Speech Synthesis Using Linear Dynamical Models

by

Gagandeep Singh

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Speech Synthesis Using Linear Dynamical Models

Gagandeep Singh

Master of Science

Graduate Department of Computer Science

University of Toronto

2017

In this work, we use Linear Dynamical Models (LDMs) for speech synthesis. We build-up on the existing work on this topic in the literature. We introduce "second order" LDMs, in which the current state of the LDM factors through the previous two state vectors. We derive the parameter estimation equations for second order LDMs using expectation maximization (EM). Second order LDMs, with some constraints, reduce the number of parameters in the model while not affecting the quality of the speech generated. We also investigate the subphone segmentation in an LDM-based TTS. A series of experiments were conducted comparing LDM-based speech synthesis with neural speech synthesis. Although, the quality of speech generated by LDMs is inferior to the one generated by neural networks, the memory and computational resources required are less in case of LDM-based synthesis.

Acknowledgements

Firstly, utmost thanks to my supervisor Prof. Gerald Penn for his invaluable guidance, support and feedback throughout, without which this work would not have been possible. I am sincerely thankful to Dr. Vassilis Tsiaras for the insightful discussions on the subject and providing some of his code. I want to thank Prof. Frank Rudzicz for agreeing to be the second reader of the paper and providing the valuable feedback.

I want to thank all the members of my research group, especially my office-mates for providing such a stimulating and motivating environment to work in. Thanks to Sean Robertson for the detailed and meaningful discussions, and helping me sort out many technical issues. Thanks to Arvie Frydenlund for sharing his knowledge about neural networks that helped with the baselining. Thanks to Chloé Pou-Prom and Muuo Wambua for helpful discussions on research and coursework. Also, thanks to Chloé for co-organizing the CL Friday lunches. Thanks to Patricia Thaine and Siavash Kazemian for the group meeting discussions and feedback.

I thank my family and all my friends for the constant moral support throughout my masters.

Contents

Chapter 1

Introduction

Text-to-speech synthesis or 'TTS' is the generation of artificial speech using a computer. Artificially generated speech has a variety of applications which include a voice for personal assistants, document readers, interactive voice response systems (IVRS), assistive technology devices, etc. There has been a gradual improvement in the quality of TTS systems over the last few decades. The factors responsible for this progress include availability of more powerful computational resources, the availability of more speech data and development of better mathematical models. Two broad goals for a TTS voice are intelligibility and naturalness. Intelligibility refers to the ability to convey the original message of the speaker with clarity to a listener. Naturalness refers to how close the voice characteristics are to a human voice. These characteristics include the continuity and flow in the utterance, the overall tone of the speech, nuances induced by accent, etc. An intelligible, yet unnatural voice will sound 'robotic'. Present day TTS systems perform well in terms of intelligibility, so the focus is more on producing a more natural sounding speech.

A TTS system, in general, can be divided into two major components: text analysis, and generation and synthesis. These components are described as follows:

- **Text analysis** : Text analysis component, also known as the front-end of a TTS system, is used for processing the text to be synthesized and convert it into linguistic features or labels. The tasks delegated to a front-end include
  - part-of-speech (POS) tagging and syntactic analysis
  - word segmentation, which is non-trivial for languages, for example Mandarin Chinese, in which words are not segmented using white space

  - text normalization, which involves converting the abbreviations, digits and quantities into their normalized form
  - prosody prediction
  - finding word pronunciations, i.e., converting each word into a string of phones along with other tags for example stress accent, POS, etc.
  - generating linguistic feature vectors, which include positional, syntactic information etc., from these phonetic transcriptions

  Not all processing steps are required in each TTS system. Some of the new deep-neural networks based systems use normalized text as input to the acoustic model, for example *Tacotron* [Wang et al., 2017].

- **Generation & synthesis** : This component, also known as the back-end of a TTS system, is responsible for generating the speech waveform from the linguistic features produced by the text analysis. There are various types of back-end designs, which differ significantly in their approach towards generating speech. Data-driven approaches to generation and synthesis can be broadly classified into two categories: concatenative speech synthesis and statistical parametric speech synthesis (SPSS).

1.1 Concatenative Speech Synthesis

As the name suggests, concatenative speech synthesis is based on concatenating together small units of speech to produce an utterance. The concatenation is typically done using a time-domain joining algorithm, for example PSOLA [Charpentier and Stella, 1986]. The concatenated utterance is smoothened by signal processing algorithms to remove the glitches at the discontinuities. These glitches could be mismatch in spectral tilt, phase, and formant frequencies and bandwidths. The resulting speech can be very natural sounding because the speech signal is directly taken from instances of the training data and there is minimal signal processing involved. The concatenative speech synthesis can further be classified into diphone synthesis and unit selection synthesis.

1.1.1 Diphone Synthesis

In diphone synthesis [Charpentier and Stella, 1986] the fundamental speech unit is typically two halves of adjacent phones taken together, called a diphone. Each di-phone corresponds to the later half of one phone and the first half of a second phone. Thus a diphone captures the transition between two phones. The advantage of using a diphone as opposed to a phone, as a fundamental unit for concatenation, is that the middle of a phone is much more stable than the boundary of a phone. It is simpler to put a join in the middle as it is much more forgiving in case there is a small error in joining the two sounds. The number of units in diphone synthesis is dependent on the phonotactics of the language, but the amount of data required is much lesser than all other data-driven approaches. As a result, the resulting speech is not of great quality but it can be easily implemented on devices with low computational and memory resources.

1.1.2 Unit Selection Synthesis

In unit selection synthesis [Hunt and Black, 1996], speech is generated from a large corpus of pre-recorded speech units by selecting the appropriate units out of this database of units. There may not be just one length of speech units. During the speech database creation, the speech utterances are broken down into a variety of units which include phones, diphones, a cluster of phones, syllables, phrases, etc. The smaller units, for example phones, are more generalizable and can thus be used in a variety of contexts, whereas the larger units, for example phrases, are specialized and thus require a larger amount of speech to be observable. The use of larger units, however, leads to better speech because such an utterance involves lesser number of concatenations, so a lesser number of artifacts at the joins. The selection of a unit in an utterance to be synthesized is dependent on a combination of two cost functions: target cost and join cost. Target cost defines if a unit is a good fit for the context under consideration. The context could be defined by a variety of features which include the previous and next phones, position in the syllable, position in the word, position in the phrase, etc. The join cost defines the joining compatibility between two speech units. This is dependent on spectral distance, fundamental frequency distance, duration and intensity mismatch between the two units, etc. After shortlisting the candidate units for concatenation, the final selection is done using a Viterbi style algorithm.

The speech produced by unit selection has been in general better than the one produced by SPSS [Zen et al., 2009], even though recently there have been SPSS systems that perform better than state-of-the-art unit selection systems [van den Oord et al., 2016]. Unit selection requires a huge speech database and is difficult to adapt to various styles and voices. Thus a new voice or style requires an entirely new dataset.

1.2 Statistical Parametric Speech Synthesis (SPSS)

In statistical parametric speech synthesis [Zen et al., 2009], the speech is generated from statistical models rather than from stored exemplars. These statistical models essentially capture the probability distribution of the acoustic features, given the linguistic context. Most of the time, the output of an acoustic model is of much lower dimension than the raw speech. The output features are used by a vocoder to generate the speech waveform. The features depend on the vocoder used but are most often the spectral envelope, fundamental frequency, and band-aperiodicity coefficients. Instead of modeling the raw spectral envelope, the acoustic model often models some other lower dimensional features, for example mel-generalized coefficients (MGCs), from which the spectral envelope could be constructed. Hidden Markov Model (HMM) based speech synthesis (HTS) has historically been the most popular SPSS model. Apart from that, there has been some work in autoregressive HMMs for speech synthesis [Shannon and Byrne, 2009, Quillen, 2012] and Linear Dynamical Model (LDM) based speech synthesis [Quillen, 2010, Tsiaras et al., 2014]. In this thesis, we explore in detail the latter two methods and their combination. Recently there has been a lot of progress in speech synthesis using deep-neural networks, often referred to as neural speech synthesis. Neural speech synthesis has been around for more than two decades [Karaali et al., 1996] but it has become popular after the advancements in the field of deep-learning. Neural speech synthesis started as yet another approach to SPSS which uses neural networks for modeling various components within a TTS system, for example a neural network is used to map linguistic features to acoustic features [Karaali et al., 1996, Ze et al., 2013]. Recently, however, there have been neural speech synthesis models which are a little different from the traditional SPSS, for example in *Wavenet* [van den Oord et al., 2016], instead of generating acoustic features, the model synthesizes raw speech waveform. In *Char2wav* [Sotelo et al., 2017], the front-end and back-end are trained jointly.

Neural networks are more powerful still than LDMs in modeling complex probability distributions and are now being extensively used in TTS. Neural networks regained popularity just around the time when LDMs were first introduced for TTS. LDMs, as a result, did not catch on and have not been studied extensively in the realm of TTS, which is one of the reasons to study LDMs in this thesis. Neural networks have more computational resources and memory requirements, thus LDMs could be a better choice in low resource devices. LDMs also use their statistics more efficiently, so they are more suited to applications with a limited amount of training data.

Before going into LDMs and autoregressive models, we describe the HMM based models in detail, in order to motivate the use of LDMs.

1.3 HMM Based Speech Synthesis

The motivation for using HMMs in TTS comes from their success in automatic speech recognition (ASR) [Gales and Young, 2008]. HMMs are stable models and have efficient well-developed algorithms [Rabiner, 1989]. They can be easily trained using expectation maximization, the most likely state sequence can be efficiently found using the Viterbi algorithm, etc. In TTS, each context-dependent phone is associated with an HMM, which is typically a three or five state left-to-right HMM. In a three-state HMM, the first state, second and third states correspond to the beginning, middle and end of the phone respectively.

The number of context-dependent phones in a language is very high because the number of phones increases exponentially with the number of features that define the context. Typically the context is defined by the one or two preceding and succeeding phones, POS tag, position in syllable, position in phrase etc. Thus it is nearly impossible to have a separate HMM for each context-dependent phone because there will not be enough training data for each context-dependent phone and some of the contexts may never be encountered in the training data. Thus a many-to-one mapping is defined from linguistic context to HMMs. Historically, the most popular method to define such a mapping is by using a decision tree to cluster together various contexts and having a single HMM for each given cluster [Young et al., 1994]. In the very early approaches to using neural networks for speech synthesis, the decision trees were replaced by neural networks which map the linguistic context to an HMM [Ze et al., 2013].

In regular HMMs, the probability distribution of the duration of each state in an HMM is exponentially distributed, which is not an accurate duration model. Thus in order to model the duration correctly, HMMs in TTS are modified to Hidden Semi Markov Models (HSMMs) [Zen et al., 2007c] which have an explicit duration component, thus they are only semi-Markov, as the state transition times are marked in the training data. However, as is the case in most of the literature, we refer to HSMMs simply as HMMs.

During utterance generation, the HMMs corresponding to each context-dependent phone in sequence are concatenated together, and observations are generated for each state of this concatenated HMM using the external duration model associated with the HMM. In the traditional HMM based TTS systems, HMMs did not really play any major role. During training time, the HMMs are used only for aligning the phones with the speech audio to generate time labels for phones in each utterance of the training data. This process is called forced-alignment. The mapping from linguistic to acoustic features is done by decision trees. During generation, the duration is modeled by an external duration model.

There are however certain issues in the above-described model. Firstly, the observation vectors are independent given the state sequence and do not directly depend on previous observation vectors. Secondly, in a given state, the most probable observation vector is the mean of a Gaussian distribution, which is the output probability for that state. Thus the parameters generated by HMMs are piece-wise constant as they remain constant in any given state. A very effective solution to this problem is the use of dynamic features (deltas and double deltas) of the observation vectors along with “static” features [Tokuda et al., 1995]. However, this is based on the assumption that the static and dynamic features in the observation vector are independent, whereas they are clearly dependent. This was addressed by the trajectory HMM [Zen et al., 2007b] which incorporates the dynamic features in the training as well, so the static and dynamic features are no longer assumed to be independent. While using dynamic features, during generation time the parameters are generated using the maximum likelihood parameter generation (MLPG) algorithm [Tokuda et al., 2000].

Unlike HMMs, LDMs have a continuous state space. Due to their inherent properties, autoregressive HMMs and LDMs do not suffer from the piece-wise constant output problem. They are more powerful models for capturing the dynamics of speech. In the next chapter, the auto-regressive HMMs and LDMs are described in detailed. Chapter 3 describes the use of these models in speech synthesis.

In this work we reappraise LDMs in view of recent developments in TTS. The novel contributions include the introduction of second order LDMs along with their parameter estimation and usage in speech synthesis. Other aspects of the LDM based speech synthesis, which include subphone segmentation and parameter sharing, are also investigated. Several experiments compare the different variations of LDMs with neural networks for speech synthesis.

Chapter 2

Linear Dynamical Models

In this chapter, we discuss the linear dynamical models, which are a class of dynamical models with a continuous state space. The state evolution in LDMs is a linear first-order Gauss-Markov random process, and the observed vectors are produced by a factor analysis model on the state vectors. LDMs have been used extensively in the field of control systems for problems which require inferring the state of a system given noisy observations [Kailath, 2000]. LDMs have well developed mathematical equations for state inference using Kalman filtering, and they are easy to learn using expectation-maximization, making them a desirable choice for modeling sequential data.

Before discussing LDMs, we briefly describe the autoregressive HMMs which can be considered as a special case of LDMs. Unlike LDMs, autoregressive HMMs do not have a separate state and observation space. We then proceed to a detailed discussion of linear dynamical models and introduce *second-order* LDMs, a novel contribution in this thesis.

2.1 Autoregressive HMMs

As explained in the previous chapter, vanilla HMM based speech synthesis generates constant observation vectors within any sub-segment, which corresponds to one state of an HMM. A variety of methods have been suggested to counter that problem. One approach is to use autoregressive HMMs instead of regular HMMs.

As is the case with regular HMMs, autoregressive HMMs were first used in ASR [Wellekens, 1987, Kenny et al., 1990, Woodland, 1992]. M. Shannon et al. published a series of papers on the use of autoregressive HMMs for speech synthesis [Shannon and Byrne, 2009, 2010, Shannon et al., 2013]. This was followed by another attempt with some modifications to the system model in [Quillen, 2012]. We will use the system equations proposed in [Shannon and Byrne, 2009]. Like any other sequence generation model with underlying hidden states, an autoregressive HMM has a hidden state sequence $\theta_{1:T}$ which produces a sequence of vector observations $X = \boldsymbol{x}_{1:T}$, T being the length of the sequence. Using Bayes rule, we can break up the conditional distribution of the observation sequence, given the state sequence as follows:

$$\mathrm{P}(X|\theta_{1:T}) = \prod_{t=1}^{T} \mathrm{P}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \theta_{1:T})$$

In regular HMMs, the observations are independent given the state sequence

$$\mathrm{P}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \theta_{1:T}) = \mathrm{P}(\boldsymbol{x}_t|\theta_{1:T})$$

whereas in case of autoregressive HMMs, the conditional probability assumes the following Gaussian distribution:

$$\mathrm{P}(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \theta_t) = \mathcal{N}(\boldsymbol{x}_t|\boldsymbol{\mu}_{\theta_t}(\boldsymbol{x}_{1:t-1}), \Sigma_{\theta_t})$$

The mean of the distribution at time t for an autoregressive HMM q is assumed to be dependent on the previous observations by the following equation:

$$\boldsymbol{\mu}_q(\boldsymbol{x}_{1:t-1}) = \sum_{d=1}^{D} A_q^d(\boldsymbol{f}^d(\boldsymbol{x}_{1:t-1}) - \boldsymbol{\mu}_q^d) + \boldsymbol{\mu}_q^0 \tag{2.1}$$

The previous observation vectors are used to produce D summary vectors $\boldsymbol{f}^d(\boldsymbol{x}_{1:t-1})$. A summary $\boldsymbol{f}_q^d$ is acted upon by a matrix A_q^d after subtracting a bias term $\boldsymbol{\mu}_q^d$ from the summary. There is an additional bias term $\boldsymbol{\mu}_q^0$ added to the result in the end. The bias vectors $\boldsymbol{\mu}_q^d$ are redundant but are used as a trick to make the estimation of the parameters easier [Woodland, 1992]. For each state q, we need to determine the parameters $A_q^1, \dots, A_q^D$, $\boldsymbol{\mu}_q^1, \dots, \boldsymbol{\mu}_q^D$ and $\boldsymbol{\mu}_q^0$. The parameter estimation equations for this system have been derived in [Shannon and Byrne, 2009].

We are free to choose the summarizers to be anything, but they are often some predefined linear combinations of the last K observation vectors:

$$\boldsymbol{f}^d(\boldsymbol{x}_{1:t-1}) = \sum_{k=1}^{K} w_k^d \boldsymbol{x}_{t-k}$$

Often certain simplifying assumptions are made, for example the summary vector $\boldsymbol{f}_q^d$ is defined to be the vector $\boldsymbol{x}_{t-d}$, with $K = D$. These types of models are called canonical autoregressive HMMs [Kenny et al., 1990]. In this case, (2.1) simplifies to:

$$\boldsymbol{\mu}_q(\boldsymbol{x}_{t-D:t-1}) = \sum_{d=1}^{D} A_q^d(\boldsymbol{x}_{t-d} - \boldsymbol{\mu}_q^d) + \boldsymbol{\mu}_q^0$$

Another common simplifying assumption is that A_q^d is a diagonal matrix. This leads to each component in the observation vector being a scalar, independent, autoregressive process. In our implementation of autoregressive HMMs for evaluation purposes, we use both of the above mentioned assumptions.

We can clearly see that in autoregressive HMMs, the observations are not piece-wise constant. They introduce the continuity and context dependence which is needed for good quality speech synthesis without deltas and double deltas. For speech synthesis, they have been found to perform better than the regular HMMs [Shannon et al., 2013].

2.2 Linear Dynamical Models

LDMs are a class of continuous state space models with a linear state evolution equation. As mentioned earlier, LDMs have a continuous vector hidden state space. LDMs do not replace the discrete state sequence $\theta_{1:T}$ in HMMs and autoregressive HMMs, with a continuous vector state sequence, however. One normally uses a set of LDMs for modeling data like acoustic features in speech. A discrete state θ_t in case of LDMs is used to select one LDM out of a set of LDMs. This is similar to autoregressive HMMs, in which the hidden state essentially picks up an autoregressive HMM, from which we produce multiple observations, before transitioning to a new state or essentially picking up a new model to produce another set of observations. LDMs, however, incorporate an additional layer of hidden variables, which are continuous vectors. In LDM based generation, a particular LDM produces a sequence of continuous hidden state vectors $X = \boldsymbol{x}_{1:T}$. Based on these states the observation vectors $Y = \boldsymbol{y}_{1:T}$ are generated. In the case of LDMs, we will use "state" to refer to the continuous valued vector $\boldsymbol{x}_t$. An LDM q can be described by the following set of equations:

$$\boldsymbol{x}_t = F_q\boldsymbol{x}_{t-1} + \boldsymbol{w}_t \ , \qquad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{g}_q, Q_q) \tag{2.2a}$$

$$\boldsymbol{y}_t = H_q\boldsymbol{x}_t + \boldsymbol{v}_t \ , \qquad \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{\mu}_q, R_q) \tag{2.2b}$$

$$\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{g}_{1,q}, Q_{1,q}) \tag{2.2c}$$

The state at time t is represented by an n-dimensional continuous valued vector $\boldsymbol{x}_t$ that linearly evolves through time according to (2.2a). F_q is the state transition matrix of size $n \times n$, $\boldsymbol{w}_t$ is a multi-dimensional

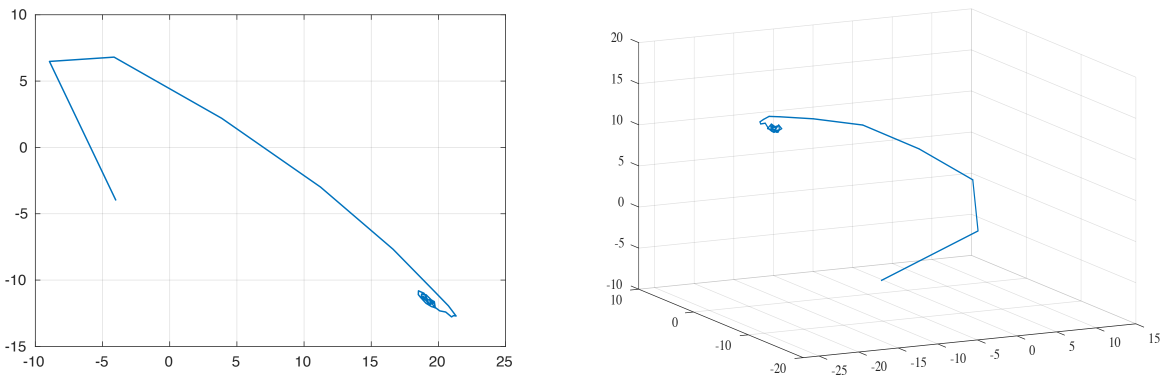

Figure 2.1: State and observation trajectory for an LDM with $n = 2$ and $m = 3$. The state and observation both converge to their target values with perturbations due to process and observation noise.

Gaussian random vector with mean $\boldsymbol{g}_q$ and variance Q_q. The state is not observable, however. An m-dimensional observation vector represented by $\boldsymbol{y}_t$ is observed at time t. The observation vector is obtained from the state vector by multiplying with an $m \times n$ dimensional observation matrix H and adding the noise vector $\boldsymbol{v}_t$, which is Gaussian distributed with mean $\boldsymbol{\mu}_q$ and variance R_q, as shown in (2.2b). The initial state is Gaussian distributed with mean $\boldsymbol{g}_{1,q}$ and variance $Q_{1,q}$. In control systems, the vectors $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$ are called process noise and observation noise respectively. All the samples of process noise, observation noise, and the initial state are independent of each other. Thus, an LDM q can be specified by the parameters $F_q, \boldsymbol{g}_q, Q_q, H_q, \boldsymbol{\mu}_q, R_q, \boldsymbol{g}_{1,q}$ and $Q_{1,q}$. In most of the subsequent discussion, we omit the subscript q.

The recursion in (2.2a) necessitates the stability of an LDM. An LDM is stable if the spectral radius of F is less than one. The state of a stable LDM converges to $(\mathrm{I} - F)^{-1}\boldsymbol{g}$. This could be considered the target state of an LDM which it eventually converges to, and $H(\mathrm{I} - F)^{-1}\boldsymbol{g} + \boldsymbol{\mu}$ is consequently the target observation of an LDM. fig. 2.1 shows a trajectory for an LDM with 2-dimensional state space and 3-dimensional observation space.

2.2.1 Autoregressive HMMs as LDMs

Autoregressive HMMs can be seen as a special case of LDMs. As illustrated in [Quillen, 2012], we can adapt LDM parameters to model autoregressive HMMs. As an example, a canonical autoregressive HMM of second order, i.e. the current observation vector factors through two previous observation vectors, can be modeled with the following set of equations:

$$\boldsymbol{x}'_t = F\boldsymbol{x}'_{t-1} + \boldsymbol{w}_t \ , \qquad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{g}, Q) \tag{2.3a}$$

$$\text{where, } \boldsymbol{x}'_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{bmatrix} , \qquad F = \begin{bmatrix} A^1 & A^2 \\ \mathrm{I} & 0 \end{bmatrix} , \qquad \boldsymbol{g} = \begin{bmatrix} \boldsymbol{\mu}^0 - A^1\boldsymbol{\mu}^1 - A^2\boldsymbol{\mu}^2 \\ 0 \end{bmatrix}$$

$$\boldsymbol{y}_t = H\boldsymbol{x}'_t \tag{2.3b}$$

$$\text{where, } H = \begin{bmatrix} \mathrm{I} & 0 \end{bmatrix}$$

All the notations used are the same as in the case of autoregressive HMM and LDM equations. A^1 and A^2 are diagonal matrices in the case of canonical autoregressive HMMs. We exclude any discussion on covariance matrices here. They will be dealt with in the subsequent sections.

2.2.2 Comparison with Recurrent Neural Networks

LDMs can in turn be seen as simpler variants of recurrent neural networks (RNNs). They are simpler because there is no non-linearity involved, and the state progresses without any external input being fed into them at each time step. LDMs are also somewhat similar to a decoder in an encoder-decoder RNN architecture [Cho et al., 2014], where the input to the decoder is only a summary vector which corresponds to the initial state in an LDM. However, unlike an LDM, a decoder uses the summary vector at every time step and the current state could depend on previous state as well as previous output, whereas in an LDM the current state depends on the previous state only. Consequently, LDMs can be trained using EM whereas RNNs are trained by back-propagating gradients. A 2-layer RNN with a given initial state $\boldsymbol{x}_1$, subsequent hidden states $\boldsymbol{x}_{2:T}$, outputs $\boldsymbol{y}_{1:T}$ is represented as:

$$\boldsymbol{x}_t = \sigma_x(F\boldsymbol{x}_{t-1} + \boldsymbol{g})$$

$$\boldsymbol{y}_t = \sigma_y(H\boldsymbol{x}_t + \boldsymbol{\mu})$$

where σ_x and σ_y are hidden-state and output activation functions. These equations are the same as the LDM equations (2.2), except that in LDMs the activation functions are identity.

2.2.3 Inference

Given an LDM with parameters Θ, and a set of observations Y, three types of hidden state inference problems can be defined:

- **filtering** : $\mathrm{P}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}, \Theta)$
- **smoothing** : $\mathrm{P}(\boldsymbol{x}_t|Y, \Theta)$
- **prediction** : $\mathrm{P}(\boldsymbol{x}_{t+\tau}|\boldsymbol{y}_{1:t}, \Theta)$

Thus, in filtering the state inference in causal, whereas, in case of smoothing, the state is inferred using all the observations. Filtering and smoothing, which are required in order to learn the LDM parameters via expectation maximization, are discussed below. Prediction is required when observation vectors are not available for some time steps, which is not the case with our problem, so we omit the discussion on prediction. For the sake of convenience, we use the following notation:

$$\begin{aligned}
\hat{\boldsymbol{x}}_{i|j} &\equiv \mathrm{E}[\boldsymbol{x}_i|\boldsymbol{y}_{1:j}] \\
\Sigma_{i|j} &\equiv \mathrm{Var}(\boldsymbol{x}_i|\boldsymbol{y}_{1:j}) \\
\Sigma_{i,i-1|j} &\equiv \mathrm{Cov}(\boldsymbol{x}_i, \boldsymbol{x}_{i-1}|\boldsymbol{y}_{1:j}) \\
\hat{P}_{i|j} &\equiv \mathrm{E}[\boldsymbol{x}_i\boldsymbol{x}_i^\top|\boldsymbol{y}_{1:j}] \\
\hat{P}_{i,i-1|j} &\equiv \mathrm{E}[\boldsymbol{x}_i\boldsymbol{x}_{i-1}^\top|\boldsymbol{y}_{1:j}]
\end{aligned}$$

Filtering

The filtering problem is defined as the causal inference of the state $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t}$. Filtering is done using Kalman Filter recursions [Kalman, 1960]. Kalman filters perform the inference recursively, where each recursion consists of two steps. The Kalman filter recursions can be written in multiple ways but one of the standard forms is stated as follows:

$$\hat{\boldsymbol{x}}_{t|t-1} = F\hat{\boldsymbol{x}}_{t-1|t-1} + \boldsymbol{g} \tag{2.4a}$$

$$\Sigma_{t|t-1} = F\Sigma_{t-1|t-1}F^\top + Q \tag{2.4b}$$

$$\boldsymbol{e}_t = \boldsymbol{y}_t - \hat{\boldsymbol{y}}_{t|t-1} = \boldsymbol{y}_t - H\hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{\mu} \tag{2.4c}$$

$$\Sigma_{e_t} = H\Sigma_{t|t-1}H^\top + R \tag{2.4d}$$

$$K_t = \Sigma_{t|t-1}H^\top\Sigma_{\boldsymbol{e}_t}^{-1} \tag{2.4e}$$

$$\hat{\boldsymbol{x}}_{t|t} = \hat{\boldsymbol{x}}_{t|t-1} + K_t\boldsymbol{e}_t \tag{2.4f}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - K_t\Sigma_{\boldsymbol{e}_t}K_t^\top \tag{2.4g}$$

$$\left(\Sigma_{t,t-1|t} = (\mathrm{I} - K_tH)F\Sigma_{t-1|t-1}\right) \tag{2.4h}$$

In the first phase, $\hat{\boldsymbol{x}}_{t|t-1}$ and $\Sigma_{t|t-1}$ are predicted given $\hat{\boldsymbol{x}}_{t-1|t-1}$ and $\Sigma_{t-1|t-1}$ as shown in (2.4a) and (2.4b) respectively. In the second phase, the error $\boldsymbol{e}_t$ in the predicted observation $\hat{\boldsymbol{y}}_{t|t-1}$ (2.4c) is used to refine the state estimation (2.4f) using the optimal Kalman gain matrix K_t (2.4e). The variance of the state vector is thus reduced as shown in (2.4g). The autocovariance matrix given in (2.4h) is used in the calculation of a statistic in the backward recursions in smoothing. The initial state and process noise are Gaussian distributed, hence the state process itself is Gaussian. The local likelihood function for observations is also Gaussian, leading to the conditional distribution of state given a set of observations being Gaussian as well.

Smoothing

In the smoothing problem, the state vectors are inferred given all the observation vectors. This is done using an RTS smoother [Rauch et al., 1965] which consists of a forward pass, which is the same as Kalman filtering, and a backward pass. The backward pass adjusts the state estimates once all the data have been observed. A gain matrix A_t, similar to the Kalman gain matrix, is introduced as shown in (2.5a). The estimated state is essentially a linear combination of the estimation made by the two filters as shown in (2.5b). The corresponding variance is shown in (2.5c). The statistics calculated in (2.5e) and (2.5f) are not a part of the standard forward-backward recursions, but are used in the parameter estimation of the LDMs. The backward recursions are stated as follows:

$$A_t = \Sigma_{t-1|t-1} F^\top \Sigma_{t|t-1}^{-1} \tag{2.5a}$$

$$\hat{\boldsymbol{x}}_{t-1|T} = \hat{\boldsymbol{x}}_{t-1|t-1} + A_t(\hat{\boldsymbol{x}}_{t|T} - \hat{\boldsymbol{x}}_{t|t-1}) \tag{2.5b}$$

$$\Sigma_{t-1|T} = \Sigma_{t-1|t-1} + A_t(\Sigma_{t|T} - \Sigma_{t|t-1})A_t^\top \tag{2.5c}$$

$$\Sigma_{t,t-1|T} = \Sigma_{t|T} A_t^\top \tag{2.5d}$$

$$\left(\hat{P}_{t-1|T} = \Sigma_{t-1|T} + \hat{\boldsymbol{x}}_{t-1|T}\hat{\boldsymbol{x}}_{t-1|T}^\top \right) \tag{2.5e}$$

$$\left(\hat{P}_{t,t-1|T} = \Sigma_{t,t-1|T} + \hat{\boldsymbol{x}}_{t|T}\hat{\boldsymbol{x}}_{t-1|T}^\top \right) \tag{2.5f}$$

2.3 Second-order LDMs

The state evolution equation in standard LDMs (2.2a) is first order, i.e. the current state's dependence on previous states factors through the immediately previous state. In order to provide more flexibility to the model, we increase the order so that the current state factors through the previous two states. This essentially leads to the state evolution being an autoregressive process, with observations still modeled as factor analysis on the state space. A second order LDM q is represented as follows:

$$\boldsymbol{x}_t = F_q\boldsymbol{x}_{t-1} + G_q\boldsymbol{x}_{t-2} + \boldsymbol{w}_t \ , \qquad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{g}_q, Q_q) \ ; \tag{2.6a}$$

$$\boldsymbol{y}_t = H_q\boldsymbol{x}_t + \boldsymbol{v}_t \ , \qquad \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{\mu}_q, R_q) \tag{2.6b}$$

$$\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{g}_{1,q}, Q_{1,q}) \ , \qquad \boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{g}_{0,q}, Q_{0,q}) \tag{2.6c}$$

Using the probability distribution of an $\boldsymbol{x}_0$ as the base case in (2.6c) rather than $\boldsymbol{x}_{2|1}$ leads to easier calculations in the EM algorithm. Similar to the set of equations for autoregressive HMMs (2.3), second order LDM equations can be transformed into a first-order LDM equation form:

$$\boldsymbol{x}'_t = F'\boldsymbol{x}'_{t-1} + \boldsymbol{w}_t \ , \qquad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{g}', Q') \tag{2.7a}$$

$$\text{where, } \boldsymbol{x}'_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{bmatrix} , \quad F' = \begin{bmatrix} F & G \\ \mathrm{I} & 0 \end{bmatrix} , \quad \boldsymbol{g}' = \begin{bmatrix} \boldsymbol{g} \\ 0 \end{bmatrix} , \quad Q' = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix}$$

$$\boldsymbol{y}_t = H'\boldsymbol{x}'_t + \boldsymbol{v}_t \tag{2.7b}$$

$$\text{where, } H' = \begin{bmatrix} H & 0 \end{bmatrix} , \quad \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{\mu}, R)$$

By converting to the above form, the Kalman filtering (2.4) and smoothing (2.5) equations can be used for inferring the states of a second order LDM as well.

The state recursive equation in second-order LDMs (2.6a) is more powerful than the recursion in first-order LDMs (2.2a). The trajectory of second-order LDMs is, in general, smoother than that of first-order LDMs because of averaging over two previous states rather than one. When restricted to diagonal state transition matrices, second order relations can model oscillations that first order relations cannot as shown in fig. 2.2. We found that LDMs with diagonal matrices F and G are sufficient to model the trajectories of acoustic features as the speech generated using these diagonal transition matrices is good as the one generated using full matrices. The evaluation metrics and the detailed results are presented in the next chapter. Thus, it uses $2n$ parameters in a state progression matrix as opposed to n^2 parameters in first-order LDMs. The parameters can be further reduced to n by using a critical damping system assumption, which will be discussed in the next chapter.

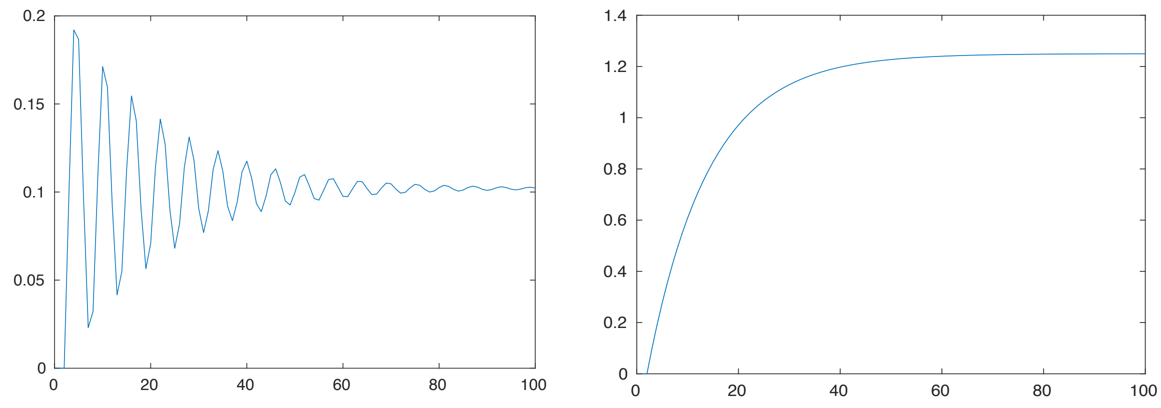

Figure 2.2: Example trajectory of a second order relation $x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + g$ is shown on left, while that for a first order relation $x_t = \phi x_{t-1} + g$ is shown on the right

2.4 Parameter Estimation

LDM parameter estimation is an unsupervised problem. An LDM can be considered as a generative model which produces a sequence of observations. The most popular method for training LDMs is the expectation-maximization (EM) algorithm which was introduced by Digalakis et al. [Digalakis et al., 1993] for first-order LDMs. In this section, we will derive the EM equations for second order LDMs. EM provides a simple and convergent solution to LDM learning.

Maximum likelihood estimation of the unknown LDM parameters Θ is obtained by minimizing the objective function, which is the negative log-likelihood of the observations. Following the generalized concept of expectation-maximization [Bishop, 2006], we introduce some distribution q over X and manipulate the log-likelihood function, $\mathcal{L}$ as follows:

$$\begin{aligned}
\mathcal{L}(\Theta) &= \log \mathrm{P}(Y|\Theta) \\
&= \int_X q(X) \log \left(\frac{\mathrm{P}(X,Y|\Theta)}{q(X)} \frac{q(X)}{\mathrm{P}(X|Y,\Theta)} \right) \mathrm{d}X \\
&= \int_X q(X) \log \left(\frac{\mathrm{P}(X,Y|\Theta)}{q(X)} \right) \mathrm{d}X - \int_X q(X) \log \left(\frac{\mathrm{P}(X|Y,\Theta)}{q(X)} \right) \mathrm{d}X \\
&= \mathcal{F}(q,\Theta) + \mathrm{KL}(q\|\mathrm{p})
\end{aligned}$$

where $\mathcal{F}(q,\Theta)$ is a functional of $q(X)$ and $\mathrm{KL}(q\|\mathrm{p})$ is the Kullback-Leibler (KL) divergence between $q(X)$ and the posterior distribution $\mathrm{P}(X|Y,\Theta)$, which has the property $\mathrm{KL}(q\|\mathrm{p}) \geq 0$, with equality being obtained if and only if $q(X) = \mathrm{P}(X|Y,\Theta)$. This provides a lower bound on the log-likelihood function:

$$\mathcal{L}(\Theta) \geq \mathcal{F}(q, \Theta).$$

EM algorithm involves an iterative two-stage optimization. In the expectation step, the lower bound is maximized with respect to $q(X)$, keeping the parameters fixed, denoted by Θ^{old}. This is achieved by setting:

$$q(X) = \mathrm{P}(X|Y, \Theta^{\text{old}}),$$

which makes the KL divergence term zero, and the log-likelihood becomes equal to the lower bound. Thus, at the end of the expectation step we have:

$$\begin{aligned}\mathcal{L}(\Theta^{\text{old}}) &= \mathcal{F}(q, \Theta^{\text{old}}) \qquad (2.8)\\ &= \int_X \mathrm{P}(X|Y, \Theta^{\text{old}}) \log \mathrm{P}(X, Y|\Theta^{\text{old}}) \mathrm{d}X - \int_X \mathrm{P}(X|Y, \Theta^{\text{old}}) \log \mathrm{P}(X|Y, \Theta^{\text{old}}) \mathrm{d}X \qquad (2.9)\end{aligned}$$

In the maximization step, the lower bound $\mathcal{F}(q, \Theta)$ is maximized with respect to Θ keeping $q(X)$ fixed:

$$\begin{aligned}\Theta^{\text{new}} &= \underset{\Theta}{\operatorname{argmax}}\, \mathcal{F}(q, \Theta)\\ &= \underset{\Theta}{\operatorname{argmax}} \int_X \mathrm{P}(X|Y, \Theta^{\text{old}}) \log \mathrm{P}(X, Y|\Theta) \mathrm{d}X - \int_X \mathrm{P}(X|Y, \Theta^{\text{old}}) \log \mathrm{P}(X|Y, \Theta^{\text{old}}) \mathrm{d}X\\ &= \underset{\Theta}{\operatorname{argmax}} \int_X \mathrm{P}(X|Y, \Theta^{\text{old}}) \log \mathrm{P}(X, Y|\Theta) \mathrm{d}X + \mathrm{H}(q)\end{aligned}$$

The second term here does not depend on Θ, so the expression can be reduced to

$$\begin{aligned}\Theta^{\text{new}} &= \underset{\Theta}{\operatorname{argmax}} \int_X \mathrm{P}(X|Y, \Theta^{\text{old}}) \log \mathrm{P}(X, Y|\Theta) \mathrm{d}X\\ &= \underset{\Theta}{\operatorname{argmax}}\, \mathrm{E}[\log \mathrm{P}(X, Y|\Theta)|Y, \Theta^{\text{old}}]\\ &\equiv \underset{\Theta}{\operatorname{argmax}}\, \mathcal{Q}(\Theta, \Theta^{\text{old}})\end{aligned}$$

The function $\mathcal{Q}(\Theta, \Theta^{\text{old}})$ is called the auxiliary function. This will increase the lower bound unless a local maximum has already been achieved. Updating Θ will make the KL divergence term non-zero. So, we repeat the procedure until $\mathcal{L}(\Theta)$ stops increasing.

The term inside the expectation of the auxiliary function, in case of a second order LDM, expands to:

$$\log \mathrm{P}(X, Y|\Theta) = \log \mathrm{P}(\boldsymbol{x}_0) + \log \mathrm{P}(\boldsymbol{x}_1) + \sum_{t=2}^{T} \log \mathrm{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}) + \sum_{t=1}^{T} \log \mathrm{P}(\boldsymbol{y}_t|\boldsymbol{x}_t)$$

The auxiliary function is evaluated as:

$$\begin{aligned}
\mathcal{Q}(\Theta, \Theta^{\text{old}}) &= \mathrm{E}[\log \mathrm{P}(\boldsymbol{x}_0)|Y, \Theta^{\text{old}}] + \mathrm{E}[\log \mathrm{P}(\boldsymbol{x}_1)|Y, \Theta^{\text{old}}] \\
&\quad + \sum_{t=2}^{T} \mathrm{E}[\log \mathrm{P}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2})|Y, \Theta^{\text{old}}] + \sum_{t=1}^{T} \mathrm{E}[\log \mathrm{P}(\boldsymbol{y}_t|\boldsymbol{x}_t)|Y, \Theta^{\text{old}}] \\
&= -\frac{1}{2}\log|Q_0| - \frac{1}{2}\mathrm{E}[(\boldsymbol{x}_0 - \boldsymbol{g}_0)^\top Q_0^{-1}(\boldsymbol{x}_0 - \boldsymbol{g}_0)|Y, \Theta^{\text{old}}] \\
&\quad - \frac{1}{2}\log|Q_1| - \frac{1}{2}\mathrm{E}[(\boldsymbol{x}_1 - \boldsymbol{g}_1)^\top Q_1^{-1}(\boldsymbol{x}_1 - \boldsymbol{g}_1)|Y, \Theta^{\text{old}}] \\
&\quad - \frac{T-1}{2}\log|Q| \\
&\quad - \frac{1}{2}\sum_{t=2}^{T} \mathrm{E}[(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})^\top Q^{-1}(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})|Y, \Theta^{\text{old}}] \\
&\quad - \frac{T}{2}\log|R| - \frac{1}{2}\sum_{t=1}^{T} \mathrm{E}[(\boldsymbol{y}_t - H\boldsymbol{x}_t - \boldsymbol{\mu})^\top R^{-1}(\boldsymbol{y}_t - H\boldsymbol{x}_t - \boldsymbol{\mu})|Y, \Theta^{\text{old}}]
\end{aligned}$$

In the E step, sufficient statistics are gathered from the forward and backward Kalman recursions. For the sake of simplicity, we provide equations for a single training sequence. When training on multiple sequences, which is normally the case, the statistics are added for each training sequence in the training data. The following are the sufficient statistics collected:

$$\boldsymbol{\zeta}_1 = \sum_{t=1}^{T-1} \hat{\boldsymbol{x}}_{t|T} \tag{2.10a}$$

$$\boldsymbol{\zeta}_2 = \sum_{t=2}^{T} \hat{\boldsymbol{x}}_{t|T} \tag{2.10b}$$

$$\boldsymbol{\zeta}_3 = \sum_{t=1}^{T} \hat{\boldsymbol{x}}_{t|T} \tag{2.10c}$$

$$\boldsymbol{\zeta}_4 = \sum_{t=1}^{T} \boldsymbol{y}_t \tag{2.10d}$$

$$\boldsymbol{\zeta}_5 = \sum_{t=0}^{T-2} \hat{\boldsymbol{x}}_{t|T} \tag{2.10e}$$

$$\Gamma_1 = \sum_{t=1}^{T-1} \hat{P}_{t|T} \tag{2.10f}$$

$$\Gamma_2 = \sum_{t=2}^{T} \hat{P}_{t|T} \tag{2.10g}$$

$$\Gamma_3 = \sum_{t=1}^{T} \hat{P}_{t|T} \tag{2.10h}$$

$$\Gamma_4 = \sum_{t=2}^{T} \hat{P}_{t,t-1|T} \tag{2.10i}$$

$$\Gamma_5 = \sum_{t=1}^{T} \boldsymbol{y}_t \hat{\boldsymbol{x}}_{t|T}^{\top} \tag{2.10j}$$

$$\Gamma_6 = \sum_{t=1}^{T} \boldsymbol{y}_t \boldsymbol{y}_t^{\top} \tag{2.10k}$$

$$\Gamma_7 = \sum_{t=0}^{T-2} \hat{P}_{t|T} \tag{2.10l}$$

$$\Gamma_8 = \sum_{t=1}^{T-1} \hat{P}_{t,t-1|T} \tag{2.10m}$$

$$\Gamma_9 = \sum_{t=2}^{T} \hat{P}_{t,t-2|T} \tag{2.10n}$$

In the M step, the auxiliary function is maximized with respect to each parameter by taking the derivative with respect to that parameter and equating it to zero. As in the case of the E step, the expressions presented below are for a single training sequence but can be trivially extended to multiple sequences. The expectations in these expressions are the statistics that were collected in the E step (2.10).

Expression for $\boldsymbol{g}_0$:

$$\begin{aligned}\frac{\partial \mathcal{Q}(\Theta, \Theta^{\text{old}})}{\partial \boldsymbol{g}_0} &= -\frac{1}{2} 2 Q_0^{-1} (\mathrm{E}[\boldsymbol{x}_0 | Y, \Theta^{\text{old}}] - \boldsymbol{g}_0) = 0 \\ \Rightarrow \boldsymbol{g}_0^{\text{new}} &= \mathrm{E}[\boldsymbol{x}_0 | Y, \Theta^{\text{old}}] \\ &= \hat{\boldsymbol{x}}_{0|T} \end{aligned} \tag{2.11}$$

In the case of multiple sequences, $\hat{\boldsymbol{x}}_{0|T}$ is averaged over multiple sequences. The expression for Q_0 is easier to calculate by differentiating with respect to Q_0^{-1}:

$$\begin{aligned}\frac{\partial \mathcal{Q}(\Theta, \Theta^{\text{old}})}{\partial Q_0^{-1}} &= \frac{1}{2} Q_0 - \frac{1}{2} \mathrm{E}[(\boldsymbol{x}_0 - \boldsymbol{g}_0)(\boldsymbol{x}_0 - \boldsymbol{g}_0)^{\top} | Y, \Theta^{\text{old}}] = 0 \\ \Rightarrow Q_0^{\text{new}} &= \mathrm{E}[\boldsymbol{x}_0 \boldsymbol{x}_0^{\top} | Y, \Theta^{\text{old}}] - \boldsymbol{g}_0^{\text{new}} \boldsymbol{g}_0^{\text{new}\top} \\ &= \hat{P}_{0|T} - \boldsymbol{g}_0^{\text{new}} \boldsymbol{g}_0^{\text{new}\top} \end{aligned} \tag{2.12}$$

In case Q_0 is assumed to be diagonal, only diagonal entries for both the terms in the expression are taken. In case of multiple sequences, $\hat{P}_{0|T}$ is averaged over all sequences. Expressions for $\boldsymbol{g}_1$ and Q_1 are similar to the ones for $\boldsymbol{g}_0$ and Q_0:

$$\begin{aligned}\boldsymbol{g}_1^{\text{new}} &= \mathrm{E}[\boldsymbol{x}_1 | Y, \Theta^{\text{old}}] \\ &= \hat{\boldsymbol{x}}_{1|T} \end{aligned} \tag{2.13}$$

$$\begin{aligned}
Q_1^{\text{new}} &= \mathrm{E}[\boldsymbol{x}_1\boldsymbol{x}_1^\top|Y,\Theta^{\text{old}}] - \boldsymbol{g}_1^{\text{new}}\boldsymbol{g}_1^{\text{new}\top} \\
&= \hat{P}_{1|T} - \boldsymbol{g}_1^{\text{new}}\boldsymbol{g}_1^{\text{new}\top} \qquad (2.14)
\end{aligned}$$

Expression for F:

$$\begin{aligned}
\frac{\partial \mathcal{Q}(\Theta,\Theta^{\text{old}})}{\partial F} &= -\frac{1}{2}(-2)Q^{-1}\sum_{t=2}^{T}\mathrm{E}[(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})\boldsymbol{x}_{t-1}^\top|Y,\Theta^{\text{old}}] \\
&\Rightarrow \sum_{t=2}^{T}E[\boldsymbol{x}_t\boldsymbol{x}_{t-1}^\top|Y,\Theta^{\text{old}}] - F\sum_{t=2}^{T}\mathrm{E}[\boldsymbol{x}_{t-1}\boldsymbol{x}_{t-1}^\top|Y,\Theta^{\text{old}}] \\
&\quad -G\sum_{t=2}^{T}\mathrm{E}[\boldsymbol{x}_{t-2}\boldsymbol{x}_{t-1}^\top|Y,\Theta^{\text{old}}] - \boldsymbol{g}\sum_{t=2}^{T}\mathrm{E}[\boldsymbol{x}_{t-1}^\top|Y,\Theta^{\text{old}}] = 0 \\
&\sum_{t=2}^{T}\hat{P}_{t,t-1|T} - F\sum_{t=1}^{T-1}\hat{P}_{t|T} - G\sum_{t=1}^{T-1}\hat{P}_{t,t-1|T}^\top - \boldsymbol{g}\sum_{t=1}^{T-1}\hat{\boldsymbol{x}}_{t|T}^\top = 0 \\
&\Gamma_4 - F\Gamma_1 - G\Gamma_8^\top - g\boldsymbol{\zeta}_1^\top = 0
\end{aligned}$$

Substituting the value of $\boldsymbol{g}$ in terms of F and G from (2.18):

$$\Rightarrow F^{\text{new}} = \left[\Gamma_4 - \frac{1}{T-1}\boldsymbol{\zeta}_2\boldsymbol{\zeta}_1^\top + G^{\text{new}}\left(\frac{1}{T-1}\boldsymbol{\zeta}_5\boldsymbol{\zeta}_1^\top - \Gamma_8^\top\right)\right]\left(\Gamma_1 - \frac{1}{T-1}\boldsymbol{\zeta}_1\boldsymbol{\zeta}_1^\top\right)^{-1} \qquad (2.15)$$

Similarly, expression for G:

$$G^{\text{new}} = \left[\Gamma_9 - \frac{1}{T-1}\boldsymbol{\zeta}_2\boldsymbol{\zeta}_5^\top + F^{\text{new}}\left(\frac{1}{T-1}\boldsymbol{\zeta}_1\boldsymbol{\zeta}_5^\top - \Gamma_8^\top\right)\right]\left(\Gamma_7 - \frac{1}{T-1}\boldsymbol{\zeta}_5\boldsymbol{\zeta}_5^\top\right)^{-1} \qquad (2.16)$$

The expressions for F^{new} and G^{new} involve each other. From (2.15) and (2.16) we can write these expressions independently as [1]:

$$\begin{bmatrix} F^{\text{new}} \\ G^{\text{new}} \end{bmatrix} = \Lambda^{-1}\Delta \qquad (2.17)$$

$$\text{where, } \Delta = \left[(\Gamma_4 - \tfrac{1}{T-1}\boldsymbol{\zeta}_2\boldsymbol{\zeta}_1^\top)(\Gamma_1 - \tfrac{1}{T-1}\boldsymbol{\zeta}_1\boldsymbol{\zeta}_1^\top)^{-1} \quad (\Gamma_9 - \tfrac{1}{T-1}\boldsymbol{\zeta}_2\boldsymbol{\zeta}_5^\top)(\Gamma_7 - \tfrac{1}{T-1}\boldsymbol{\zeta}_5\boldsymbol{\zeta}_5^\top)^{-1}\right]^\top$$

$$\Lambda = \begin{bmatrix} \mathrm{I} & -\left(\tfrac{1}{T-1}\boldsymbol{\zeta}_5\boldsymbol{\zeta}_1^\top - \Gamma_8^\top\right)\left(\Gamma_1 - \tfrac{1}{T-1}\boldsymbol{\zeta}_1\boldsymbol{\zeta}_1^\top\right)^{-1} \\ -\left(\tfrac{1}{T-1}\boldsymbol{\zeta}_1\boldsymbol{\zeta}_5^\top - \Gamma_8^\top\right)\left(\Gamma_7 - \tfrac{1}{T-1}\boldsymbol{\zeta}_5\boldsymbol{\zeta}_5^\top\right)^{-1} & \mathrm{I} \end{bmatrix}$$

[1] In our implementation, we iteratively find F^{new} and G^{new} instead of finding the matrix inverse as in (2.17). This is followed by restricting the spectral radius to be less than 1 using the method described in [Quillen, 2012]

Expression for $\boldsymbol{g}$:

$$\begin{aligned}
\frac{\partial \mathcal{Q}(\Theta,\Theta^{\text{old}})}{\partial \boldsymbol{g}} &= -\frac{1}{2}(-2)Q^{-1}\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - g|Y,\Theta^{\text{old}}] \\
\Rightarrow \boldsymbol{g}^{\text{new}} &= \frac{1}{T-1}(\sum_{t=2}^{T}E[\boldsymbol{x}_t|Y,\Theta^{\text{old}}] - F^{\text{new}}\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_{t-1}|Y,\Theta^{\text{old}}] - G^{\text{new}}\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_{t-2}|Y,\Theta^{\text{old}}]) \\
&= \frac{1}{T-1}\left(\sum_{t=2}^{T}\hat{\boldsymbol{x}}_{t|T} - F^{\text{new}}\sum_{t=1}^{T-1}\hat{\boldsymbol{x}}_{t|T} - G^{\text{new}}\sum_{t=0}^{T-2}\hat{\boldsymbol{x}}_{t|T}\right) \\
&= \frac{1}{T-1}(\boldsymbol{\zeta}_2 - F^{\text{new}}\boldsymbol{\zeta}_1 - G^{\text{new}}\boldsymbol{\zeta}_5) \qquad (2.18)
\end{aligned}$$

Expression for Q is easier to find by differentiating with respect to Q^{-1}:

$$\frac{\partial \mathcal{Q}}{\partial Q^{-1}} = \frac{T-1}{2}Q - \frac{1}{2}\sum_{t=2}^{T}\text{E}[(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})^\top|Y,\Theta^{\text{old}}] = 0.$$

By assuming that $\frac{\partial \mathcal{Q}}{\partial Q^{-1}}$, $\frac{\partial \mathcal{Q}}{\partial F}$, $\frac{\partial \mathcal{Q}}{\partial G}$ and $\frac{\partial \mathcal{Q}}{\partial \boldsymbol{g}}$ are simultaneously 0, many terms in the product $(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})(\boldsymbol{x}_t - F\boldsymbol{x}_{t-1} - G\boldsymbol{x}_{t-2} - \boldsymbol{g})^\top$ cancel, leaving:

$$\begin{aligned}
Q^{\text{new}} &= \frac{1}{T-1}\left(\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_t\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}] - F^{\text{new}}\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_{t-1}\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}]\right. \\
&\quad \left. -G^{\text{new}}\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_{t-2}\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}] - \boldsymbol{g}^{\text{new}}\sum_{t=2}^{T}\text{E}[\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}]\right) \\
&= \frac{1}{T-1}[\Gamma_2 - F^{\text{new}}\Gamma_4^\top - G^{\text{new}}\Gamma_9^\top - \boldsymbol{g}^{\text{new}}\boldsymbol{\zeta}_2^\top] \qquad (2.19)
\end{aligned}$$

Expression for H, in which we substitute $\boldsymbol{\mu}$ in terms of H from (2.21):

$$\begin{aligned}
\frac{\partial \mathcal{Q}(\Theta,\Theta^{\text{old}})}{\partial H} &= -\frac{1}{2}(-2)R^{-1}\sum_{t=1}^{T}\text{E}[(\boldsymbol{y}_t - H\boldsymbol{x}_t - \boldsymbol{\mu})\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}] = 0 \\
\Rightarrow \sum_{t=1}^{T}\boldsymbol{y}_t\text{E}[\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}] - H\sum_{t=1}^{T}\text{E}[\boldsymbol{x}_t\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}] &- \boldsymbol{\mu}\sum_{t=1}^{T}\text{E}[\boldsymbol{x}_t^\top|Y,\Theta^{\text{old}}] = 0 \\
\Rightarrow H^{\text{new}} &= \left(\Gamma_5 - \frac{1}{T}\boldsymbol{\zeta}_4\boldsymbol{\zeta}_3^\top\right)\left(\Gamma_3 - \frac{1}{T}\boldsymbol{\zeta}_3\boldsymbol{\zeta}_3^\top\right)^{-1} \qquad (2.20)
\end{aligned}$$

Expression for $\boldsymbol{\mu}$:

$$\begin{aligned}
\frac{\partial \mathcal{Q}}{\partial \boldsymbol{\mu}} &= -\frac{1}{2}(-2)R^{-1}\sum_{t=1}^{T}\text{E}[\boldsymbol{y}_t - H\boldsymbol{x}_t - \boldsymbol{\mu}|Y,\Theta^{\text{old}}] = 0 \\
\Rightarrow \boldsymbol{\mu}^{\text{new}} &= \frac{1}{T}\left(\sum_{t=1}^{T}\boldsymbol{y}_t - H^{\text{new}}\sum_{t=1}^{T}\hat{\boldsymbol{x}}_{t|T}\right)
\end{aligned}$$

$$= \frac{1}{T}(\boldsymbol{\zeta}_4 - H^{\text{new}}\boldsymbol{\zeta}_3) \tag{2.21}$$

Expression for R:

$$\begin{aligned}
\frac{\partial \mathcal{Q}(\Theta, \Theta^{\text{old}})}{\partial R^{-1}} &= \frac{T}{2}R - \frac{1}{2}\sum_{t=1}^{T} \mathrm{E}[(\boldsymbol{y}_t - H\boldsymbol{x}_t - \boldsymbol{\mu})(\boldsymbol{y}_t - H\boldsymbol{x}_t - \boldsymbol{\mu})^\top | Y, \Theta^{\text{old}}] = 0 \\
\Rightarrow R^{\text{new}} &= \frac{1}{T}\left(\sum_{t=1}^{T} \boldsymbol{y}_t\boldsymbol{y}_t^\top - H^{\text{new}}\sum_{t=1}^{T}\mathrm{E}[\boldsymbol{x}_t|Y,\Theta^{\text{old}}]\boldsymbol{y}_t^\top - \boldsymbol{\mu}^{\text{new}}\sum_{t=1}^{T}\boldsymbol{y}_t^\top\right) \\
&= \frac{1}{T}\left(\Gamma_6 - H^{\text{new}}\Gamma_5^\top - \boldsymbol{\mu}^{\text{new}}\boldsymbol{\zeta}_4^\top\right) \qquad (2.22)
\end{aligned}$$

Chapter 3

LDMs for Text-to-Speech Synthesis

In this chapter we will discuss the application of LDMs in text-to-speech synthesis. An initial attempt to use LDMs for speech synthesis was made by C. Quellin [2010]. This was followed by a series of papers by V. Tsiaras et al. [2014, 2015, 2016] that further investigated LDM-based TTS. We start by providing a motivation for the use of LDMs in acoustic modeling in section 3.1. In speech synthesis, a phone segment is generally subdivided into sub-segments. The resulting entities are called subphones, which are the fundamental speech units modeled. Each context-dependent subphone is modeled by a single LDM, so a set of LDMs jointly model the speech. As is the case with HMMs, several context-dependent subphones are tied together and share the same LDM. This clustering of subphones is done using binary decision trees which is discussed in section 3.2. The algorithms for subphone segmentation are discussed in section 3.3. We provide the implementation details of our LDM-based TTS system in section 3.4. All the results are complied in the next chapter.

3.1 Motivation

Production of speech is a multi-step process. It involves the creation of airflow by the compression of the pulmonic cavity. This airflow is acted upon by the larynx which may provide voicing through vibrations of the vocal folds. The vocal tract then acts as a filter on the source excitation and produces different vowels and consonants. In SPSS, speech is generally decomposed into at least two components which model the source excitation and the vocal tract filter. The source is characterized with a binary variable which determines whether the sound is voiced or unvoiced, and for voiced sounds, the fundamental frequency (f_0). The vocal tract filter is characterized by the impulse response of the filter or equivalently the Fourier transform of the impulse response. We are mostly interested in the ability of LDMs to model the vocal tract filter.

The vocal tract's shape is determined by the position of various articulators like lips, teeth, alveloar ridge, hard palate, etc. The movement of the articulators over time changes the vocal tract's shape and thus changes the filter's impulse response. The movement of the various articulators can be characterized by a critically damped spring-mass system [Saltzman and Munhall, 1989]. A critically damped spring-mass system is governed by the following equation:

$$\frac{\mathrm{d}^2x(t)}{\mathrm{d}t^2} + 2\phi\frac{\mathrm{d}x(t)}{\mathrm{d}t} + \phi^2(x(t) - u) = w(t) \tag{3.1}$$

where $x(t)$ is the position of the articulator at time t, u is the target parameter of the system, ϕ^2 is the stiffness parameter and $w(t)$ is a zero-mean Gaussian noise. This can be written in a canonical form [Deng, 1999] (where $\dot{x}(t) = \frac{\mathrm{d}x(t)}{\mathrm{d}t}$):

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\phi^2 & -2\phi \end{bmatrix}\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \phi^2 u \end{bmatrix} + \begin{bmatrix} 0 \\ w(t) \end{bmatrix} \tag{3.2}$$

This continuous-time system of equations can be expressed by a discrete-time dynamic-system state equation [Deng, 1998]:

$$x_{t+1} = 2\phi x_t - \phi^2 x_{t-1} + (1-\phi)^2 u + w_t \tag{3.3}$$

which is an equation for an autoregressive process. The notation has been abused to use t to denote time in the continuous time equations (3.1) and (3.2), as well as in the discrete time equation (3.3). The above equations can be vectorized to simultaneously model multiple articulators. The acoustic parameters that characterize the filter's impulse response, can be obtained from a non-linear operation on the articulator state [Deng, 1998]. However we approximate it with a linear function that maps the state space of the articulators to the state space of the acoustic parameters using a matrix H and a noisy bias vector $\boldsymbol{v}_t$ with mean $\boldsymbol{\mu}$ and variance R. This leads to a second order LDM for speech production:

$$\boldsymbol{x}'_t = F'\boldsymbol{x}'_{t-1} + \boldsymbol{w}_t \ , \qquad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{g}', Q') \tag{3.4a}$$

$$\text{where, } \boldsymbol{x}'_t = \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{x}_{t-1} \end{bmatrix} , \qquad F' = \begin{bmatrix} 2\varphi & -\varphi^2 \\ \mathrm{I} & 0 \end{bmatrix} , \qquad \boldsymbol{g}' = \begin{bmatrix} (\mathrm{I}-\varphi)^2\boldsymbol{u} \\ 0 \end{bmatrix} ,$$

$$Q' = \begin{bmatrix} Q & 0 \\ 0 & 0 \end{bmatrix} , \qquad \varphi = \mathrm{diag}(\phi_1, \phi_2, ..., \phi_n) \ , \qquad Q = \mathrm{diag}(q_1, q_2, ..., q_n)$$

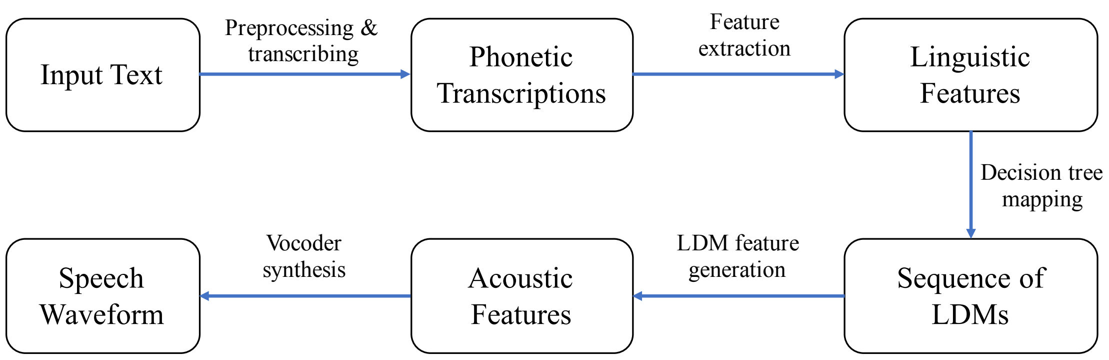

Figure 3.1: Schematic diagram of an LDM-based TTS system

$$\boldsymbol{y}_t = H'\boldsymbol{x}'_t + \boldsymbol{v}_t, \qquad \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{\mu}, R) \tag{3.4b}$$
$$\text{where, } H' = \begin{bmatrix} H & 0 \end{bmatrix}$$

φ and Q are diagonal matrices, with ϕ_i^2 and q_i being the stiffness parameter and variance associated with i^{th} articulator. This implies that the motion of articulators are independent. We found using full matrices do not provide any improvement in the speech, so we follow this assumption, unless stated otherwise. Apart from this, the advantages of using LDMs to model sequential data have already been discussed in the earlier chapters.

3.2 Decision Tree Clustering

A schematic diagram of an LDM-based TTS system is shown in fig. 3.1. The input is a text string and the output is a speech waveform. The speech waveform is generally generated from acoustic features which are a denser representation than the raw speech. The LDMs are responsible for generating these acoustic features. On the input side, the raw text is preprocessed and phonetic transcriptions are extracted from the clean normalized text. Some linguistic features are extracted from phonetic transcriptions which form the input to the backend of a TTS system. Generally one feature vector is extracted per subphone in the utterance. The linguistic features typically include the current phone, two preceding and two succeeding phones, position in the syllable, position in the word, position in the utterance, etc. The various features used in our model are listed in the appendix. Many of these features are the ones used in HTK [Young et al., 2006].

A statistical model is required to map the linguistic feature vector to an LDM, which will generate the corresponding acoustic features. This is done using a binary decision tree as used by V. Tsiaras et al. [2015]. The number of context-dependent subphones is much more than the number of LDMs, so the mapping is many to one. Thus a decision tree solves the problem of limited or non-existent data for many context-dependent subphones by clustering together various context-dependent phones. A separate decision tree is created for each sub-segment of each phone for each acoustic feature. For example our implementation segments phones into three sub-segments, starting from 51 phones and three acoustic features, so we create a total of $3 \times 51 \times 3$ decision trees.

PThe decision tree clustering is performed with the minimum description length (MDL) criterion [Rissanen, 1984]. The linguistic features associated with the subphone are used in the clustering process. This involves creating a root node which points to all the examples for a given subphone. An LDM is fit to these examples, and the data likelihood is calculated given that LDM. This is followed by recursively splitting the leaf nodes, beginning with the root node. For a given leaf node, data associated with that node are split into "yes" and "no" examples according to a binary question based on the linguistic features. A separate LDM is fit to the "yes" and "no" examples respectively. There should be an increase in the data likelihood relative to a weighted combination of these two LDMs as opposed to the single previous LDM. This is done for each available question for a given leaf node. If the increase in log-likelihood on splitting a node with some question is lesser than a certain threshold or if the number of examples in either of the "yes" or "no" categories are lesser than a threshold, that question is not considered for splitting. The question which provides the maximum increase of likelihood is selected for splitting this given leaf node. This is done recursively for each leaf node. Ultimately, for each leaf node, there will be no question available which could split the leaf any further. Further details about the algorithm can be found in V. Tsiaras et al. [2015].

The computational time required by this algorithm is very high because in order to split a node, LDMs of the order $2n_q$, where n_q are the number binary questions, need to be trained. A binary tree with l leaves will have $l - 1$ internal nodes. There are multiple decision trees which depend on the number of phones and sub-segments. All this contributes to a large number of LDMs being trained in order to cluster. Apart from providing an initial model to the EM algorithm, the sufficient statistics collected for the parent node cannot be used to directly train the LDMs for the potential child nodes. Instead of clustering the data using LDMs, we used autoregressive HMMs for clustering because they have a closed form solution as opposed to the iterative EM algorithm learning for LDMs. The quality of speech produced from autoregressive-HMM-based trees is found to be at par with the LDM-based trees.

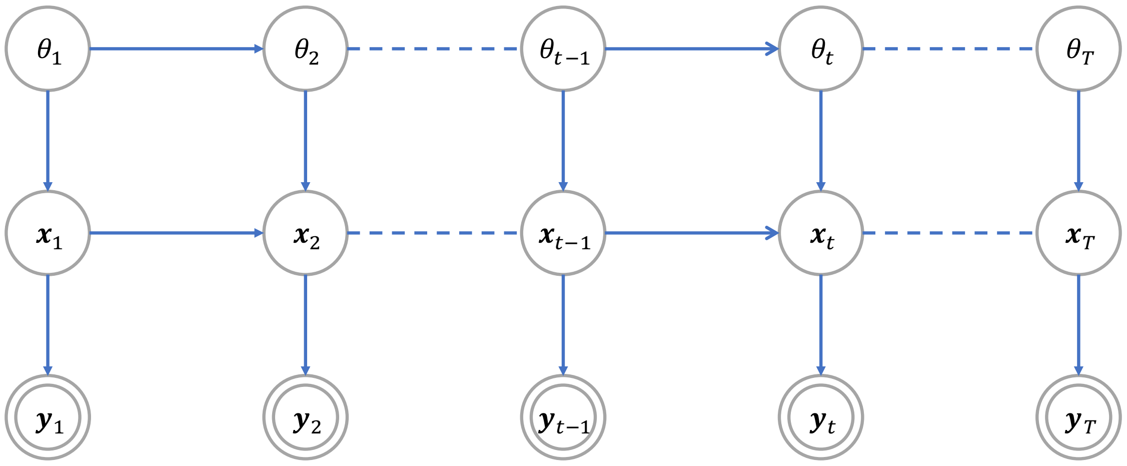

Figure 3.2: Bayesian network representation of the switching LDM used. $\theta_{1:T}$ represent the LDM selection variables, $\boldsymbol{x}_{1:T}$ are the state vectors, $\boldsymbol{y}_{1:T}$ are the observations and are thus the only observed variables

3.3 Subphone Segmentation

A phone is generally divided into 3 or 5 subphones. The vocal tract parameters are relatively stationary in a subphone as opposed to a phone, which may have different dynamics in the beginning, middle and end. This is especially true for phones like diphthongs and stops. Subphones correspond to different regions of a phone. In our implementation, we divided a phone into three subphones. Dividing a phone into more subphones leads to very short sub-segments, often of length 1 or 2 frames, thus defeating the purpose of using LDMs to model dynamics. In earlier approaches to LDM-based synthesis, a phone was divided into three equal segments [Tsiaras et al., 2015], which is suboptimal. Another approach was to use the segmentation provided by HMM models as a part of forced alignment [Tsiaras et al., 2014].

As each subphone maps to an LDM, after dividing a phone segment into three sub-segments, three LDMs together generate the acoustic features in that segment, one LDM per sub-segment. Thus each phone is essentially modeled by a switching-state-space model (switching LDM). There has been a lot of work on switching-state-space models in the area of control and estimation [Ghahramani and Hinton, 2000]. The various models proposed differ in the underlying assumptions of the generation process, and consequently the complexity of the model. In our problem there are two conditions that dictate the underlying assumptions. Firstly, there is a left to right switch between the LDMs, i.e. the model cannot switch back to a previous LDM once it has switched to the next LDM. Secondly, the member LDMs are not necessarily unique to one switching LDM, so one LDM can be a member of multiple switching LDMs. For example, two context-dependent phones might be modeled by LDMs $\{l_1, l_2, l_3\}$ and $\{l_7, l_2, l_9\}$

respectively, where l_2 belongs to both of them. Based on these two conditions, our switching LDM is loosely based on the model introduced in section 11.6.3 of Y. Bar-Shalom et al. [2002], which provides an approximate solution to the state estimation problem, but does not provide any algorithm to learn the parameters. The model can be represented by the graph depicted in fig. 3.2. θ_t selects the LDM at time t. Due to the first condition mentioned above, it only depends on θ_{t-1}. The state vector $\boldsymbol{x}_t$ is dependent on θ_t as well as on previous state $\boldsymbol{x}_{t-1}$. The observation $\boldsymbol{y}_t$ depends only on $\boldsymbol{x}_t$. We learn the parameters using expectation maximization. Expectation involves inferring the state vectors $x_{1:T}$ and the selection variables $\theta_{1:T}$. In the maximization step, the LDMs are again learned using the new segmentation derived by the current model parameters. This is done iteratively. Hard assignment is done for LDM selection, i.e. $\mathrm{P}(\theta_t)$ is a one-hot vector, rather than a soft assignment, which simplifies the model.

In the E step all the phone segments in the training data are internally resegmented, i.e. subphone boundaries for each phone segment are learned again using the current set of LDMs. Thus θ_t within each segment can only take 3 values $\{l_1, l_2, l_3\}$. Two methods are used to resegment. The first one is a brute-force search for the best segmentation by calculating the likelihood for all the possible segmentations of the phone (with the condition that each sub-segment is at least one frame long), and then choosing the best one. The number of possible segmentations is quadratic in the length of a segment ($\binom{n-1}{2}$ segmentations in a segment with n frames). Thus the time taken to find the best segmentation grows quickly as segments become long. We introduce an approximate method which grows linearly with the length of a segment. This second method involves doing an approximate Viterbi-like search, the basic philosophy of which is that in order to evaluate the state vector $\boldsymbol{x}_t$, Kalman filter recursions are not performed from the beginning of the sequence, but in a fixed interval. The presence of two LDM switches in the segment leads to an exponential number of paths all of which start with $\theta_1 = l_1$ and end with $\theta_T = l_3$, where T is the segment length, not the complete utterance length. In order to evaluate $\boldsymbol{x}_t$, we perform the Kalman filter recursions from $t-d$ till t, where d is the depth parameter. At each time step t, a state vector $\boldsymbol{x}_t^{\theta_t}$ is determined for all of the possible values of θ_t. So we determine three different state vectors corresponding to the three values of θ_t, along with the log-likelihoods of reaching those states, and obtain two $T \times 3$ tables in the end. There are multiple paths which start from $t-d$ and lead to θ_t. These multiple paths correspond to different starting states θ_{t-d}, and also different paths with the same starting state. The path which maximizes the log-likelihood of reaching θ_t from $t-d$ is chosen for each value of θ_t. The state $\boldsymbol{x}_t^{\theta_t}$ corresponding to the best path is stored per θ_t. Along with that, the second-to-last state of the best path, θ_{t-1}, is also stored per θ_t, resulting in yet another $T \times 3$ table. Similar to the Viterbi algorithm, this will be used for backtracking in the end. In order

Algorithm 1: Sub-optimal Subphone Segmentation

```
Data: $\boldsymbol{y}_{1:T}$; parameters $\Theta^{l_1}, \Theta^{l_2}, \Theta^{l_3}$ for LDMs $l_1, l_2, l_3$ respectively; $d$
Result: $\theta^*_{1:T}, \boldsymbol{x}^*_{1:T}$
Initialization: $\boldsymbol{x}^1_1 = \text{K.F.}(\boldsymbol{y}_1|\boldsymbol{g}^{l_1}_1, \theta_1 = l_1), \delta^1_1 = \text{L}(\boldsymbol{y}_1|\boldsymbol{g}^{l_1}_1, \theta_1 = l_1), \delta^2_1 = \delta^3_1 = -\infty$
for $t = 2 : T$ do
    $d' = \min(t-1, d)$
    for $j = 1 : 3$ do
        for $k = 1 : j$ do
            $\mathcal{P} = \{\theta_{t-d':t} : \theta_{t-d'} = l_k, \theta_t = l_j\}$
            $p^*_k = p \in \mathcal{P} : \text{L}(\boldsymbol{y}_{t-d':t}|\boldsymbol{x}^k_{t-d'}, p) > \text{L}(\boldsymbol{y}_{t-d':t}|\boldsymbol{x}^k_{t-d'}, p') \, \forall \, p' \in \mathcal{P}$
            $q_k = \text{L}(\boldsymbol{y}_{t-d':t}|\boldsymbol{x}^k_{t-d'}, p^*)$
        end
        $\delta^j_t = \max_k(q_k + \delta^k_{t-d'})$
        $r = \operatorname{argmax}_k(q_k + \delta^k_{t-d'})$
        $\psi^j_t = p^*_r(-2)$ (second last element of $p^*_r$)
        $\boldsymbol{x}^j_t = \text{K.F.}(\boldsymbol{y}_{t-d':t}|\boldsymbol{x}^r_{t-d'}, p^*_r)$
    end
end
$\theta^*_T = l_3$
$\boldsymbol{x}^*_T = \boldsymbol{x}^3_T$
for $t = T-1 : -1 : 1$ do
    $\theta^*_t = \psi^{\theta^*_{t+1}}_{t+1}$
    $\boldsymbol{x}^*_t = \boldsymbol{x}^{\theta^*_t}_t$
end
```

to find the path with the maximum log-likelihood and to do Kalman recursions, the log-likelihood at θ_{t-d} and the corresponding state vector $\boldsymbol{x}^{\theta_{t-d}}_{t-d}$ for each value of θ_{t-d} are required. These would have been stored in the table earlier when $\boldsymbol{x}_{t-d}$ had been calculated. This is done for all time steps and the best path is found by tracing back through the trellis. The pseudo-code is provided in algorithm 1. K.F.$(\boldsymbol{y}_{t_1:t_2}|\boldsymbol{x}_{t_1}, \theta_{t_1:t_2})$ refers to the Kalman filter recursions which infer the state $\boldsymbol{x}_{t_2|t_2}$ from $\boldsymbol{y}_{t_1:t_2}$, and require the initial state $\boldsymbol{x}_{t_1}$ and the path followed $\theta_{t_1:t_2}$. The only difference from regular Kalman recursions is that the LDM parameters at time t are given by Θ^{θ_t}, and thus they do remain constant throughout the recursion. $\text{L}(\boldsymbol{y}_{t_1:t_2}|\boldsymbol{x}_{t_1}, \theta_{t_1:t_2})$ refers to the log-likelihood that is actually derived from the Kalman filter but is written separately in the algorithm for clarity. The algorithm can easily be changed in order to accommodate a different number of sub-segments than 3. It was observed that the brute-force segmentation method starts taking more time to run than the approximate segmentation method when the number of frames in a segment exceeds 25. Thus we follow the brute-force segmentation method for phone segments with length less than 26 frames.

In the M step, the segmentation derived in the E step is used to individually re-train all the LDMs.

3.4 Implementation Details

In this section we will discuss the implementation details and design choices of the TTS system that we developed. We used the Nick corpus speech data released for the Hurricane Challenge [Hurricane Corpus 2012]. A total of 2643 utterances sampled at 16 kHz from the *herald* and *hvd* sets of the corpus were used for training the models.

3.4.1 Front-end

We used the Festival [Taylor et al., 1998] front end to generate HTS-style [Zen et al., 2007a] full context labels. The lexicon used was unilex-rpx [Fitt and Isard, 1999]. The corresponding phoneset had 49 phones. We used 2 phones for short and long pause respectively, so a total of 51 phones were used. The resulting full context labels were aligned with the speech audio using the HMM Toolkit (HTK) [Young et al., 2006]. For the forced alignment, MFCCs were extracted from frames of length 10 ms, with a frame shift of 5 ms.

3.4.2 Acoustic Features

The WORLD vocoder [Morise et al., 2016] with the D4C band-aperiodicity estimation [Morise, 2016] was used to obtain the acoustic features. The acoustic features include 40 Mel-cepstral coefficients (MCEPs), 1 log fundamental frequency ($\ln(f_0)$) and 1 band-aperiodicity coefficient, extracted every 5 ms. Thus we have a 42-dimensional acoustic feature vector per frame. WORLD outputs a raw spectral envelope which was converted into the Mel-cepstral coefficients using SPTK [SPTK 2016].

3.4.3 Back-end

The back-end used LDMs for acoustic modeling and was implemented in MATLAB. For parameter estimation, two methods were proposed by V. Tsiaras [2014]. Our implementation used the segment-level EM algorithm as opposed to the phone-level EM algorithm because it was shown to perform better than the latter [Tsiaras et al., 2014]. In this method, the EM algorithm is carried out for each label individually. Here, each label corresponds to an individual clustered subphone state obtained form a decision tree. For a particular label, the Kalman recursions are performed individually for each segment assigned that label and sufficient statistics are calculated in the E step. In the M step, the parameters are re-estimated using the accumulated statistics. The consequence of training each label segment-wise is that the state vector is not propagated from one segment to the next in an utterance. Even though one might expect this algorithm to perform worse because of the disruption in the continuity of state vector progression through the utterance, in practice it performs better because LDMs are quite sensitive to the initial state. So getting the initial state vector of a new segment from the last state vector of the previous segment does not work very well.

As mentioned in section 3.2, we created a binary decision tree for each segment of each phone. This results in 3 trees per phone, so there are a total of 153 decision trees for modeling Mel-cepstral coefficients alone. Clustering aperiodicity and $\ln(\mathrm{f}_0)$ will require 153 decision trees each as well. 1586 binary questions were used to create these binary decision trees. Out of these, 755 corresponded to binary features. The rest of the questions were created in order to specify the continuous valued features.

The Mel-cepstral coefficients were modeled using several LDM variants including the second order LDMs. The LDMs however do not provide any advantage in modeling one dimensional features, i.e., $\ln(\mathrm{f}_0)$ and BAP. So we used autoregressive HMMs to model those. However, in most of our experiments, we modeled only the Mel-cepstral coefficients in order to study the performance of LDMs only. In those cases, the original $\ln(\mathrm{f}_0)$ and BAP as obtained from the vocoder analysis were used. We did not model duration and used the durations obtained from HMM forced alignment for synthesis as well.

The subphone segmentation was done by three different methods. The first one is dividing each phone segment into three equal sub-segments. The second one involves using the subphone segmentation obtained from the HMM forced alignment, where the HMMs themselves are three-state left-to-right models. The third method is doing the re-segmentation as described in section 3.3 while starting from one of the first two segmentations mentioned above.

Some of the LDM parameters can be tied for all the LDMs. There is degeneracy in the LDMs. The state covariance Q and initial state covariance Q_1 can be moved into matrices F and H [Roweis and Ghahramani, 1999]. For an LDM with a given set of parameters, there exists an equivalent LDM with different F and H but with identity covariance matrices Q and Q_1. Thus without loss of generality we can assume Q and Q_1 to be identity matrices, which has been followed in all our experiments. The same cannot be said about R, though. Another method to reduce the number of parameters is to tie H for all the LDMs. This drastically reduces the total number of parameters by Nnm, where N is the total number of LDMs, n is the state dimension and m is the observation dimension. Tying H degrades the quality of speech but the reduction in the number of parameters is more significant. Reducing the number of clusters while keeping the number of parameters constant provides lower quality than tying H does.

3.4.4 Speech Parameter Generation

In order to generate an utterance, the system requires full context labels and the subphone segment durations corresponding to the utterance. The full context label for each subphone segment is used to determine the decision tree state and thus the corresponding LDM for that segment. The duration for each subphone segment is used to calculate the number of frames for that segment. With this we obtain $\theta_{1:T}$ where θ_t is the LDM used at time t and T is the total duration of the utterance in terms of the number of frames. The parameter generation equations are as follows:

$$\boldsymbol{x}_1 = \boldsymbol{g}_1^{\theta_1} \tag{3.5a}$$

$$\boldsymbol{x}_t = \begin{cases} F^{\theta_t}\boldsymbol{x}_{t-1} + \boldsymbol{g}^{\theta_t}, & \text{if } \theta_t = \theta_{t-1} \\ \rho\boldsymbol{g}^{\theta_t} + (1-\rho)\boldsymbol{x}_{t-1}, & \text{otherwise} \end{cases} \tag{3.5b}$$

$$\boldsymbol{y}_t = H^{\theta_t}\boldsymbol{x}_t + \boldsymbol{\mu}^{\theta_t} \tag{3.5c}$$

We can use the same set of equations for the second order LDMs with the help of the conversion technique illustrated in (2.7). There are two cases in (3.5b) because, in the middle of a segment, we use the usual LDM equation for state progression, but at a segment boundary, the first state of the next segment is a linear combination of the last state of the previous segment and the initial learned state. The parameter generation for a toy example with only 3 subphones in the utterance is shown in fig. 3.3. In our experiments, we observed that $\rho = 1$ always performs better than any other value. This could be attributed to two reasons. First is that even though the final state of an LDM is not dependent on the initial state, the beginning state trajectory is dependent on the initial state. It was observed that starting from an unusual initial state leads to large swings in the state vectors. The second reason is that in our implementation, the LDMs are trained such that the initial state of a segment is not dependent on the last state of the previous segment. After generating the Mel-cepstral feature vectors, global variance [Tsiaras et al., 2016] is applied, which marginally improves the speech quality.

3.4.5 Training Optimization Techniques

LDM training is a time consuming process. One of the techniques to speed-up the training process, introduced by J. Frankel and S. King [2007], is to pre-compute some of the quantities in the Kalman recursions. In the forward Kalman recursions, the quantities $\Sigma_{t|t-1}, \Sigma_{t|t}, \Sigma_{e_t}, K_t$, and, in the backward recursions A_t are independent of data. All these quantities can be calculated once for each subphone up to the maximum segment length of that subphone and reused in the Kalman recursions for all the

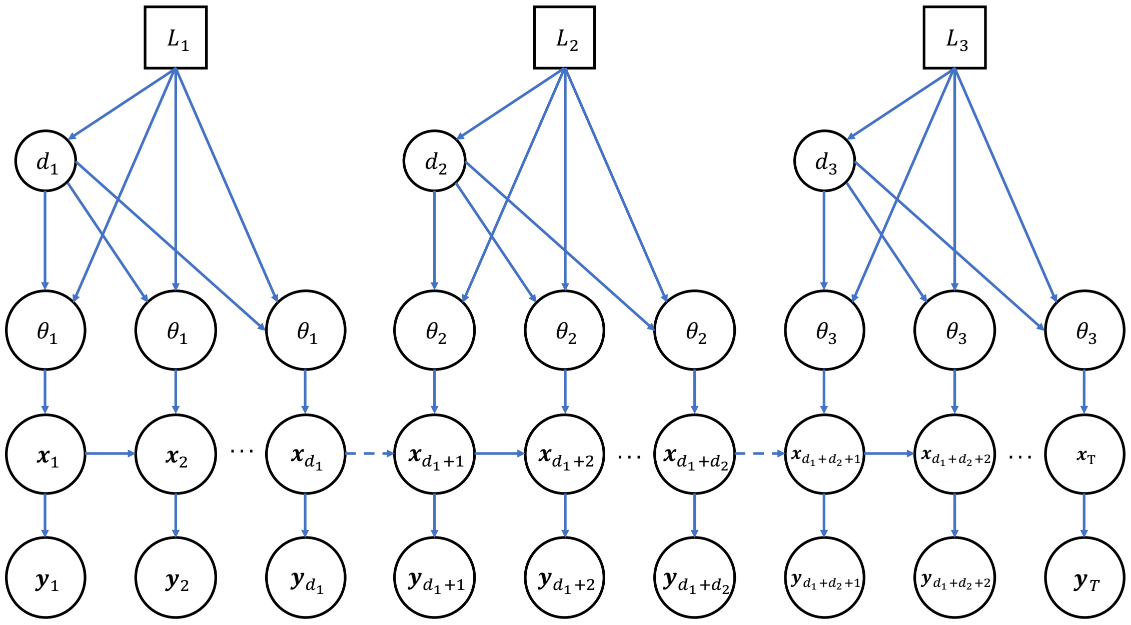

Figure 3.3: A toy generation example with three subphones of durations d_1, d_2, d_3 respectively in an utterance of length T. The regular arrows show dependency and dotted arrows show dependency where ρ is involved. $\rho = 1$ removes that dependency link.

segments. The decision tree clustering as well as LDM training can easily be parallelized on multiple CPUs. During decision tree clustering, the node split corresponding to the various questions can be carried out on different CPUs. In LDM training, different LDMs can be trained on different CPUs because the training process is independent for each LDM. We used 28 CPU cores to train. Both the above mentioned techniques significantly speed up the training process.

Chapter 4

Results and Conclusions

In this chapter, we will evaluate various aspects of the LDM-based speech synthesis that have been discussed in the previous chapters. We compare this approach to autoregressive HMMs and various configurations of neural networks. A total of 100 utterances from the *hvd* and *herald* sets of the Nick Hurricane corpus [Hurricane Corpus 2012] were synthesized in all the experiments.

4.1 Evaluation Metrics

In all our experiments the evaluation is performed using two metrics: Mel-Cepstral Distance (MCD) [Kubichek, 1993] and Perceptual Evaluation of Speech Quality (PESQ) [Rix et al., 2001]. These are described as follows:

- Mel-cepstral distance : Mel-cepstral distance (in dB) between two vector sequences of Mel-cepstral coefficients $\boldsymbol{c}_1, \boldsymbol{c}_2$ of lengths T is given by:

$$\mathrm{d}(\boldsymbol{c}_1, \boldsymbol{c}_2) = \frac{10}{T\ln(10)} \sum_{t=1}^{T} \sqrt{\sum_{i=1}^{m} [c_{t,1}(i) - c_{t,2}(i)]^2}$$

  where m is the Mel-cepstral vectors' dimension, $c_{t,1}(i)$ and $c_{t,2}(i)$ are the i^{th} Mel-cepstral coefficients (MCEPs) at time step t for the first and second sequences respectively. To evaluate a test sentence, the MCEPs generated by the acoustic model and the ones derived from vocoder analysis of the gold standard are used to calculate the MCD. The scoring metric is the average MCD over all test sentences. MCD evaluates the modeling ability of a generative model.

- perceptual evaluation of speech quality score : PESQ was originally introduced to evaluate the speech quality of a telephony system. Some studies have found a correlation between the raw PESQ score and subjective measures [Hu and Loizou, 2008]. Raw PESQ scores correspond to the perceptual similarity between the original and synthesized waveform. Thus PESQ has been argues to be a reasonable substitute for a subjective evaluation. Subjective study is time-consuming, so we use PESQ to do a quick evaluation in a large number of experiments.

4.2 Experiments

A number of experiments are conducted to evaluate different aspects of LDM based speech synthesis, each of which is described next.

4.2.1 Spectral modelling

As mentioned in previous chapters, LDMs are useful to model vector sequences. They do not provide any advantage over autoregressive HMMs in modelling scalar sequences. In our implementation, MCEPs are vector sequences whereas $\ln(\mathrm{f}_0)$ and band aperiodicity (BAP) are scalar sequences. In order to evaluate the speech modelling power of LDMs only, we obtain the $\ln(\mathrm{f}_0)$ and BAP sequences for each of the test sentences by vocoder analysis. Only MCEPs, which are responsible for providing spectral information, are obtained by the various acoustic models. The acoustic models that are compared are:

- First order LDMs : $n = 10$
- Second order LDMs : Two types of second order LDMs are used: with and without diagonal transition matrices, $n = 10$ in both. Henceforth we refer to these as SO-LDM$_\mathrm{d}$ and SO-LDM respectively.
- Autoregressive HMMs
- Feed-forward neural networks : Two models are used: one with 6 layers, each layer of 512 units and a second with 4 layers of 1024 units each. The hidden units used tanh activation. In all the neural networks, the input layer is of size 613 (corresponds to the number of linguistic features) and the output is a linear layer of size 40, because of the 40 MCEPs.
- Hybrid model : Hybrid (LSTM + feed-forward) neural network with one feed-forward layer of size 512, 3 LSTM layers of size 384 each and one feed forward layer of size 512 on top. All the neural networks are implemented using the Merlin speech synthesis system [Wu et al., 2016].

Model	Details	No. Param.	MCD	PESQ
2nd order LDM	diagonal F, G	2,804,230	4.06	2.62
2nd order LDM	full F, G	3,756,610	4.06	2.61
1st order LDM	–	3,174,600	4.08	2.61
Autoregressive HMM	–	1,058,200	4.43	2.53
Feed-forward NN	4 layers × 1024 units	3,819,560	3.71	2.68
Feed-forward NN	6 layers × 512 units	1,648,680	3.71	2.69
Hybrid LSTM + FF	512 FF + 384 × 3 LSTM + 512 FF units	4,070,440	**3.62**	**2.73**

Table 4.1: Comparison of various acoustic models. No. param. is the number of parameters in the acoustic model.

We did not include HMMs in the comparison because it has already been shown [Tsiaras et al., 2014, 2015] that LDMs perform better than HMMs. Besides, we did not use dynamic features for any of the models, while HMMs require dynamic features to synthesize decent-quality speech. The initial results are shown in table 4.1.

Amongst the non-neural-network models, all the LDM-based systems perform nearly identically. The SO-LDM$_\mathrm{d}$ based system has the least number of parameters but performs at least as well as the others. Autoregressive HMMs have the least number of parameters overall but their performance is worse than that of LDMs, even though the clustering used in all the non-neural-network models was derived using autoregressive HMMs, thus giving them an edge over the others. Figure 4.1 compares the trajectories of the 2nd Mel-cepstral coefficient in part of an utterance generated by various state-space models. The trajectories generated by the variations of LDMs are nearly the same. The trajectory generated by the autoregressive HMMs is a little different from the others, and seems piece-wise linear. The faster variations in the original trajectory have not been captured by any of the models. This is true in case of neural networks as well.

Neural networks perform better than the LDMs. This is expected because they are more flexible in modeling non-linearities. Neural networks learn distributed representations whereas in LDMs, a separate LDM is learned for each cluster-state. However, the working memory required by LDMs is much smaller than by neural networks because they involve two small matrix-vector products per time step. In neural networks, the number of computations is much larger, and generation of parameters for each time step involves using all the neural network parameters. This will not be a problem in systems with adequate resources but in devices with limited working memory, LDMs are much easier to use. In this experiment we did not try to restrict the number of parameters in any of the models. Figure 4.2 compares the trajectory generated by SO-LDM$_\mathrm{d}$ with the ones generated by the two neural network models. The trajectory generated by the hybrid (LSTM + feed-forward) model tracks the original trajectory most closely. In some of the regions, for example around 350 ms, all the generated trajectories differ significantly from the original one. In general, the LDM-generated trajectory is quite similar to the one generated by the neural-networks. One of the reasons for the better performance of neural networks might be that the neural networks were trained using penta-phone labels, thus more fine grained linguistic context information was supplied to the model at any time-step. LDMs do not benefit from segmenting the phone into 5 subphones rather than 3 because then each sub-segment will be too short to take advantage of the dynamics of the LDMs.

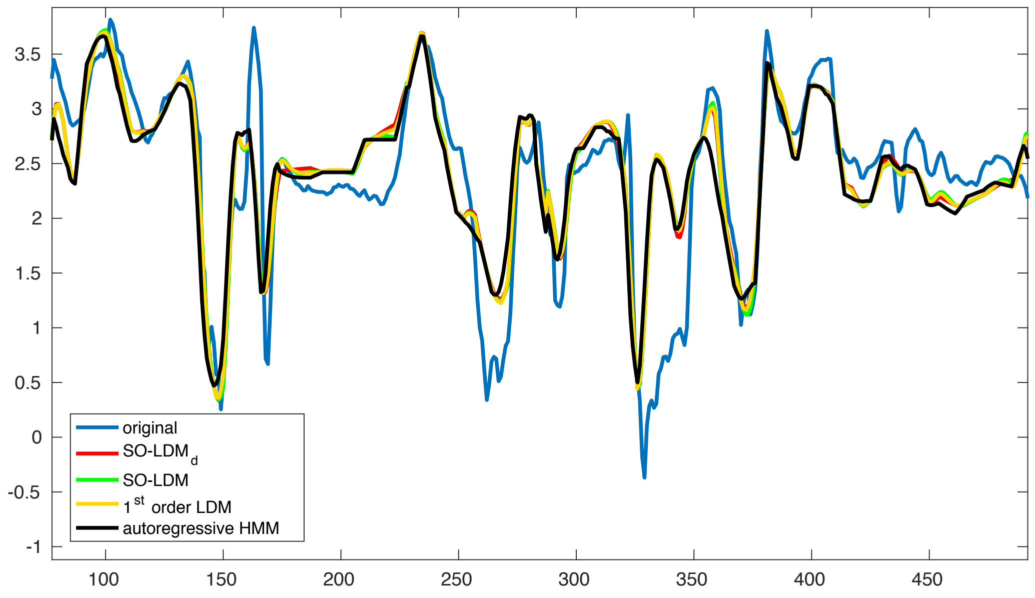

Figure 4.1: Trajectories of 2nd Mel-cepstral coefficient in part of an utterance generated by various types of state space models

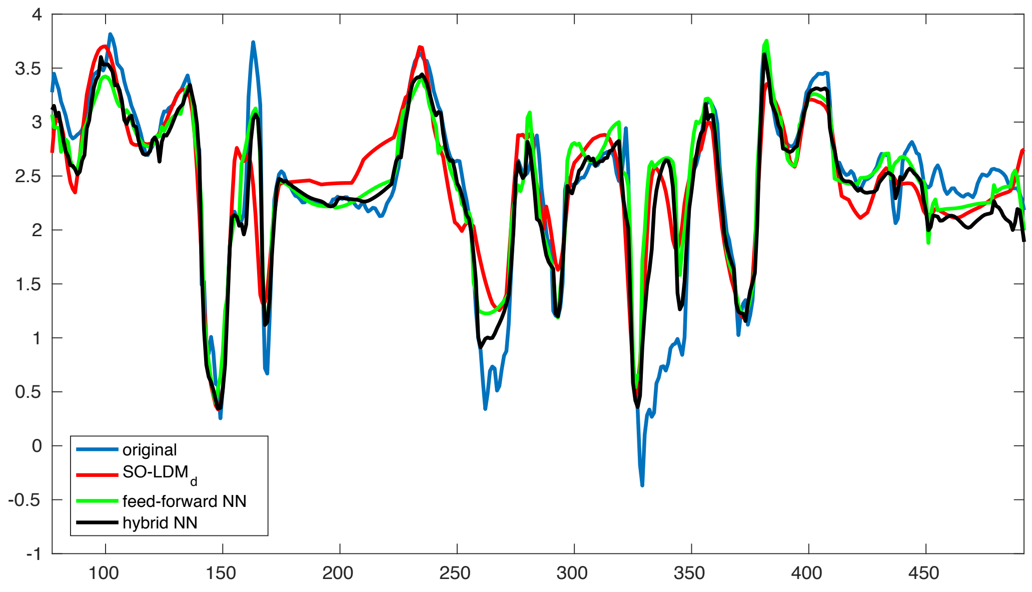

Figure 4.2: Comparison of trajectory of 2nd Mel-cepstral coefficient in part of an utterance generated by SO-LDM$_\mathrm{d}$ with the ones generated by neural networks

Model	No. clusters/layers	No. Param.	MCD	PESQ
SO-LDM$_\mathrm{d}$	5291	2,804,230	4.06	2.62
SO-LDM$_\mathrm{d}$, tied H	5291	688,230	4.08	2.60
SO-LDM$_\mathrm{d}$	1937	1,026,610	4.14	2.59
SO-LDM$_\mathrm{d}$	1264	669,920	4.21	2.57
SO-LDM$_\mathrm{d}$, tied H	1264	164,720	4.34	2.50
Feed-forward NN	6 layers $\times$ 512 units	1,648,680	3.71	2.69
Feed-forward NN	4 layers $\times$ 512 units	1,123,368	3.76	2.68
Feed-forward NN	2 layers $\times$ 512 units	598,056	3.94	2.59

Table 4.2: Variation of MCD and PESQ on varying the number of parameters in the models.

4.2.2 Clustering & Parameter Tying

There are two major ways of reducing the number of parameters in LDM-based TTS. The first one is to reduce the number of clusters in each decision tree. The other one is to tie the projection matrix H for all the LDMs. We tried both methods. The results are presented in table 4.2. In order to compare to neural networks with a similar number of parameters, we include the results for feed-forward neural networks of various sizes.

It is observed that reducing the parameters by tying H is less damaging than reducing the number of clusters. The number of parameters with 5291 clusters but tied H are nearly the same as with 1264 clusters, but the former performs better in terms of MCD as well as PESQ. The MCD for neural-networks is still lesser than the LDMs after reducing the number of parameters. The PESQ scores, however,

Model	No. Param.	MCD	PESQ
SO-LDM$_\mathrm{d}$	3,758,760	3.96	2.66
SO-LDM	5,035,320	3.94	2.68
1st order LDM	4,255,200	3.96	2.66

Table 4.3: MCD and PESQ using SO-LDM$_\mathrm{d}$-based clustering.

become nearly identical to those of LDMs.

So far we have used autoregressive HMMs to perform clustering which took around one day to complete by parallelizing on 28 cores of an Intel Xeon E5-2470. This clustering will, however, be sub-optimal for LDMs. We clustered using SO-LDM$_\mathrm{d}$ as well. The number of clusters is not directly controlled, but rather are controlled by the stopping criteria of the clustering. A threshold criterion which resulted in 1264 clusters in autoregressive HMM based clustering, provided 7092 clusters in SO-LDM$_\mathrm{d}$ based clustering. The results are presented in table 4.3 The results are better than the ones presented in the previous sections for each model. However, it took two weeks to cluster using 14 cores of an Intel Xeon E7520. Nevertheless, it is recommended to use LDMs to cluster for LDMs.

4.2.3 Segmentation

In section 3.3, we discussed various methods of dividing a phone segment into sub-segments. In all the experiments described so far, we divided a phone segment into three equal-length sub-segments, which is a suboptimal segmentation method. We alternatively used the segmentation achieved by forced alignment to three-state left-to-right HMMs. Apart from these, we also attempted the a new method for segmentation introduced in section 3.3. The comparison of performance among these three SO-LDM$_\mathrm{d}$-based systems with different segmentation methods is shown in table 4.4. The clustering was done using autoregressive HMMs.

We can observe that HMM-based segmentation performs much better than the sub-optimal equal sized segmentation. The new method performed slightly better than the equal-sized segmentation but worse than the HMM-based segmentation on the test data. It was observed during training that, the log-likelihood of each phone segment increased after each iteration of resegmentation. On the training data, it performed much better than the other methods. The new segmentation overfits the training data and thus does not provide any advantage on the test data. This might be attributed to the relatively small size of our training data.

Segmentation	No. clusters	MCD	PESQ
3 equal segments	5291	4.06	2.62
tri-state HMM segmentation	3759	3.94	2.66
new segmentation method	5291	4.04	2.63

Table 4.4: MCD and PESQ on using three different segmentation methods with SO-LDM$_{\mathrm{d}}$ and autoregressive HMM based clustering.

Model	Details	MCD	PESQ
2nd order LDM	diagonal F, G	4.06	2.24
2nd order LDM	full F, G	4.06	2.24
1st order LDM	–	4.08	2.25
Autoregressive HMM	–	4.43	2.18
Feed-forward NN	6 layers $\times$ 512 units	3.70	2.42
Feed-forward NN	4 layers $\times$ 512 units	3.76	2.39
Feed-forward NN	3 layers $\times$ 512 units	3.81	2.36
Feed-forward NN	2 layers $\times$ 512 units	3.93	2.35

Table 4.5: Comparison of complete TTS systems with different acoustic models.

4.2.4 Complete synthesis

In all the experiments so far, only MCEPs were modelled by the acoustic models. In this section, we present the results when $\ln(\mathrm{f}_0)$ and BAP are synthesized as well. In the case of autoregressive HMMs and LDMs, we synthesized $\ln(\mathrm{f}_0)$ and BAP using two separate sets of autoregressive HMMs. The MCEPs for LDMs were obtained using the LDM clustering tree, i.e. the same as the one used in table 4.3. In neural-network-based synthesis, the same neural network predicts $\ln(\mathrm{f}_0)$ and BAP as well. Thus, the output vector's size is increased from 40 to 42. The results are presented in table 4.5.

We can observe that the performance of both the LDM-based and neural-network-based systems goes down because of the artificially generated prosody and band-aperiodicity features. The generated samples sound more monotonous and robotic. The performance of the LDMs, however, deteriorates more than the neural networks. It can be inferred that autoregressive HMMs are not powerful enough to model the prosody properly.

4.3 Conclusions

In this work, we investigated LDM-based speech synthesis. There is a limited amount of work in the literature concerning the use of LDMs for speech synthesis. It has, however, been shown that LDMs provide better speech than the HMMs. We introduced second-order LDMs, which have a second-order autoregressive state process and a factor-analysis observation process. It was shown that the critically damped motion of articulators for speech production leads to a second order LDM speech synthesis process, which is the primary motivation for using second-order LDMs. Furthermore, it was found that the second-order LDMs with diagonal transition matrices are sufficient to model speech. These models perform as well as the first-order LDMs but have a reduced number of parameters. Similar to HMMs, LDMs require decision-tree-based clustering, which maps the linguistic features to a particular LDM from the given set of LDMs. This results in one LDM per each decision-tree cluster. During feature generation, a single LDM is deployed per segment in the utterance. Thus, the working memory required by the LDMs is really small as they require two small matrix-vector products per time step. However, they require a comparable amount of total space to the neural networks to store the complete acoustic model. We looked into methods for reducing the number of parameters further. It was observed that tying the projection matrix H is a good way to do that. There has been no study in the literature on subphone segmentation for LDMs. It was observed that using the subphone segmentation obtained from HMMs leads to an improvement in the speech quality relative to an equal-length baseline. A new segmentation algorithm was introduced which did not perform very well because it overfits the training data.

The quality of LDM-based speech synthesis was compared to neural-network-based speech synthesis. Neural networks are better at acoustic modelling, as revealed by their lower MCD. Also, perceptually the speech obtained from LDMs is inferior to the one obtained using neural networks as LDM synthesized speech obtained lower PESQ scores. However, as mentioned earlier, LDMs require much less working memory so they are useful in low-memory devices. LDMs could potentially replace HMM-based speech synthesis because they have similar computational and memory requirements, but are superior in performance.

4.4 Future Work

There are numerous directions of further investigation, some of which are discussed in this section. Currently one of the weakest links in LDM-based speech synthesis is the usage of decision trees to map linguistic features to LDMs, which are then used to generate output features. Decision trees are inefficient at expressing dependencies such as XOR or parity. Expressing these sorts of relations will result in prohibitively large decision trees [Esmeir and Markovitch, 2007]. Also, they partition the data into small clusters, corresponding to the leaves of these decision trees, from which individual models are learned. This reduces the amount of training data for each cluster as well as for clustering the other contexts [Yu et al., 2011]. A very large tree, with a small amount of data per terminal node, leads to overfitting and degrades the quality of the synthesized speech. HMMs formerly suffered from a similar problem because they also used decision trees. In the case of HMMs, neural networks replaced the decision trees, which lead to a significant improvement in the speech quality [Zen et al., 2013]. For LDMs as well, a more powerful, distributed model, like a neural network, would perhaps be better suited to this mapping, thus creating a hybrid of LDMs and neural networks. This could also reduce training time by avoiding decision-tree clustering. However, this is not a trivial problem because if we replace a decision tree directly with a neural network, the objective of such a neural network is not clearly defined. In such an implementation, a neural network acts as a discriminator and picks up one LDM given the input linguistic features. The LDMs are themselves trained on the data that gets mapped to each individual LDM. Besides, there is no way to determine the number of LDMs required. As described in section 2.2.2, an LDM is a simplified RNN. Thus combining an LDM with a neural network might essentially lead to a hybrid neural network with feed-forward layers at the bottom and an RNN layer at the top. However, the advantages of LDMs such as simplified training using the EM algorithm will not be possible in such a hybrid. Furthermore, if the model is trained using stochastic gradient descent, then we are free to choose any type of neural-network configuration, without worrying whether the implementation is close to an LDM or not.

Subphone segmentation currently has not been properly optimized. The new segmentation algorithm proposed in this document could be further investigated. In theory, the algorithm should work because the log-likelihood is increased after each iteration. It overfits the training data, however. Thus the first step could be to try it with more training data, or to add some sort of regularization to avoid overfitting.

In our work, a duration model was not implemented. One possible duration model is the same as the one in HSMMs. However, it is not a very powerful model. Another possibility is a neural-network-based duration model. Though more complex, it could lead to better speech than the HSMM-based duration model. It was also observed that the autoregressive-HMM-based prosody and aperiodicity models did not perform very well. Alternatives to these need to be developed to create a good-quality, full-fledged LDM-based TTS system. LDMs could be useful in articulatory speech synthesis as LDMs could be good at modelling the dynamics of the articulators, and could learn the models from statistical data while respecting the physical constraints on the dynamics.

Bibliography

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

F. Charpentier and M. Stella. Diphone synthesis using an overlap-add technique for speech waveforms concatenation. In *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pages 2015–2018, Apr 1986. doi: 10.1109/ICASSP.1986.1168657.

K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D14-1179`.

L. Deng. A dynamic, feature-based approach to the interface between phonology and phonetics for speech modeling and recognition. *Speech Communication*, 24(4):299 – 323, 1998. ISSN 0167-6393. doi: https://doi.org/10.1016/S0167-6393(98)00023-5. URL `http://www.sciencedirect.com/science/article/pii/S0167639398000235`.

L. Deng. *Computational Models for Speech Production*, pages 199–213. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. ISBN 978-3-642-60087-6. doi: 10.1007/978-3-642-60087-6_20. URL `https://doi.org/10.1007/978-3-642-60087-6_20`.

V. Digalakis, J. R. Rohlicek, and M. Ostendorf. Ml estimation of a stochastic linear system with the em algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442, Oct 1993. ISSN 1063-6676. doi: 10.1109/89.242489.

S. Esmeir and S. Markovitch. Anytime learning of decision trees. *Journal of Machine Learning Research*, 8:891–933, 2007. URL `http://dl.acm.org/citation.cfm?id=1314531`.

S. Fitt and S. Isard. Synthesis of regional english using a keyword lexicon. In *Proceedings: Eurospeech 99*, pages 823–826, 1999.

M. Gales and S. Young. The application of hidden markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304, 2008. ISSN 1932-8346. doi: 10.1561/2000000004. URL `http://dx.doi.org/10.1561/2000000004`.

Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000. doi: 10.1162/089976600300015619. URL `https://doi.org/10.1162/089976600300015619`.

Y. Hu and P. C. Loizou. Evaluation of objective quality measures for speech enhancement. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):229–238, Jan 2008. ISSN 1558-7916. doi: 10.1109/TASL.2007.911054.

A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376 vol. 1, May 1996. doi: 10.1109/ICASSP.1996.541110.

Hurricane Corpus 2012. Data release for the Hurricane challenge. `http://www.cstr.ed.ac.uk/projects/hurricane/1/index.html`, 2012. Last accessed: 2017-11-17.

T. Kailath. *Linear estimation.* Prentice Hall, Upper Saddle River, N.J, 2000. ISBN 0130224642.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

O. Karaali, G. Corrigan, and I. Gerson. Speech Synthesis with Neural Networks. In *Proc. of World Congress on Neural Networks*, pages 45–50, September 1996.

P. Kenny, M. Lennig, and P. Mermelstein. A linear predictive HMM for vector-valued observations with applications to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(2):220–225, Feb 1990. ISSN 0096-3518. doi: 10.1109/29.103057.

R. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE Pacific Rim Conference on Communications Computers and Signal Processing*, volume 1, pages 125–128 vol.1, May 1993. doi: 10.1109/PACRIM.1993.407206.

M. Morise. D4c, a band-aperiodicity estimator for high-quality speech synthesis. *Speech Communication*, 84(Supplement C):57 – 65, 2016. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2016.09.001. URL `http://www.sciencedirect.com/science/article/pii/S0167639316300413`.

M. Morise, F. Yokomori, and K. Ozawa. World: A vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, E99.D(7):1877–1884, 2016. doi: 10.1587/transinf.2015EDP7457.

C. Quillen. Kalman filter based speech synthesis. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4618–4621, March 2010. doi: 10.1109/ICASSP.2010.5495547.

C. Quillen. Autoregressive HMM speech synthesis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4021–4024, March 2012. doi: 10.1109/ICASSP.2012.6288800.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989. ISSN 0018-9219. doi: 10.1109/5.18626.

H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, Jul 1984. ISSN 0018-9448. doi: 10.1109/TIT.1984.1056936.

A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra. Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, volume 2, pages 749–752 vol.2, 2001. doi: 10.1109/ICASSP.2001.941023.

S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345, 1999. doi: 10.1162/089976699300016674. URL `https://doi.org/10.1162/089976699300016674`.

E. L. Saltzman and K. G. Munhall. A dynamical approach to gestural patterning in speech production. *Ecological Psychology*, 1(4):333–382, 1989. doi: 10.1207/s15326969eco0104_2. URL `http://dx.doi.org/10.1207/s15326969eco0104_2`.

M. Shannon and W. Byrne. Autoregressive HMMs for speech synthesis. In *Intespeech 2009: 10th Annual Conference of the International Speech Communication Association*, September 2009.

M. Shannon and W. Byrne. Autoregressive clustering for HMM speech synthesis. In *Intespeech 2010: 11th Annual Conference of the International Speech Communication Association*, September 2010.

M. Shannon, H. Zen, and W. Byrne. Autoregressive models for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(3):587–597, March 2013. ISSN 1558-7916. doi: 10.1109/TASL.2012.2227740.

J. Sotelo, S. Mehri, K. Kumar, J. F. Santos, K. Kastner, A. Courville, and Y. Bengio. Char2wav: End-to-end speech synthesis. In *5th International Conference on Learning Representations*, April 2017.

SPTK 2016. Speech signal processing toolkit (sptk) version 3.10. `http://sp-tk.sourceforge.net/`, Dec 2016. Last accessed: 2017-11-17.

P. Taylor, A. W. Black, and R. Caley. The architecture of the festival speech synthesis system. In *IN THE THIRD ESCA WORKSHOP IN SPEECH SYNTHESIS*, pages 147–151, 1998.

K. Tokuda, T. Kobayashi, and S. Imai. Speech parameter generation from HMM using dynamic features. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 660–663 vol.1, May 1995. doi: 10.1109/ICASSP.1995.479684.

K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, volume 3, pages 1315–1318 vol.3, 2000. doi: 10.1109/ICASSP.2000.861820.

V. Tsiaras, R. Maia, V. Diakoloukas, Y. Stylianou, and V. Digalakis. Linear dynamical models in speech synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 300–304, May 2014. doi: 10.1109/ICASSP.2014.6853606.

V. Tsiaras, R. Maia, V. Diakoloukas, Y. Stylianou, and V. Digalakis. Towards a linear dynamical model based speech synthesizer. In *Intespeech 2015: 16th Annual Conference of the International Speech Communication Association*, September 2015.

V. Tsiaras, R. Maia, V. Diakoloukas, Y. Stylianou, and V. Digalakis. Global variance in speech synthesis with linear dynamical models. *IEEE Signal Processing Letters*, 23(8):1057–1061, Aug 2016. ISSN 1070-9908. doi: 10.1109/LSP.2016.2580672.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *ArXiv e-prints*, September 2016.

Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous. Tacotron: Towards End-to-End Speech Synthesis. *ArXiv e-prints*, March 2017.

C. Wellekens. Explicit time correlation in hidden markov models for speech recognition. In *ICASSP '87. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 12, pages 384–386, Apr 1987. doi: 10.1109/ICASSP.1987.1169614.

P. C. Woodland. Hidden markov models using vector linear prediction and discriminative output distributions. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 509–512 vol.1, Mar 1992. doi: 10.1109/ICASSP.1992.225860.

Z. Wu, O. Watts, and S. King. Merlin: An open source neural network speech synthesis system. In *9th ISCA Speech Synthesis Workshop (SSW9)*, Sunnyvale, CA, USA, September 2016.

S. J. Young, J. J. Odell, and P. C. Woodland. Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 307–312, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics. ISBN 1-55860-357-3. doi: 10.3115/1075812.1075885. URL `https://doi.org/10.3115/1075812.1075885`.

S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK, 2006.

K. Yu, H. Zen, F. Mairesse, and S. Young. Context adaptive training with factorized decision trees for HMM-based statistical parametric speech synthesis. *Speech Communication*, 53(6):914 – 923, 2011. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2011.03.003. URL `http://www.sciencedirect.com/science/article/pii/S0167639311000379`.

H. Ze, A. Senior, and M. Schuster. Statistical parametric speech synthesis using deep neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7962–7966, May 2013. doi: 10.1109/ICASSP.2013.6639215.

H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda. The HMM-based speech synthesis system (HTS) version 2.0. In *SSW*, 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 & Language*, 21 (1):153 – 173, 2007b. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2006.01.002. URL `http://www.sciencedirect.com/science/article/pii/S0885230806000052`.

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, May 2007c. ISSN 0916-8532. doi: 10.1093/ietisy/e90-d.5.825. URL `http://dx.doi.org/10.1093/ietisy/e90-d.5.825`.

H. Zen, K. Tokuda, and A. W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039 – 1064, 2009. ISSN 0167-6393. doi: https://doi.org/10.1016/j.specom.2009.04.004. URL `http://www.sciencedirect.com/science/article/pii/S0167639309000648`.

H. Zen, A. Senior, and M. Schuster. Statistical parametric speech synthesis using deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7962–7966, 2013.

Appendix

List of linguistic features used:

- features for current phone, previous phone, next phone, previous to previous phone, and next to next phone:
  - identity
  - is vowel
  - is consonant
  - is voiced consonant
  - is unvoiced consonant
  - is pulmonic consonant
  - is voiced pulmonic consonant
  - is unvoiced pulmonic consonant
  - is syllabic consonant
  - is plosive
  - is voiced plosive
  - is unvoiced plosive
  - is bilabial plosive
  - is coronal plosive
  - is velar plosive
  - is fricative
  - is voiced fricative
  - is unvoiced fricative
  - is labiodental fricative
  - is coronal fricative
  - is dental fricative
  - is alveolar fricative
  - is post-alveolar fricative
  - is approximant
  - is pulmonic approximant
  - is nasal
  - is pulmonic nasal
  - is syllabic nasal
  - is affricate
  - is lateral
  - is voiced lateral
  - is labial
  - is voiced labial
  - is unvoiced labial
  - is bilabial
  - is voiced bilabial
  - is unvoiced bilabial

- is pulmonic bilabial
- is pulmonic voiced bilabial
- is coronal
- is general voiced coronal
- is general unvoiced coronal
- is pulmonic coronal
- is voiced coronal
- is unvoiced coronal
- is dorsal
- is voiced dorsal
- is dorsal velar
- is labialized velar approximant
- is fortis consonant
- is lenis consonant
- is general front vowel
- is front vowel
- is front close vowel
- is front open-mid vowel
- is front near-open vowel
- is general front or near-front vowel
- is front or near-front vowel
- is general front or near-back vowel
- is front or near-back vowel
- is general or near-front vowel
- is near-front vowel
- is general central vowel
- is central vowel or rhotic consonant
- is central vowel
- is central mid vowel
- is central open-mid vowel
- is general near-back vowel
- is near-back vowel
- is general back vowel
- is back vowel
- is back open vowel
- is back open-mid vowel
- is back close vowel
- is back vowel or rhotic consonant
- is long vowel
- is long back vowel
- is long close vowel
- is long open-mid vowel
- is close vowel
- is near-close vowel
- is open-mid vowel
- is open vowel
- is open-mid near-close vowel
- is general open near-close vowel
- is open near-close vowel
- is open near-close mid vowel
- is near-close mid vowel
- is dipthong
- is rounded
- is unrounded

  - is unround-schwa
- vowel in the central syllable
  - identity
  - is general front vowel
  - is front vowel
  - is front close vowel
  - is front open-mid vowel
  - is front near-open vowel
  - is general front or near-front vowel
  - is front near-front vowel
  - is general front or near-back vowel
  - is front or near-back vowel
  - is general near-front vowel
  - is near-front vowel
  - is general central vowel
  - is central vowel or rhotic consonant
  - is central vowel
  - is central mid vowel
  - is central open-mid vowel
  - is general near-back vowel
  - is near-back vowel
  - is general back vowel
  - is back vowel
  - is back open vowel
  - is back open-mid vowel
  - is back close vowel
  - is back vowel or rhotic consonant
  - is long vowel
  - is long back vowel
  - is long close vowel
  - is long open-mid vowel
  - is close vowel
  - is near-close vowel
  - is open-mid vowel
  - is open vowel
  - is open-mid or near-close vowel
  - is general open or near-close vowel
  - is open or near-close vowel
  - is open, near-close or mid vowel
  - is near-close or mid vowel
  - is diphthong
  - is rounded
  - is unrounded
  - is unrounded-schwa
- part of speech of the current, left and right word
- position of phone in syllable from the front
- position of phone in syllable from the back
- left syllable stress
- left syllable accent
- left syllable number of phones
- current syllable stress
- current syllable accent
- current syllable number of phone
- position current syllable in word from front

- position of current syllable in word from back
- position of current syllable in phrase from front
- position of current syllable in phrase from back
- number of stressed syllables before current syllable in phrase
- number of stressed syllables after current syllable in phrase
- number of accented syllables before current syllable in phrase
- number of accented syllables after current syllable in phrase
- number of syllables from previous stressed syllable
- number of syllables from next stressed syllable
- number of syllables from previous accented syllable
- number of syllables from next accented syllable
- right syllable stress
- right syllable accent
- right syllable number of segments
- left word number of syllables
- current word number of syllables
- part of speech of current word
- number of content words before current word in current phrase
- number of content words after current word in current phrase
- number of words from previous content word
- number of words from next content word
- right word number of syllables
- left phrase number of syllables
- left phrase number of words
- current phrase number of syllables
- current phrase number of words
- part of speech of current phrase in utterance from front
- part of speech of current phrase in utterance from back
- right phrase number of syllables
- right phrase number of words
- number of syllables in utterance
- number of words in utterance
- number of phrases in utterance