DOCTORAL DISSERTATION

ACOUSTIC AND WAVEFORM MODELING FOR STATISTICAL SPEECH SYNTHESIS

(統計的音声合成のための音響・波形モデリング)

DOCTOR OF ENGINEERING

JUNE 2018

Takenori YOSHIMURA

Supervisor : Dr. Keiichi TOKUDA
Dr. Yoshihiko NANKAKU

Department of Scientific and Engineering Simulation
Nagoya Institute of Technology

Abstract

Text-to-speech (TTS) is the artificial generation of human speech from texts. Since speech is one of the most important ways of human communication, TTS systems have been widely used in many applications, e.g., bus announcements, smartphone applications, smart speakers, and speech-to-speech translation systems. With the spread of services using TTS systems, they are expected to generate synthetic speech with not only high naturalness but also various speaker characteristics, emotions, and speaking styles.

To achieve that goal, many researchers have been tackled the issues of TTS over the past decades. One of the most successful approaches is corpus-based statistical parametric speech synthesis (SPSS). In this approach, the relationship between the linguistic features extracted from texts and the acoustic features extracted from the corresponding speech waveforms is modeled by a statistical model. The major advantage is that the characteristics of synthetic speech can be easily controlled by manipulating the parameters of the statistical model. As the statistical model, the hidden Markov model (HMM) has been widely used thanks to the well-defined algorithms and its flexibility for modeling sequential data. One of the major problems of the HMM is that it cannot fully exploit large collections of heterogeneous speech data. In order to solve the problem, the factor analyzed HMM (FAHMM), which is a probabilistic version of *eigenvoice*, has been proposed. In the framework of the FAHMM, the speech characteristics can be altered by changing a low-dimensional variable, which is called factor, rather than very high-dimensional model parameters. It should be noted that the factor is automatically extracted through model training of the FAHMM. While the effectiveness of the SPSS methods has been shown in various experiments, novel deep-neural-network (DNN)-based TTS approaches have been recently proposed. They attempt to directly model the relationship between the linguistic features and the raw audio speech waveforms using a specially-designed neural network architecture. One of the most successful approaches is the WaveNet generative model. By using a stack of convolutional neural networks (CNNs), the WaveNet can capture long-term temporal dependencies in speech signals. The WaveNet model outperformed the state-of-the-art TTS systems in subjective evaluation tests.

In the paper, techniques for improving the naturalness of synthetic speech and controlling the speech characteristics are proposed. For controlling the speech characteristics, the model structure to appropriately model heterogeneous speech data is studied using the FAHMM. Although the model structure of the FAHMM has no constraints under its framework, a single binary decision tree is typically used so that a simple algorithm for building a model structure can be used. However, it would prevent to capture the complex speaker/emotion/speaking-style-dependent relationship between the linguistic and acoustic features. To flexibly model the relationship, a more complex multiple-tree structure is proposed. The multiple trees are grown simultaneously rather than sequentially because building trees one by one makes it difficult to find the optimal model structure. However, since the possible combinations of tree structures exponentially increase according to growing trees, simultaneously optimizing the model structures is computationally infeasible. In order to avoid the computational explosion of the optimization, two computational complexity reduction algorithms inspired by techniques used in HMM-based speech synthesis are introduced.

For improving the naturalness of synthetic speech, the quantization used in neural-network-based speech waveform synthesis is investigated. One of the key techniques of neural-network-based speech waveform synthesis is modeling speech signals composed of a set of discrete values instead of continuous ones. In other words, speech waveform modeling is formulated as a classification problem rather than as a regression one. This enables more flexible waveform modeling because a categorical distribution has no assumptions about the shape. Simple linear quantization or nonlinear quantization with μ-law companding is typically used to obtain the discrete-valued speech signals. However, the quantization scheme introduces white noise into the original signals. Since the white noise is uniformly distributed over the entire frequency range, the quantization noise is easily perceived by human listeners. This paper presents a quantization noise shaping method in which a time-variant mask derived from mel-cepstrum is applied to the white quantization noise. Since mel-cepstrum can be based on the human auditory system, some of the quantization noise should be difficult for a human listener to perceive.

Keywords: Speech synthesis, signal processing, hidden Markov model, WaveNet, factor analysis, mel-cepstrum

Abstract in Japanese

音声は我々にとって最も身近なコミュニケーション手段の一つであることから，近年，音声信号処理技術の発展に伴って，バスの運行案内やスマートフォンのアプリケーション，スマートスピーカーなど，音声を利用したサービスが増加してきている．このようなサービスにおいては，発話の内容が多岐に渡るため，人の発声を収録したものを再生するのではなく，人工的に作られた音声を用いることが多い．そのため，人間らしい自然な音声を合成するための音声合成システムの需要が高まってきている．また，合成音声を利用した円滑な意思疎通を実現するため，音声の個人性や感情，発話スタイルなどを自由に表現可能な音声合成技術の実現が期待されている．

音声合成の問題は，与えられた任意のテキストから，対応する音声を求めることである．この問題を解決するため，この数十年の間に様々な研究が成されてきた．最も代表的な手法の一つとして，テキストから抽出した言語特徴量と，音声波形から抽出した音響特徴量との関係を，統計モデルにより表現する統計的パラメトリック音声合成方式が挙げられる．ここで，言語特徴量とは，音声に影響を与える音素や品詞などの文脈的な情報を表し，音響特徴量とは，音声波形を良く表現するような低次元の特徴量を表す．統計モデルのパラメータを変化させることによって，声を混ぜる，声を真似る，声を作り出すといったことが容易に可能であることが，統計的パラメトリック音声合成方式の大きな利点の一つである．統計モデルとしては，隠れマルコフモデル (Hidden Markov Model; HMM) が広く用いられてきた．HMM は，時間とともに変動する観測系列を統計的な枠組みで扱うことができるため，音声のモデル化に適していると言える．しかし，HMM では，様々な話者，感情，発話スタイルを含む大量の音声データを有効利用することが難しく，極少量の音声データから音声合成システムを構築することが困難であるという問題がある．このような問題を解決する手法として，HMM を拡張した因子分析に基づく HMM (Factor Analyzed HMM; FAHMM) が提案されている．因子分析に基づく HMM では，音声の話者性や感情を制御する低次元のパラメータ（因子）をモデルに組み込んでおり，この因子を変化させることで，音声の多様性を制御できる枠組みとなっている．このような統計的パラメトリック音声合成方式の有用性が示されている一方で，ディープニューラルネットワーク (Deep Neural Network; DNN) と呼ばれるモデルを用いて，言語特徴量

と音声波形との関係を直接モデル化する手法が最近提案され，注目が集まっている．特に，WaveNet と呼ばれるモデルは，特殊な構造を持った多層の畳み込みニューラルネットワークを利用することで，長期の依存関係を持つ音声波形を効率的にモデル化しており，WaveNet から合成された音声は人間が発話した音声と比べてほとんど劣化が見られないことが報告されている．

このような背景を踏まえて，本論文では，合成音声の自然性の向上および多様性の制御の両方を目指す．まず，多様性においては，因子分析に基づく HMM のモデル構造に着目する．これまで，因子分析に基づく HMM のモデル構造は，学習アルゴリズムを簡単にするため，話者や感情に依存しないものとして捉えられていた．しかし，言語特徴量と音響特徴量の対応関係は明らかに話者や感情に依存するものと考えられるため，このようなモデル構造では，多様な音声データを精度良くモデル化することが困難であり，合成音声の品質の劣化に繋がると考えられる．そこで，言語特徴量と音響特徴量，および話者や感情の関係を柔軟に表現するためのモデル構造と，そのモデル構造を自動獲得するためのアルゴリズムを提案し，合成音声の品質の向上を目指す．提案法では，考慮すべきモデル構造の数が爆発的に増加するため，モデル構造を評価するために必要な計算量が年単位となり，そのままでは実現が困難である．そこで，計算量を大幅に削減するためのアルゴリズムを導入し，現実的な計算時間によるモデル構造の選択を実現する．

また，自然性の向上においては，WaveNet における量子化に着目する．WaveNet を始めとして，音声波形を直接モデル化する手法においては，音声波形に対して量子化を行うことで，音声波形のモデル化を回帰問題ではなく分類問題として捉えている．これによって，音声波形を柔軟にモデル化することが可能であることが実験によって確かめられている．しかし，量子化を行うことにより発生する量子化ノイズが白色であるため，量子化ノイズの特に高域の周波数成分が知覚されやすく，合成音声の自然性が低下する要因となっていた．この問題に対して，メルケプストラムに基づくプリフィルタリングおよびノイズシェーピング量子化法を提案する．プリフィルタリングおよびノイズシェーピングの操作はメルケプストラムを介して行うため，これらの効果は人間の聴覚特性に合ったものとなっており，量子化ノイズが音声によって効率良くマスクされ，知覚されにくくなることが期待される．

以上のように，本論文では，人間と人間，人間と機械との円滑なコミュニケーションを実現するために，合成音声の自然性の向上と，多様性を制御するための技術を提案し，評価実験によってその有効性を検証する．

Acknowledgement

First of all, I would like to express my sincere gratitude to Keiichi Tokuda, my advisor, for his support, encouragement, and guidance.

I would like to thank Akinobu Lee, Yoshihiko Nankaku, Keiichiro Oura, and Kei Hashimoto for their technical supports and helpful discussions. Special thanks go to all the members of Tokuda and Nankaku laboratories for their technical supports and encouragement. If somebody was missed among them, my work would not be completed. I would be remiss if I did not thank Natsuki Kuromiya, a secretary of the laboratory, for her kind assistance.

Finally, I would sincerely like to thank my family and my friends for their encouragement.

Contents

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

2 Text-to-speech synthesis **5**

2.1 Overview 5

2.2 Hidden Markov models 6

2.2.1 Definition 6

2.2.2 Forward-backward algorithm 8

2.2.3 Viterbi algorithm 10

2.2.4 Expectation-maximization algorithm 11

2.2.5 HMM-based speech synthesis system 12

2.3 Factor analyzed hidden Markov models 19

2.3.1 Eigenvoice 19

2.3.2 Factor analysis 20

2.3.3 Definition 21

2.3.4 Variational expectation-maximization algorithm 22

2.3.5 FAHMM-based speech synthesis system 26

2.4 Deep neural networks 26

2.4.1 Definition 26

2.4.2 Backpropagation algorithm 27

2.4.3 DNN-based speech synthesis system 28

2.4.4 WaveNet 29

3 Multiple decision tree-based context clustering for FAHMM 31

3.1 Multiple decision tree-based context clustering 31

3.1.1 Reducing computational complexity by tying noise covariance matrices 34

3.1.2 Reducing computational complexity by using matrix inversion lemma 35

3.2 Experiments 37

3.2.1 Experimental setups 37

3.2.2 Objective evaluation 38

3.2.3 Subjective evaluation 41

3.3 Summary 42

4 Mel-cepstrum-based quantization noise shaping 44

4.1 Quantization in raw audio waveform modeling 44

4.2 Mel-cepstral analysis 46

4.3 Mel-cepstrum-based quantization noise shaping 48

4.3.1 Implementation using FIR filter 48

4.3.2 Implementation using MLSA filter structure 50

4.3.3 Mel-cepstrum-based prefiltering 53

4.4 Experiments 54

4.4.1 Experimental setups 54

4.4.2 Objective evaluation 54
4.4.3 Subjective evaluation 55
4.5 Summary 58

5 Conclusions **60**

List of Publications **70**

Journal papers 70
International conference proceedings 70
Technical reports 71
Domestic conference proceedings 71

Appendix A Software **73**

List of Tables

2.1 An example of contexts used in HMM-based speech synthesis. 15

4.1 Methods compared by using MOS. 58

List of Figures

2.1 Typical framework of text-to-speech synthesis. 6

2.2 A left-to-right three-state HMM. 7

2.3 Overview of HMM-based speech synthesis system. 13

2.4 An example of the decision tree-based context clustering. 14

2.5 An example of the relationship between the static feature vector sequence and the speech parameter vector sequence in a matrix form. 18

2.6 Overview of FAHMM. 21

2.7 A three-layer feed-forward neural network. 27

2.8 A stack of dilated causal convolutional layers. 29

2.9 Overview of WaveNet architecture with residual block. 30

3.1 Model structure of FAHMM. 32

3.2 Decomposition of $\boldsymbol{G}$. 35

3.3 MCD vs. number of training speakers, under the condition that the number of training sentences is 500 and the number of basis vectors is 5. . . . 39

3.4 MCD vs. number of training sentences, under the condition that the number of training speakers is 100 and the number of basis vectors is 5. 39

3.5 MCD vs. number of basis vectors, under the condition that the number of training sentences is 2000 and the number of training speakers is 100. . . 40

3.6 MCD vs. number of adaptation sentences, under the condition that the number of training speakers, sentences, and basis vectors are 2000, 100, and 30, respectively. 40

3.7 Number of leaf nodes for each HMM state. 41

3.8 MOS of naturalness with 95% confidence intervals. 43

3.9 DMOS of speaker similarity with 95% confidence intervals. 43

4.1 Phase characteristic $\tilde{\omega}$ of the first order all-pass transfer function $\tilde{z}^{-1}$. . . . 46

4.2 Block diagram of quantization noise shaping. 47

4.3 Power spectra of spectral envelope $H(z)$ and filter $D^{\gamma}(z)$ $(\gamma = 0.4)$. . . . 47

4.4 Spectrogram of a speech waveform and its quantization noise. 49

4.5 Block diagram of quantization noise shaping using $S^{\gamma}(z)$ for approximation of $D^{\gamma}(z) - 1$. 50

4.6 Block diagram of basic filter $F(z)$ $(M = 3)$. 52

4.7 Block diagram of noise shaping filter $D(z) - 1$ $(L = 4)$. 52

4.8 Two-stage cascaded structure for noise shaping filter $D(z) - 1$. 52

4.9 Block diagram of quantization noise shaping with prefiltering. 53

4.10 MOS-LQO value of speech quantized using mel-cepstrum-based quantization noise shaping. 55

4.11 Relationship between MOS and MNRU values where target speech is quantized speech. 57

4.12 Relationship between MOS and MNRU values where target speech is generated by WaveNet model. 57

4.13 MOS on speech quality with 95% confidence intervals. 59

4.14 Result of preference test of speech quality with a 95% confidence interval. 59

A.1 HTS: http://hts.sp.nitech.ac.jp/ 73

A.2 SPTK: http://sp-tk.sourceforge.net/ 74

Chapter 1

Introduction

Since speech is one of the most important ways for human communication, a number of research topics for human-machine communication have been proposed. Text-to-speech (TTS), which is the artificial generation of human speech from text, is a fundamental technology for human-machine communication, and a practical way to improve our quality of life. In recent years, TTS synthesis systems are used in many application, e.g., bus announcements, smartphone applications, smart speakers, and speech-to-speech translation systems.

The goal of TTS synthesis is to let machines speak naturally like humans with various speaker characteristics. In the past decade, the two main approaches, i.e., statistical parametric [1, 2] and concatenative [3] synthesis, have been widely developed to achieve that goal, and the speech generated by the synthesizers has been gradually approaching human speech in terms of naturalness [4]. However, it is expected to synthesize speech with not only high naturalness but also specific speaker characteristics. The amount of speech data uttered by a specific speaker with specific style is typically not enough to build a high-quality TTS system. Thus, leveraging a large amount of speech data consisting of multiple conditions is the key idea to solve the problem of a lack of speech data of a specific condition, but it makes building TTS systems challenging due to their acoustic variation.

For exploiting large collections of heterogeneous speech data, many approaches have been proposed especially in the hidden Markov model (HMM) [1] framework because speaker characteristics can be easily altered by transforming its model parameters. Some of those approaches are based on the idea of a *speaker subspace*. They assume that speaker characteristics can be sufficiently captured in a low-dimensional subspace rather than in a very high-dimensional model parameter space. By estimating the low-dimensional weights of the basis vectors spanning the subspace, the desired voice can be created even if only a few utterances of the desired speaker are available. These subspace-based approaches have become sophisticated over the years, and the three main approaches are described as follows. 1) Interpolation [5, 6]: speaker characteristics are controlled by linearly interpolating several speaker-dependent models. Each of the speaker-dependent models can be viewed as the basis vectors. The speaker similarity of the synthetic speech generated by the interpolated model with the actual speaker strongly depends on the representative speakers modeled by the speaker-dependent models. 2) Eigenvoice [7, 8]: the basis vectors are obtained by applying principal component analysis (PCA) to a set of parameters of speaker-dependent models. Dimensionality reduction by PCA makes the model compact, and the resulting subspace can effectively capture speaker characteristics more than the interpolation-based method. 3) Probabilistic eigenvoice [9]: one of the critical drawbacks of the interpolation- and eigenvoice-based approaches is that training an acoustic model is independent of building a speaker subspace. That is, the basis vectors are estimated so that they can reconstruct the parameters of speaker-dependent models rather than speech data. To solve the problem, factor analyzed HMM (FAHMM) has been proposed [9] as a well-defined probabilistic version of eigenvoice. Although several subspace-based approaches, e.g., maximum likelihood eigenspace (MLES) [10], multiple-regression HMM (MRHMM) [11, 12], and cluster adaptive training (CAT) [13, 14], have been proposed, they can be viewed as the variants of FAHMM. The difference between FAHMM and those approaches will be discussed in the paper. FAHMM has been shown to be effective [9], but its structure has much room for improvement.

Considering various speaker-independent linguistic contexts is important to synthesize high-quality speech because it is well-known that spectral and prosodic features in human speech are affected by contextual factors such as lexical stress, pitch accent, tone, and part-of-speech information. Unfortunately, it is almost impossible to reliably estimate the basis vectors for all possible combinations of the contexts with a finite set of training data. Thus, the basis vectors must be shared within similar contexts based on their model structures representing the relation between linguistic contexts and their acoustic realization. Although the model structures have no constraints under the FAHMM framework, a single binary decision tree is typically used as the model structures [9] so that decision tree-based context clustering [15], which is widely used in HMM-based speech synthesis, can be used. However, since it can be seen that each of the basis vectors represents a representative speaker, they must be based on their own structure rather than the same one to capture the complex speaker-dependent relationship between linguistic and acoustic features. In this study, to flexibly model the relationship, a context clustering technique for building the multiple-tree structure of the basis vectors is proposed. The multiple trees are grown simultaneously rather than sequentially. Building trees one by one makes it difficult to find the optimal model structures because the basis vectors depend on each other. However, since the possible combinations of tree structures exponentially increase according to growing trees, simultaneously optimizing the model structures is computationally infeasible. To avoid the computational explosion of the optimization, computational complexity reduction algorithms inspired by techniques used in HMM-based speech synthesis [16, 17] are introduced. Since they significantly reduce computational cost, the proposed clustering can be considered a feasible technique.

Recent statistical parametric speech synthesis methods have achieved remarkable success in producing high-quality synthetic speech. Most of them statistically model the relationship between the linguistic features extracted from the text and the acoustic features extracted from the corresponding speech [1, 2]. Due to the difficulty of directly modeling raw audio waveforms having long-term temporal dependencies, the relationship between the acoustic features and the raw speech waveforms is externally represented using a vocoder [18, 19] based on expert knowledge for human speech production process. Although the vocoder can easily produce synthetic speech from the acoustic features, it inevitably introduces degradation in speech quality. Efforts to prevent this degradation include focusing on directly modeling speech waveforms using the power of neural network machine learning [20, 21]. The most successful approaches have been the WaveNet generative model [22] and the SampleRNN audio generation model [23]. They use a specially-designed convolutional or recurrent neural network to capture the long-term temporal dependencies in speech signals. The WaveNet model is well able to model raw audio waveforms and outperformed the best TTS systems in subjective evaluation tests.

One of the key techniques of neural-network-based speech waveform synthesis is modeling speech signals composed of a set of discrete values instead of continuous ones. In other words, speech waveform modeling is formulated as a classification problem rather than as a regression one. This enables more flexible waveform modeling because a categorical distribution has no assumptions about the shape. Simple linear quantization or nonlinear quantization with μ-law companding [24] is typically used to obtain the discrete-valued speech signals. Although reconstructed speech signals after quantization have been reported to sound very similar to the original ones [22,23], the quantization scheme introduces white noise into the original signals. Since the white noise is uniformly distributed over the entire frequency range, the quantization noise is easily perceived by human listeners.

One possible solution for overcoming the problem caused by the quantization is to use a noise shaping technique. Noise shaping techniques alter the spectral characteristics of the quantization noise so that the noise is masked by speech. Parameters representing the envelope of speech spectrum such as linear predictive coding (LPC) coefficients are used for noise shaping [25]. The parameters should accurately capture the spectral peaks and valleys in the low-frequency range because the human ear is more sensitive to the low-frequency range than to the higher frequency range. Mel-cepstral coefficients [26, 27] meet this requirement and are well suited for noise shaping.

This paper presents a quantization noise shaping method in which a time-variant mask derived from mel-cepstrum is applied to the white quantization noise. Since mel-cepstrum can be based on the human auditory system, some of the quantization noise should be difficult for a human listener to perceive. The noise shaping filter can be easily implemented with a finite impulse response (FIR) filter. However, this is computationally expensive because the impulse response of the noise shaping filter is infinite and must be calculated sample-by-sample. Another approach is to use the structure of the mel-log spectrum approximation (MLSA) filter [28]. Although the latter approach includes an approximation, the filter can be applied in a computationally very efficient manner, and the accuracy of the approximation is sufficient.

Chapter 2

Text-to-speech synthesis

2.1 Overview

TTS synthesis is the process of converting an arbitrary text to intelligible and natural-sounding speech. Figure 2.1 shows the typical frameworks of TTS.

In the past decades, sophisticated approaches have been proposed with the advancement of hardware technology. One of the most successful approaches is corpus-based HMM-based speech synthesis [1]. In this approach, the relationship between linguistic features and acoustic features are statistically modeled using HMMs (see Figure 2.1). The linguistic features, e.g., phoneme identities, lexical stress, pitch accent, tone, and part-of-speech information, are extracted from text by a text analyzer using natural language processing techniques. On the other hand, acoustic features are compact representation of speech waveform and are extracted by a vocoder based on expert knowledge for human speech production process. Due to the rapidly advancing of deep learning, HMMs have been gradually replaced by DNN [2]. Since DNN can flexibly capture the relationship between linguistic and acoustic features, DNN-based speech synthesis approaches achieved significant progress [29].

There are some attempts to integrate some components of a TTS synthesis system. For example, text analysis and acoustic modeling are simultaneously modeled in a unified framework [30–32]. Acoustic feature extraction and acoustic modeling are integrated using signal processing knowledge [20,21,33]. Most remarkable techniques are to directly predict raw audio speech waveform from linguistic features based on specially-designed neural networks [22, 23]. Since they avoid to use a vocoder, natural speech like humans can be generated.

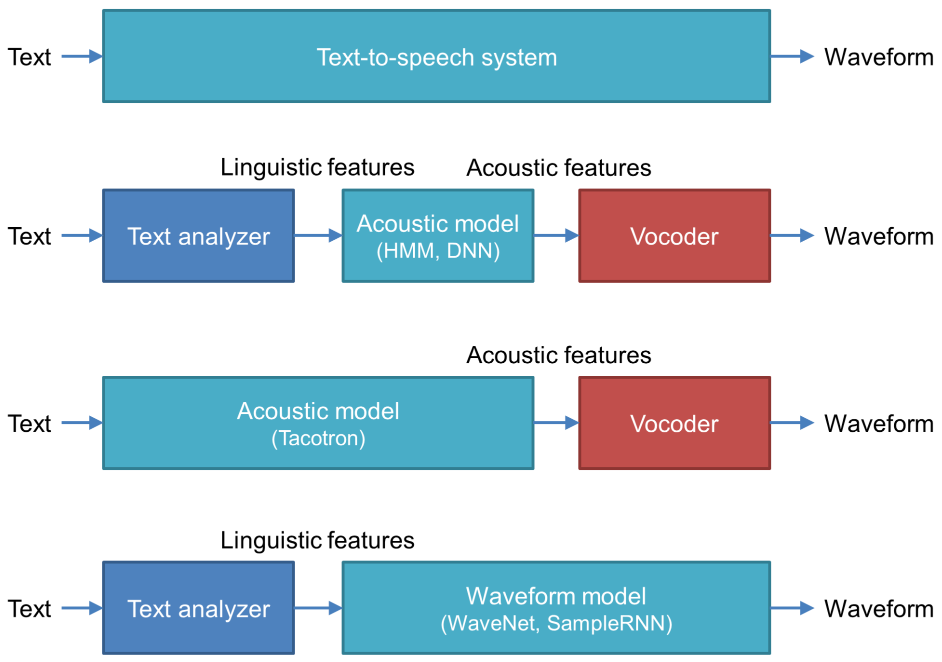


Figure 2.1: Typical framework of text-to-speech synthesis.

In the next subsections, models typically used in TTS synthesis systems are described in detail.

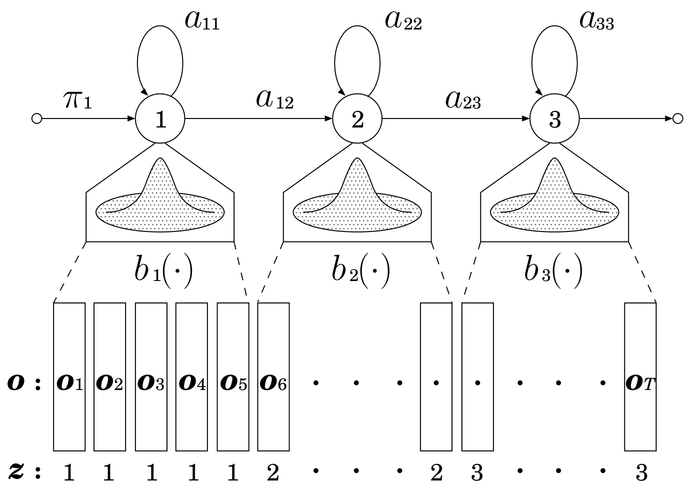
2.2 Hidden Markov models

2.2.1 Definition

A hidden Markov model (HMM) is a finite state machine that generates a sequence of discrete time observations [34–36]. The model parameters $\boldsymbol{\Lambda}$ of an N-state HMMs is represented as

$$\boldsymbol{\Lambda} = \{\boldsymbol{\Pi}, \boldsymbol{A}, \boldsymbol{B}\}\,, \tag{2.1}$$

where $\boldsymbol{\Pi} = \{\pi_i\}_{i=1}^{N}$ is a set of initial state probabilities, π_i is a probability of being in state i at time $t = 1$, $\boldsymbol{A} = \{a_{ij}\}_{i,j=1}^{N}$ is a set of state transition probabilities, a_{ij} is a probability of going from state i to state j, $\boldsymbol{B} = \{b_i(\cdot)\}_{i=1}^{N}$ is a set of output probability distributions, and $b_i(\boldsymbol{o}_t)$ is a probability producing an observation $\boldsymbol{o}_t \in \mathbb{R}^D$ at time t in state i. The



a_{11}
a_{22}
a_{33}
π_1
a_{12}
a_{23}
1
2
3
$b_1(\cdot)$
$b_2(\cdot)$
$b_3(\cdot)$
$\boldsymbol{O}$:
$\boldsymbol{o}_1$ $\boldsymbol{o}_2$ $\boldsymbol{o}_3$ $\boldsymbol{o}_4$ $\boldsymbol{o}_5$ $\boldsymbol{o}_6$ $\boldsymbol{o}_T$
$\boldsymbol{z}$: 1 1 1 1 1 2 2 3 3

Figure 2.2: A left-to-right three-state HMM.

model parameters $\boldsymbol{\Pi}$, $\boldsymbol{A}$, and $\boldsymbol{B}$ satisfy the following constraints:

$$\sum_{i=1}^{N} \pi_i = 1, \tag{2.2}$$

$$\forall i \quad \sum_{j=1}^{N} a_{ij} = 1, \tag{2.3}$$

$$\forall i \quad \int b_i(\boldsymbol{o}_t)\, d\boldsymbol{o}_t = 1, \tag{2.4}$$

respectively. The output probability distributions $\boldsymbol{B}$ can be discrete or continuous depending on the observations. In continuous distribution HMMs, each output probability distribution is usually modeled by a Gaussian distribution:

$$\begin{aligned} b_i(\boldsymbol{o}_t) &= \mathcal{N}\left(\boldsymbol{o}_t \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \\ &= \frac{1}{\sqrt{(2\pi)^D \left|\boldsymbol{\Sigma}_i\right|}} \exp\left(-\frac{1}{2}\left(\boldsymbol{o}_t - \boldsymbol{\mu}_i\right)^{\mathsf{T}} \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{o}_t - \boldsymbol{\mu}_i\right)\right), \end{aligned} \tag{2.5}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean vector and the covariance matrix of the Gaussian distribution, respectively. The covariance matrix $\boldsymbol{\Sigma}_i$ is often assumed to be diagonal.

Figure 2.2 shows a three-state left-to-right HMM without skips over states. The left-to-right model is useful for modeling temporal or sequential structures of stochastic signals such as speech signals. At each time, the HMM makes a transition to a state according to the state transition probability distribution, and then generates an observation $\boldsymbol{o}_t$ according to the output probability distribution of the current state.

Let us assume that an observation $\boldsymbol{o}_t$ is generated in a state $z_t \in \{1, 2, \dots, N\}$. The joint likelihood of an observation sequence $\boldsymbol{o} = [\ \boldsymbol{o}_1 \quad \boldsymbol{o}_2 \quad \cdots \quad \boldsymbol{o}_T\ ]$ and a state sequence $\boldsymbol{z} = [\ z_1 \quad z_2 \quad \cdots \quad z_T\ ]$ can be written as

$$\begin{aligned} p\left(\boldsymbol{o}, \boldsymbol{z} \,|\, \boldsymbol{\Lambda}\right) &= p\left(\boldsymbol{o} \,|\, \boldsymbol{z}, \boldsymbol{\Lambda}\right) p\left(\boldsymbol{z} \,|\, \boldsymbol{\Lambda}\right) \\ &= \left(\prod_{t=1}^{T} b_{z_t}\left(\boldsymbol{o}_t\right)\right) \left(\pi_{z_1} \prod_{t=1}^{T-1} a_{z_t z_{t+1}}\right). \end{aligned} \tag{2.6}$$

The total output probability of the observation sequence from the HMMs is calculated by marginalizing Eq. (2.6) over all possible state sequences:

$$\begin{aligned} p\left(\boldsymbol{o} \,|\, \boldsymbol{\Lambda}\right) &= \sum_{\text{all } \boldsymbol{z}} p\left(\boldsymbol{o}, \boldsymbol{z} \,|\, \boldsymbol{\Lambda}\right) \\ &= \sum_{\text{all } \boldsymbol{z}} \left(\prod_{t=1}^{T} b_{z_t}\left(\boldsymbol{o}_t\right)\right) \left(\pi_{z_1} \prod_{t=1}^{T-1} a_{z_t z_{t+1}}\right). \end{aligned} \tag{2.7}$$

The likelihood calculation requires $\mathcal{O}(T \cdot N^T)$ because at every time $t = 1, 2, \dots, T$ there are N possible states that can be reached, i.e., there are N^T possible state sequences. This calculation is computationally infeasible, even for small values of N and T. For example, when $T = 100$ and $N = 5$, the computational complexity is $\mathcal{O}(100 \cdot 5^{100}) \approx \mathcal{O}(10^{72})$. Fortunately, there is a very efficient algorithm to calculate Eq. (2.7) using forward and backward procedures.

2.2.2 Forward-backward algorithm

The forward-backward algorithm is widely used to calculate $p\left(\boldsymbol{o} \,|\, \boldsymbol{\Lambda}\right)$, which is the probability of an observation sequence $\boldsymbol{o}$ given the model parameters $\boldsymbol{\Lambda}$. The detail of the forward-backward algorithm is described in the following part.

Let us define the probability of a partial observation sequence from time 1 to t where $\boldsymbol{o}_t$ is generated in j-th state:

$$\alpha_j(t) = p\left(\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_t, z_t = j \,|\, \boldsymbol{\Lambda}\right). \tag{2.8}$$

The forward probability $\alpha_j(t)$ is recursively calculated as follows:

1. Initialization

$$\alpha_j(1) = \pi_j b_j(\boldsymbol{o}_1), \quad (1 \le j \le N) \tag{2.9}$$

2. Recursion

$$\alpha_j(t) \;=\; \left[\sum_{i=1}^{N} \alpha_i(t-1)a_{ij}\right] b_j(\boldsymbol{o}_t), \quad \begin{pmatrix} 1 \le j \le N \\ 1 < t \le T \end{pmatrix} \tag{2.10}$$

3. Termination

$$p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right) \;=\; \sum_{i=1}^{N} \alpha_i(T). \tag{2.11}$$

The computational complexity is $\mathcal{O}(T \cdot N^2)$. This is significant reduction from the method of exploring all the possible states with a complexity of $\mathcal{O}(T \cdot N^T)$. The left-to-right constraint can further reduce the computational cost.

As opposite to the forward probability, we can define the probability of a partial observation sequence from time T to $t+1$ where $\boldsymbol{o}_t$ is generated in i-th state:

$$\beta_i(t) \;=\; p\left(\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T \mid z_t = i, \boldsymbol{\Lambda}\right). \tag{2.12}$$

In the similar way as the forward algorithm, the backward probability $\beta_i(t)$ can be calculated in a recursive manner as follows:

1. Initialization

$$\beta_i(T) \;=\; a_{iN}, \quad (1 \le i \le N) \tag{2.13}$$

2. Recursion

$$\beta_i(t) \;=\; \sum_{j=1}^{N} a_{ij} b_j(\boldsymbol{o}_{t+1})\beta_j(t+1), \quad \begin{pmatrix} 1 \le i \le N \\ 1 \le t < T \end{pmatrix} \tag{2.14}$$

3. Termination

$$p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right) \;=\; \sum_{j=1}^{N} \pi_j b_j(\boldsymbol{o}_1)\beta_j(1). \tag{2.15}$$

The forward and backward probabilities can be used to compute the total output probability as follows:

$$\forall t \;\; p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right) \;=\; \sum_{i=1}^{N} \alpha_i(t)\beta_i(t). \tag{2.16}$$

The probabilities can also be used to compute the probability of being in state i at time t, and the probability of being state i at time t and state j at time $t+1$:

$$\begin{aligned} \gamma_i(t) &= p\left(z_t = i \,|\, \boldsymbol{o}, \boldsymbol{\Lambda}\right) \\ &= \frac{\alpha_i(t)\beta_i(t)}{\sum_{i'=1}^{N} \alpha_{i'}(t)\beta_{i'}(t)}, \end{aligned} \tag{2.17}$$

$$\begin{aligned} \xi_{ij}(t) &= p\left(z_t = i, z_{t+1} = j \,|\, \boldsymbol{o}, \boldsymbol{\Lambda}\right) \\ &= \frac{\alpha_i(t) a_{ij} b_j(\boldsymbol{o}_{t+1})\beta_j(t+1)}{\sum_{i'=1}^{N}\sum_{j'=1}^{N} \alpha_{i'}(t) a_{i'j'} b_{j'}(\boldsymbol{o}_{t+1})\beta_{j'}(t+1)}, \end{aligned} \tag{2.18}$$

where

$$\gamma_i(t) = \sum_{j=1}^{N} \xi_{ij}(t). \tag{2.19}$$

2.2.3 Viterbi algorithm

The problem to finding the most probable state sequence $\hat{\boldsymbol{z}} = [\ \hat{z}_1 \quad \hat{z}_2 \quad \cdots \quad \hat{z}_T\ ]$ given the model parameters $\boldsymbol{\Lambda}$ and an observation sequence $\boldsymbol{o}$ is called the decoding task. This problem is efficiently solved by the Viterbi algorithm [37] which is similar to the forward algorithm. Let $\delta_j(t)$ be the likelihood of the most likely state sequence ending in state j at time t:

$$\delta_j(t) = \max_{z_1, z_2, \ldots, z_{t-1}} p\left(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t, z_1, z_2, \ldots, z_{t-1}, z_t = j \,|\, \boldsymbol{\Lambda}\right), \tag{2.20}$$

and $\psi_j(t)$ be a trace-back pointer to store the best path information. Using these variables, the complete procedure for finding the most likely state sequence can be written as follows:

1. Initialization

$$\delta_j(1) = \pi_j b_j(\boldsymbol{o}_1), \quad (1 \le j \le N) \tag{2.21}$$

$$\psi_j(1) = 0, \quad (1 \le j \le N) \tag{2.22}$$

2. Recursion

$$\delta_j(t) = \max_i \left[\delta_i(t-1) a_{ij}\right] b_j(\boldsymbol{o}_t), \quad \begin{pmatrix} 1 \le j \le N \\ 1 < t \le T \end{pmatrix} \tag{2.23}$$

$$\phi_j(t) = \operatorname*{argmax}_i \left[\delta_i(t-1)a_{ij}\right] b_j(\boldsymbol{o}_t), \quad \begin{pmatrix} 1 \leq j \leq N \\ 1 < t \leq T \end{pmatrix} \tag{2.24}$$

3. Termination

$$\max_{\boldsymbol{z}} p\left(\boldsymbol{o}, \boldsymbol{z} \mid \boldsymbol{\Lambda}\right) = \max_i \delta_i(T), \tag{2.25}$$

$$\hat{z}_T = \operatorname*{argmax}_i \delta_i(T), \tag{2.26}$$

4. Backtracing

$$\hat{z}_t = \psi_{\hat{z}_{t+1}}(t+1). \quad (1 \leq t < T) \tag{2.27}$$

It should be noted that the Viterbi algorithm is identical to the forward algorithm except that it takes the maximum over the previous path probabilities whereas the forward algorithm takes the summation. Note also that the Viterbi algorithm has backpointers that is not used in the forward algorithm. This is because the Viterbi algorithm must produce not only a probability but also the most likely state sequence.

2.2.4 Expectation-maximization algorithm

For a given observation sequence $\boldsymbol{o}$, there is no known method to analytically obtain the optimal model parameters based on the maximum likelihood (ML) criterion, i.e., the $\boldsymbol{\Lambda}$ that globally maximizes the likelihood $p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right)$. This is because HMMs have hidden variables, $\boldsymbol{z}$. However, the model parameters $\boldsymbol{\Lambda}$ that *locally* maximizes the likelihood $p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right)$ can be obtained using an iterative procedure. The well-known procedure is the expectation-maximization (EM) algorithm [38]. The key idea behind the EM algorithm is to calculate the ML estimate for the incomplete data by using the complete data likelihood instead of the observed likelihood. The algorithm can appropriately estimate $\boldsymbol{\Lambda}$ if a good initial estimate is provided.

In the EM algorithm, an auxiliary function (so-called $\mathcal{Q}$-function), $\mathcal{Q}(\boldsymbol{\Lambda}, \hat{\boldsymbol{\Lambda}})$, of the current model parameters $\boldsymbol{\Lambda}$ and the new model parameters $\hat{\boldsymbol{\Lambda}}$ is defined as

$$\mathcal{Q}(\boldsymbol{\Lambda}, \hat{\boldsymbol{\Lambda}}) = \sum_{\text{all}\, \boldsymbol{z}} p\left(\boldsymbol{z} \mid \boldsymbol{o}, \boldsymbol{\Lambda}\right) \log p(\boldsymbol{o}, \boldsymbol{z} \mid \hat{\boldsymbol{\Lambda}}). \tag{2.28}$$

The EM algorithm starts with some initial model parameters and iterates between the following two steps:

$$\begin{aligned} &\text{E-step:} \quad && \text{compute } \mathcal{Q}(\boldsymbol{\Lambda}, \hat{\boldsymbol{\Lambda}}) \\ &\text{M-step:} \quad && \hat{\boldsymbol{\Lambda}} \leftarrow \operatorname*{argmax}_{\boldsymbol{\Lambda}} \mathcal{Q}(\boldsymbol{\Lambda}, \hat{\boldsymbol{\Lambda}}) \end{aligned}$$

The E-step computes the posterior probabilities of the hidden variables $p(\boldsymbol{z} \mid \boldsymbol{o}, \boldsymbol{\Lambda})$ while keeping the model parameters $\boldsymbol{\Lambda}$, and then computes $\mathcal{Q}(\boldsymbol{\Lambda}, \hat{\boldsymbol{\Lambda}})$. The M-step estimates the new model parameters $\hat{\boldsymbol{\Lambda}}$ by maximizing the $\mathcal{Q}$-function. Note that this iterative procedure is guaranteed to monotonically increase the observed data log likelihood at each iteration until it reaches a local maximum.

In HMM, the posterior probabilities of the state sequence $p(\boldsymbol{z} \mid \boldsymbol{o}, \boldsymbol{\Lambda})$ can be effectively computed by using the forward-backward algorithm as Eqs. (2.17) and (2.18). The new model parameters $\hat{\boldsymbol{\Lambda}}$ can be derived by using the method of Lagrange multipliers [39] to find the local maxima under the stochastic constraints shown in Eqs (2.2) and (2.3):

$$\hat{\pi}_i = \gamma_i(1), \tag{2.29}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}, \tag{2.30}$$

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{t=1}^{T} \gamma_i(t) \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_i(t)}, \tag{2.31}$$

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_i &= \frac{\operatorname{diag}\left[\sum_{t=1}^{T} \gamma_i(t) \left(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i\right) \left(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i\right)^{\mathsf{T}}\right]}{\sum_{t=1}^{T} \gamma_i(t)} \\ &= \frac{\operatorname{diag}\left[\sum_{t=1}^{T} \gamma_i(t)\, \boldsymbol{o}_t \boldsymbol{o}_t^{\mathsf{T}} - \hat{\boldsymbol{\mu}}_i \hat{\boldsymbol{\mu}}_i^{\mathsf{T}}\right]}{\sum_{t=1}^{T} \gamma_i(t)}. \end{aligned} \tag{2.32}$$

2.2.5 HMM-based speech synthesis system

Figure 2.3 shows the HMM-based speech synthesis system [40]. It consists of the training and synthesis parts. In the training part, acoustic features such as spectrum and fundamental frequency are extracted from speech waveform. The relationship between the extracted acoustic features and the linguistic features extracted from the corresponding text is mod-

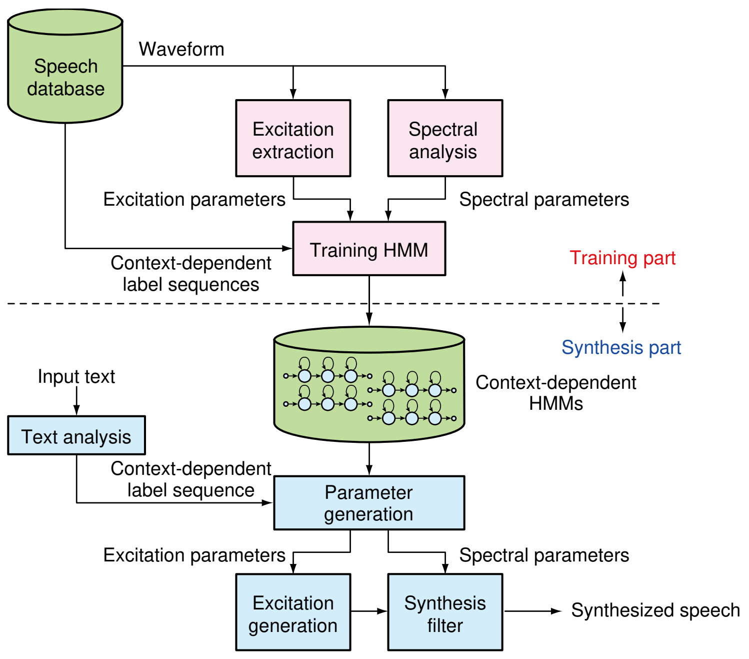


Figure 2.3: Overview of HMM-based speech synthesis system.

eled by context-dependent HMMs, which will be described later. In the synthesis part, a sentence HMM is constructed by concatenating the context-dependent HMMs from a given text to be synthesized. The sequences of spectrum and excitation parameters are generated from the sentence HMM using the well-established speech parameter generation algorithm [41–43]. Finally, speech waveform is synthesized using a synthesis filter such as mel-log spectrum approximation (MLSA) filter [28].

Context-dependent models

Considering various linguistic contexts is important to synthesize high-quality speech because it is well-known that spectral and prosodic features in human speech are affected by contextual factors, e.g., phoneme identities, lexical stress, pitch accent, tone, and part-of-speech information. Unfortunately, it is almost impossible to reliably estimate the HMM parameters for all possible combinations of the contexts with a finite set of training data. The decision tree-based context clustering [15, 44] can effectively sovle the problem.

Figure 2.4 shows an example of the decision tree-based context clustering. In this technique, top-down clustering is performed to locally maximize the likelihood of parameters with respect to training data using pre-defined linguistic questions about contexts. Table 2.1 shows an example of contexts used in an English HMM-based speech synthesis

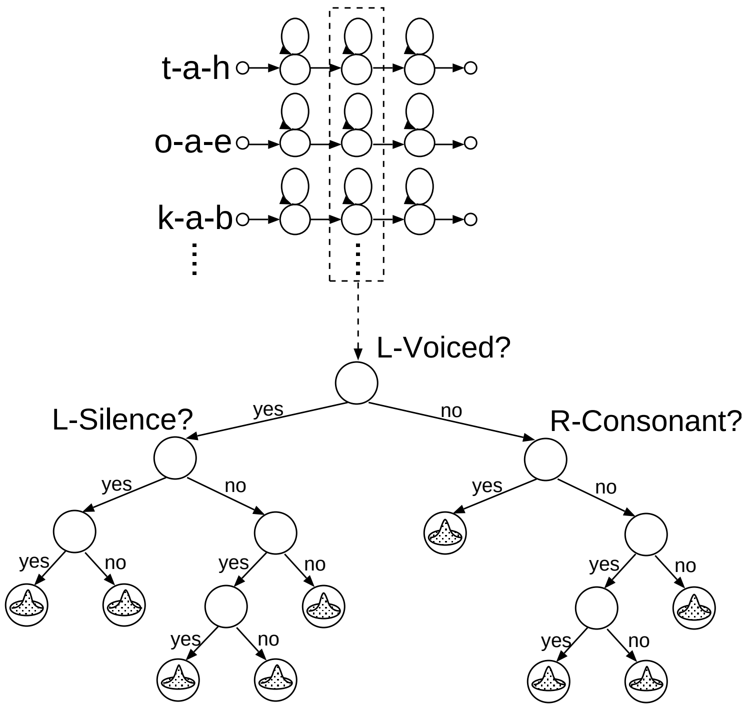


Figure 2.4: An example of the decision tree-based context clustering.

system. The total log likelihood to be maximized while clustring is represented as

$$\mathcal{L} = \sum_{t=1}^{T} \sum_{m \in \mathcal{M}} \gamma_m(t) \log \mathcal{N}\left(\boldsymbol{o}_t \mid \boldsymbol{\mu}_{f^{-1}(m)}, \boldsymbol{\Sigma}_{f^{-1}(m)}\right), \tag{2.33}$$

where $\mathcal{M}$ denotes a set of all contexts ($|\mathcal{M}|$ corresponds to the number of leaf nodes in the decision tree), $\gamma_m(t)$ is the state occupancy probability with respect to a context m, and $f^{-1}(\cdot)$ is the function gives a index of a leaf node that contains a context m. The state index is omitted for simplicity. The mean vectors and the covariance matrices of HMM states are shared on the same leaf node of the built tree, and reesimated using Eqs. (2.31) and (2.32). The following steps are the procedure for the decision tree-based context clustering algorithm:

Step 1. Create a root node.

Step 2. Evaluate all questions using Eq. (2.33) at the root node for the first time or two nodes created by the previous split.

Step 3. Select a pair of node and question that maximizes the likelihood, and then split the selected node into two by applying the selected question.

Step 4. If the change of likelihood after splitting the node falls below a predefined threshold, stop the procedure. Otherwise, go to **Step 2**.

In order to build the decision tress with less computational cost, the sufficient statistics $\gamma_m(t)$ is fixed in the clustering. Thus, this algorithm can be viewed as a kind of the EM algorithm without E-step while changing model structures. For automatically controlling the size of the decision tree, minimum description length (MDL) criterion [45] is widely used in HMM-based speech synthesis. The MDL criterion mainly consists of two terms: the first term is negative log likelihood used in the ML criterion, and the second one is a penalty imposed for having a large number of parameters.

Table 2.1: An example of contexts used in HMM-based speech synthesis.

the phoneme identity before the previous phoneme
the previous phoneme identity
the current phoneme identity
the next phoneme identity
the phoneme after the next phoneme identity
position of the current phoneme identity in the current syllable (forward)
position of the current phoneme identity in the current syllable (backward)
whether the previous syllable stressed or not (0: not stressed, 1: stressed)
whether the previous syllable accented or not (0: not accented, 1: accented)
the number of phonemes in the previous syllable
whether the current syllable stressed or not (0: not stressed, 1: stressed)
whether the current syllable accented or not (0: not accented, 1: accented)
the number of phonemes in the current syllable
position of the current syllable in the current word (forward)
position of the current syllable in the current word (backward)
position of the current syllable in the current phrase (forward)
position of the current syllable in the current phrase (backward)
the number of stressed syllables before the current syllable in the current phrase
the number of stressed syllables after the current syllable in the current phrase
the number of accented syllables before the current syllable in the current phrase
the number of accented syllables after the current syllable in the current phrase
the number of syllables from the previous stressed syllable to the current syllable
the number of syllables from the current syllable to the next stressed syllable
the number of syllables from the previous accented syllable to the current syllable
the number of syllables from the current syllable to the next accented syllable
name of the vowel of the current syllable
whether the next syllable stressed or not (0: not stressed, 1: stressed)

Continued on next page

whether the next syllable accented or not (0: not accented, 1: accented)
the number of phonemes in the next syllable
gpos (guess part-of-speech) of the previous word
the number of syllables in the previous word
gpos (guess part-of-speech) of the current word
the number of syllables in the current word
position of the current word in the current phrase (forward)
position of the current word in the current phrase (backward)
the number of content words before the current word in the current phrase
the number of content words after the current word in the current phrase
the number of words from the previous content word to the current word
the number of words from the current word to the next content word
gpos (guess part-of-speech) of the next word
the number of syllables in the next word
the number of syllables in the previous phrase
the number of words in the previous phrase
the number of syllables in the current phrase
the number of words in the current phrase
position of the current phrase in the utterance (forward)
position of the current phrase in the utterance (backward)
ToBI endtone of the current phrase
the number of syllables in the next phrase
the number of words in the next phrase
the number of syllables in the utterance
the number of words in the utterance
the number of phrases in the utterance

Speech parameter generation

For a sentence HMM $\boldsymbol{\Lambda}$, the problem of speech synthesis is to obtain an observation sequence consisted of spectral and excitation parameters $\hat{\boldsymbol{o}} = [\ \hat{\boldsymbol{o}}_1 \quad \hat{\boldsymbol{o}}_2 \quad \cdots \quad \hat{\boldsymbol{o}}_T\ ]$ that maximizes the posterior probability with respect to $\boldsymbol{o}$:

$$\hat{\boldsymbol{o}} \;=\; \underset{\boldsymbol{o}}{\operatorname{argmax}} \sum_{\text{all}\,\boldsymbol{z}} p\left(\boldsymbol{o} \,|\, \boldsymbol{z}, \boldsymbol{\Lambda}\right) p\left(\boldsymbol{z} \,|\, \boldsymbol{\Lambda}\right). \tag{2.34}$$

There is no method to analytically obtain $\boldsymbol{o}$ that maximizes $p\left(\boldsymbol{o} \,|\, \boldsymbol{\Lambda}\right)$ in a closed form. However, this problem is approximated by separating into two stages using the Viterbi approximation: finding the best state sequence $\boldsymbol{z}$ for given $\boldsymbol{\Lambda}$, and obtaining $\boldsymbol{o}$ that maximizes $p(\boldsymbol{o} \mid \boldsymbol{z}, \boldsymbol{\Lambda})$ with respect to $\boldsymbol{o}$, i.e.,

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}}{\operatorname{argmax}}\, p(\boldsymbol{z} \mid \boldsymbol{\Lambda}), \tag{2.35}$$

$$\hat{\boldsymbol{o}} = \underset{\boldsymbol{o}}{\operatorname{argmax}}\, p(\boldsymbol{o} \mid \hat{\boldsymbol{z}}, \boldsymbol{\Lambda}). \tag{2.36}$$

The optimization of Eq. (2.35) is performed using explicit state duration models [46, 47] in the HMM-based speech synthesis system.

If an observation $\boldsymbol{o}_t$ is independent from previous and next time, the predicted observation sequence $\hat{\boldsymbol{o}}$ would contain discontinuities at transition of HMM states. To avoid this problem, dynamic features [41] have been introduced. By letting a static feature vector be $\boldsymbol{c}_t \in \mathbb{R}^M$, the dynamic features, i.e., delta and delta-delta parameters, can be represented as

$$\Delta \boldsymbol{c}_t = \sum_{\tau=-J_-^{(1)}}^{J_+^{(1)}} w_\tau^{(1)} \boldsymbol{c}_{t+\tau}, \tag{2.37}$$

$$\Delta^2 \boldsymbol{c}_t = \sum_{\tau=-J_-^{(2)}}^{J_+^{(2)}} w_\tau^{(2)} \boldsymbol{c}_{t+\tau}, \tag{2.38}$$

where $w_\tau^{(\cdot)}$, $K_-^{(\cdot)}$ and $K_+^{(\cdot)}$ are a window coefficient, the left length of window, and the right length of window, respectively. The observation vector $\boldsymbol{o}_t \in \mathbb{R}^{3M}$ to be modeled consists of the static and dynamic features:

$$\boldsymbol{o}_t = \left[\begin{array}{ccc} \boldsymbol{c}_t^\top & \Delta \boldsymbol{c}_t^\top & \Delta^2 \boldsymbol{c}_t^\top \end{array} \right]^\top. \tag{2.39}$$

Conditions (2.37) and (2.38) can be represented in a matrix form (see Figure 2.5):

$$\boldsymbol{o} = \boldsymbol{W}\boldsymbol{c}, \tag{2.40}$$

where $\boldsymbol{o} = \left[\begin{array}{cccc} \boldsymbol{o}_1^\top & \boldsymbol{o}_2^\top & \cdots & \boldsymbol{o}_T^\top \end{array} \right]^\top$, $\boldsymbol{c} = \left[\begin{array}{cccc} \boldsymbol{c}_1^\top & \boldsymbol{c}_2^\top & \cdots & \boldsymbol{c}_T^\top \end{array} \right]^\top$, and $\boldsymbol{W}$ is a sparse regression window matrix given by

$$\boldsymbol{W} = \left[\begin{array}{cccc} \boldsymbol{W}_1 & \boldsymbol{W}_2 & \cdots & \boldsymbol{W}_T \end{array} \right]^\top \otimes \boldsymbol{I}_{M \times M}, \tag{2.41}$$

$$\boldsymbol{W}_t = \left[\begin{array}{ccc} \boldsymbol{w}_t^{(0)} & \boldsymbol{w}_t^{(1)} & \boldsymbol{w}_t^{(2)} \end{array} \right], \tag{2.42}$$

$$\boldsymbol{w}_t^{(0)} = \Big[\underbrace{0, \ldots, 0}_{t-1}, 1, \underbrace{0, \ldots, 0}_{T-t}\Big]^\top, \tag{2.43}$$

$$\boldsymbol{w}_t^{(1)} = \Big[\underbrace{0, \ldots, 0}_{t-J_-^{(1)}-1}, w_{-J_-^{(1)}}^{(1)}, \ldots, w_0^{(1)}, \ldots, w_{J_+^{(1)}}^{(1)}, \underbrace{0, \ldots, 0}_{T-\left(t+J_+^{(1)}\right)}\Big]^\top, \tag{2.44}$$

$$\boldsymbol{w}_t^{(2)} = \Big[\underbrace{0, \ldots, 0}_{t-J_-^{(2)}-1}, w_{-J_-^{(2)}}^{(2)}, \ldots, w_0^{(2)}, \ldots, w_{J_+^{(2)}}^{(2)}, \underbrace{0, \ldots, 0}_{T-\left(t+J_+^{(2)}\right)}\Big]^\top, \tag{2.45}$$

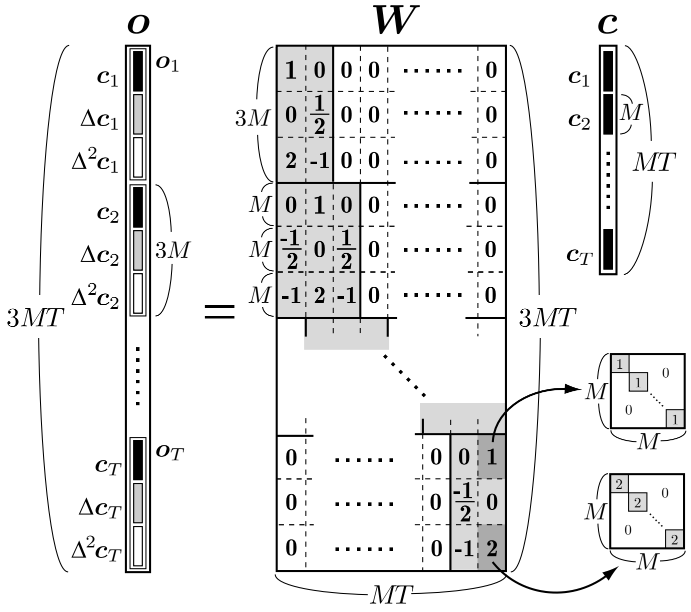


Figure 2.5: An example of the relationship between the static feature vector sequence and the speech parameter vector sequence in a matrix form.

where $\otimes$ denotes the Kronecker product. The probability of $\boldsymbol{o}$ conditioned on $\boldsymbol{z}$ is calculated by multiplying the output probabilities of entire observation vectors, and is represented as a single Gaussian component:

$$\begin{aligned} p(\boldsymbol{o} \mid \boldsymbol{z}, \boldsymbol{\Lambda}) &= \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{o}_t \mid \boldsymbol{\mu}_{z_t}, \boldsymbol{\Sigma}_{z_t}) \\ &= \mathcal{N}(\boldsymbol{o} \mid \boldsymbol{\mu}_{\boldsymbol{z}}, \boldsymbol{\Sigma}_{\boldsymbol{z}}), \end{aligned} \tag{2.46}$$

where $\boldsymbol{\mu}_{\boldsymbol{z}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{z}}$ are the supervector and the supermatrix corresponding to entire state sequence $\boldsymbol{z}$:

$$\boldsymbol{\mu}_{\boldsymbol{z}} = \begin{bmatrix} \boldsymbol{\mu}_{z_1}^{\top} & \boldsymbol{\mu}_{z_2}^{\top} & \cdots & \boldsymbol{\mu}_{z_T}^{\top} \end{bmatrix}^{\top}, \tag{2.47}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{z}} = \operatorname{diag}\begin{bmatrix} \boldsymbol{\Sigma}_{z_1} & \boldsymbol{\Sigma}_{z_2} & \cdots & \boldsymbol{\Sigma}_{z_T} \end{bmatrix}, \tag{2.48}$$

respectively. Under the constraint in Eq. (2.40), the static feature sequence $\hat{\boldsymbol{c}}$ that maximizes $p(\boldsymbol{o} \mid \boldsymbol{z}, \boldsymbol{\Lambda})$ can be derived by solving a set of linear equations:

$$\boldsymbol{R}_{\boldsymbol{z}} \boldsymbol{c} = \boldsymbol{r}_{\boldsymbol{z}}, \tag{2.49}$$

where

$$\boldsymbol{R}_{\boldsymbol{z}} = \boldsymbol{W}^{\top} \boldsymbol{\Sigma}_{\boldsymbol{z}}^{-1} \boldsymbol{W}, \tag{2.50}$$

$$\boldsymbol{r}_{\boldsymbol{z}} = \boldsymbol{W}^{\top} \boldsymbol{\Sigma}_{\boldsymbol{z}}^{-1} \boldsymbol{\mu}_{\boldsymbol{z}}. \tag{2.51}$$

As a result, a smoothly varying speech parameter trajectory is obtained. Although $\mathcal{O}(M^3T^3)$ operations are required for solving Eq. (2.49), the computational cost can be significantly reduced by using the Cholesky decomposition thanks to the symmetric band structure of $\boldsymbol{R}_{\boldsymbol{z}}$.

2.3 Factor analyzed hidden Markov models

2.3.1 Eigenvoice

One of the advantage of HMM-based speech synthesis system is that voice characteristics of synthesized speech can be easily modified by transforming HMM parameters. The well-known techniques include speaker adaptation [48], speaker interpolation [5, 6], and eigenvoice [8].

The eigenvoice method was originally proposed for very fast speaker adaptation in HMM-based speech recognition [7,10]. Then, it was introduced to HMM-based speech synthesis to overcome the problems with the interpolation method [6]. The basic idea behind the eigenvoice method is to find a small set of basis vectors from a diverse set of speaker's voices by assuming that speaker characteristics can be sufficiently represented as a point in a low-dimensional subspace rather than in a very high-dimensional model parameter space. Since a speaker's voice can be described as a linear combination of the basis vectors, a new voice with desired speaker characteristics is easily obtained by estimating the weights of the basis vectors from a small amount of speech data of the desired speaker.

With the eigenvoice method, R speaker-dependent HMM sets are first trained. Then, R supervectors are created by concatenating all mean parameters of the HMM set. By applying principal component analysis (PCA) to the mean-subtracted supervectors, $R-1$ eigenvectors are derived. Consequently, supervector $\hat{\boldsymbol{\mu}}$ for a new speaker is calculated from the first K eigenvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_K$ as

$$\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{\mu}} + \sum_{k=1}^{K} \omega_k \boldsymbol{e}_k, \tag{2.52}$$

where $\bar{\boldsymbol{\mu}}$ is the mean supervector of R supervectors and $\omega_1, \ldots, \omega_K$ are the weights representing speaker characteristics. The weights can be estimated from adaptation data based on the maximum likelihood criterion [7]. A new speaker-adapted HMM set is reconstructed from the generated supervector $\hat{\boldsymbol{\mu}}$.

The critical problems with the eigenvoice method are as follows: 1) The eigenvoice model accurately represents the model parameters of the representative HMM sets rather than the training speech data. 2) Speaker-dependent HMMs must be based on the same model structure to create fixed-dimensional supervectors. 3) Each representative HMM set should be well-trained using enough speech data to construct reliable supervectors. The first and second problems prevent to generate high-quality speech, and the third one makes difficult to build TTS systems when only a small amount of speech data is available for each training speaker. While the first and third problems have been solved by FAHMM described in this section, the second one, which is the very important part in acoustic modeling, is addressed in this study.

2.3.2 Factor analysis

Factor analysis (FA) is a statistical method for modeling the covariance structure of high-dimensional data by using a small number of latent variables. According to this model, an observation vector, $\boldsymbol{o} \in \mathbb{R}^D$, is generated as follows:

$$\boldsymbol{o} = \boldsymbol{L}\boldsymbol{x} + \boldsymbol{n}, \tag{2.53}$$

where $\boldsymbol{x} \in \mathbb{R}^Q$ is the factor that cannot be observed, and $\boldsymbol{n} \in \mathbb{R}^D$ is the noise vector. The relationship between each observation variable to the underlying factor is expressed by the so-called factor loading matrix, $L \in \mathbb{R}^{D \times Q}$, which is composed of Q basis vectors. Traditionally, the factor $\boldsymbol{x}$ and the noise vector $\boldsymbol{n}$ are assumed to be distributed according to a Gaussian distribution:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{2.54}$$

$$\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.55}$$

where $\boldsymbol{0}$ and $\boldsymbol{I}$ are a zero vector and an identity matrix, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the noise mean vector and the noise diagonal covariance matrix, respectively. In addition, it is usually assumed that factor $\boldsymbol{x}$ has a lower dimension than that of the observation vector $\boldsymbol{o}$ ($Q < D$). Thus, FA is basically a method for reducing the number of dimensions of the data by choosing the dimensionality of the subspace spanned by the basis vectors, in a similar way to PCA. The likelihood of the observation vector $\boldsymbol{o}$ given the factor $\boldsymbol{x}$ is

$$p(\boldsymbol{o} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{o} \mid \boldsymbol{L}\boldsymbol{x} + \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{2.56}$$

Furthermore, the joint likelihood of them is represented as

$$p(\boldsymbol{o}, \boldsymbol{x}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{o} \\ \boldsymbol{x} \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{L}\boldsymbol{L}^\top + \boldsymbol{\Sigma} & \boldsymbol{L} \\ \boldsymbol{L}^\top & \boldsymbol{I} \end{bmatrix}\right). \tag{2.57}$$

This equation shows that FA can be viewed as a model whose variance is constrained to be $\boldsymbol{L}\boldsymbol{L}^\top + \boldsymbol{\Sigma}$. Thus, the covariance structure of the observation vector can be represented using a smaller number of parameters than that of full covariance.

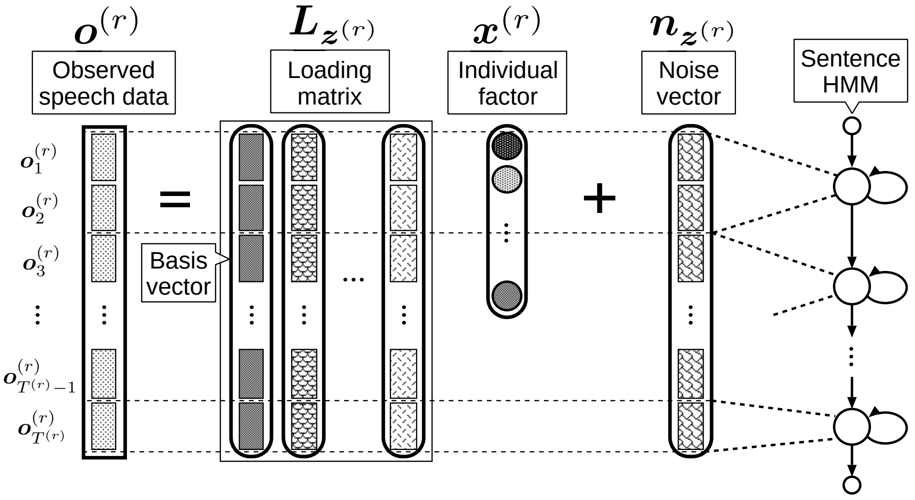


Figure 2.6: Overview of FAHMM.

2.3.3 Definition

The eigenvoice method has been reformulated as a probabilistic model for HMM-based speech synthesis. It is referred to as FAHMM because the generating process of observation sequences is based on FA. Figure 2.6 shows the overview of FAHMM. FAHMM assumes that an observation sequence $\boldsymbol{o}^{(r)}$ belonging to a speaker r is generated as

$$\boldsymbol{o}^{(r)} = \boldsymbol{L}_{\boldsymbol{z}^{(r)}} \boldsymbol{x}^{(r)} + \boldsymbol{n}_{\boldsymbol{z}^{(r)}}, \tag{2.58}$$

where

$$\boldsymbol{x}^{(r)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{2.59}$$
$$\boldsymbol{n}_{\boldsymbol{z}^{(r)}} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{z}^{(r)}}, \boldsymbol{\Sigma}_{\boldsymbol{z}^{(r)}}\right). \tag{2.60}$$

The loading matrix $\boldsymbol{L}_{\boldsymbol{z}^{(r)}}$ and the noise vector $\boldsymbol{n}_{\boldsymbol{z}^{(r)}}$ are composed by concatenating the parameters of context-dependent HMMs aligned according to a state sequence $\boldsymbol{z}^{(r)}$. Thus, FAHMM directly represents a variable-length utterance rather than a fixed-length supervector consisting of model parameters, unlike the eigenvoice method, Since the factor is shared among each speaker while the loading matrices and the noise vectors are shared among all speakers, speaker-dependent characteristics are automatically extracted as the factor through model training.

The joint log likelihood function of observation sequences $\boldsymbol{o} = \left\{\boldsymbol{o}^{(1)}, \boldsymbol{o}^{(2)}, \ldots, \boldsymbol{o}^{(R)}\right\}$, state sequences $\boldsymbol{z} = \left\{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(R)}\right\}$, and factors $\boldsymbol{x} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(R)}\right\}$ can be written as

$$\begin{aligned} p(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\Lambda}) &= p(\boldsymbol{o} \mid \boldsymbol{z}, \boldsymbol{x}, \boldsymbol{\Lambda})\, p(\boldsymbol{z} \mid \boldsymbol{\Lambda})\, p(\boldsymbol{x}) \\ &= \prod_{r=1}^{R} \left(\prod_{t=1}^{T^{(r)}} b_{z_t^{(r)}} \left(\boldsymbol{o}_t^{(r)} \right) \right) \left(\pi_{z_1^{(r)}} \prod_{t=1}^{T^{(r)}-1} a_{z_t^{(r)} z_{t+1}^{(r)}} \right) \cdot \mathcal{N}\left(\boldsymbol{x}^{(r)} \mid \boldsymbol{0}, \boldsymbol{I} \right), \end{aligned} \tag{2.61}$$

where

$$\begin{aligned} b_i\left(\boldsymbol{o}_t^{(r)}\right) &= \mathcal{N}\left(\boldsymbol{o}_t^{(r)} \,\middle|\, \boldsymbol{L}_i \boldsymbol{x}^{(r)} + \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \\ &= \mathcal{N}\left(\boldsymbol{o}_t^{(r)} \,\middle|\, \boldsymbol{W}_i \boldsymbol{\zeta}^{(r)}, \boldsymbol{\Sigma}_i\right), && (2.62) \\ \boldsymbol{W}_i &= \begin{bmatrix} \boldsymbol{\mu}_i & \boldsymbol{L}_i \end{bmatrix}, && (2.63) \\ \boldsymbol{\zeta}^{(r)} &= \begin{bmatrix} 1 \\ \boldsymbol{x}^{(r)} \end{bmatrix}, && (2.64) \end{aligned}$$

and $\boldsymbol{L}_i$, $\boldsymbol{\mu}_i$, and $\boldsymbol{\Sigma}_i$ are the factor loading matrix, the noise mean vector, and the noise diagonal covariance matrix in state i, respectively.

From Eq. (2.62), if $\boldsymbol{L}_i \boldsymbol{x}^{(r)} = \boldsymbol{0}$, FAHMM is clearly same as the standard HMM. This means that many techniques proposed in the standard HMM can be applied to FAHMM. Eqs (2.74) and (2.75) are similar to the maximum likelihood eigen-decomposition (MLED) estimator in the eigenvoice method [7]. However, in the MLED, the weight coefficients $\omega_1, \ldots, \omega_K$ are not random variables, and the basis vectors would not be updated. Although maximum likelihood eigenspace (MLES) [10] re-estimates the basis vectors based on the ML criterion, variance parameters are not considered. In the multiple-regression HMM (MRHMM) [11, 12], the auxiliary vectors, which correspond to the factors in FAHMM, are given and fixed through model training. Cluster adaptive training (CAT) [13, 14, 49] does not marginalize over the interpolation weights, which can be viewed as the factors in FAHMM, to obtain a likelihood function. In contrast to these models, FAHMM estimates the basis vectors and the factors in a unified framework. Thus, it is expected that FAHMM outperforms these approaches.

2.3.4 Variational expectation-maximization algorithm

In the model training of FAHMM, a set of model parameters $\boldsymbol{\Lambda}$ is estimated by maximizing the joint probability over the observation vectors by marginalization over the factors and the state sequences:

$$\hat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Lambda}}{\operatorname{argmax}} \prod_{r=1}^{R} \sum_{\text{all}\, \boldsymbol{z}^{(r)}} \int p\left(\boldsymbol{o}^{(r)}, \boldsymbol{z}^{(r)}, \boldsymbol{x}^{(r)} \,\middle|\, \boldsymbol{\Lambda}\right) d\boldsymbol{x}^{(r)}. \tag{2.65}$$

As a result, the observation vectors and the speaker subspace are simultaneously modeled within a unified framework. However, since the calculation of Eq. (2.65) requires averaging over all configurations of the two latent variables $\boldsymbol{x}$ and $\boldsymbol{z}$, directly computing the likelihood is computationally intractable. To solve that problem, the variational EM algorithm [50] can be used. According to this algorithm, the lower bound of log likelihood is maximized instead of true likelihood. The lower bound of the log likelihood of FAHMM, $\mathcal{F}$, is defined by using Jensen's inequality:

$$\begin{aligned} \log p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right) &= \log \sum_{\text{all } \boldsymbol{z}} \int q\left(\boldsymbol{z}, \boldsymbol{x}\right) \frac{p\left(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\Lambda}\right)}{q\left(\boldsymbol{z}, \boldsymbol{x}\right)} d\boldsymbol{x} \\ &\geq \sum_{\text{all } \boldsymbol{z}} \int q\left(\boldsymbol{z}, \boldsymbol{x}\right) \log \frac{p\left(\boldsymbol{o}, \boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{\Lambda}\right)}{q\left(\boldsymbol{z}, \boldsymbol{x}\right)} d\boldsymbol{x} \\ &\equiv \mathcal{F}, \end{aligned} \tag{2.66}$$

where $\boldsymbol{o} = \{\boldsymbol{o}^{(r)}\}_{r=1}^{R}$, $\boldsymbol{z} = \{\boldsymbol{z}^{(r)}\}_{r=1}^{R}$, $\boldsymbol{x} = \{\boldsymbol{x}^{(r)}\}_{r=1}^{R}$, and $q\left(\boldsymbol{z}, \boldsymbol{x}\right)$ is an arbitrary distribution. The relation between the true likelihood and the lower bound $\mathcal{F}$ is represented as

$$\begin{aligned} \log p\left(\boldsymbol{o} \mid \boldsymbol{\Lambda}\right) - \mathcal{F} &= \sum_{\text{all } \boldsymbol{z}} \int q\left(\boldsymbol{z}, \boldsymbol{x}\right) \log \frac{p\left(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{o}, \boldsymbol{\Lambda}\right)}{q\left(\boldsymbol{z}, \boldsymbol{x}\right)} d\boldsymbol{x} \\ &= D_{\text{KL}}(q\left(\boldsymbol{z}, \boldsymbol{x}\right) \,||\, p\left(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{o}, \boldsymbol{\Lambda}\right)), \end{aligned} \tag{2.67}$$

where the right-hand side is the Kullback-Leibler divergence (KLD) between the arbitrary distribution $q\left(\boldsymbol{z}, \boldsymbol{x}\right)$ and the true posterior distribution $p\left(\boldsymbol{z}, \boldsymbol{x} \mid \boldsymbol{o}, \boldsymbol{\Lambda}\right)$. It can be seen that maximizing the lower bound $\mathcal{F}$ is equivalent to minimizing the KLD between the two distributions. Thus, by maximizing the lower bound $\mathcal{F}$, the optimal (approximate) posterior distribution $q\left(\boldsymbol{z}, \boldsymbol{x}\right)$ can be estimated. To further relax the computational complexity, the latent variables are assumed to be conditionally independent given observation $\boldsymbol{o}$:

$$q\left(\boldsymbol{z}, \boldsymbol{x}\right) = q\left(\boldsymbol{z}\right) q\left(\boldsymbol{x}\right). \tag{2.68}$$

Under this assumption, the optimal posterior distributions that maximize the objective function $\mathcal{F}$ are given by using the variational method as

$$\begin{aligned} q\left(\boldsymbol{x}\right) &= \prod_{r=1}^{R} C_{\boldsymbol{x}^{(r)}} p\left(\boldsymbol{x}^{(r)}\right) \\ &\quad \times \exp \left\langle \log p\left(\boldsymbol{o}^{(r)} \,\middle|\, \boldsymbol{z}^{(r)}, \boldsymbol{x}^{(r)}, \boldsymbol{\Lambda}\right) \right\rangle_{q\left(\boldsymbol{z}^{(r)}\right)}, \end{aligned} \tag{2.69}$$

$$\begin{aligned} q\left(\boldsymbol{z}\right) &= \prod_{r=1}^{R} C_{\boldsymbol{z}^{(r)}} p\left(\boldsymbol{z}^{(r)} \mid \boldsymbol{\Lambda}\right) \\ &\quad \times \exp \left\langle \log p\left(\boldsymbol{o}^{(r)} \,\middle|\, \boldsymbol{z}^{(r)}, \boldsymbol{x}^{(r)}, \boldsymbol{\Lambda}\right) \right\rangle_{q\left(\boldsymbol{x}^{(r)}\right)}, \end{aligned} \tag{2.70}$$

where $\langle\cdot\rangle_{q(\cdot)}$ denotes the expectation with respect to distribution $q(\cdot)$, and $C_{\boldsymbol{x}^{(r)}}$ and $C_{\boldsymbol{z}^{(r)}}$ are the normalization terms that satisfy the following probabilistic constraints:

$$\sum_{\text{all}\,\boldsymbol{z}^{(r)}} q\left(\boldsymbol{z}^{(r)}\right) = 1, \tag{2.71}$$

$$\int q\left(\boldsymbol{x}^{(r)}\right) d\boldsymbol{x}^{(r)} = 1. \tag{2.72}$$

Since the obtained posterior distributions $q(\boldsymbol{x})$ and $q(\boldsymbol{z})$ depend on each other, they should be updated iteratively. The variational EM algorithm consists of two steps and guarantees that the value of an objective function monotonically increases at each iteration in the same manner as the standard EM algorithm [38]. In the E-step, sufficient statistics derived from $q\left(\boldsymbol{x}^{(r)}\right)$ and $q\left(\boldsymbol{z}^{(r)}\right)$ are calculated. Then, the set of model parameters $\boldsymbol{\Lambda}$ is updated to maximize the lower bound $\mathcal{F}$ in the M-step using the sufficient statistics.

As the product of two Gaussian distributions, the posterior distribution of factor $q(\boldsymbol{x}^{(r)})$ is also represented as a Gaussian:

$$q\left(\boldsymbol{x}^{(r)}\right) = \mathcal{N}\left(\boldsymbol{x}^{(r)} \,\middle|\, \hat{\boldsymbol{\mu}}_{\boldsymbol{x}^{(r)}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(r)}}\right), \tag{2.73}$$

where

$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(r)}} = \left(\boldsymbol{I} + \sum_{i=1}^{N}\sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t)\boldsymbol{L}_i^{\mathsf{T}}\boldsymbol{\Sigma}_i^{-1}\boldsymbol{L}_i\right)^{-1}, \tag{2.74}$$

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{x}^{(r)}} = \hat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(r)}} \left(\sum_{i=1}^{N}\sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t)\boldsymbol{L}_i^{\mathsf{T}}\boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{o}_t^{(r)} - \boldsymbol{\mu}_i\right)\right), \tag{2.75}$$

and $\gamma_i^{(r)}(t)$ is the state occupancy in HMM state i at time t given observation $\boldsymbol{o}^{(r)}$ which can be obtained by using the standard forward-backward algorithm:

$$\gamma_i^{(r)}(t) = \left\langle \delta_{iz_t^{(r)}} \right\rangle_{q\left(\boldsymbol{z}^{(r)}\right)}, \tag{2.76}$$

where $\delta_{iz_t^{(r)}}$ is the Kronecker delta:

$$\delta_{iz_t^{(r)}} = \begin{cases} 1, & \text{if } z_t^{(r)} = i \\ 0. & \text{if } z_t^{(r)} \neq i \end{cases} \tag{2.77}$$

The first- and second-order sufficient statistics of factor $\boldsymbol{x}^{(r)}$ can be written as

$$\langle \boldsymbol{\zeta}^{(r)} \rangle = \begin{bmatrix} 1 \\ \langle \boldsymbol{x}^{(r)} \rangle \end{bmatrix} = \begin{bmatrix} 1 \\ \hat{\boldsymbol{\mu}}_{\boldsymbol{x}^{(r)}} \end{bmatrix}, \tag{2.78}$$

$$\begin{aligned} \langle \boldsymbol{\zeta}^{(r)} \boldsymbol{\zeta}^{(r)\mathsf{T}} \rangle &= \begin{bmatrix} 1 & \langle \boldsymbol{x}^{(r)\mathsf{T}} \rangle \\ \langle \boldsymbol{x}^{(r)} \rangle & \langle \boldsymbol{x}^{(r)} \boldsymbol{x}^{(r)\mathsf{T}} \rangle \end{bmatrix} \\ &= \begin{bmatrix} 1 & \hat{\boldsymbol{\mu}}^{\mathsf{T}}_{\boldsymbol{x}^{(r)}} \\ \hat{\boldsymbol{\mu}}_{\boldsymbol{x}^{(r)}} & \hat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(r)}} + \hat{\boldsymbol{\mu}}_{\boldsymbol{x}^{(r)}} \hat{\boldsymbol{\mu}}^{\mathsf{T}}_{\boldsymbol{x}^{(r)}} \end{bmatrix}. \end{aligned} \tag{2.79}$$

Using these sufficient statistics and equating the partial derivative of $\mathcal{F}$ with respect to each of the model parameters, the following updating formulae are derived:

$$\hat{\pi}_i = \frac{\sum_{r=1}^{R} \gamma_i^{(r)}(1)}{\sum_{i'=1}^{N} \sum_{r=1}^{R} \gamma_{i'}^{(r)}(1)}, \tag{2.80}$$

$$\hat{a}_{ij} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}-1} \zeta_{ij}^{(r)}(t)}{\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}-1} \gamma_i^{(r)}(t)}, \tag{2.81}$$

$$\hat{\boldsymbol{W}}_i = \left(\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t) \boldsymbol{o}_t^{(r)} \left\langle \boldsymbol{\zeta}^{(r)\mathsf{T}} \right\rangle \right) \left(\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t) \left\langle \boldsymbol{\zeta}^{(r)} \boldsymbol{\zeta}^{(r)\mathsf{T}} \right\rangle \right)^{-1}, \tag{2.82}$$

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_i &= \frac{\operatorname{diag} \left[\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t) \left\langle \left(\boldsymbol{o}_t^{(r)} - \hat{\boldsymbol{W}}_i \boldsymbol{\zeta}^{(r)} \right) \left(\boldsymbol{o}_t^{(r)} - \hat{\boldsymbol{W}}_i \boldsymbol{\zeta}^{(r)} \right)^{\mathsf{T}} \right\rangle_{q\left(\boldsymbol{x}^{(r)}\right)} \right]}{\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t)} \\ &= \frac{\operatorname{diag} \left[\left(\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t) \boldsymbol{o}_t^{(r)} \boldsymbol{o}_t^{(r)\mathsf{T}} - \sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t) \boldsymbol{o}_t^{(r)} \left\langle \boldsymbol{\zeta}^{(r)\mathsf{T}} \right\rangle \hat{\boldsymbol{W}}_i^{\mathsf{T}} \right) \right]}{\sum_{r=1}^{R} \sum_{t=1}^{T^{(r)}} \gamma_i^{(r)}(t)}. \end{aligned} \tag{2.83}$$

2.3.5 FAHMM-based speech synthesis system

The framework of the FAHMM-based speech synthesis is similar to the HMM-based speech synthesis system described in Section 2.2.5. In the synthesis stage, using the trained model parameters $\boldsymbol{\Lambda}$ and given adaptation data $\boldsymbol{o}^{(r')}$ belonging to a new speaker r', the optimal parameter sequence, $\hat{\boldsymbol{o}}'$, is obtained as

$$\begin{aligned}\hat{\boldsymbol{o}}' &= \operatorname*{argmax}_{\boldsymbol{o}'} \sum_{\text{all}\,\boldsymbol{z}'} \int p\left(\boldsymbol{o}', \boldsymbol{z}', \boldsymbol{x}' \,\middle|\, \boldsymbol{o}^{(r')}, \boldsymbol{\Lambda}\right) d\boldsymbol{x}' \\ &= \operatorname*{argmax}_{\boldsymbol{o}'} \sum_{\text{all}\,\boldsymbol{z}'} \int p\left(\boldsymbol{o}' \,\middle|\, \boldsymbol{z}', \boldsymbol{x}', \boldsymbol{\Lambda}\right) p\left(\boldsymbol{z}' \,\middle|\, \boldsymbol{\Lambda}\right) p\left(\boldsymbol{x}' \,\middle|\, \boldsymbol{o}^{(r')}, \boldsymbol{\Lambda}\right) d\boldsymbol{x}'. \end{aligned} \tag{2.84}$$

Instead of the true posterior distribution, the approximate posterior distribution derived from the variational EM algorithm is used for the calculation in Eq. (2.84). Consequently, the suboptimal parameter sequence is calculated from

$$\hat{\boldsymbol{o}}' \approx \operatorname*{argmax}_{\boldsymbol{o}'} \sum_{\text{all}\,\boldsymbol{z}'} \int p\left(\boldsymbol{o}' \,\middle|\, \boldsymbol{z}', \boldsymbol{x}', \boldsymbol{\Lambda}\right) q\left(\boldsymbol{z}'\right) q\left(\boldsymbol{x}'\right) d\boldsymbol{x}'. \tag{2.85}$$

The above equation is further approximated using the Viterbi approximation in the same way as described in Section 2.2.5:

$$\hat{\boldsymbol{x}}' = \operatorname*{argmax}_{\boldsymbol{x}'} q\left(\boldsymbol{x}'\right), \tag{2.86}$$

$$\hat{\boldsymbol{z}}' = \operatorname*{argmax}_{\boldsymbol{z}'} q\left(\boldsymbol{z}'\right), \tag{2.87}$$

$$\hat{\boldsymbol{o}}' = \operatorname*{argmax}_{\boldsymbol{o}'} p\left(\boldsymbol{o}' \,\middle|\, \hat{\boldsymbol{z}}', \hat{\boldsymbol{x}}', \boldsymbol{\Lambda}\right). \tag{2.88}$$

2.4 Deep neural networks

2.4.1 Definition

A neural network is basically a sequence of operations applied to a matrix of input data. These operations usually consist of additions and multiplications followed by non-linear functions. Figure 2.7 shows a example of a simple neural network, which is called feed-forward neural network that has no cycle. As shown in the figure, the common type of neural network is composed of three groups: input layer, hidden layer, and output layer. Nodes in the input layer receive input data, and nodes in the output layer give output to the user. Nodes in the hidden layer(s) receive inputs from the previous layer and deliver their output to the next layer. Each unit in a layer is connected via a trainable weight to each

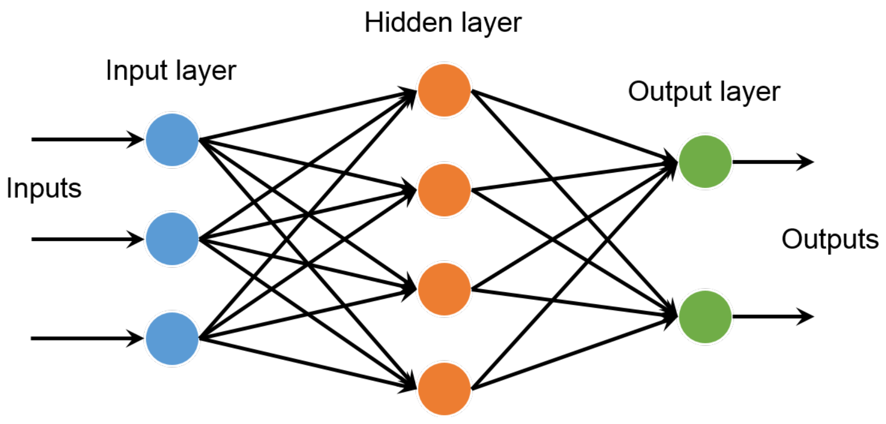


Figure 2.7: A three-layer feed-forward neural network.

unit in the next layer. The weights can be effectively updated through the well-known backpropagation algorithm described latter. The complex relationship between input and output data can be captured by stacking hidden layers followed by non-linear activation functions such as

1. Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}, \tag{2.89}$$

2. Hyperbolic tangent

$$\tanh(x) = 2\sigma(2x) - 1, \tag{2.90}$$

3. Rectified linear units (ReLU) [51]

$$ReLU(x) = \max(0, x)\,. \tag{2.91}$$

2.4.2 Backpropagation algorithm

The backpropagation is a common algorithm for learning of neural networks using gradient descent [52]. Given a neural network and a loss function to be minimized, the method calculates the gradient of the loss function with respect to the weights of the network. The basic concept behind backpropagation is to calculate error derivatives. After the forward pass through the network, the network output is compared to the true output and a prediction error is calculated. The prediction error is then propagated backwards through the network and changes are made to the weights in each layer. The cycle is repeated until the overall error value drops below some pre-determined threshold. After then, the network assumed to learn the relationship between input and output data well enough.

Standard gradient descent computes the gradient of the loss function with respect to the parameters and updates the parameters as

$$\mathbf{\Lambda} = \mathbf{\Lambda} - \eta \cdot \nabla_{\mathbf{\Lambda}} \mathcal{L}(\mathbf{\Lambda}), \tag{2.92}$$

where $\mathcal{L}(\mathbf{\Lambda})$ is an objective function parametrized by a set of model parameters $\mathbf{\Lambda}$ and η is a learning late that determines the size of the steps to reach a minimum. Recently, the update rule has been sophisticated [53, 54], resulting in stabilization and speedup of convergence in training neural networks.

Letting true and predicted outputs be y_i and $\hat{y}_i$ respectively, the following objective functions are usually used:

1. Squared error

$$\mathcal{L}(\mathbf{\Lambda}) = \sum_i (y_i - \hat{y}_i)^2, \tag{2.93}$$

2. Absolute error

$$\mathcal{L}(\mathbf{\Lambda}) = \sum_i |y_i - \hat{y}_i| \tag{2.94}$$

3. Cross entropy

$$\mathcal{L}(\mathbf{\Lambda}) = -\sum_i y_i \log \hat{y}_i. \tag{2.95}$$

2.4.3 DNN-based speech synthesis system

Thanks to the power of DNN, DNN has been introduced in some of the components of TTS synthesis, e.g., F0 prediction, duration prediction, grapheme-to-phoneme [55], and postfilter [56]. One of the most successful applications is acoustic modeling [2,57] where DNN is trained to represent the mapping function from linguistic features to acoustic features, which is typically modeled by a decision tree in HMM-based synthesis systems. The acoustic modeling using DNN has been studied and proposed using special structure such as mixture density and recurrent [58,59] for several years. However, novel generative models that directly model raw audio speech waveform and linguistic features has been recently proposed. The WaveNet model is well able to model raw audio waveforms and outperformed the best TTS systems in subjective evaluation tests. In the next subsection, the WaveNet generative model is briefly described.

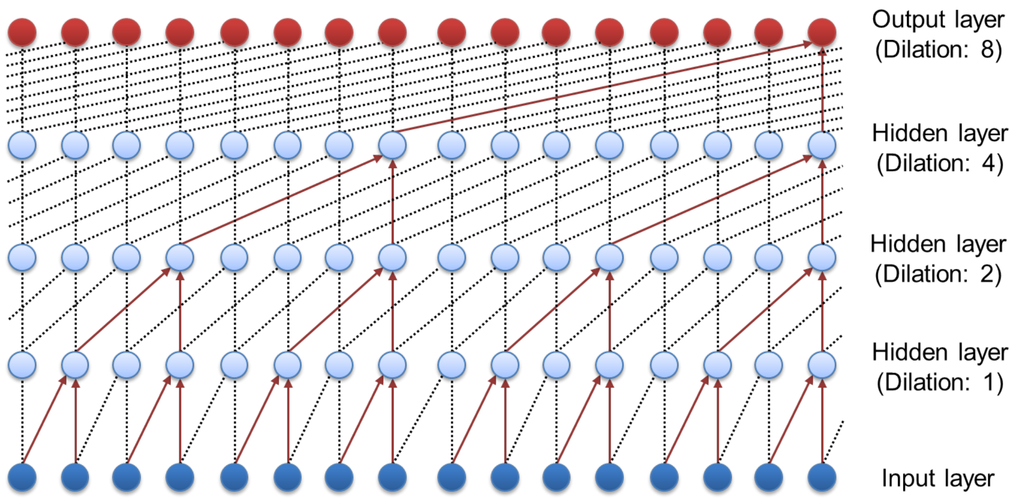


Figure 2.8: A stack of dilated causal convolutional layers.

2.4.4 WaveNet

WaveNet is a neural network that directly generates audio waveform signals. It mainly consists a specially-designed convolutional layers, i.e., dilated causal convolutional layers. Figure 2.8 shows a stack of dilated causal convolutional layers. In causal convolutions, the current output of networks is not depend on future timesteps. This constraint is suitable to represent autoregressive models. One of the problems of causal convolutions is that they require many layers to increase the receptive field. A dilated convolution is a convolution applied to input with certain gaps. By stacking dilated convolutions, very large receptive fields can be obtained with relatively few parameters. As shown in Figure 2.8, the dilation is doubled for every layer in the WaveNet. This results in exponential receptive field growth with depth.

The entire architecture of WaveNet is shown in Figure 2.9 where the 1×1 block means a one by one convolution. It can be seen a nonlinear predictive analysis with a very large structure. First, input waveform signals are fed into a causal convolutional layers, and then they pass through several dozens of *residual blocks*. The outputs of the residual blocks are pooled and nonlinearly transformed by ReLU. Finally, the conditional discrete probability distributions are obtained by a softmax output layer. The residual blocks consists of two inputs, i.e., outputs of previous layer and auxiliary features, and two outputs for residual connections and skip connections. The auxiliary features $\boldsymbol{h}$ affects to the dilated outputs of previous layer $\boldsymbol{s}$ in gated activation units:

$$\boldsymbol{z} = \tanh\left(\boldsymbol{W}^{(f)} * \boldsymbol{s} + \boldsymbol{V}^{(f)} * \boldsymbol{y}\right) \odot \sigma\left(\boldsymbol{W}^{(g)} * \boldsymbol{s} + \boldsymbol{V}^{(g)} * \boldsymbol{y}\right), \tag{2.96}$$

$$\boldsymbol{y} = u\left(\boldsymbol{h}\right), \tag{2.97}$$

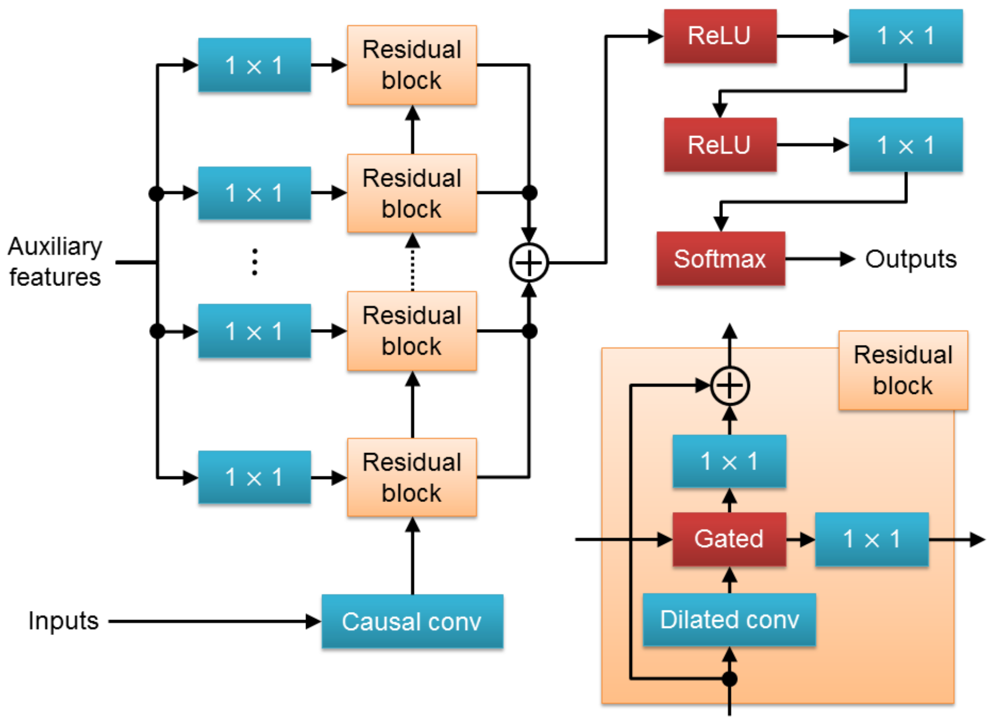


Figure 2.9: Overview of WaveNet architecture with residual block.

where $\boldsymbol{W}^{(\cdot)}$ and $\boldsymbol{V}^{(\cdot)}$ are learnable convolution filters, $*$ denotes a 1×1 convolution, and $\odot$ denotes element-wise product. Since the time resolution of $\boldsymbol{s}$ and $\boldsymbol{h}$ would differ, it is necessary to match the sequence length between the auxiliary feature sequence and the speech waveform signals. Hence, the function $u(\cdot)$ is introduced for upsampling $\boldsymbol{h}$. A transposed convolutional network or a function copying $\boldsymbol{V}^{(\cdot)} * \boldsymbol{h}$ is typically used as $u(\cdot)$.

Chapter 3

Multiple decision tree-based context clustering for FAHMM

3.1 Multiple decision tree-based context clustering

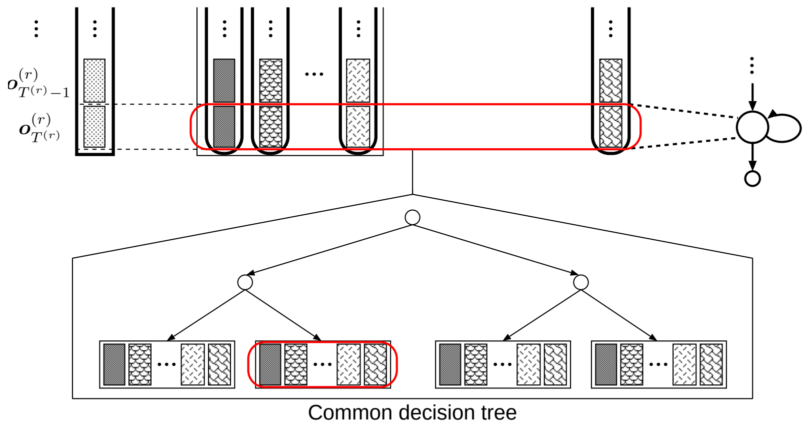
The goal of training FAHMM is to estimate the basis vectors that can represent any speaker's voice in all possible linguistic contexts. To estimate such basis vectors, the model structures of the basis vectors for all linguistic contexts should be carefully designed. In previous researches [9], a common single decision tree is used for the model structures by assuming that all the basis vectors are based on the same structure. This assumption enables to use well-known decision tree-based context clustering in the same way as the standard HMM. The model structure is illustrated in Figure 3.1(a). Although the context clustering needs a light computational load, the speaker-independent structure makes it difficult to model speech of multiple speakers. Thus, the multiple-tree structure shown in Figure 3.1(b) is required. (In the Figure, the noise mean vector and the noise covariance matrix have the same model structure for simplification.) In this section, we propose a multiple decision tree-based context clustering technique to overcome the limitation imposed by a simple model structure.

In terms of the output probability of FAHMM, the lower bound $\mathcal{F}$ to be maximized can be rewritten as follows:

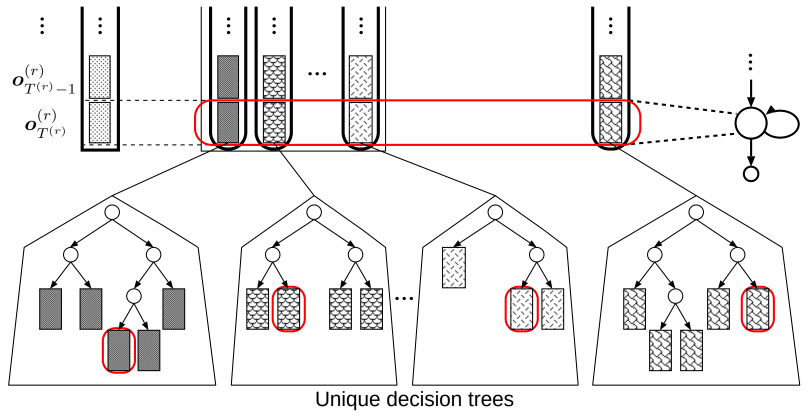
$$\mathcal{F} \propto \sum_{r,m,t} \gamma_m^{(r)}(t) \left\langle \log \mathcal{N}\left(\boldsymbol{o}_t^{(r)} \,\middle|\, \boldsymbol{W}_m \boldsymbol{\zeta}^{(r)}, \boldsymbol{\Sigma}_m\right)\right\rangle_{q\left(\boldsymbol{x}^{(r)}\right)}, \tag{3.1}$$

where $\gamma_m^{(r)}(t)$ is the posterior probability of context[1] m generating observation $\boldsymbol{o}_t^{(r)}$ given the current model parameters. The HMM state index is omitted to simplify description.

[1]A context corresponds to a combination of $Q+2$ leaf nodes belonging to each decision tree.



(a) The basis vectors and the noise vector are based on the same single decision tree.



(b) The basis vectors and the noise vector are based on their own decision tree.

Figure 3.1: Model structure of FAHMM.

Here, we assume that the noise covariance matrix, as well as the basis vectors and the noise mean vector, have its own decision tree. Since each of the basis vectors and the noise vector depend on each other, multiple trees must be built simultaneously. To evaluate all of them simultaneously, let $\boldsymbol{w}$ be a vector that concatenates all the basis vectors and the noise mean vector:

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_1^\top & \boldsymbol{w}_2^\top & \cdots & \boldsymbol{w}_V^\top \end{bmatrix}^\top, \tag{3.2}$$

where $\boldsymbol{w}_v$ is a basis vector or a noise mean vector in leaf node v, and V is the sum of all leaf nodes of decision trees for the basis vectors and the noise mean vector. Differentiating (3.1) with respect to $\boldsymbol{w}$ and equating it to zero gives a set of linear equations for updating $\boldsymbol{w}$ as

$$\boldsymbol{G}\hat{\boldsymbol{w}} = \boldsymbol{k}, \tag{3.3}$$

where

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}_{1,1} & \boldsymbol{G}_{1,2} & \cdots & \boldsymbol{G}_{1,V} \\ \boldsymbol{G}_{2,1} & \boldsymbol{G}_{2,2} & \cdots & \boldsymbol{G}_{2,V} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{G}_{V,1} & \boldsymbol{G}_{V,2} & \cdots & \boldsymbol{G}_{V,V} \end{bmatrix}, \tag{3.4}$$

$$\boldsymbol{k} = \begin{bmatrix} \boldsymbol{k}_1^\mathsf{T} & \boldsymbol{k}_2^\mathsf{T} & \cdots & \boldsymbol{k}_V^\mathsf{T} \end{bmatrix}^\mathsf{T}. \tag{3.5}$$

The each element is defined as

$$\boldsymbol{G}_{v_1,v_2} = \sum_{m \in f(v_1) \wedge f(v_2)} \boldsymbol{\Sigma}^{-1}_{f_{\boldsymbol{\Sigma}}^{-1}(m)} \sum_{r,t} \gamma_m^{(r)}(t) \langle \zeta^{(r)}_{\rho(v_1)} \zeta^{(r)}_{\rho(v_2)} \rangle, \tag{3.6}$$

$$\boldsymbol{k}_{v_1} = \sum_{m \in f(v_1)} \boldsymbol{\Sigma}^{-1}_{f_{\boldsymbol{\Sigma}}^{-1}(m)} \sum_{r,t} \gamma_m^{(r)}(t) \langle \zeta^{(r)}_{\rho(v_1)} \rangle \boldsymbol{o}_t^{(r)}, \tag{3.7}$$

where $f(\cdot)$ provides a set of contexts included in a given leaf node index of the tree corresponding to a basis vector or the noise mean vector, $f_{\boldsymbol{\Sigma}}^{-1}(\cdot)$ returns a leaf node index in the tree of the noise covariance matrix where the node includes a given context, $\rho(\cdot)$ is a function to get a tree index where the tree includes a given leaf node, and $\zeta_q^{(r)}$ is the qth element of $\boldsymbol{\zeta}^{(r)}$. The update formula for the noise covariance matrix is derived by differentiating (3.1) with respect to $\boldsymbol{\Sigma}_u$ and setting its result to zero:

$$\hat{\boldsymbol{\Sigma}}_u = \frac{\operatorname{diag}\left[\sum_{\substack{m \in f_{\boldsymbol{\Sigma}}(u) \\ r,t}} \gamma_m^{(r)}(t) \left(\boldsymbol{o}_t^{(r)} \boldsymbol{o}_t^{(r)\mathsf{T}} - 2\boldsymbol{o}_t^{(r)} \langle \boldsymbol{\zeta}^{(r)\mathsf{T}} \rangle \boldsymbol{W}_m^\mathsf{T} + \boldsymbol{W}_m \langle \boldsymbol{\zeta}^{(r)} \boldsymbol{\zeta}^{(r)\mathsf{T}} \rangle \boldsymbol{W}_m^\mathsf{T} \right) \right]}{\sum_{\substack{m \in f_{\boldsymbol{\Sigma}}(u) \\ r,t}} \gamma_m^{(r)}(t)}, \tag{3.8}$$

where u is a leaf node index in the tree of the noise covariance matrix, and $f_{\boldsymbol{\Sigma}}(\cdot)$ gives a set of contexts included in a given leaf node index of the noise covariance matrix.

In the proposed clustering algorithm, multiple trees are built greedily and simultaneously as follows:

Step 1. Create $Q + 2$ root nodes for each of the basis vectors, the noise mean vector, and the noise covariance matrix.

Step 2. Evaluate all questions at all leaf nodes of all trees using Eqs. (3.1), (3.3), and (3.8).

Step 3. Select a pair of node and question that maximizes Eq. (3.1), and then split the selected node into two by applying the selected question.

Step 4. If the change of likelihood after splitting the node falls below a predefined threshold, stop the procedure. Otherwise, go to **Step 2**.

Step 2 makes this procedure computationally intractable for the following three reasons. First, solving (3.3) involves a matrix inversion where the size of matrix $\boldsymbol{G}$ depends on the size of the trees. The complexity of matrix inversion is $\mathcal{O}(V^3)$. Note that the number of leaf nodes V typically exceeds 1000. Second, the inverse matrix $\boldsymbol{G}^{-1}$ needs to be calculated for all possible combinations of linguistic questions and leaf nodes. The number of linguistic questions is about 2000. Third, vector $\boldsymbol{w}$ and noise covariance matrix $\boldsymbol{\Sigma}_u$ must be iteratively updated until convergence because they depend on each other, as shown in (3.6), (3.7) and (3.8). These reasons exponentially increase computational complexity as the trees grow, and such increased complexity makes the proposed clustering infeasible. To solve the problem, two algorithms for reducing computational complexity inspired by additive structure models [17], whose structure is quite similar to that of FAHMM, are introduced in the following sections.

3.1.1 Reducing computational complexity by tying noise covariance matrices

The basis vectors and the noise covariance need to be iteratively updated until a convergence is reached. To reduce the computational cost coming from iterative updating, all the noise covariance matrices are globally tied while clustering:

$$\forall_m \left(\boldsymbol{\Sigma}_m = \boldsymbol{\Sigma}_g\right), \tag{3.9}$$

where $\boldsymbol{\Sigma}_g$ is the globally-tied noise covariance matrix. By tying all the noise covariance parameters, iterative updates are no longer required because the noise covariance matrix in Eqs (3.6) and (3.7) are canceled out as follows:

$$\begin{aligned}
\boldsymbol{G}_{v_1,v_2} &= \sum_{\substack{m \in f(v_1) \wedge f(v_2) \\ r,t}} \gamma_m^{(r)}(t) \langle \zeta_{\rho(v_1)}^{(r)} \zeta_{\rho(v_2)}^{(r)} \rangle \boldsymbol{I}, && (3.10) \\
\boldsymbol{k}_{v_1} &= \sum_{\substack{m \in f(v_1) \\ r,t}} \gamma_m^{(r)}(t) \langle \zeta_{\rho(v_1)}^{(r)} \rangle \boldsymbol{o}_t^{(r)}. && (3.11)
\end{aligned}$$

Furthermore, the number of trees to be built is reduced to $Q+1$ from $Q+2$. After clustering, the globally-tied noise covariance matrix $\boldsymbol{\Sigma}_g$ is untied and then retied according

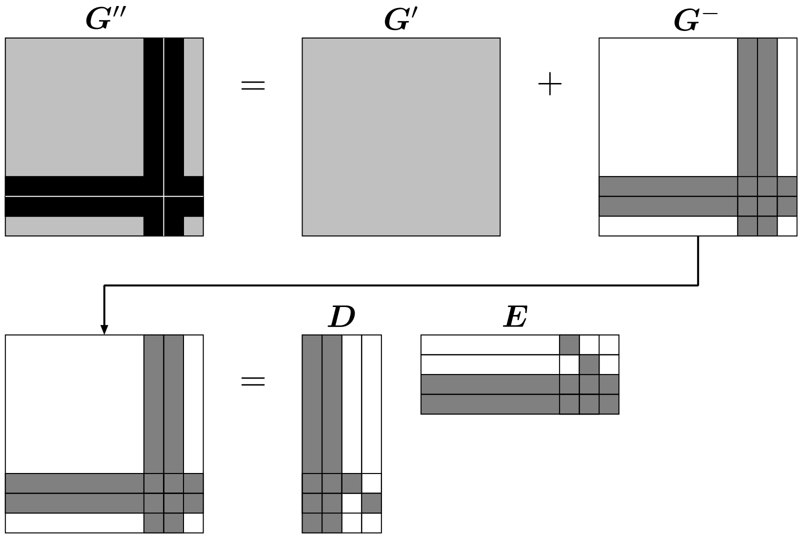


Figure 3.2: Decomposition of $\boldsymbol{G}$.

to the tree of the noise mean vector. The improvement by re-tying them is small in early experiments. Thus, the impact on speech quality of tying noise covariance parameters during clustering is not critical. In fact, it has been reported that covariance parameters are less important than mean parameters in terms of speech quality in HMM-based speech synthesis [16].

3.1.2 Reducing computational complexity by using matrix inversion lemma

The inverse matrix $\boldsymbol{G}^{-1}$ is calculated $V \times N_Q$ times every **Step 2**, where N_Q is the number of linguistic questions. This calculation requires a huge computational time. However, that computation time can be significantly reduced because the matrix inversions partially include the same calculation. When a leaf node is split by a question, the sufficient statistics are only changed in contexts related to newly created nodes by the split[2]; that is, almost all elements of $\boldsymbol{G}$ are unchanged even if a different question is applied to that leaf node. This fact can be exploited for computational reduction with the matrix inversion lemma in a similar way to the additive structure models [17].

Let $\boldsymbol{G}'$ and $\boldsymbol{G}''$ are matrices of $\boldsymbol{G}$ obtained by applying one question and another question

[2]This assumption is valid when the statistics of the factor represented as Eqs. (2.78) and (2.79) are fixed, and the noise covariance matrices are globally tied while clustering.

to a node, respectively. Then, $\boldsymbol{G}''$ can be represented as

$$\boldsymbol{G}'' = \boldsymbol{G}' + \boldsymbol{G}^{-}, \tag{3.12}$$

where $\boldsymbol{G}^{-}$ is the following symmetric matrix:

$$\boldsymbol{G}^{-} = \begin{bmatrix} \mathbf{0} & g^{-}_{1,v} & g^{-}_{1,v+1} & \mathbf{0} \\ & \vdots & \vdots & \\ g^{-}_{v,1} \cdots & g^{-}_{v,v} & g^{-}_{v,v+1} & \cdots g^{-}_{v,V} \\ g^{-}_{v+1,1} \cdots & g^{-}_{v+1,v} & g^{-}_{v+1,v+1} & \cdots g^{-}_{v+1,V} \\ & \vdots & \vdots & \\ \mathbf{0} & g^{-}_{V,v} & g^{-}_{V,v+1} & \mathbf{0} \end{bmatrix}, \tag{3.13}$$

where v and $v+1$ are indexes of the leaf nodes newly created by the split. For simplicity of notation, only one dimension of the observation vector is focused on here. The matrix $\boldsymbol{G}^{-}$ can be decomposed into two matrices:

$$\boldsymbol{G}^{-} = \boldsymbol{D}\boldsymbol{E}, \tag{3.14}$$

where

$$\begin{aligned} \boldsymbol{D} &= \begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 & \boldsymbol{D}_3 & \boldsymbol{D}_4 \end{bmatrix} \\ &= \begin{bmatrix} g^{-}_{1,v} & g^{-}_{1,v+1} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g^{-}_{v-1,v} & g^{-}_{v-1,v+1} & 0 & 0 \\ g^{-}_{v,v}/2 & g^{-}_{v,v+1}/2 & 1 & 0 \\ g^{-}_{v+1,v}/2 & g^{-}_{v+1,v+1}/2 & 0 & 1 \\ g^{-}_{v+2,v} & g^{-}_{v+2,v+1} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ g^{-}_{V,v} & g^{-}_{V,v+1} & 0 & 0 \end{bmatrix}, &(3.15) \\ \boldsymbol{E} &= \begin{bmatrix} \boldsymbol{D}_3 & \boldsymbol{D}_4 & \boldsymbol{D}_1 & \boldsymbol{D}_2 \end{bmatrix}^{\mathsf{T}}. &(3.16) \end{aligned}$$

The relation between $\boldsymbol{G}''$ and $\boldsymbol{G}'$ is illustrated in Figure 3.2. If $\boldsymbol{G}'^{-1}$ is known, $\boldsymbol{G}''^{-1}$ can be efficiently calculated using the matrix inversion lemma:

$$\begin{aligned} \boldsymbol{G}''^{-1} &= (\boldsymbol{G}' + \boldsymbol{D}\boldsymbol{E})^{-1} \\ &= \boldsymbol{G}'^{-1} - \boldsymbol{G}'^{-1}\boldsymbol{D}\left(\boldsymbol{I} + \boldsymbol{E}\boldsymbol{G}'^{-1}\boldsymbol{D}\right)^{-1}\boldsymbol{E}\boldsymbol{G}'^{-1} \\ &= \boldsymbol{G}'^{-1} - \boldsymbol{G}'^{-1}\boldsymbol{D}\boldsymbol{\Psi}\boldsymbol{E}\boldsymbol{G}'^{-1}. \end{aligned} \tag{3.17}$$

Matrix $\boldsymbol{\Psi}$ is the inverse of $\boldsymbol{I}+\boldsymbol{E}\boldsymbol{G}'^{-1}\boldsymbol{D}$, whose size is 4×4. As a result, the complexity of the clustering $\mathcal{O}(N_Q N_I D V^4)$ becomes $\mathcal{O}(N_Q V^2)$, where N_I is the number of iterations of updating the basis vectors and the noise covariance matrix. Thus, the clustering is regarded as computationally feasible.

3.2 Experiments

3.2.1 Experimental setups

To evaluate the proposed method, speaker adaptation in speech synthesis was set as a task. A Japanese speech database, which was constructed by our research group, was used for the experimental evaluation. The database contains sets of 503 phonetically balanced sentences uttered by more than 100 college students. At most 89 male and 11 female speakers were chosen for training, and the other eight male speakers and two female speakers were used for the evaluation. The number of training sentences per speaker was at most 50, and the number of adaptation sentences was at most 10. Forty test sentences, which were included in neither the training nor the adaptation data, were used for the evaluation. The speech signals were sampled at a rate of 16 kHz and windowed by a 25-ms Hamming window with a 5-ms shift. Each feature vector consisted of 24 mel-cepstral coefficients including the zeroth coefficient, log-fundamental frequency ($\log F_0$), and their first- and second-time derivatives. A five-state left-to-right MSD-HSMM [47,60] without skip paths was used. The number of basis vectors for $\log F_0$ and duration were both one. Decision trees were constructed based on the MDL criterion. The following three methods were compared:

- **PCA**: The basis vectors were constructed by applying PCA to a set of supervectors of speaker-dependent models trained by using constrained maximum likelihood linear regression (CMLLR)-based speaker adaptive training (SAT) [61,62].
- **FA**: The basis vectors were trained on the FAHMM framework. The model structure was based on a single tree built by the standard decision-tree based context clustering.
- **FA_MT**: The basis vectors were trained on the FAHMM framework. The model structure was based on multiple trees built by the proposed context clustering algorithm.

The proposed multiple tree-based context clustering was used only for spectral parameters, and the standard decision tree-based context clustering was used for $\log F_0$ and duration modeling. To avoid estimating unreliable basis vectors, a constraint was applied to the split in clustering: every node of a decision tree must always have training data from at least 10 speakers in a similar way to the shared decision tree context clustering (STC) [63].

The computational time for building trees without GPU in **FA** and **FA_MT** were about 40 minutes and three weeks, respectively, when the number of training sentences was 2000 and the number of basis vectors was 30.

3.2.2 Objective evaluation

Mel-cepstrum distance (MCD) [64] was used as the objective measure. MCDs for 10 training speakers and 10 test speakers were calculated. They were then averaged for each speaker set. First, the number of training speakers was varied (10, 20, 50, and 100), while the number of training sentences was fixed to 500 and the number of basis vectors for mel-cepstrum was fixed to five. The results of this experiment are shown in Figure 3.3. As the number of training speakers increases, MCDs of test speakers decrease while those of training speakers increase. This result indicates that a more robust speaker subspace can be spanned by using multiple speakers. As expected, **FA** outperformed **PCA**, and **FA_MT** further improved MCDs in comparison with that of **FA**.

In the second experimental evaluation, the number of training sentences was varied (500, 1000, 1500, and 2000), while the number of training speakers was fixed to 100 and the number of basis vectors was fixed to five. The results of the evaluation are shown in Figure 3.4. As for **FA** and **FA_MT**, MCD monotonically decreases with increasing number of training sentences. It can thus be said that FAHMM is able to exploit large, heterogeneous speech data.

In the next objective evaluation, the number of basis vectors was varied (5, 10, 20, and 30), and 2000 training sentences uttered by 100 speakers were used. The relation between MCD and the number of basis vectors is plotted in Figure 3.5. It is clear from the figure that **FA** achieves lower MCD than that achieved by **PCA**, but MCD saturates at 10 basis vectors. On the other hand, MCD attained by **FA_MT** decreased as the number of basis vectors increased, and **FA_MT** achieved the lowest MCD for both the training and test speakers. This result is due to the fact that the flexible multiple tree structures make it possible to model the complex relations between linguistic contexts, acoustic features, and speaker characteristics.

In the final objective evaluation, the number of adaptation sentences was varied (1, 2, 5, and 10), while the number of training speakers, training sentences, and basis vectors were 100, 2000, and 30, respectively. The relation between MCD and the number of adaptation sentences is illustrated in Figure 3.6. It can be seen that five sentences are enough to adapt models. This is because only 30 parameters can be changed for a new speaker, resulting in very fast adaptation. Combining FAHMM with powerful but data-hungry adaptation techniques such as MLLR [48, 61] and maximum a posteriori (MAP) [65, 66] could well decrease MCD when a large amount of adaptation data is available.

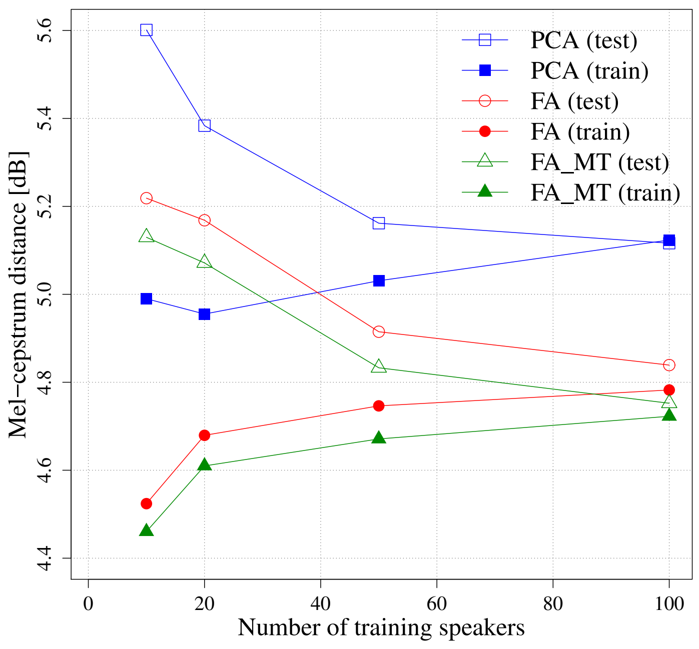


Figure 3.3: MCD vs. number of training speakers, under the condition that the number of training sentences is 500 and the number of basis vectors is 5.

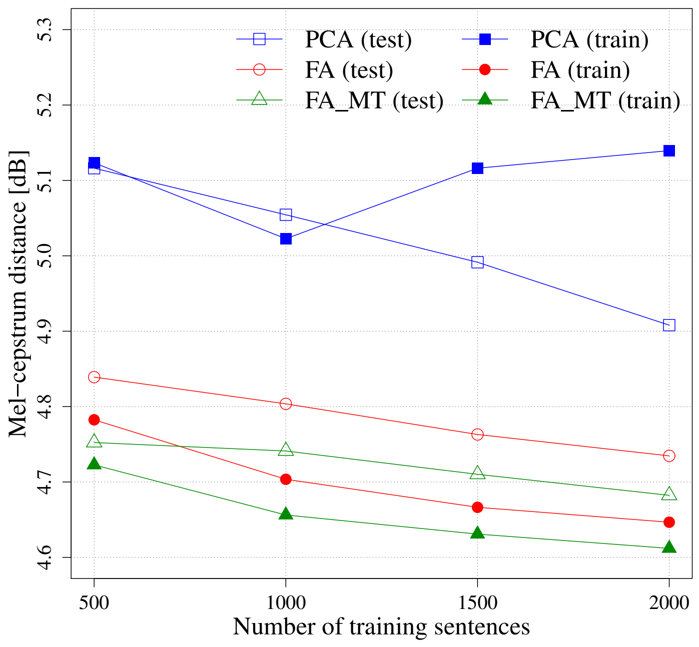


Figure 3.4: MCD vs. number of training sentences, under the condition that the number of training speakers is 100 and the number of basis vectors is 5.

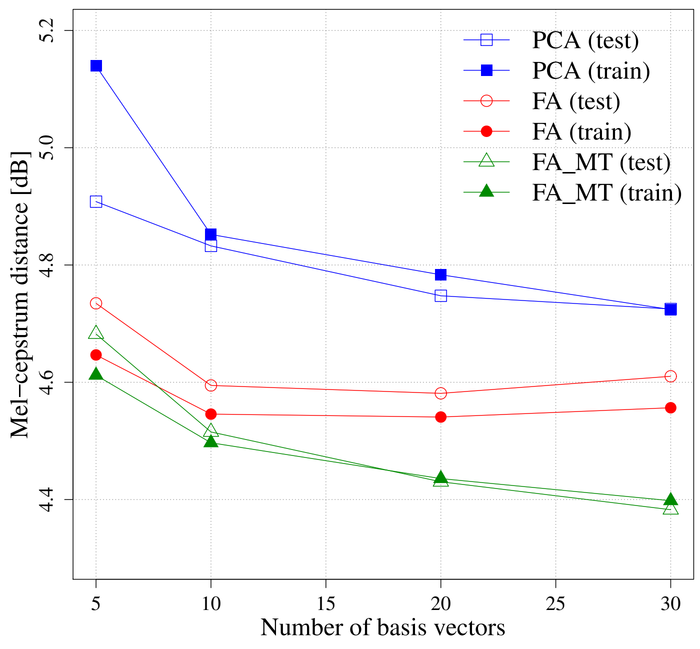


Figure 3.5: MCD vs. number of basis vectors, under the condition that the number of training sentences is 2000 and the number of training speakers is 100.

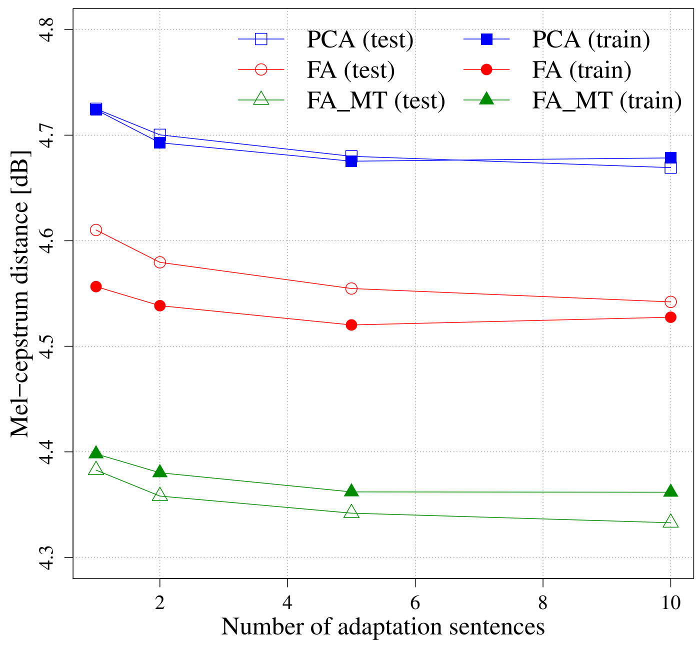


Figure 3.6: MCD vs. number of adaptation sentences, under the condition that the number of training speakers, sentences, and basis vectors are 2000, 100, and 30, respectively.

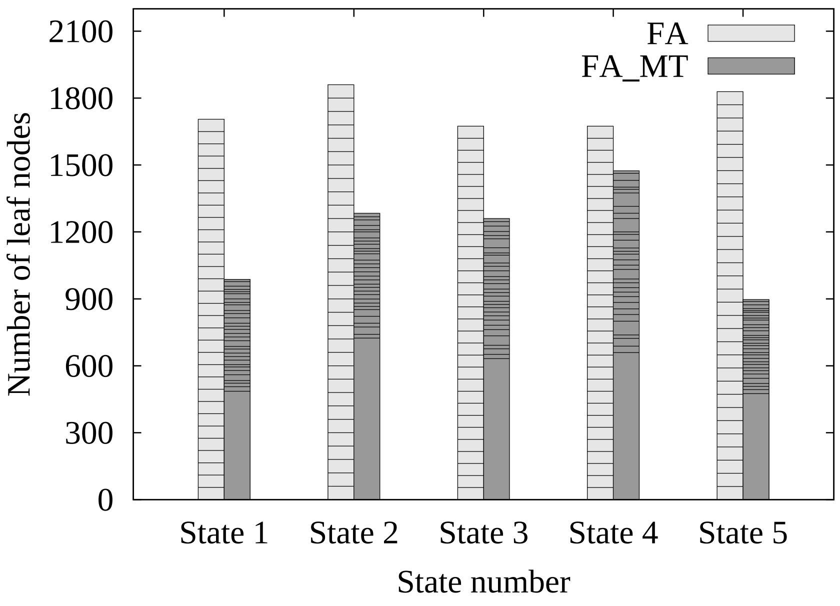


Figure 3.7: Number of leaf nodes for each HMM state.

The number of leaf nodes for each HMM state is shown in Figure 3.7 where the bottom bars denote the number of leaf nodes of noise mean vector, and the 30 stacked bars represent the number of leaf nodes of each basis vector. The tree sizes in the case of **FA_MT** differ, while they were fixed in the case of **FA**. This result means that the proposed algorithm can take into account the importance of each of the basis vectors while clustering.

3.2.3 Subjective evaluation

Two subjective listening tests were conducted. The first test evaluated the naturalness of synthetic speech by the mean opinion score (MOS) test method, and the second one evaluated the speaker similarity between target speech and the synthetic speech by the differential MOS (DMOS) test method. In the MOS test, after the subjects had listened to a test sample, they were asked to assign it a naturalness score on a five-point scale. In the DMOS test, after the subjects had listened to the natural speech of the target speaker and a test sample, they were asked to assign it a similarity score on a five-point scale. Ten native Japanese speakers evaluated 10 sentences, which were randomly chosen from 400 sentences (40 sentences $\times$ 10 test speakers), in both tests. Synthetic speech was generated from models obtained under the condition that the number of training speakers, training sentences, basis vectors, adaptation sentences were 100, 2000, 30, and 1, respectively.

The results of the MOS and DMOS listening tests are shown in Figures 3.8 and 3.9, respectively. Figure 3.8 shows that **FA** significantly improved the naturalness of synthetic speech compared with **PCA**. **PCA** could not estimate reliable basis vectors because it requires an adequate amount of speech data for each training speaker to build speaker-dependent models. It can be said that FAHMM enables to effectively use speech data even if only a few sentences per speaker are available. **FA_MT** achieved better performance than **FA**. It can thus be suggested that flexible spectral modeling based on multiple-tree structures is essential in subspace-based methods.

As for the DMOS test, **FA** significantly outperformed **PCA**. It can be said that **FA** can reliably estimate a speaker subspace even if a small amount of training data is available for each training speaker. In addition, **FA_MT** achieved a slightly higher score than **FA**, though there was no significant difference. Since prosody strongly affects speaker similarity, applying the proposed method to F_0 and duration parameters as well as spectral ones could be helpful for synthesizing speech with desired speaker characteristics.

3.3 Summary

A multiple decision tree-based context clustering technique for factor analyzed HMM (FAHMM)-based speech synthesis was proposed and evaluated. Although the proposed multiple tree-based clustering is computationally infeasible, two computational complexity reduction algorithms made the proposed method computationally tractable. Both objective and subjective experiments showed that the proposed method significantly outperforms the conventional method based on a single model structure. It can be said that taking speaker characteristics into account modeling the relation between linguistic and acoustic features is essential when using speech data consists of multiple speakers. The comparison of FAHMM and other subspace-based approaches such as cluster adaptive training [14] is future work.

Recently, DNN-based approaches have been shown to be effective for speech synthesis, and the adaptation techniques have begun to be investigated [67, 68] including subspace-based approaches [69–71]. However, it is not clear which approach is the most effective. When a small amount of adaptation data is available, subspace-based methods are expected to be effective. Future work includes introducing the idea of FAHMM into DNN framework based on the result of the experiments in this paper.

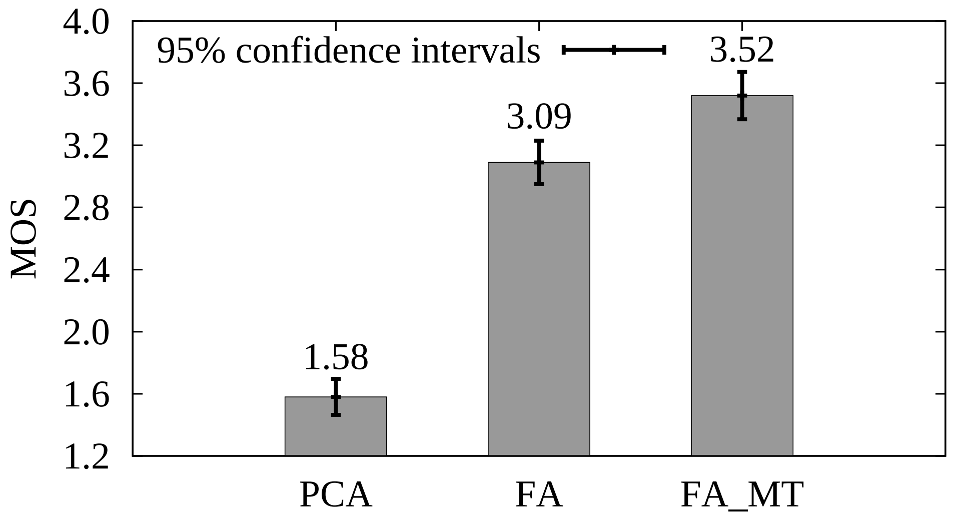


Figure 3.8: MOS of naturalness with 95% confidence intervals.

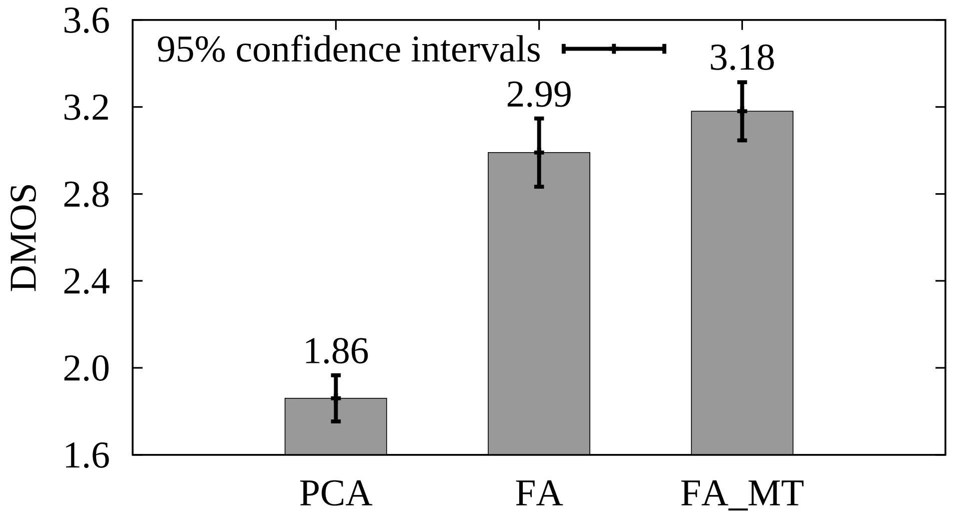


Figure 3.9: DMOS of speaker similarity with 95% confidence intervals.

Chapter 4

Mel-cepstrum-based quantization noise shaping

4.1 Quantization in raw audio waveform modeling

Let us assume an autoregressive generative model that predicts the current value of the real-valued time series $\boldsymbol{x} = [\, x[0] \, \cdots \, x[N-1] \,]$ using the past samples:

$$p\,(\boldsymbol{x} \,|\, \boldsymbol{h}) \;=\; \prod_{n=0}^{N-1} p\,(x[n] \,|\, x[0], \ldots, x[n-1])\,, \tag{4.1}$$

where N is the number of speech samples. In order to produce waveforms with specified characteristics, auxiliary features, $\boldsymbol{h}$, that affect the distribution, e.g., linguistic features in TTS, are typically introduced to the model:

$$p\,(\boldsymbol{x} \,|\, \boldsymbol{h}) \;=\; \prod_{n=0}^{N-1} p\,(x[n] \,|\, x[0], \ldots, x[n-1], \boldsymbol{h})\,, \tag{4.2}$$

To optimize the model, which is represented as a conditional probability distribution in Eq. (4.2), the difference between the actual and predicted values of a speech sample must be measured on the basis of some criterion. Although it is reasonable to use a continuous criterion such as the minimum mean squared error (MSE) because $x[n]$ takes a real value, this implicitly imposes an assumption about the shape of the distribution. The WaveNet and SampleRNN models are more flexible due to avoiding this problem. They quantize $x[n]$ and are optimized by minimizing the cross-entropy between the real and predicted probability distributions. Then, the quantized sequence $\boldsymbol{i} = [\, i[0] \, \cdots \, i[N-1] \,]$ is modeled as

$$p(\boldsymbol{i} \,|\, \boldsymbol{h}) \;=\; \prod_{n=0}^{N-1} p\left(i[n] \,|\, i[0], \ldots, i[n-1], \boldsymbol{h}\right) \tag{4.3}$$

rather than as Eq. (4.2). The softmax function is used as the output layer of the neural networks. A softmax distribution tends to work better than a Gaussian mixture distribution [23, 72] even when the data is implicitly continuous. To quantize speech samples, a simple linear uniform quantization with μ-law companding [24] is used.

In μ-law companding, a speech sample is non-linearly transformed as follows:

$$f(x[n]) \;=\; \operatorname{sgn}(x[n])\,\frac{\log(1+\mu|x[n]|)}{\log(1+\mu)}, \tag{4.4}$$

where $|x[n]| \leq 1$ and μ is the compression factor. The typical value for μ is 255 when converting from 16-bit linear PCM to 8-bit sample. The sample $x[n]$ is then quantized it into q levels:

$$i[n] = \operatorname{round}\left[(2^q-1)\,\frac{f(x[n])+1}{2}\right], \tag{4.5}$$

where $0 \leq i[n] < 2^q$ and $\operatorname{round}(\cdot)$ denotes rounding. The quantized sample $i[n]$ can be reconstructed by using μ-law decompression:

$$\hat{x}[n] \;=\; f^{-1}\left(\frac{2}{2^q-1}i[n]-1\right), \tag{4.6}$$

where

$$f^{-1}(x[n]) = \operatorname{sgn}(x[n])\,\frac{(1+\mu)^{|x[n]|}-1}{\mu}. \tag{4.7}$$

The z-transform of the reconstructed speech sample $\hat{x}[n]$ is represented as

$$\hat{X}(z) \;=\; X(z)+E(z), \tag{4.8}$$

where $X(z)$ and $E(z)$ are the z-transforms of $x[n]$ and $e[n]$, respectively, and $e[n]$ is the error caused by quantization:

$$e[n] \;=\; \hat{x}[n]-x[n]. \tag{4.9}$$

The quantization noise $e[n]$ can be considered to be uncorrelated with $x[n]$ when $x[n]$ is fairy complicated and q is large [73–75]. Thus, the frequency spectrum of $E(z)$ is assumed to be white. White noise is not very pleasant on the human ears as it contains many high frequency components. The quantization bits q should be large enough to prevent the degradation of speech quality caused by the quantization. However, there is a trade-off between the amount of quantization noise and the difficulty of neural network optimization. Current state-of-the-art techniques use 8-bit quantization [22, 23].

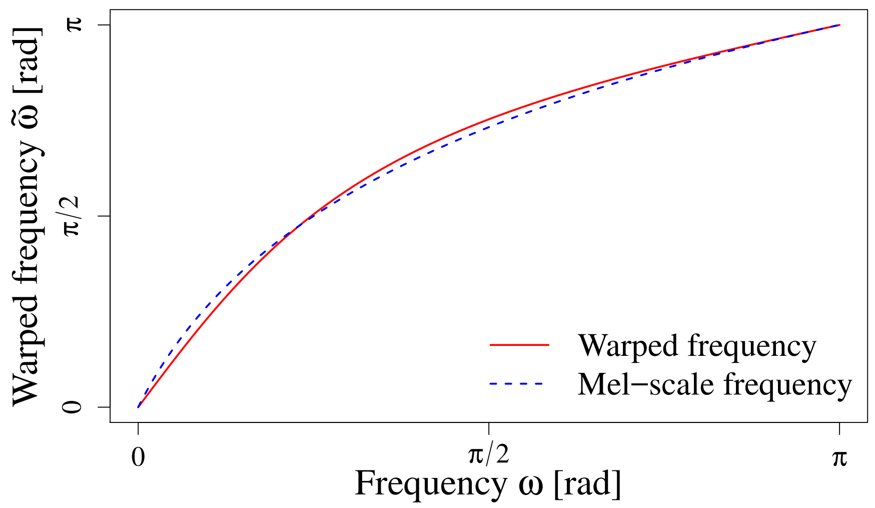


Figure 4.1: Phase characteristic $\tilde{\omega}$ of the first order all-pass transfer function $\tilde{z}^{-1}$.

4.2 Mel-cepstral analysis

In the mel-cepstral analysis [26], spectral envelope $H(z)$ is modeled using M-th order mel-cepstral coefficients $\{\tilde{c}(m)\}_{m=0}^{M}$:

$$H(z) = \exp \sum_{m=0}^{M} \tilde{c}(m)\, \tilde{z}^{-m}, \tag{4.10}$$

where

$$\tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}} \quad (|\alpha| < 1)\,. \tag{4.11}$$

The phase characteristic $\tilde{\omega}$ of the first order all-pass transfer function $\tilde{z}^{-1} = \exp(-j\tilde{\omega})$ is

$$\tilde{\omega} = \tan^{-1} \frac{(1-\alpha^2)\sin\omega}{(1+\alpha^2)\cos\omega - 2\alpha}. \tag{4.12}$$

For example, when the sampling frequency is 16 kHz, the phase characteristic $\tilde{\omega}$ of the all-pass function $\tilde{z}^{-1}$ for $\alpha = 0.42$ is a good approximation to the mel-frequency scale [76] (See Figure 4.1). Thus, the spectral envelope $H(z)$ can imitate the non-uniform frequency resolution of the human ear. Shaping the spectrum of the quantization noise using $H(z)$ should effectively mask the quantization noise by speech.

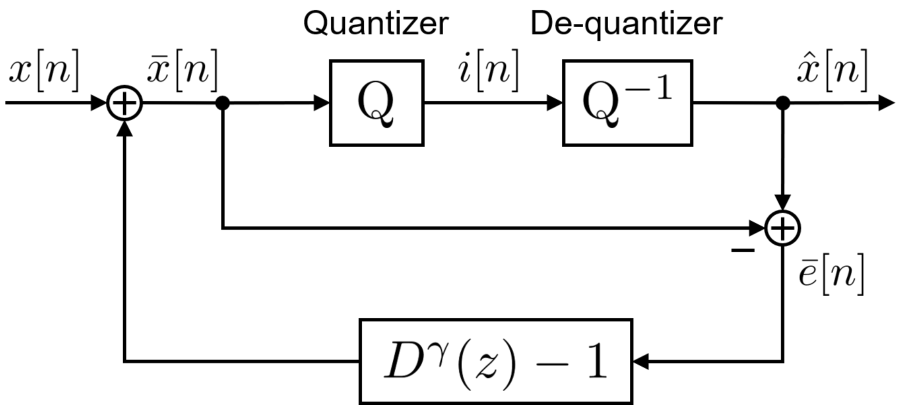


Figure 4.2: Block diagram of quantization noise shaping.

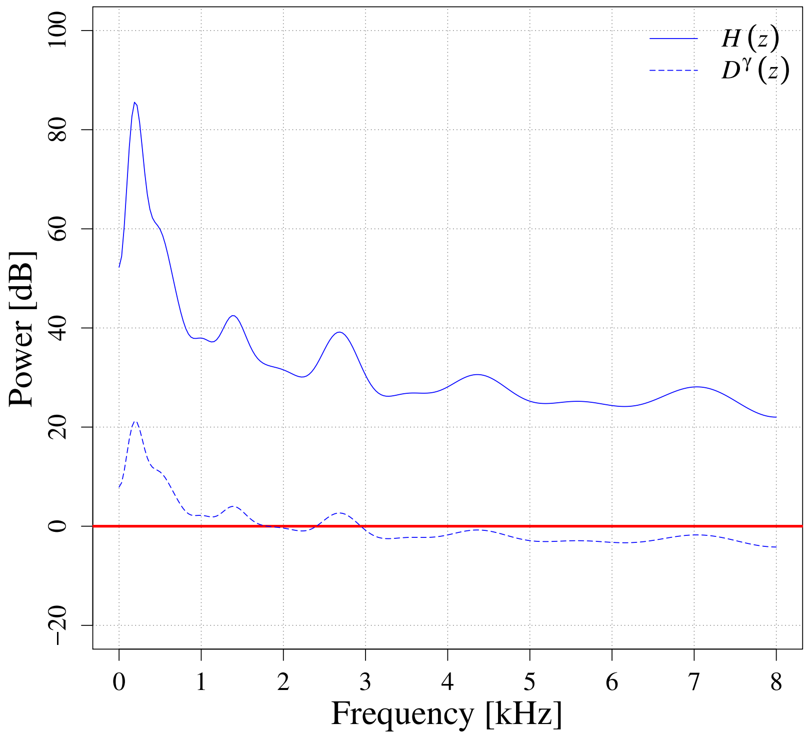


Figure 4.3: Power spectra of spectral envelope $H(z)$ and filter $D^{\gamma}(z)$ $(\gamma = 0.4)$.

4.3 Mel-cepstrum-based quantization noise shaping

Figure 4.2 shows the basic structure of the proposed quantization noise shaping. This structure was inspired by previous work [77] on adaptive differential pulse code modulation (ADPCM) speech coding. The basic concept of noise shaping is feedback of the quantization error to the input of the quantizer. This enables the spectral characteristics of the quantization noise to be changed. In the figure, $D(z)$ is a minimum phase transfer function derived from $H(z) = K \cdot D(z)$ where the gain factor K is factored out from $H(z)$. Hence, the impulse response of $D^\gamma(z)$ at time $n = 0$ equals unity, and $D^\gamma(z) - 1$ has a delay. The z-transform of the de-quantized speech sample $\hat{x}[n]$ is represented as

$$\hat{X}(z) = X(z) + D^\gamma(z)\bar{E}(z), \tag{4.13}$$

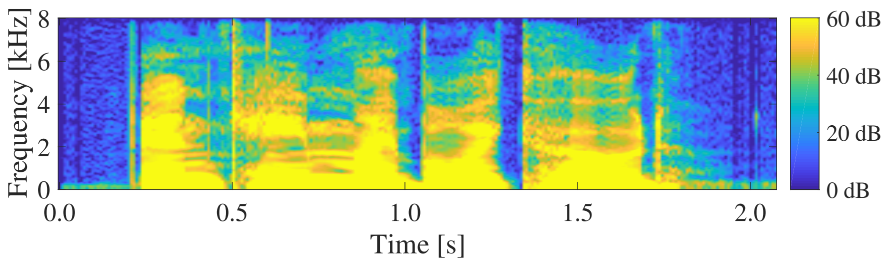
where $\bar{E}(z)$ is the z-transform of $\bar{e}[n] = \hat{x}[n] - \bar{x}[n]$ (See Figure 4.2). It can be seen that the noise spectrum $\bar{E}(z)$ is shaped by filter $D^\gamma(z)$. Tunable parameter $\gamma\,(0 \leq \gamma \leq 1)$ controls the effect of noise shaping ($\gamma = 0$ corresponds to conventional quantization).

Figure 4.3 shows the example of the power spectra of $H(z)$ and $D^\gamma(z)$ in a frame where red horizontal line denotes 0 dB. It can be seen from the figure that the $H(z)$ represents a spectrum envelope for which the shape is well captured in the low-frequency region. The $D^\gamma(z)$ has the same spectrum as $H(z)$ multiplied by γ. Since $\bar{E}(z)$ is multiplied by $D^\gamma(z)$, the altered quantization noise $D^\gamma(z) \cdot \bar{E}(z)$ should be masked by speech. To demonstrate the effect of the proposed quantization noise shaping, we applied it to a speech waveform. The spectrograms of the quantization noise as well as of the input speech waveform are shown in Figure 4.4. While the quantization noise is distributed over the entire frequency range without noise shaping (See Figure 4.4(b)), it is concentrated at low frequencies by the mel-cepstrum-based noise shaping with $\gamma = 0.4$ (See Figure 4.4(c)).

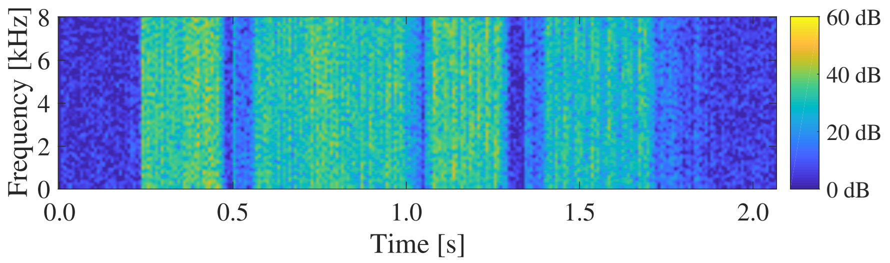
4.3.1 Implementation using FIR filter

A reasonable way to implement the proposed method is to use a FIR filter, which is easy to implement. The $D(z) - 1$ can be represented using the impulse response:

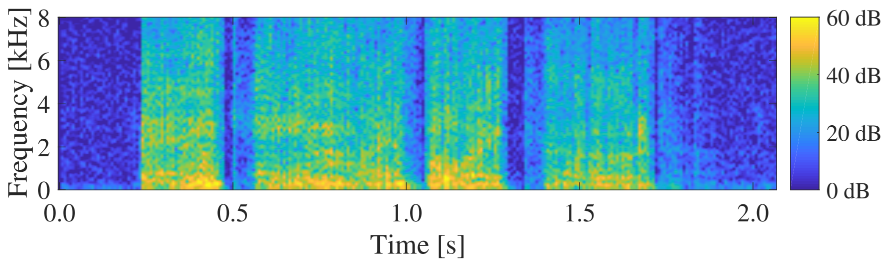
$$\begin{aligned} D(z) - 1 &= \sum_{n=0}^{\infty} h[n] z^{-n} - 1 \\ &= \sum_{n=1}^{\infty} h[n] z^{-n}, \end{aligned} \tag{4.14}$$



(a) Input signal



(b) Quantization noise (without noise shaping)



(c) Quantization noise (with noise shaping)

Figure 4.4: Spectrogram of a speech waveform and its quantization noise.

where $h[n]$ is the impulse response of $D(z)$ and $h[0] = 1$. The $D(z) - 1$ can be approximated as follows by truncating the infinite impulse response:

$$\begin{aligned} D(z) - 1 &\simeq \sum_{n=1}^{I} h[n] z^{-n} \\ &= z^{-1} \sum_{n=0}^{I-1} h[n+1] z^{-n} \\ &\equiv z^{-1} S(z), \end{aligned} \tag{4.15}$$

where I is the FIR length, which is typically more than a few hundred. A block diagram of this implementation is shown in Fig. 4.5. This implementation is computationally expensive due to the sample-by-sample conversion of $D(z)$. The conversion consists of two steps: from mel-cepstrum to cesptrum, and from cepstrum to impulse response. The

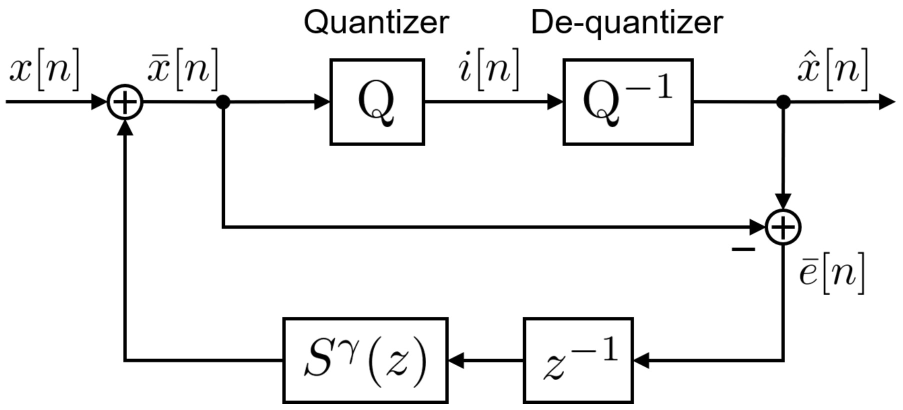


Figure 4.5: Block diagram of quantization noise shaping using $S^{\gamma}(z)$ for approximation of $D^{\gamma}(z) - 1$.

computational complexities are $\mathcal{O}(MI)$ and $\mathcal{O}(I^2)$. Since I is usually greater than M, the overall computational complexity for filtering is $\mathcal{O}(I^2)$.

4.3.2 Implementation using MLSA filter structure

The proposed quantization noise shaping can be efficiently implemented by using the MLSA filter structure [28]. First, the $H(z)$ is decomposed as $K \cdot D(z)$ using one of the methods proposed in [28]:

$$K = \exp b(0), \tag{4.16}$$

$$D(z) = \exp \sum_{m=1}^{M} b(m)\, \varPhi_m(z), \tag{4.17}$$

where

$$b(m) = \begin{cases} \tilde{c}(M) & (m = M), \\ \tilde{c}(m) - \alpha b(m+1) & (m < M), \end{cases} \tag{4.18}$$

$$\varPhi_m(z) = \begin{cases} 1 & (m = 0), \\ \dfrac{(1-\alpha^2)\, z^{-1}}{1 - \alpha z^{-1}} \tilde{z}^{-(m-1)} & (m > 0). \end{cases} \tag{4.19}$$

The $D^{\gamma}(z)$ is clearly

$$D^{\gamma}(z) = \exp \sum_{m=1}^{M} \gamma b(m)\, \varPhi_m(z). \tag{4.20}$$

The $D(z)$ is implemented as a digital filter by approximating exponential function $\exp(\cdot)$ as an L-th order rational function $R_L(\cdot)$ using the modified Padé approximation [78]:

$$\begin{aligned} D(z) &= \exp F(z) \simeq R_L(F(z)) \\ &= \frac{1 + \sum_{l=1}^{L} A_{L,l} \{F(z)\}^l}{1 + \sum_{l=1}^{L} A_{L,l} \{-F(z)\}^l}, \end{aligned} \tag{4.21}$$

where $\{A_{L,l}\}_{l=1}^{L}$ are the coefficients of the modified Padé approximation, L is the order of the modified Padé approximation (usually four or five), and $F(z)$ is the basic filter for the MLSA filter:

$$\begin{aligned} F(z) &= \sum_{m=1}^{M} b(m)\,\Phi_m(z) \\ &\equiv z^{-1}G(z). \end{aligned} \tag{4.22}$$

The structure of $G(z)$ and of basic filter $F(z)$ are shown in Fig. 4.6. The $D(z) - 1$ can be rewritten as

$$\begin{aligned} D(z) - 1 &\simeq R_L(F(z)) - 1 \\ &= \frac{2 \sum_{l=1,3,\cdots}^{L} A_{L,l} \{z^{-1}G(z)\}^l}{1 + \sum_{l=1}^{L} A_{L,l} \{-z^{-1}G(z)\}^l} \\ &\equiv z^{-1}S(z). \end{aligned} \tag{4.23}$$

Figure 4.7 shows a block diagram of the filter $D(z) - 1$. Since there is no need to convert it into the impulse response, this implementation is computationally faster than that using a FIR filter described in Section 4.3.1. The computational complexity for filtering is $\mathcal{O}(LM)$.

In the MLSA filter, the accuracy of the approximation of the exponential function can be improved by decomposing the basic filter:

$$F(z) = F_1(z) + F_2(z). \tag{4.24}$$

Although there are various ways to define $F_1(z)$ and $F_2(z)$, we use

$$F_1(z) = b(1)\Phi_1(z), \tag{4.25}$$

$$F_2(z) = \sum_{m=2}^{M} b(m)\Phi_m(z). \tag{4.26}$$

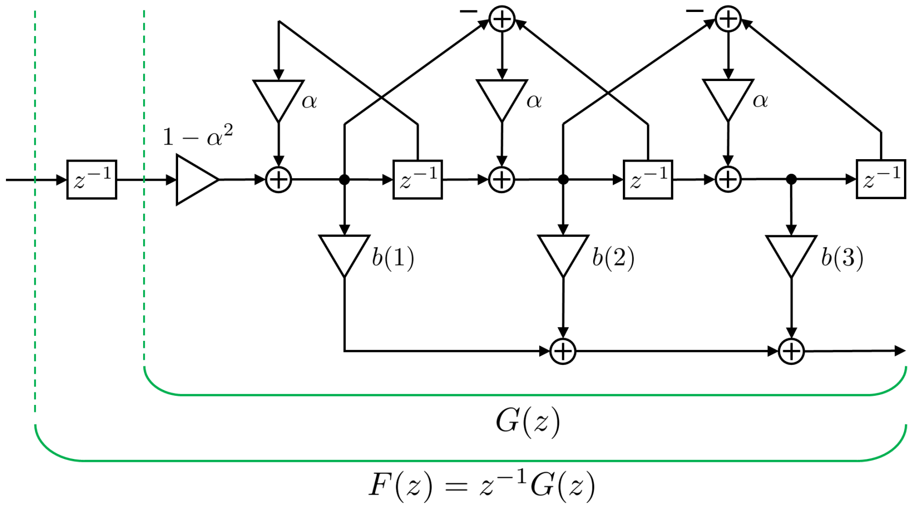


Figure 4.6: Block diagram of basic filter $F(z)$ $(M = 3)$.

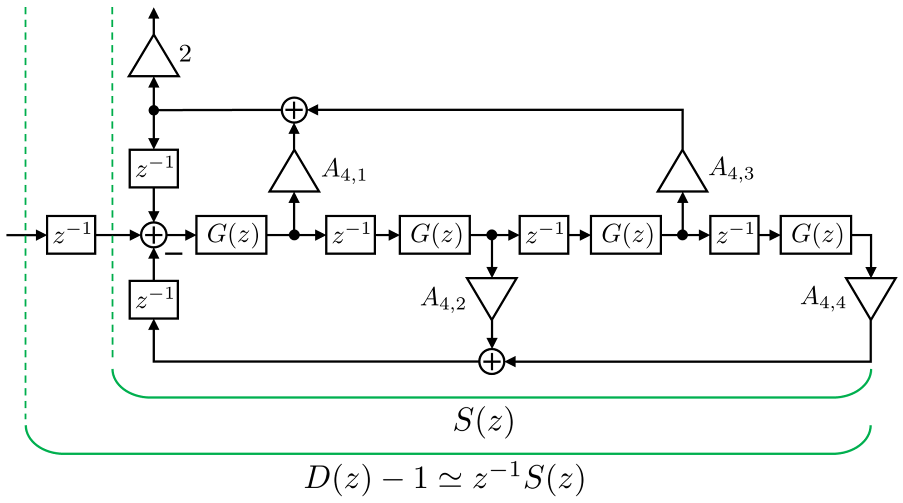


Figure 4.7: Block diagram of noise shaping filter $D(z) - 1$ $(L = 4)$.

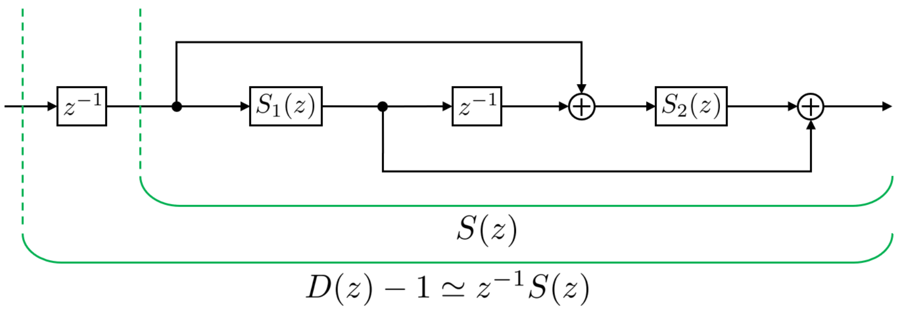


Figure 4.8: Two-stage cascaded structure for noise shaping filter $D(z) - 1$.

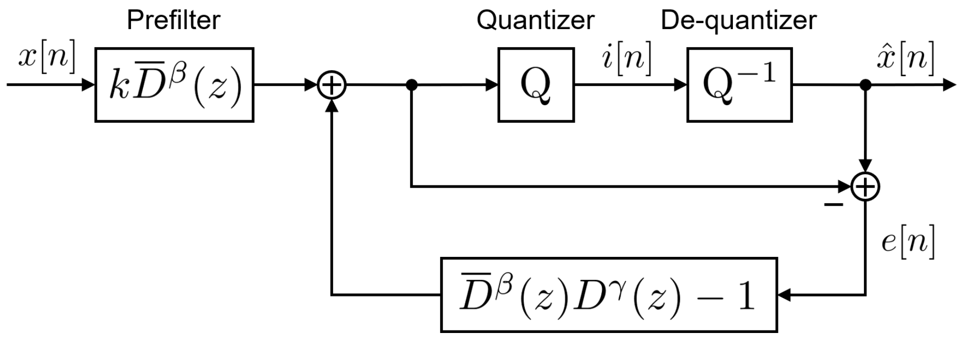


Figure 4.9: Block diagram of quantization noise shaping with prefiltering.

On the basis of decomposed basic filters $F_1(z)$ and $F_2(z)$, filter $D(z)-1$ can be represented as

$$\begin{aligned} D(z)-1 &= D_1(z)D_2(z)-1 \\ &\simeq \{R_L(F_1(z))-1\}+\{R_L(F_2(z))-1\} \\ &\quad \times [\{R_L(F_1(z))-1\}+1] \\ &\equiv z^{-1}S_1(z)+z^{-1}S_2(z) \\ &\quad \times \left[z^{-1}S_1(z)+1\right]. \end{aligned} \tag{4.27}$$

It is implemented as shown in Fig. 4.8. The structure illustrated in Fig. 4.5 is used for noise shaping in a similar manner as described in Section 4.3.1. Using the MLSA filter structure instead of a FIR filter enables to apply quantization noise shaping more than 30 times faster without any degradation for $L=4$, $M=24$, and $I=256$.

4.3.3 Mel-cepstrum-based prefiltering

Figure 4.9 shows the structure of mel-cepstrum-based noise shaping with prefiltering. In this Figure, z-transform of the de-quantized speech sample $\hat{x}[n]$ is represented as

$$\hat{X}(z) \;=\; \{kX(z)+D^{\gamma}(z)\,E(z)\}\,\overline{D}^{\beta}(z), \tag{4.28}$$

where k is a constant to normalize the power of signal, $\overline{D}^{\beta}$ is a filter for prefiltering, and β is a tunable parameter ($0 \le \beta \le 1$) controls the effect of prefiltering ($\beta = 0$ corresponds to no prefiltering). The transfer function $\overline{D}(z)$ is obtained by setting $\tilde{c}(1)=0$ and normalizing the gain as 1:

$$\overline{D}(z) \;=\; \exp\sum_{m=1}^{M}\bar{b}(m)\,\Phi_m(z), \tag{4.29}$$

where

$$\bar{b}(m) = \begin{cases} -\alpha b(2), & (m = 1) \\ b(m). & (m > 1) \end{cases} \tag{4.30}$$

The prefitering filter $\overline{D}^{\beta}$ can be derived by multiplying $\bar{b}(m)$ by β:

$$\overline{D}^{\beta}(z) = \exp \sum_{m=1}^{M} \beta \, \bar{b}(m) \, \Phi_m(z). \tag{4.31}$$

The reason for replacing $\tilde{c}(1)$ with 0 is to avoid the change of voice quality by the emphasis of the global slope of the spectrum. It has been reported that this kind of filtering is effective to reduce noise in ADPCM [77].

4.4 Experiments

4.4.1 Experimental setups

The CMU ARCTIC databases [79] were used to evaluate the proposed quantization noise shaping method. We used a female speaker (SLT) from the databases for the experiments. The number of sentences used for evaluation was 40. The speech signals were downsampled to 16 kHz. They were amplified on the basis of the maximum value of the signals in the databases to reduce quantization error. The downsampled speech signals were quantized using μ-law compression with/without the proposed noise shaping method implemented using a FIR filter. The filter coefficients were derived from 24-th order mel-cepstral coefficients including the zeroth coefficient extracted from the downsampled speech signals windowed using a 25-ms Hamming window with a 5-ms shift. Two methods were compared:

- **NS-off**: Conventional quantization with μ-law companding. This corresponds to the proposed method for $\gamma = 0.0$.
- **NS-on**: Mel-cepstrum-based quantization noise shaping using μ-law companding.

4.4.2 Objective evaluation

Before modeling quantized speech waveform using the WaveNet model, we evaluated the quantized speech waveform using the mean opinion score-listening quality objective (MOS-LQO) [80, 81] measure calculated from the perceptual evaluation of speech

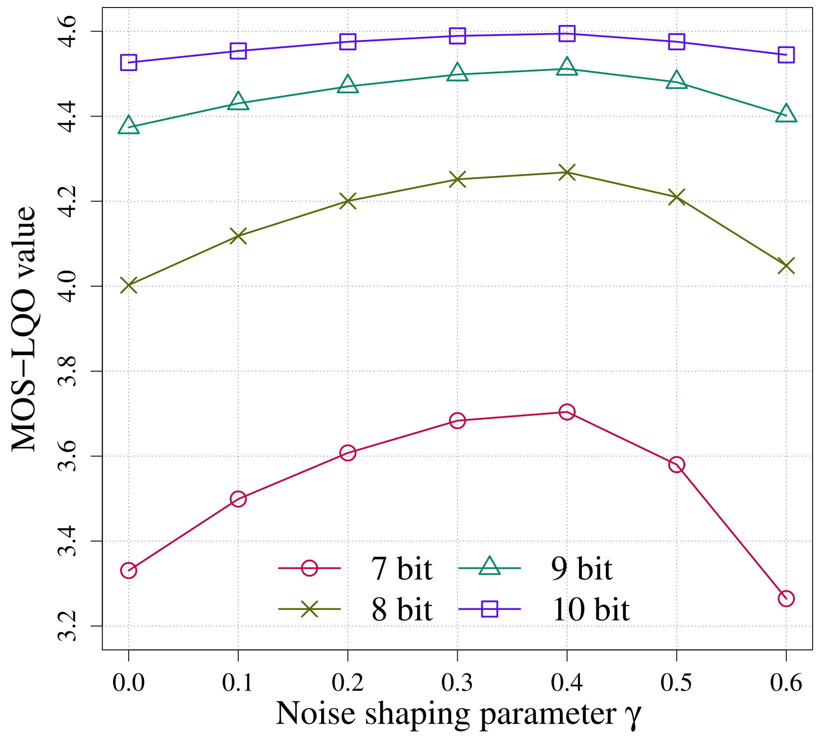


Figure 4.10: MOS-LQO value of speech quantized using mel-cepstrum-based quantization noise shaping.

quality (PESQ) [82], which is a widely used perceptual measure for evaluating voice quality in telecommunications. Figure 4.10 shows the results for different values of the noise shaping parameter γ and the quantization level q. They are consistent with those of our informal listening test. The proposed noise shaping method achieved a high score when an appropriately tuned γ was used. When γ was too large, the score was low. This is because the signal power was excessively amplified by the noise shaping filter. As expected, the noise shaping was more effective for lower quantization levels. However, improvement in the score was observed even when it was applied to 10-bit quantization. In subsequent experiments, we set γ to 0.4 and used 8-bit quantization in accordance with the previous work [22, 23].

4.4.3 Subjective evaluation

We subjectively assessed speech quality on the basis of opinion equivalent-Q [83]. The participants rated not only target speech of **NS-off** and **NS-on** but also original speech degraded by passing through a modified noise reference unit (MNRU) [84] with various

Q (30, 35, 40, and 45 dB) on a 5-point scale, where Q is the ratio of speech power to modulated noise power. The participants were 10 Japanese. Each of participants evaluated 10 sentences randomly chosen from the 40 test sentences, i.e., each of participants rated 60 sentences (10 sentences $\times$ 6 methods). Although MOS [85] is a simple and convenient indicator of speech quality, it is easily influenced by various internal/external experimental factors. While the absolute value of MOS may differ from experiment to experiment, the relative quality between the MNRU and target speech should be preserved over different subjective experiments. Thus, the use of MNRU removes the experimental dependence of the MOS test [86]. The subjective experiments were performed in a soundproof room. Stimuli were played to participants through headphones. The average duration of stimuli was about three seconds. Speech volume was adjusted appropriately for each participants.

First, we evaluated raw quantized speech signals to measure the effectiveness of the proposed method without prefiltering. The relationship between MOS and MNRU values is shown in Fig. 4.11. **NS-on** significantly increased MOS by more than 0.6 from **NS-off**. This is equivalent to a 4.0 dB improvement in equivalent-Q value, indicating the effectiveness of mel-cepstrum-based noise shaping. The main reason for the gain is that quantization noise spectrally shaped by the proposed method was more difficult for the participants to perceive.

In a subsequent experiment, we evaluated synthetic speech generated by the WaveNet model. The number of sentences used for training the WaveNet was 1091, which is not included in the test sentences. The dilations of the WaveNet model were set to 1, 2, 4, . . ., 512. The 10 dilation layers were stacked three times, resulting in a receptive field with a size of 3072. The channel size for dilation, residual block, and skip-connection were 128, 256, and 256, respectively. We used the Adam solver [54] with an initial learning late of 0.001 for training the WaveNet model. The WaveNet model was used as a vocoder [87], not as a TTS model. The auxiliary features, $\boldsymbol{h}$, were composed of 24-th order mel-cepstral coefficients, which is the same as used in noise shaping, a log-fundamental frequency ($\log F_0$) value, and a voiced/unvoiced binary symbol. The $\log F_0$ were extracted using a robust algorithm for pitch tracking (RAPT) [88] with a 5-ms shift.

Figure 4.12 shows the relationship between MOS and MNRU values. As with the previous experiment, **NS-on** significantly increased MOS by more than 0.6 from **NS-off**. This is equivalent to a 4.0 dB improvement in equivalent-Q value. Although the equivalent-Q values were lower than those in the previous experiment due to the statistical modeling process, the relative improvement was almost the same. This indicates that the proposed method is effective for neural-network-based raw audio speech waveform modeling.

In the next evaluation, mel-cepstrum-based prefiltering was evaluated using MOS instead of the equivalent-Q method due to a number of samples to be evaluated. We compared

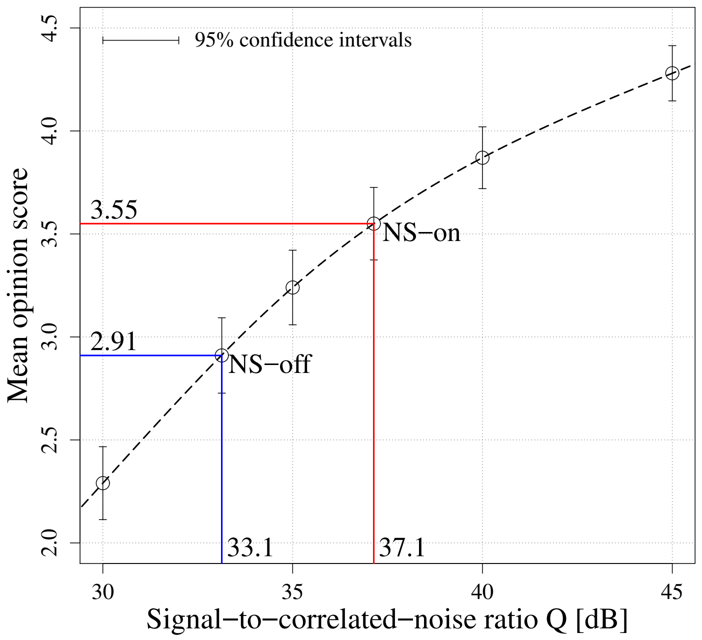


Figure 4.11: Relationship between MOS and MNRU values where target speech is quantized speech.

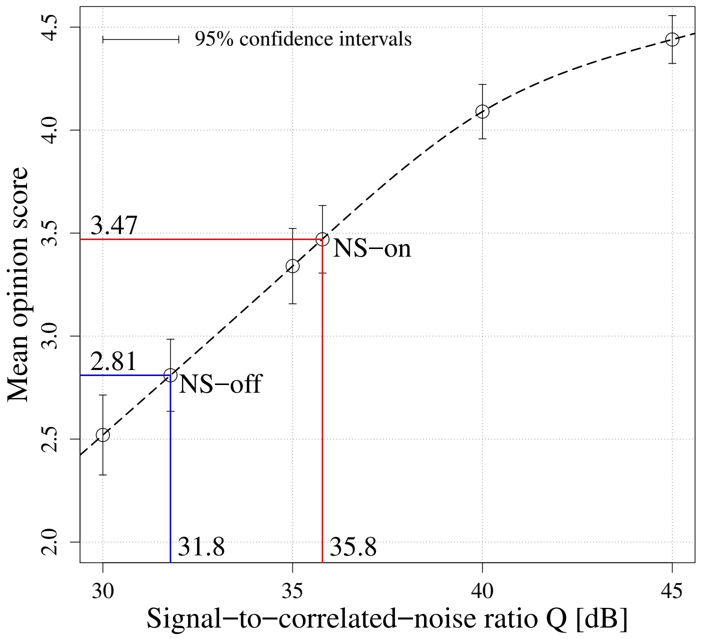


Figure 4.12: Relationship between MOS and MNRU values where target speech is generated by WaveNet model.

each combination of with/without mel-cepstrum-based noise shaping and prefiltering. Table 4.1 shows the details of the four methods. The experimental result is shown in Figure 4.13. **NS-on_PF-off** and **NS-off_PF-on** outperformed **NS-off_PF-off**. This indicates that both the mel-cepstrum-based noise shaping and prefiltering are effective to improve speech quality. Alghouth the MOSs of **NS-on_PF-off** and **NS-off_PF-on** are almost same, **NS-on_PF-on** achieved a higher score than these methods. It can be seen that the proposed mel-cepstrum-based noise shaping and prefiltering have different effects on human perceptual system, and the combination of them is effective for neural-network-based speech waveform speech syntheis.

Finally, the proposed method was evaluated in a TTS task. The WaveNet was feed with the acoustic features $\boldsymbol{h}$ predicted by a feed-forward neural network with three hidden layers. The phone durations of speech were predicted by five-state left-to-right hidden semi-Markov models [47]. The recipe for training these models was the same as the HTS demo script[1]. The WaveNet model was trained using the predicted acoustic features rather than those extracted from natural speech. The other experimental condition was the same as the previous experiment. Since prosody of natural speech and one of speech synthesized from text differs, a preference test (AB test) was conducted instead of the equivalent-Q method for evaluation. The result of the preference test is shown in Fig. 4.14. **NS-on** significantly outperformed **NS-off**. It can be said that the proposed method is also effective in a TTS task.

4.5 Summary

We have developed a mel-cepstrum-based quantization noise shaping method and applied it to a neural-network-based speech waveform synthesis system using the WaveNet model. Objective and subjective experiments showed that the proposed quantization noise shaping significantly improves speech quality by masking quantization noise. Future work

Table 4.1: Methods compared by using MOS.

Name	γ (for noise shaping)	β (for prefiltering)
NS-off_PF-off	0.0	0.0
NS-on_PF-off	0.4	0.0
NS-off_PF-on	0.0	0.2
NS-on_PF-on	0.4	0.2

[1]`http://hts.sp.nitech.ac.jp/?Download`

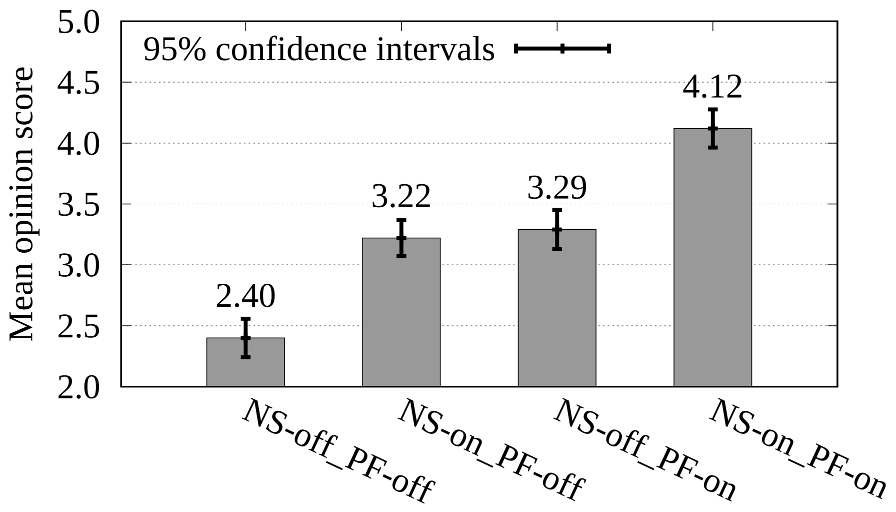


Figure 4.13: MOS on speech quality with 95% confidence intervals.

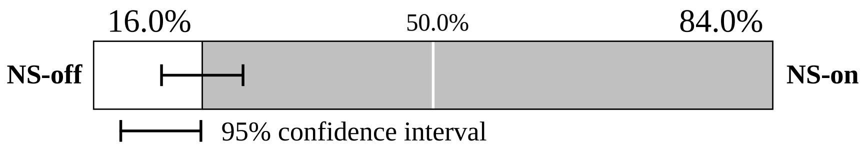


Figure 4.14: Result of preference test of speech quality with a 95% confidence interval.

includes integrating the training of the WaveNet with noise shaping.

Chapter 5

Conclusions

The paper described techniques to let machines speak naturally like humans with various speaker characteristics.

In the past decades, hidden Markov model (HMM)-based speech synthesis systems have been widely developed. The basic theories and the fundamental algorithms of the HMM were reviwed in Chapter 2. Chapter 2 also described the basic idea and the basic algorithms of the factor analyzed HMM, which is a probabilistic version of the eigenvoice method. The FAHMM is a well-established framework that can control characteristics of synthetic speech. Although the model structures have no constraints under the FAHMM framework, a single binary decision tree is typically used. Chapter 3 also proposed a novel context clustering technique for building the multiple-tree structure of the FAHMM with computational complexity reduction algorithms. Both objective and subjective experiments showed that the proposed method significantly outperforms the conventional method based on a single model structure. It can be said that taking speaker characteristics into account modeling the relation between linguistic and acoustic features is essential for modeling heterogenious speech data.

The paper also tackled to produce synthetic speech with high naturalness using the WaveNet generative model, which is a state-of-the-art model for neural-network-based speech waveform synthesis. The overview of the WaveNet model was presented in Chapter 2. One of the key techniques of neural-network-based speech waveform synthesis is modeling speech signals composed of a set of discrete values instead of continuous ones using quantization. However, this introduces white noise, which is easily perceived by human listeners. To solve the problem, mel-cepstral-based quantization noise shaping technique was proposed and investigated in Chapter 4. Using the proposed method, some of the quantization noise should be difficult for a human listener to perceive because mel-cepstrum is based on the human auditory system. Objective and subjective experiments showed that the proposed quantization noise shaping significantly improves speech quality by masking quantization noise. The combination of the mel-cepstrum-based quantization noise shaping and prefiltering achieved a significant improvement.

Future work includes integrating techniques for generating speech with high naturalness and controlling the characteristics.

Bibliography

[1] K. Tokuda, Y. Nankaku, T. Toda, H. Zen, J. Yamagishi, and K. Oura. Speech synthesis based on hidden Markov models. *Proceedings of the IEEE*, Vol. 101, No. 5, pp. 1234–1252, 2013.

[2] H. Zen, A. Senior, and M. Schuster. Statistical parametric speech synthesis using deep neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7962–7966, 2013.

[3] A. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. *1996 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 373–376, 1996.

[4] S. King. Measuring a decade of progress in text-to-speech. *Loquens*, Vol. 1, No. 1, 2014.

[5] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Speaker interpolation in HMM-based speech synthesis system. *Proceedings of Eurospeech 1997*, pp. 2523–2526, 1997.

[6] M. Tachibana, J. Yamagishi, T. Masuko, and T. Kobayashi. Speech synthesis with various emotional expressions and speaking styles by style interpolation and morphing. *Proceedings of IEICE Transactions on Information and Systems*, Vol. E88–D, No. 11, pp. 2484–2491, 2005.

[7] Roland Kuhn, Jean-Claude Junqua, Patrick Nguyen, and Nancy Niedzielski. Rapid speaker adaptation in eigenvoice space. *Proceedings of IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 6, pp. 695–707, 2000.

[8] K. Shichiri, A. Sawabe, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Eigenvoices for HMM-based speech synthesis. *Proceedings of ICSLP*, pp. 1269–1272, 2002.

[9] K. Kazumi, Y. Nankaku, and K. Tokuda. Factor analyzed voice models for HMM-based speech synthesis. *Proceedings of ICASSP 2010*, pp. 4234–4237, 2010.

[10] P. Nguyen, C. Wellekens, and J.-C. Junqua. Maximum likelihood eigenspace and MLLR for speech recognition in noisy environments. *Proceedings of Eurospeech 1999*, Vol. 6, pp. 2519–2522, 1999.

[11] Katsuhisa Fujinaga, Mitsuru Nakai, Hiroshi Shimodaira, and Shigeki Sagayama. Multiple-regression hidden Markov model. *Proceedings of ICASSP 2001*, Vol. 1, pp. 513–516, 2001.

[12] Takashi Nose, Makoto Tachibana, and Takao Kobayashi. HMM-based style control for expressive speech synthesis with arbitrary speaker's voice using model adaptation. *IEICE Transactions on Information and Systems*, Vol. E92–D, No. 3, pp. 489–497, 2009.

[13] M. J. F. Gales. Cluster adaptive training of hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 4, pp. 417–428, 2000.

[14] Javier Latorre, Vincent Wan, Mark J. F. Gales, Langzhou Chen, K. K. Chin, Kate Knill, and Masami Akamine. Speech factorization for HMM-TTS based on cluster adaptive training. *Proceedings of Interspeech 2012*, pp. 971–974, 2012.

[15] J. J. Odell. The use of context in large vocabulary speech recognition. *PhD dissertation, Cambridge University*, 1995.

[16] K. Oura, H. Zen, A. Lee, and K. Tokuda. A covariance-tying technique for HMM-based speech synthesis. *Proceedings of IEICE*, Vol. E93–D, No. 3, pp. 595–601, 2010.

[17] Shinji Takaki, Yoshihiko Nankaku, and Keiichi Tokuda. Contextual additive structure for HMM-based speech synthesis. *IEEE Journal of Selected Topics in Signal Processing*, Vol. 8, No. 2, pp. 229–238, 2014.

[18] H. Kawahara, I. Masuda-Katsuse, and A. Cheveigne. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech Communication*, Vol. 27, pp. 187–207, 1999.

[19] 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*, Vol. E99-D, No. 7, pp. 1877–1884, 2016.

[20] Keiichi Tokuda and Heiga Zen. Directly modeling speech waveforms by neural networks for statistical parametric speech synthesis. *Proceedings of ICASSP 2015*, pp. 4215–4219, 2015.

[21] Keiichi Tokuda and Heiga Zen. Directly modeling voiced and unvoiced components in speech waveforms by neural networks. *Proceedings of ICASSP 2016*, pp. 5640–5644, 2016.

[22] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[23] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv:1612.07837*, 2016.

[24] Pulse code modulation (PCM) of voice frequencies. *ITU-T Recommendation G.711*, 1988.

[25] 7 kHz audio-coding within 64 kbit/s. *ITU-T Recommendation G.722*, 1988.

[26] Keiichi Tokuda, Takao Kobayashi, Takeshi Chiba, and Satoshi Imai. Spectral estimation of speech by mel-generalized cepstral analysis. *Electronics and Communications in Japan (Part 3)*, Vol. 76, No. 2, pp. 30–43, 1993.

[27] T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai. An adaptive algorithm for mel-cepstral analysis of speech. *Proceedings of ICASSP 1992*, pp. 137–140, 1992.

[28] Satoshi Imai, Kazuo Sumita, and Chieko Furuichi. Mel log spectrum approximation (MLSA) filter for speech synthesis. *Electronics and Communications in Japan*, Vol. 66, No. 2, pp. 11–18, 1983.

[29] Oliver Watts, Gustav Eje Henter, Thomas Merritt, Zhizheng Wu, and Simon King. From HMMs to DNNs: where do the improvements come from? *Proceedings of ICASSP 2016*, pp. 5505–5509, 2016.

[30] Keiichiro Oura, Yoshihiko Nankaku, Tomoki Toda, Keiichi Tokuda, Ranniery Maia, Shinsuke Sakai, and Satoshi Nakamura. Simultaneous acoustic, prosodic, and phrasing model training for TTS conversion systems. *Proceedings of ISCSLP2008*, pp. 1–4, 2008.

[31] Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le,

Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. *Proceedings of Interspeech 2017*, pp. 4006–4010, 2017.

[32] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. *arXiv:1712.05884*, 2017.

[33] Kazuhiro Nakamura, Kei Hashimoto, Yoshihiko Nankaku, and Keiichi Tokuda. Integration of spectral feature extraction and modeling for HMM-based speech synthesis. *IEICE Transactions on Information and Systems*, Vol. E97-D, No. 6, pp. 1438–1448, 2014.

[34] X. D. Huang, Y. Ariki, and M. A. Jack. Hidden Markov models for speech recognition. *Edinburgh University Press*, 1990.

[35] L. Rabiner and B. H. Juang. Fundamentals of speech recognition. *Prentice-Hall*, 1993.

[36] S. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. The HTK book (for HTK Version 3.3). *Cambridge University*, 2005.

[37] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, Vol. 13, pp. 260–269, 1967.

[38] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B (Methodological)*, Vol. 39, No. 1, pp. 1–38, 1977.

[39] B. H. Juang. Maximum likelihood estimation for mixture of multivariate stochastic observations of Markov chains. *AT&T Technical Journal*, Vol. 64, No. 6, pp. 1235–1249, 1985.

[40] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, H. Zen, J. Yamagishi, and K. Oura. Speech synthesis based on hidden Markov models. *Proceedings of the IEEE*, Vol. 101, No. 5, pp. 1234–1252, 2013.

[41] K. Tokuda, T. Kobayashi, and S. Imai. Speech parameter generation from HMM using dynamic features. *Proceedings of International Conference on Acoustics, Speech, and Signal Processing'95*, pp. 660–663, 1995.

[42] K. Tokuda, T. Masuko, Y. Yamada, T. Kobayashi, and S. Imai. An algorithm for speech parameter generation from continuous mixture HMMs with dynamic features. *Proceedings of European Conference on Speech Communication and Technology'95*, pp. 757–760, 1995.

[43] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. *2000 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 936–939, 2000.

[44] S. Young, J. J. Odell, and P. Woodland. Tree-based state tying for high accuracy acoustic modelling. *Proceedings of ARPA Workshop on Human Language Technology*, pp. 307–312, 1994.

[45] K. Shinoda and T. Watanabe. MDL-based context-dependent subword modeling for speech recognition. *Acoustical Science and Technology*, Vol. 21, No. 2, pp. 79–86, 2000.

[46] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Duration modeling for HMM-based speech synthesis. *Proceedings of International Conference on Spoken Language Processing'98*, Vol. 2, pp. 29–32, 1998.

[47] H. Zen, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Hidden semi-Markov model based speech synthesis. *8th International Conference on Spoken Language Processing*, pp. 1185–1180, 2004.

[48] M. Tamura, T. Masuko, K. Tokuda, and T. Kobayashi. Adaptation of pitch and spectrum for HMM-based speech synthesis using MLLR. *Proceedings of ICASSP 2001*, pp. 805–808, 2001.

[49] Heiga Zen and Norbert Braunschweiler. Context-dependent additive log F0 model for HMM-based speech synthesis. *Proceedings of Interspeech 2009*, pp. 2091–2094, 2009.

[50] H. Attias. Inferring parameters and structure of latent variable models by variational Bayes. *Proceedings of UAI 15*, 1999.

[51] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 807–814, 2010.

[52] R. Hecht-Nielsen. Theory of the backpropagation neural network. *International 1989 Joint Conference on Neural Networks*, Vol. 1, pp. 593–605, 1989.

[53] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159, 2011.

[54] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[55] K. Rao, F. Peng, H. Sak, and F. Beaufays. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. *Proceedings of ICASSP 2015*, pp. 4225–4229, 2015.

[56] Ling-Hui Chen, Tuomo Raitio, Cassia Valentini-Botinhao, Junichi Yamagishi, and Zhen-Hua Ling. DNN-based stochastic postfilter for HMM-based speech synthesis. *Proceedings of Interspeech 2014*, pp. 1954–1958, 2014.

[57] Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda. The effect of neural networks in statistical parametric speech synthesis. *Proceedings of ICASSP 2015*, pp. 4455–4459, 2015.

[58] Heiga Zen and Andrew Senior. Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. *Proceedings of ICASSP 2014*, pp. 3872–3876, 2014.

[59] Heiga Zen and Hasim Sak. Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis. *Proceedings of ICASSP 2015*, pp. 4470–4474, 2015.

[60] K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi. Hidden Markov models based on multi-space probability distribution for pitch pattern modeling. *Proceedings of ICASSP*, pp. 229–232, 1999.

[61] M. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, Vol. 12, No. 2, pp. 75–98, 1998.

[62] J. Yamagishi and T. Kobayashi. Average-voice-based speech synthesis using HSMM-based speaker adaptation and adaptive training. *IEICE Transactions on Information & Systems*, Vol. E90-D, No. 2, pp. 533–543, 2007.

[63] J. Yamagishi, T. Masuko, K. Tokuda, and T. Kobayashi. A training method for average voice model based on shared decision tree context clustering and speaker adaptive training. *Proceedings of ICASSP 2003*, Vol. 1, pp. 716–719, 2003.

[64] R. F. Kubichek. Mel-cepstral distance measure for objective speech quality assessment. *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Vol. 1, pp. 125–128, 1993.

[65] J.-L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 2, pp. 291–298, 1994.

[66] T. Masuko, K. Tokuda, T. Kobayashi, and S. Imai. Voice characteristics conversion for HMM-based speech synthesis system. *Proceedings of ICASSP 1997*, Vol. 3, pp. 1611–1614, 1997.

[67] Zhizheng Wu, Pawel Swietojanski, Christophe Veaux, Steve Renals, and Simon King. A study of speaker adaptation for DNN-based speech synthesis. *Proceedings of Interspeech 2015*, pp. 879–883, 2015.

[68] Yuchen Fan, Yao Qian, Frank K. Soong, and Lei He. Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis. *Proceedings of ICASSP 2015*, pp. 4475–4479, 2015.

[69] Chunyang Wu and M. J. F. Gales. Multi-basis adaptive neural network for rapid adaptation in speech recognition. *Proceedings of ICASSP 2015*, pp. 4315–4319, 2015.

[70] Tian Tan, Yanmin Qian, Maofan Yin, Yimeng Zhuang, and Kai Yu. Cluster adaptive training for deep neural network. *Proceedings of ICASSP 2015*, pp. 4325–4329, 2015.

[71] Santiago Pascual and Antonio Bonafonte. Multi-output RNN-LSTM for multiple speaker speech synthesis with alpha-interpolation model. *Proceedings of 9th ISCA Speech Synthesis Workshop*, pp. 119–124, 2016.

[72] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv:1601.06759*, 2016.

[73] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. Discrete-time signal processing (second edition). *Prentice-Hall*, 1999.

[74] Jae S. Lim and A. V. Oppenheim. Reduction of quantization noise in PCM speech coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 28, No. 1, pp. 107–110, 1980.

[75] Valeri Mladenov, Panagiotis Karampelas, Georgi Tsenov, and Vassiliki Vita. Approximation formula for easy calculation of signal-to-noise ratio of sigma-delta modulators. *ISRN Signal Processing*, Vol. 2011, , 2011.

[76] Gunnar Fant. Speech sounds and features. *The MIT Press*, 1973.

[77] Keiichi Tokuda, Hidetoshi Matsumura, Takao Kobayashi, and Satoshi Imai. Speech coding based on adaptive mel-cepstral analysis. *Proceedings of ICASSP 1994*, Vol. 1, pp. 197–200, 1994.

[78] S. Imai. Cepstral analysis synthesis on the mel frequency scale. *Proceedings of International Conference on Acoustics, Speech, and Signal Processing'83*, pp. 93–96, 1983.

[79] John Kominek and Alan W Black. CMU ARCTIC databases for speech synthesis. *Technical Report CMU-LTI-03-177*, pp. 1–19, 2003.

[80] Mapping function for transforming P.862 raw result scores to MOS-LQO. *ITU-T Recommendation P.862.1*, 2003.

[81] Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs. *ITU-T Recommendation P.862.2*, 2007.

[82] Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *ITU-T Recommendation P.862*, 2001.

[83] Subjective performance assessment of telephone-band and wideband digital codecs. *ITU-T Recommendation P.830*, 1996.

[84] Modulated noise reference unit (MNRU). *ITU-T Recommendation P.810*, 1996.

[85] Methods for subjective determination of transmission quality. *ITU-T Recommendation P.800*, 1996.

[86] Syed A. Ahson and Mohammad Ilyas. Voip handbook: Applications, technologies, reliability, and security. *CRC Press*, 2008.

[87] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda. Speaker-dependent WaveNet vocoder. *Proceedings of Interspeech 2017*, pp. 1118–1122, 2017.

[88] D. Talkin. A robust algorithm for pitch tracking (RAPT). pp. 497–518. Elsevier, 1995.

List of Publications

Journal papers

[1] **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Simultaneous optimization of multiple tree-based factor analyzed HMM for speech synthesis," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 25, no. 9, pp. 1532–1541, Sep. 2017.

[2] **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Mel-cepstrum-based quantization noise shaping applied to neural-network-based speech waveform synthesis," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 26, no. 7, pp. 1173–1180, Jul. 2018.

International conference proceedings

[3] **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Cross-lingual speaker adaptation based on factor analysis using bilingual speech data for HMM-based speech synthesis," Proceedings of 8th ISCA Speech Synthesis Workshop, pp. 297–302, Barcelona, Spain, Sep. 2013.

[4] **Takenori Yoshimura**, Kei Hashimoto, Yoshihiko Nankaku, and Keiichi Tokuda, "Simultaneous optimization of multiple tree structures for factor analyzed HMM-based speech synthesis," Proceedings of Interspeech 2015, pp. 1196–1200, Dresden, Germany, Sep. 2015.

[5] **Takenori Yoshimura**, Gustav Eje Henter, Oliver Watts, Mirjam Wester, Junichi Yamagishi, and Keiichi Tokuda, "A hierarchical predictor of synthetic speech naturalness using neural networks," Proceedings of Interspeech 2016, pp. 342–346, San Francisco, USA, Sep. 2016.

[6] Amelia J. Gully, **Takenori Yoshimura**, Damian T. Murphy, Kei Hashimoto, Yoshihiko Nankaku, and Keiichi Tokuda, "Articulatory text-to-speech synthesis using the digital waveguide mesh driven by a deep neural network," Proceedings of Interspeech 2017, pp. 234–238, Stockholm, Sweden, Aug. 2017.

[7] Jumpei Niwa, **Takenori Yoshimura**, Kei Hashimoto, Yoshihiko Nankaku, and Keiichi Tokuda, "Statistical voice conversion based on WaveNet," Proceedings of ICASSP 2018, pp. 5289–5293, Calgary, Canada, Apr. 2018.

Technical reports

[7] **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Mel-cepstrum based quantization noise shaping applied to speech synthesis based on WaveNet," IEICE Technical Report, vol. 117, no. 393, SP2017-83, pp. 93–98, Jan. 2018 (in Japanese).

[8] Jumpei Niwa, **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "A study on voice conversion based on WaveNet," IEICE Technical Report, vol. 117, no. 393, SP2017-84, pp. 99–104, Jan. 2018 (in Japanese).

Domestic conference proceedings

[9] **Taknenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Cross-lingual speaker adaptation based on factor analysis using bilingual speech data," Proceedings of ASJ 2013 Spring Meeting, pp. 267–268, Mar. 2013 (in Japanese).

[**10**] **Takenori Yoshimura**, Kei Hashimoto, Yoshihiko Nankaku, and Keiichi Tokuda, "A clustering technique for factor analyzed HMM-based speech synthesis," Proceedings of ASJ 2014 Autumn Meeting, pp. 239–240, Sep. 2014 (in Japanese).

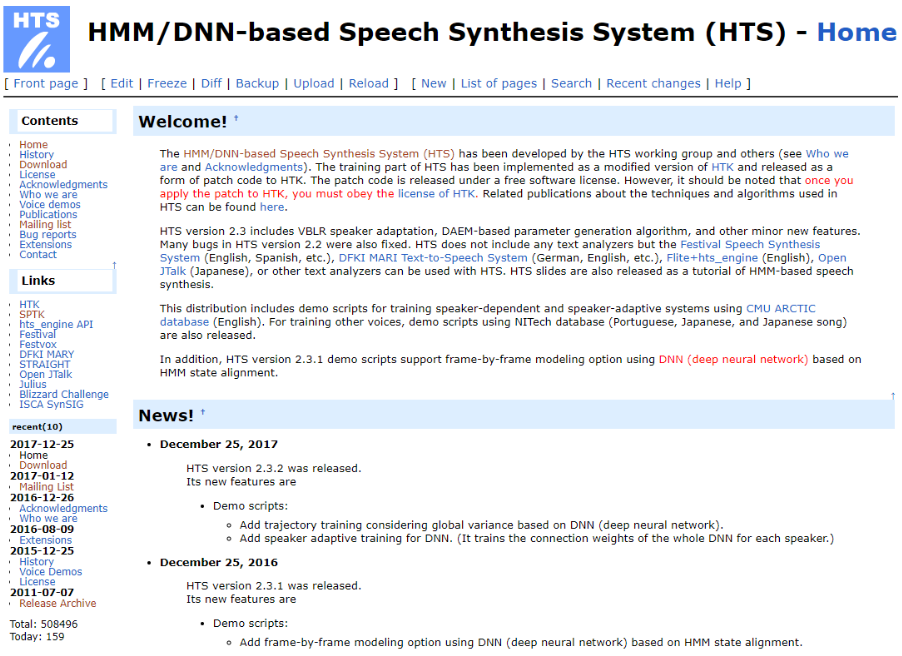
[**11**] **Taknenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "A design of voice recording software tool to effectively collect speech data based on crowdsourcing," Proceedings of ASJ 2016 Spring Meeting, pp. 307–308, Mar. 2016 (in Japanese).

[**12**] **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "Mel-cepstrum based quantization noise shaping applied for WaveNet," Proceedings of ASJ 2017 Autumn Meeting, pp. 193–194, Sep. 2017 (in Japanese) (**Student Presentation Award**).

[**13**] Jumpei Niwa, **Takenori Yoshimura**, Kei Hashimoto, Keiichiro Oura, Yoshihiko Nankaku, and Keiichi Tokuda, "WaveNet-based voice conversion," Proceedings of ASJ 2017 Autumn Meeting, pp. 207–208, Sep. 2017.

Appendix A

Software



HMM/DNN-based Speech Synthesis System (HTS) - Home

[Front page] [Edit | Freeze | Diff | Backup | Upload | Reload] [New | List of pages | Search | Recent changes | Help]

Contents

- Home
- History
- Download
- License
- Acknowledgments
- Who we are
- Voice demos
- Publications
- Mailing list
- Bug reports
- Extensions
- Contact

Links

- HTK
- SPTK
- hts_engine API
- Festival
- Festvox
- DFKI MARY
- STRAIGHT
- Open JTalk
- Julius
- Blizzard Challenge
- ISCA SynSIG

recent(10)

2017-12-25
- Home
- Download

2017-01-12
- Mailing List

2016-12-26
- Acknowledgments
- Who we are

2016-08-09
- Extensions

2015-12-25
- History
- Voice Demos
- License

2011-07-07
- Release Archive

Total: 508496
Today: 159

Welcome! †

The HMM/DNN-based Speech Synthesis System (HTS) has been developed by the HTS working group and others (see Who we are and Acknowledgments). The training part of HTS has been implemented as a modified version of HTK and released as a form of patch code to HTK. The patch code is released under a free software license. However, it should be noted that once you apply the patch to HTK, you must obey the license of HTK. Related publications about the techniques and algorithms used in HTS can be found here.

HTS version 2.3 includes VBLR speaker adaptation, DAEM-based parameter generation algorithm, and other minor new features. Many bugs in HTS version 2.2 were also fixed. HTS does not include any text analyzers but the Festival Speech Synthesis System (English, Spanish, etc.), DFKI MARI Text-to-Speech System (German, English, etc.), Flite+hts_engine (English), Open JTalk (Japanese), or other text analyzers can be used with HTS. HTS slides are also released as a tutorial of HMM-based speech synthesis.

This distribution includes demo scripts for training speaker-dependent and speaker-adaptive systems using CMU ARCTIC database (English). For training other voices, demo scripts using NITech database (Portuguese, Japanese, and Japanese song) are also released.

In addition, HTS version 2.3.1 demo scripts support frame-by-frame modeling option using DNN (deep neural network) based on HMM state alignment.

News! †

- **December 25, 2017**

  HTS version 2.3.2 was released.
  Its new features are

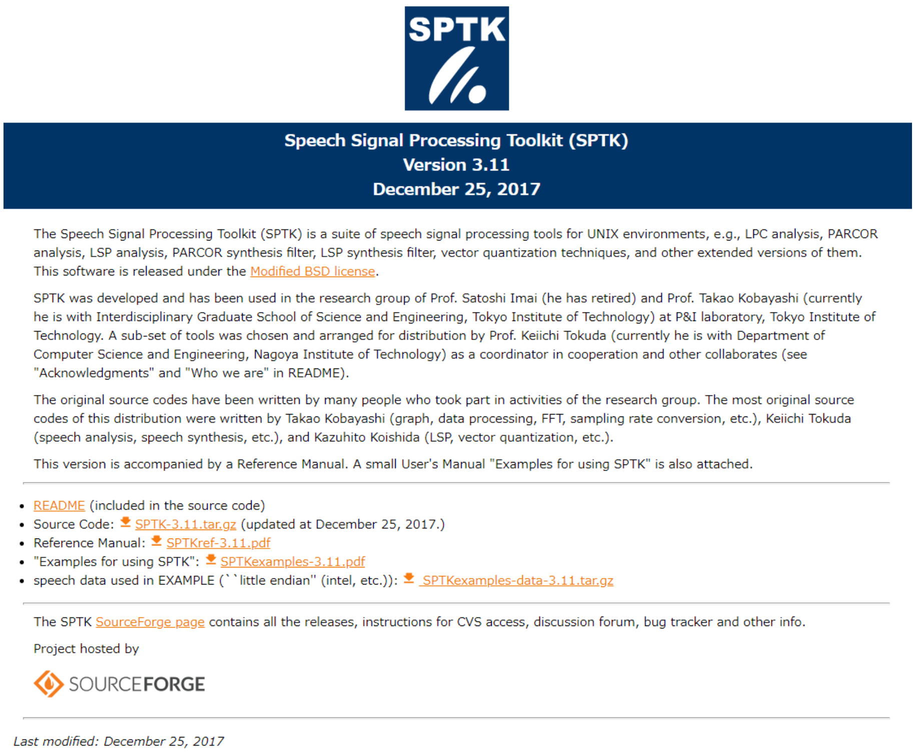
  - Demo scripts:
    - Add trajectory training considering global variance based on DNN (deep neural network).
    - Add speaker adaptive training for DNN. (It trains the connection weights of the whole DNN for each speaker.)

- **December 25, 2016**

  HTS version 2.3.1 was released.
  Its new features are

  - Demo scripts:
    - Add frame-by-frame modeling option using DNN (deep neural network) based on HMM state alignment.

Figure A.1: HTS: http://hts.sp.nitech.ac.jp/


Speech Signal Processing Toolkit (SPTK)
Version 3.11
December 25, 2017

The Speech Signal Processing Toolkit (SPTK) is a suite of speech signal processing tools for UNIX environments, e.g., LPC analysis, PARCOR analysis, LSP analysis, PARCOR synthesis filter, LSP synthesis filter, vector quantization techniques, and other extended versions of them. This software is released under the Modified BSD license.

SPTK was developed and has been used in the research group of Prof. Satoshi Imai (he has retired) and Prof. Takao Kobayashi (currently he is with Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology) at P&I laboratory, Tokyo Institute of Technology. A sub-set of tools was chosen and arranged for distribution by Prof. Keiichi Tokuda (currently he is with Department of Computer Science and Engineering, Nagoya Institute of Technology) as a coordinator in cooperation and other collaborates (see "Acknowledgments" and "Who we are" in README).

The original source codes have been written by many people who took part in activities of the research group. The most original source codes of this distribution were written by Takao Kobayashi (graph, data processing, FFT, sampling rate conversion, etc.), Keiichi Tokuda (speech analysis, speech synthesis, etc.), and Kazuhito Koishida (LSP, vector quantization, etc.).

This version is accompanied by a Reference Manual. A small User's Manual "Examples for using SPTK" is also attached.

- README (included in the source code)
- Source Code: SPTK-3.11.tar.gz (updated at December 25, 2017.)
- Reference Manual: SPTKref-3.11.pdf
- "Examples for using SPTK": SPTKexamples-3.11.pdf
- speech data used in EXAMPLE (``little endian'' (intel, etc.)): SPTKexamples-data-3.11.tar.gz

The SPTK SourceForge page contains all the releases, instructions for CVS access, discussion forum, bug tracker and other info.

Project hosted by

SOURCEFORGE

Last modified: December 25, 2017

Figure A.2: SPTK: http://sp-tk.sourceforge.net/