMSD generation as a faster alternative to parameter generation considering global variance. We have also provided insight into various facets of existing and new models and generation methods.

The LGLAR HMM addresses the inconsistency in the treatment of the dynamics of speech parameter sequences present in the standard framework. Compared to the trajectory HMM, it has the advantage of supporting efficient parameter estimation using expectation maximization and decision tree clustering. Ultimately this tractability is due to the fact that the linear regression model at the heart of the LGLAR HMM is a conditionally additive exponential family. The LGLAR HMM supports existing effective speech parameter generation methods such as parameter generation considering global variance, as well as supporting a simple and exact low latency parameter generation algorithm not available for the standard framework or the trajectory HMM. Experimentally we have investigated the best way to set model structure parameters such as depth for the LGLAR HMM and made recommendations for how to set the model structure parameters to achieve good TSLP, MCD and naturalness. We found that a moderate amount of overfitting improved MCD scores and naturalness. In a subjective evaluation we found that the LGLAR HMM is capable of producing speech that is as natural as that of the standard framework with its conventional settings, but not as natural as the trajectory HMM.

Fixed-multiplier LSPA GMSD generation is the culmination of two separate threads of investigation. Firstly it builds on a mathematical analysis of GMSD and GV generation, showing that fixed-multiplier GMSD generation is a very fast approximation to corpus-level expected-spread GMSD generation. Secondly it builds on an investigation into why GV

generation sometimes introduces artifacts, showing that LSPA is an effective way to prevent their occurrence. The combination, fixed-multiplier LSPA GMSD generation, is as fast as standard speech parameter generation but improves naturalness as much as parameter generation considering global variance, without introducing artifacts.

We have also investigated and hopefully developed a better understanding of several aspects of existing and new models. We have seen that the standard framework, due its lack of consistency, greatly underestimates predictive variance. We have observed that the trajectory HMM acoustic model is a conditional exponential family, implying that the log likelihood function is concave in a particular parameterization and has no non-global local maxima in any parameterization, and that both sampled trajectories and the mean trajectory match certain leaf-specific statistics. We have identified future state blindness as a weakness in the autoregressive HMM and investigated ways in which the LGLAR HMM might be compensating for this weakness. We have described two ways to view the trajectory HMM and the LGLAR HMM within a common framework, including a view of the trajectory HMM as a directed graphical model which shows explicitly how the trajectory HMM avoids suffering from future state blindness by passing information about future states backwards in time. We have shown that GMSD and GV generation are equivalent to standard generation on a modified model. We have shown theoretically and experimentally that consistent models such as the trajectory HMM and autoregressive HMM model global variance very well. This implies that, for consistent models, parameter generation considering global variance is perhaps best viewed as compensating for a deficiency in standard generation, not a deficiency in the model. Finally we have investigated the causes of artifacts in some detail. We have shown that GV generation suffers from a pathology which can result in excursions in the generated trajectory, and found that excursions in different cepstral components often occur simultaneously. We have provided evidence that these excursions are the cepstral-level phenomenon responsible for the perceptual-level phenomenon of artifacts. We have also provided a partial explanation of why early stopping helps to prevent artifacts, and shown that a previously suggested hypothesis about the causes of artifacts has only a small effect for our experimental systems. We have presented our hypothesis about the causes of artifacts, suggesting that they are due to a pathology in GV generation exacerbated by a deficiency in the decision tree clustering process.

Appendix A

Proof of the variance boost bound

We mentioned in §5.4.1 that the optimal variance boost for a standard HMM synthesis framework system trained on a fixed alignment is always less than or equal to the number of windows used for that system. Here we give a proof of this fact. We assume a fixed state sequence θ throughout.

First we review notation. For each vector component of the speech parameters the probability distribution used by the trajectory HMM is a Gaussian with b-value b and precision matrix P given by (3.39) and (3.40), where these values depend on the state sequence θ and model parameters λ. The mean trajectory μ satisfies $b = P\mu$. Applying a collection of D windows to trajectory $c = [c_t]_{t=1}^T$ results in a trajectory tuple. As discussed in §3.2.2 we may view a trajectory tuple as a (DT)-dimensional vector o, rather than a $T \times D$ matrix as we have in the majority of the thesis. The process of computing o from c is linear, so $o = Wc$ for a $DT \times T$ matrix W. In terms of these quantities P and b may be expressed as

$$P = W^\mathsf{T}\tilde{P}W \tag{A.1}$$

$$b = W^\mathsf{T}\tilde{P}\tilde{\mu} \tag{A.2}$$

where $\tilde{\mu}$ is the result of flattening a $T \times D$ matrix of mean parameters to obtain a vector of length DT, and $\tilde{P}$ is a diagonal matrix whose diagonal is the result of flattening a $T \times D$ matrix of precision parameters to obtain a vector of length DT (Tokuda et al., 2000). The above formulae, rather than (3.39) and (3.40), are in fact the standard way to describe the standard speech parameter generation method and the trajectory HMM in the literature (see for example (Zen et al., 2009)).

We saw in §5.4.1 that amongst the family of variance boost transformations $(b, P) \mapsto (kb, kP)$ for $k > 0$, the optimal k is given by

$$\frac{1}{\hat{k}} = \frac{1}{T}(c - \mu)^\mathsf{T}P(c - \mu) \tag{A.3}$$

This is optimal in the sense of maximizing the log probability of the natural trajectory c. Our claim is that for a trained standard HMM synthesis framework system $1/\hat{k} \leq D$, that is

$$(c-\mu)^{\mathsf{T}} P (c-\mu) \leq DT \tag{A.4}$$

Now (A.3) applies to any Gaussian distribution, and in particular to the Gaussian acoustic model used by the standard HMM synthesis framework during training. Thus the optimal variance boost for this model is given by

$$\frac{1}{DT}(o-\tilde{\mu})^{\mathsf{T}} \tilde{P} (o-\tilde{\mu}) \tag{A.5}$$

where $o = Wc$ for the natural trajectory c. However if the model has been trained using o as its training data (where we are following the convention of only considering a single training example for sequential models as discussed in §2.5.1) then the optimal variance boost for this trajectory must be 1, since otherwise we could obtain a better log likelihood by using the variance-boosted model. This variance-boosted model, say with variance boost $1/k$, is an instance of the Gaussian acoustic model defined by (3.10) since it may be obtained by multiplying all b-value and precision parameters in the model by k. Therefore for a trained Gaussian acoustic model we have

$$\begin{aligned} DT &= (o-\tilde{\mu})^{\mathsf{T}} \tilde{P} (o-\tilde{\mu}) && \text{(A.6)} \\ &= (Wc-\tilde{\mu})^{\mathsf{T}} \tilde{P} (Wc-\tilde{\mu}) && \text{(A.7)} \\ &= c^{\mathsf{T}} P c - 2b^{\mathsf{T}} c + \tilde{\mu}^{\mathsf{T}} \tilde{P} \tilde{\mu} && \text{(A.8)} \\ &= (c-\mu)^{\mathsf{T}} P (c-\mu) + \tilde{\mu}^{\mathsf{T}} \tilde{P} \tilde{\mu} - \mu^{\mathsf{T}} P \mu && \text{(A.9)} \end{aligned}$$

Thus our claim is equivalent to $\tilde{\mu}^{\mathsf{T}} \tilde{P} \tilde{\mu} - \mu^{\mathsf{T}} P \mu \geq 0$.

As discussed in §3.2.2 the set of all realizable o forms a T-dimensional subspace of the (DT)-dimensional vector space of all possible o. This realizable subspace is the image of W. Thinking geometrically, $\tilde{P}$ defines an inner product on $\mathbb{R}^{DT}$, so we may consider the orthogonal projection operator Q on to the realizable subspace with respect to this inner product. The explicit formula for Q is

$$Q = W(W^{\mathsf{T}} \tilde{P} W)^{-1} W^{\mathsf{T}} \tilde{P} \tag{A.10}$$

It it easy to verify that $Q^2 = Q$ and that the image of Q is equal to the image of W, so Q is a projection on to the realizable subspace. To verify that it is orthogonal with respect to the inner product defined by $\tilde{P}$ requires checking that $Q^{\mathsf{T}} \tilde{P} = \tilde{P} Q$. Geometrically Q takes a trajectory tuple o and computes the nearest realizable trajectory tuple, where "nearest" means with respect to the Mahalanobis distance defined by $\tilde{P}$.

It is interesting to note that the standard parameter generation method may be expressed very simply in this geometric view: it finds the member $Q\tilde{\mu}$ of the realizable subspace which is closest to the vector $\tilde{\mu}$ of mean parameters. This can be seen by noting that $Q\tilde{\mu} = W\mu$, where $W\mu$ is the result of taking the mean trajectory μ and embedding it in o-space by computing the corresponding trajectory tuple. The distance between the vector $\tilde{\mu}$ of mean parameters and its realizable projection $W\mu$ is a measure of how much the standard speech parameter generation algorithm has to compromise when trading-off the conflicting soft constraints imposed by the different windows.

Returning to our proof, we have that

$$\tilde{\mu}^{\mathsf{T}}\tilde{P}\tilde{\mu} = \|\tilde{\mu}\|^2_{\tilde{P}} \tag{A.11}$$

where $\|\cdot\|_{\tilde{P}}$ is the norm defined by $\tilde{P}$, and since $W\mu = Q\tilde{\mu}$ we have

$$\mu^{\mathsf{T}}P\mu = \tilde{\mu}^{\mathsf{T}}Q^{\mathsf{T}}\tilde{P}Q\tilde{\mu} = \|Q\tilde{\mu}\|^2_{\tilde{P}} \tag{A.12}$$

Since Q is an orthogonal projection we have $\|Q\tilde{\mu}\|^2_{\tilde{P}} \leq \|\tilde{\mu}\|^2_{\tilde{P}}$. This establishes our claim.

We can go slightly further and obtain a limited amount of insight into when the optimal variance boost is likely to be close to D. The slack

$$\|\tilde{\mu}\|^2_{\tilde{P}} - \|Q\tilde{\mu}\|^2_{\tilde{P}} \tag{A.13}$$

in the inequality (A.4) is precisely the squared distance that $\tilde{\mu}$ moves when projected on to the realizable subspace. If we normalize this per frame by dividing by T then we obtain the amount $D - 1/\hat{k}$ by which the optimal variance boost $1/\hat{k}$ differs from the number of windows D. Thus the amount by which the optimal variance boost is less than D depends on how far the vector of trained mean parameters is from the realizable subspace. This distance is also the quantity minimized by the standard speech parameter generation method, and so the optimal variance boost will be close to D when the standard parameter generation method (operating on the training corpus) does not have to compromise too greatly when balancing the conflicting soft constraints imposed by the different windows.

Appendix B

End effects for trajectory-level acoustic models

For many sequential probabilistic models, care is needed to define precisely what happens at the beginning and end of a sequence. We use *end effects* as a catch-all term for this sort of consideration. The issue of how to cope with end effects for trajectory-level acoustic models is a subtle one, and it is hard to find a completely satisfying solution. In this appendix we briefly discuss the issue and mention a number of possible solutions for the trajectory HMM acoustic model and the LGLAR acoustic model.

That end effects are an issue for the trajectory HMM acoustic model and LGLAR acoustic model can be seen in their definitions. In §3.2.2 we saw that there is an issue with how to choose the window coefficients to use while computing the first few and last few frames of dynamic trajectories, i.e. how to choose the first few and last few rows of the window matrix. The trajectory HMM acoustic model defined in §3.3.2 uses this window matrix to specify the b-value b and precision matrix P. For the LGLAR acoustic model, $\mathbb{P}(c \,|\, \theta, \lambda)$ is mostly specified by $\mathbb{P}(c_t \,|\, c_{t-K:t-1}, \theta_t, \lambda)$ in (4.6), but to fully define the model it is necessary to specify the initial acoustic context $c_{-(K-1):0}$. In fact, as we outline below, for trajectory-level acoustic models with banded precision matrices the end effect issue centres on how to choose the top-left and bottom-right corners of the precision matrix P and the first few and last few elements of the b-value vector b. In terms of the decomposition into local contributions presented in §5.3.1, this corresponds to how to choose the first few and last few local contributions to the precision matrix and b-value vector. Note that the influence of the first and last few frames of b and P can in principle extend over much longer timescales than just a few frames. For example, for a degenerate case of the trajectory HMM acoustic model where the delta expert at every time says that the derivative should be exactly zero, end effects determine the entire trajectory.

The following are some potential solutions to coping with the issue of end effects for the

trajectory HMM acoustic model:

1. Ignore, i.e. evaluate as a constant value 1, any experts which we cannot evaluate properly. This corresponds to using zero-output windows.

2. Condition on the trajectory values before the start and after the end of the observed values being zero. This corresponds to using zero-input windows.

3. Condition on the trajectory values before c_1 being some fixed value, not necessarily zero, and similarly for the trajectory values after c_T. In principle these fixed values could be treated as model parameters and learned.

4. Use alternative finite difference window coefficients. The conventional windows are motivated as finite difference approximations to the derivative and second derivative, so alternative finite difference approximations could be used for the first few and last few frames. For example the first frame could use a delta window of $(0.0, -1.0, 1.0)$ instead of $(-0.5, 0.0, 0.5)$. Note that further additional windows are needed if we wish to cope with the case of short utterances (those with $T < K$).

5. Assume that what we actually observe is a snapshot of a longer process. Extend the state sequence to start earlier than 1 and end later than T. One way to do this is to add "before initial" and "after end" states. Assume the trajectory extends similarly but that the trajectory values before c_1 and after c_T are not observed. Thus we marginalize over the unobserved trajectory values, but the extra states still have a (possibly weak) coupling to the observed values via these unobserved values.

6. Allow a special initial local contribution from time 1 to time K, and a special final local contribution from time $T - K$ to time K. Allow these contributions to be arbitrary, i.e. P^{LC} is an arbitrary positive semi-definite $K \times K$ matrix and b^{LC} is an arbitrary vector, both learned from data. The first few and last few states are now ignored. This adds $O(K^2)$ extra model parameters. Note that further special local contributions are needed if we wish to cope with the case of short utterances (those with $T < K$).

As a reminder $K = K^{\mathrm{L}} + K^{\mathrm{R}}$ is the sum of the left and right extents of a given window or collection of windows. The choice of solution above only affects the first and last K elements of b and the $K \times K$ submatrices in the top left and bottom right of P, as can be seen by examining the formula for conditioning and marginalization for Gaussian distributions (e.g. as given by Bishop (2006, chapter 2)). In this sense solution 6 may be viewed as the most general.

We default to using solution 2 due to its simplicity. In preliminary experiments with solutions 1, 2 and 3, we found only minor differences in TSLP and MCD performance. This may be partly because, for median alignments derived from system S, the first state in all utterances corresponded to a particular leaf which only ever appeared at the start of utterances and almost always only lasted one frame, so a restricted form of solution 6 was effectively already being used. The situation with the final state was not quite as clear cut.

For solution 5 the effect of the additional states before θ_1 on the observable trajectory is likely to diminish as we go further back in time, and similarly for the additional states after θ_T. This means that we may view the state sequence as essentially infinite.

For solutions 2 and 3, which condition on unobserved trajectory values, it is possible to view the state sequence as extended in a similar way to solution 5, and again we may conceptually consider the state sequence to be infinite. However now only the K^{R} extra states before θ_1 and the K^{L} extra states after θ_T have any effect. Viewed from this perspective, the trajectory HMM acoustic model defined in §3.3.2 appears to be missing some local contributions, and this situation is rectified by using the model defined in §5.2.

All of the above proposed solutions except 5 result in a probabilistic model that is a conditional exponential family. The model parameters b_{qd} and τ_{qd} for each leaf q and window d are natural parameters corresponding to feature functions f_{qd} and g_{qd}, and the difference between solutions 1, 2, 3 and 4 is just in the precise form of the feature functions f_{qd} and g_{qd}. For example f_{qd} and g_{qd} are given by (3.45) and (3.46) for solutions 1 and 2. The solution 6 effectively adds additional feature functions and corresponding model parameters but may still be viewed as a conditional exponential family.

As an alternative to the above proposed solutions for modelling end effects, we can instead condition on the actual values of the first few frames and last few frames, treating the prediction of these values as outside the scope of the model. This has the advantage that we can compare different models on how they perform for the majority of the trajectory (according to whatever is our chosen metric) without worrying that a difference in how well they cope with end effects will affect the comparison fundamentally. This has the disadvantage that in principle it is no longer possible to synthesize audio for a new utterance, and that in practice we probably do care about having a good model for the first few and last few frames. Note that the first few and last few states are now ignored.

Solutions 2 and 3, and the condition-on-the-actual-values solution presented in the previous paragraph, are also applicable to an autoregressive acoustic model. In this case the only issue is with initial end effects, not final end effects, since an autoregressive acoustic model effectively already marginalizes over all possible future unobserved trajectory values. Solution 5 is also applicable to the LGLAR acoustic model, but exact analytic parameter estimation is no longer possible, so an approach such as gradient ascent or variational Bayes would be required for parameter estimation.

Bibliography

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK users' guide*. Society for Industrial and Applied Mathematics, third edition, 1999. ISBN 0898714478.

E. W. Barankin and A. P. Maitra. Generalization of the Fisher-Darmois-Koopman-Pitman theorem on sufficient statistics. *Sankhya: The Indian Journal of Statistics, Series A*, 25 (3):217–244, 1963.

Y. Bengio and P. Frasconi. An input output HMM architecture. In *Proc. Advances in Neural Information Processing Systems*, pages 427–434, 1995.

C. L. Bennett and A. W. Black. The Blizzard Challenge 2006. In *Proc. Blizzard Challenge Workshop 2006*, 2006.

J. Bilmes. Graphical models and automatic speech recognition. In M. Johnson, S. P. Khudanpur, M. Ostendorf, and R. Rosenfeld, editors, *Mathematical foundations of speech and language processing*. Springer, 2004. ISBN 0387203265.

C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. ISBN 9780387310732.

A. W. Black and K. Tokuda. The Blizzard Challenge 2005: Evaluating corpus-based speech synthesis on common datasets. In *Proc. Interspeech 2005*, pages 77–80, 2005.

P. F. Brown. *The acoustic-modeling problem in automatic speech recognition*. PhD thesis, Carnegie Mellon University, USA, 1987.

K. K. Chin and P. C. Woodland. Maximum mutual information training of hidden Markov models with vector linear predictors. In *Proc. Interspeech 2002*, pages 997–1000, 2002.

R. A. J. Clark, M. Podsiadlo, M. Fraser, C. Mayo, and S. King. Statistical analysis of the Blizzard Challenge 2007 listening test results. In *Proc. Blizzard Challenge Workshop 2007*, 2007.

C. Courcoubetis and R. Weber. Appendix A: Lagrangian methods for constrained optimization. In *Pricing communication networks: economics, technology and modelling*. Wiley, 2003. ISBN 9780470851302.

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)*, pages 1–38, 1977.

R. E. Donovan. *Trainable speech synthesis*. PhD thesis, Department of Engineering, University of Cambridge, UK, 1996.

Y. Ephraim, D. Malah, and B.-H. Juang. On the application of hidden Markov models for enhancing noisy speech. *IEEE Trans. Acoust., Speech, Signal Process.*, 37(12):1846–1856, 1989.

R. L. Eubank and S. Wang. The equivalence between the Cholesky decomposition and the Kalman filter. *The American Statistician*, 56(1):39–43, 2002.

H. Everett III. Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, 11(3):399–417, 1963.

P. D. Feigin. Conditional exponential families and a representation theorem for asympotic inference. *The Annals of Statistics*, 9(3):597–603, 1981.

T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai. An adaptive algorithm for mel-cepstral analysis of speech. In *Proc. ICASSP 1992*, pages 137–140, 1992.

C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. STOC 2008 (ACM symposium on the theory of computing)*, pages 197–206, 2008.

S. M. Goldfeld and R. E. Quandt. A Markov model for switching regressions. *Journal of Econometrics*, 1:3–15, 1973.

Google. gperftools. https://code.google.com/p/gperftools/, 2014. Accessed 29 September 2014.

J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57(2):357–384, 1989.

W. Han, L. Wang, F. Soong, and B. Yuan. Improved minimum converted trajectory error training for real-time speech-to-lips conversion. In *Proc. ICASSP 2012*, pages 4513–4516, 2012.

K. Hashimoto, H. Zen, Y. Nankaku, T. Masuko, and K. Tokuda. A Bayesian approach to HMM-based speech synthesis. In *Proc. ICASSP 2009*, pages 4029–4032, 2009.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

HTS working group. HMM-based speech synthesis system (HTS). http://hts.sp.nitech.ac.jp/, 2012. Accessed 21 March 2012.

HTS working group. HTS speaker dependent training demo (English). http://hts.sp.nitech.ac.jp/?Download, 2013. Accessed 12 June 2013.

HTS working group. hts_engine API. http://hts-engine.sourceforge.net/, 2014. Accessed 29 September 2014.

M. F. Hutchinson and F. R. de Hoog. Smoothing noisy data with spline functions. *Numerische Mathematik*, 47(1):99–106, 1985.

B. H. Juang and L. Rabiner. Mixture autoregressive hidden Markov models for speech signals. *IEEE Trans. Acoust., Speech, Signal Process.*, 33(6):1404–1413, 1985.

H. Kawahara, I. Masuda-Katsuse, and A. de Cheveign. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech Communication*, 27:187–207, 1999.

P. Kenny, M. Lennig, and P. Mermelstein. A linear predictive HMM for vector-valued observations with applications to speech recognition. *IEEE Trans. Acoust., Speech, Signal Process.*, 38(2):220–225, 1990.

S. King. An introduction to statistical parametric speech synthesis. *Sadhana*, 36(5):837–852, 2011.

K. Koishida, K. Tokuda, T. Masuko, and T. Kobayashi. Vector quantization of speech spectral parameters using statistics of static and dynamic features. *IEICE Trans. Inf. Syst.*, E84-D(10):1427–1434, 2001.

J. Kominek and A. W. Black. The CMU ARCTIC databases for speech synthesis. Technical Report CMU-LTI-03-177, Carnegie Mellon University, USA, 2003.

R. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proc. IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, pages 125–128, 1993.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. ICML 2001*, pages 282–289, 2001.

G. Lindgren. Markov regime models for mixed distributions and switching regressions. Technical Report 1976-11, Institute of Mathematics and Statistics, University of Umeå, Sweden, 1976. http://www.maths.lth.se/matstat/staff/georg/Publications/MarkovMixture_text.pdf.

G. Lindgren. Markov regime models for mixed distributions and switching regressions. *Scandinavian Journal of Statistics*, 5:81–91, 1978.

Z.-H. Ling, Y.-J. Wu, Y.-P. Wang, L. Qin, and R.-H. Wang. USTC system for Blizzard Challenge 2006 an improved HMM-based speech synthesis method. In *Proc. Blizzard Challenge Workshop 2006*, 2006.

D. J. C. MacKay. *Information theory, inference, and learning algorithms.* Cambridge University Press, 2003. ISBN 9780521642989. Available from http://www.inference.phy.cam.ac.uk/mackay/itila/.

R. Maia, H. Zen, and M. J. F. Gales. Statistical parametric speech synthesis with joint estimation of acoustic and excitation model parameters. In *Seventh ISCA Tutorial and Research Workshop on Speech Synthesis (SSW 7)*, pages 88–93, 2010.

B. A. Maxwell and P. C. Woodland. Hidden Markov models using shared and global vector linear predictors. In *Proc. Eurospeech 1993*, pages 819–822, 1993.

A. McCallum, D. Freitag, and F. Pereira. Maximum entropy Markov models for information extraction and segmentation. In *ICML 2000*, pages 591–598, 2000.

T. Muramatsu, Y. Ohtani, T. Toda, H. Saruwatari, and K. Shikano. Low-delay voice conversion based on maximum likelihood estimation of spectral parameter trajectory. In *Proc. Interspeech 2008*, pages 1076–1079, 2008.

K. Nakamura, K. Hashimoto, Y. Nankaku, and K. Tokuda. Integration of acoustic modeling and mel-cepstral analysis for HMM-based speech synthesis. In *Proc. ICASSP 2013*, pages 7883–7887, 2013.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in graphical models*, pages 355–368. Kluwer Academic Publishers, 1998. ISBN 0262600323.

J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006. ISBN 9780387400655.

A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*, page 15. Prentice Hall, first edition, 1975. ISBN 0132146355.

A. Poritz. Linear predictive hidden Markov models and the speech signal. In *Proc. ICASSP 1982*, volume 7, pages 1291–1294, 1982.

C. Quillen. Autoregressive HMM speech synthesis. In *Proc. ICASSP 2012*, pages 4021–4024, 2012.

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

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

S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11:305–345, 1999.

M. S. Seigel, P. C. Woodland, and M. J. F. Gales. A confidence-based approach for improving keyword hypothesis scores. In *Proc. ICASSP 2013*, pages 8565–8569, 2013.

M. Shannon and W. Byrne. Autoregressive HMMs for speech synthesis. In *Proc. Interspeech 2009*, pages 400–403, 2009a.

M. Shannon and W. Byrne. A formulation of the autoregressive HMM for speech synthesis. Technical Report CUED/F-INFENG/TR.629, Department of Engineering, University of Cambridge, UK, 2009b. http://mi.eng.cam.ac.uk/~sms46/papers/shannon2009fah.pdf.

M. Shannon and W. Byrne. Autoregressive clustering for HMM speech synthesis. In *Proc. Interspeech 2010*, pages 829–832, 2010.

M. Shannon and W. Byrne. Viewing the trajectory HMM as a generalized autoregressive HMM. Technical Report CUED/F-INFENG/TR.677, Department of Engineering, University of Cambridge, UK, 2012. http://mi.eng.cam.ac.uk/~sms46/papers/shannon2012viewing.pdf.

M. Shannon and W. Byrne. Fast, low-artifact speech synthesis considering global variance. In *Proc. ICASSP 2013*, pages 7869–7873, 2013a.

M. Shannon and W. Byrne. Partially analytic speech parameter generation considering global variance. Technical Report CUED/F-INFENG/TR.682, Department of Engineering, University of Cambridge, UK, 2013b. http://mi.eng.cam.ac.uk/~sms46/papers/shannon2013partially.pdf.

M. Shannon, H. Zen, and W. Byrne. The effect of using normalized models in statistical speech synthesis. In *Proc. Interspeech 2011*, pages 121–124, 2011.

M. Shannon, H. Zen, and W. Byrne. Autoregressive models for statistical parametric speech synthesis. *IEEE Trans. Audio Speech Language Process.*, 21(3):587–597, 2013.

K. Shinoda and T. Watanabe. MDL-based context-dependent subword modeling for speech recognition. *J. Acoust. Soc. Jpn. (E)*, 21(2):79–86, 2000.

H. Silén, E. Helander, J. Nurminen, and M. Gabbouj. Ways to implement global variance in statistical speech synthesis. In *Proc. Interspeech 2012*, 2012.

K. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirschberg. ToBI: a standard for labeling English prosody. In *Proc. ICSLP 1992*, pages 867–870, 1992.

P. Taylor. *Text-to-speech synthesis*. Cambridge University Press, 2009. ISBN 9780521899277.

T. Toda. Modeling of speech parameter sequence considering global variance for HMM-based speech synthesis. In P. Dymarski, editor, *Hidden Markov models, theory and applications*. InTech, 2011. ISBN 9789533072081.

T. Toda and K. Tokuda. A speech parameter generation algorithm considering global variance for HMM-based speech synthesis. *IEICE Trans. Inf. Syst.*, E90-D(5):816–824, 2007.

T. Toda and S. Young. Trajectory training considering global variance for HMM-based speech synthesis. In *Proc. ICASSP 2009*, pages 4025–4028, 2009.

K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proc. ICASSP 2000*, pages 1315–1318, 2000.

K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi. Multi-space probability distribution HMM. *IEICE Trans. Inf. Syst.*, E85-D(3):455–464, 2002a.

K. Tokuda, H. Zen, and A. W. Black. An HMM-based speech synthesis system applied to English. In *Proc. IEEE Workshop on Speech Synthesis*, pages 227–230, 2002b.

K. S. van Horn. Rethinking derived acoustic features in speech recognition. In *Proc. Interspeech 2002*, pages 1009–1012, 2002.

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

P. Whittle. *Optimization under constraints: theory and applications of nonlinear programming*. Wiley, 1971. ISBN 0471941301.

P. C. Woodland. Hidden Markov models using vector linear prediction and discriminative output distributions. In *Proc. ICASSP 1992*, pages 509–512, 1992.

Y.-J. Wu, H. Zen, Y. Nankaku, and K. Tokuda. Minimum generation error criterion considering global/local variance for HMM-based speech synthesis. In *Proc. ICASSP 2008*, pages 4621–4624, 2008.

J. Yamagishi, H. Zen, Y.-J. Wu, T. Toda, and K. Tokuda. The HTS-2008 system: yet another evaluation of the speaker-adaptive HMM-based speech synthesis system in the 2008 Blizzard Challenge. In *Proc. Blizzard Challenge Workshop 2008*, 2008.

T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Duration modeling for HMM-based speech synthesis. In *Proc. ICSLP 1998*, 1998.

T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Incorporation of mixed excitation model and postfilter into HMM-based text-to-speech synthesis. *IEICE Trans. Inf. Syst. (Japanese edition)*, J87-D-II(8):1565–1571, 2004.

T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Incorporating a mixed excitation model and postfilter into HMM-based text-to-speech synthesis. *Systems and Computers in Japan*, 36(12):43–50, 2005.

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

S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. J. Odell, D. G. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK book (for HTK version 3.4)*. Cambridge University Engineering Department, 2006.

K. Yu. Continuous F0 modeling for HMM based statistical parametric speech synthesis. *IEEE Trans. Audio Speech Language Process.*, 19(5):1071–1079, 2011.

H. Zen. Implementing an HSMM-based speech synthesis system using an efficient forward-backward algorithm. Technical Report TR-SP-0001, Nagoya Institute of Technology, 2007.

H. Zen, T. Toda, and K. Tokuda. The Nitech-NAIST HMM-based speech synthesis system for the Blizzard Challenge 2006. In *Proc. Blizzard Challenge Workshop 2006*, 2006.

H. Zen, T. Toda, M. Nakamura, and K. Tokuda. Details of the Nitech HMM-based speech synthesis system for the Blizzard Challenge 2005. *IEICE Trans. Inf. Syst.*, E90-D(1): 325–333, 2007a.

H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Computer Speech and Language*, 21(1):153–173, 2007b.

H. Zen, K. Tokuda, T. Masuko, T. Kobayasih, and T. Kitamura. A hidden semi-Markov model-based speech synthesis system. *IEICE Trans. Inf. Syst.*, E90-D(5):825–834, 2007c.

H. Zen, K. Tokuda, and A. W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039–1064, 2009.

H. Zen, M. J. F. Gales, Y. Nankaku, and K. Tokuda. Statistical parametric speech synthesis based on product of experts. In *Proc. ICASSP 2010*, pages 4242–4245, 2010.

H. Zen, M. J. F. Gales, Y. Nankaku, and K. Tokuda. Product of experts for statistical parametric speech synthesis. *IEEE Trans. Audio Speech Language Process.*, 20(3):794–805, 2012.

L. Zhang. *Modelling speech dynamics with trajectory HMMs.* PhD thesis, University of Edinburgh, UK, 2009.