ROBUST SPEECH FEATURES AND ACOUSTIC MODELS FOR SPEECH RECOGNITION

XIAO XIONG

SCHOOL OF COMPUTER ENGINEERING

2009
ROBUST SPEECH FEATURES AND ACOUSTIC MODELS FOR SPEECH RECOGNITION

XIAO XIONG

School of Computer Engineering

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirement for the degree of Doctor of Philosophy

2009
Acknowledgments

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Eng Siong Chng (Nanyang Technological University, NTU), and co-supervisor, Dr. Haizhou Li (Institute for Infocomm Research, I²R) for their invaluable guidance, support and suggestions. Their knowledge, suggestions, and discussions help me to become a capable researcher. Their encouragement also helps me to overcome the difficulties encountered in my research.

My thanks also go to Prof. Chin-Hui Lee and Dr. Jinyu Li for their help, discussions, and experimental setup for soft-margin estimation (SME) during my internship in Georgia Institute of Technology. Without their help, my work on SME will be much more difficult to accomplish.

I also want to thank my colleagues in NTU, Dr. Jinjun Wang, Lei Wang, Eugene Koh, Nguyen Trung Hieu, Omid Dehzangi, Ehsan Younessian, Di Gao, Jian Lai, and Shengkui Zhao. My PhD life is much more comfortable with their friendship.

I also want to thank my colleagues in the Speech and Dialogue Processing lab of I²R, including Dr. Bin Ma, Shuanghu Bai, Dr. George White, Swee Lan, Dr. Yeow Kee Tan, Rong Tong, Hendra, Dr. Tin Lay Nwe, Yu Chen and Boon Pang for their generous help.

Last but not least, I want to thank my family in China, for their constant love and encouragement.
# Contents

Acknowledgments .......................................................... 1  
Abstract ................................................................. 6  
List of Publications .................................................... 8  
List of Figures .......................................................... 10  
List of Tables ........................................................... 13  
List of Abbreviations ................................................... 15

1 Introduction ............................................................. 17  
  1.1 Contributions ....................................................... 18  
  1.2 Thesis Outline ...................................................... 19

2 Introduction of Speech Recognition .................................. 21  
  2.1 Overview of Statistical Speech Recognition System .......... 23  
  2.2 Feature Extraction ................................................ 24  
  2.3 Acoustic Model Training .......................................... 29  
      2.3.1 Hidden Markov Models ..................................... 30  
      2.3.2 Model Training Methods ................................... 31  
  2.4 Summary .......................................................... 39

3 Review of Robust Speech Recognition Techniques .................. 40  
  3.1 Overview of Noise Robust Speech Recognition ................. 40  
      3.1.1 Mismatches and Robust Approaches ...................... 40  
      3.1.2 About Notations Used in This Thesis ................... 43  
  3.2 Noise Effect in Filterbank and Cepstral Domains ............. 44  
      3.2.1 The phase-insensitive model ........................... 46
3.2.2 The phase-sensitive model ........................................ 47
3.2.3 Summary ......................................................... 48
3.3 Feature Domain Techniques ........................................... 48
  3.3.1 Speech Enhancement ............................................. 48
  3.3.2 Feature Compensation .......................................... 52
  3.3.3 Feature Normalization ......................................... 60
  3.3.4 Temporal Filters ................................................ 63
3.4 Model Adaptation Techniques ....................................... 66
  3.4.1 Data-driven-based Adaptation ................................ 66
  3.4.2 Environment Model-based Adaptation ....................... 70
3.5 Summary .......................................................... 72

4 Normalizing the Temporal Structure of Feature Trajectories 73
  4.1 Noise Effect in Modulation Spectra Domain ................... 74
    4.1.1 Definition of Modulation Spectra .......................... 75
    4.1.2 Importance of Modulation Spectra for ASR ............... 79
    4.1.3 Modulation Spectra of Clean Speech and Noises .......... 80
    4.1.4 Modulation Spectra of Noisy Speech ....................... 84
    4.1.5 Summary .................................................... 88
  4.2 Temporal Structure Normalization ............................... 88
    4.2.1 Overview of TSN Framework ................................ 88
    4.2.2 Filter Design ............................................. 89
    4.2.3 Training of Reference PSD Functions ..................... 93
    4.2.4 Comparison of TSN and other temporal filters .......... 95
  4.3 Segment-based Implementation ................................... 97
    4.3.1 Implementation Scheme .................................... 98
    4.3.2 Considerations ........................................... 99
  4.4 Effect of Temporal Structure Normalization on Features .... 99
    4.4.1 TSN alone ............................................... 99
    4.4.2 TSN combined with ARMA .................................. 103
  4.5 Speech Recognition Experiments ................................. 106
5 Improving Acoustic Model Generalization Capability

5.1 Introduction ............................................. 122
5.2 Noise Effect in Log Likelihood Domain .......................... 124
  5.2.1 A Two-Class Example ................................ 125
  5.2.2 Investigation of Noise Effect Using Monte Carlo Methods ... 128
  5.2.3 Empirical Study of Noise Effect on Speech Recognition ..... 132
  5.2.4 Summary ............................................. 140
5.3 Generalization Capability and Margin .......................... 140
5.4 Improving the Margin Using SME, MCE, and MMI .................. 143
5.5 Comparison of SME, MCE, and MMI ............................ 146
5.6 Experiments .............................................. 152
  5.6.1 System Description .................................. 152
  5.6.2 Effect of SME on separation measures .................... 155
  5.6.3 Effect of Margin Size ................................ 157
  5.6.4 Speech Recognition Performance on Aurora-2 ............... 159
  5.6.5 Interaction with MVN .................................. 161
  5.6.6 Interaction with HLDA ................................ 163
  5.6.7 Interaction with TSN .................................. 164
  5.6.8 Speech Recognition Performance on Aurora-3 ............... 166
5.7 Summary ............................................... 166
6 Conclusions

6.1 Contributions .............................................................. 168
    6.1.1 Temporal structure normalization filter ......................... 168
    6.1.2 Study of model generalization capability ........................ 169

6.2 Suggestion for Future Work ............................................. 170

Appendix

A. Mathematical Representation of Noisy Modulation Spectra ............ 172

References ........................................................................... 175
Abstract

This thesis examines techniques to improve the robustness of automatic speech recognition (ASR) systems against noise distortions. The study is important as the performance of ASR systems degrades dramatically in adverse environments, and hence greatly limits the speech recognition application deployment in realistic environments. Towards this end, we examine a feature compensation approach and a discriminative model training approach to improve the robustness of speech recognition system.

The degradation of recognition performance is mainly due to the statistical mismatch between clean-trained acoustical model and noisy testing speech features. To reduce the feature-model mismatch, we propose to normalize the temporal structure of both training and testing speech features. Speech features’ temporal structures are represented by the power spectral density (PSD) functions of feature trajectories. We propose to normalize the temporal structures by applying equalizing filters to the feature trajectories. The proposed filter is called temporal structure normalization (TSN) filter. Compared to other temporal filters used in speech recognition, the advantage of the TSN filter is its adaptability to changing environments. The TSN filter can also be viewed as a feature normalization technique that normalizes the PSD function of features, while other normalization methods, such as histogram equalization (HEQ), normalize the probability density function (p.d.f.) of features. Experimental study shows that the TSN filter produces better performance than other state-of-the-art temporal filters on both small vocabulary Aurora-2 task and large vocabulary Aurora-4 task.

In the second study, we improve the robustness of speech recognition by improving the generalization capability of acoustic model rather than reducing the feature-model mismatch. In the log likelihood score domain, noise distortion causes the log likelihood score of noisy features to deviate from that of clean features. The deviation may move
the noisy features to the wrong side of the decision boundary that is trained from clean features, and hence causes recognition error. To improve performance, discriminative training (DT) methods, including minimum classification error (MCE), maximum mutual information (MMI) and soft-margin estimation (SME), are applied to improve the generalization capability of the acoustic model, which in turn is implemented by increasing the margin, i.e. the desired minimum distance from training samples to the decision boundary. Experimental study shows that by improving the acoustic model’s generalization capability with SME and other DT training, speech recognition performance can be improved even when the testing data is mismatched from the training data. In addition, the margin-based SME is slightly more effective than MCE and MMI in terms of increasing the margin and robustness. It is also observed that DT methods are less effective on highly mismatched testing data. This limitation can be reduced by applying MVN on speech features before DT. In our experimental study, further improvement is obtained by applying DT methods to speech features already normalized by mean and variance normalization (MVN). This suggests that it may be a good strategy to combine SME with feature domain robust ASR techniques to produce good robustness.
List of Publications

Journal


Conference


List of Figures

2.1 The six modules of a modern ASR system ............................. 24
2.2 The common procedures of feature extraction (MFCC). .............. 26
2.3 The illustration of Mel windows. ..................................... 28
2.4 An illustration of HMM with three true states .......................... 30
2.5 Building of recognition network. ................................. 32

3.1 Distortions in feature and model spaces .............................. 41
3.2 Three approaches to reduce the effects of noise distortions ......... 42
3.3 Distortions in a typical speech communication system ............. 43
3.4 The environment model with additive noise and linear distortion. .. 45
3.5 Comparison of the magnitude responses of common temporal filters... 64

4.1 Illustration of the amplitude modulation characteristics of speech signal . 76
4.2 Illustration of the modulation spectra of speech signal ............. 77
4.3 Comparison of spectrogram and corresponding modulation spectra ... 78
4.4 Comparison of the count histogram of speech syllables and modulation spectrum ......................................................... 79
4.5 Comparison between speech and noise: their feature trajectories and modulation spectra ......................................................... 81
4.6 Comparison of the modulation spectra of speech and noise .......... 83
4.7 Speech feature trajectories and modulation spectra in different SNR levels 85
4.8 Speech modulation spectra in different SNR levels, from log filterbank coefficients and MFCC features ................................. 87
4.9 Illustration of the TSN framework ..................................... 88
4.10 Illustration of normalizing the trend of the PSD function only .......... 94
4.11 Demonstration of the variability of clean modulation spectra ........................................ 96
4.12 The processing scheme of the segment-based implementation of the TSN framework ................................................................. 98
4.13 The original feature trajectories ................................................................................. 100
4.14 Effect of TSN filter on feature trajectories ................................................................. 100
4.15 Effect of TSN filter on modulation spectra with/without weight normalization ........................................................................... 101
4.16 Magnitude response of the ARMA filter with order 3. ........................................... 103
4.17 Effect of the combination of TSN and ARMA filters on feature trajectories 104
4.18 Effect of the combination of TSN and ARMA filters on modulation spectra of a test utterance .............................................................. 105
4.19 Performance of segment-based TSN with different segment length on the Aurora-4 task ........................................................................ 112
4.20 The performance of two TSN configurations with different filter length 114
4.21 Illustration of modifying the TSN filter’s magnitude response ............................. 115
4.22 The performance of the TSN filter with different filter edge frequency ........ 116

5.1 Illustration of the two-class classification problem in log likelihood domain 126
5.2 Distribution of deviations in the log likelihood domain ............................................. 130
5.3 Illustration of noise effect in the two-class classification problem in the log likelihood domain ........................................................................ 131
5.4 Noise effect on phoneme pairs (sil-au) and (sil-f) in the log likelihood domain 135
5.5 Noise effect on phoneme pairs (eh-ey) and (v-f) in the log likelihood domain 136
5.6 Noise effect on phoneme pairs (s-th) and (uw-v) in the log likelihood domain 137
5.7 Histogram of separation measures for different SNR levels ................................ 139
5.8 Illustration of increasing margin to improve the generalization capability of acoustic model ........................................................................ 142
5.9 Contribution of a single training utterance $O_i$ to loss functions of DT methods ................................................................. 148
5.10 Contributions to the gradient of loss functions of DT methods .................................. 151
5.11 Effect of SME, MCE, and MMI training methods on the separation measures 154
5.12 Histogram of separation measure of training data with different $\lambda$. \ldots 159
5.13 System with MVN and HLDA \ldots \ldots \ldots \ldots \ldots \ldots \ldots 163
# List of Tables

4.1 Summary of TSN filter design procedures ........................................ 91
4.2 The settings for the Aurora-2 task. .................................................. 107
4.3 The settings for the Aurora-4 task. .................................................. 108
4.4 Performance of TSN on the Aurora-2 task ........................................... 109
4.5 Performance of TSN filter with the ARMA filter on the Aurora-2 task ...... 110
4.6 Performance of TSN on the Aurora-4 task ........................................... 111
4.7 Performance of applying the TSN filter to different features .................. 112
4.8 Details of baseline performance on the Aurora-2 task ........................... 118
4.9 Details of TSN performance on the Aurora-2 task ............................... 119
4.10 Details of TSN and ARMA filter performance on the Aurora-2 task ...... 120
4.11 Details of performance on the Aurora-4 task ..................................... 121

5.1 The settings for the Aurora-3 task. .................................................. 153
5.2 Comparison of SME effects on correctly and wrongly classified utterances 156
5.3 Performance of SME with MVN processed MFCC features with different
    \( \lambda \) values on Aurora-2 task ..................................................... 158
5.4 Performance of SME, MCE, and MMI with raw MFCC features on Aurora-
    2 task ......................................................................................... 161
5.5 Performance of SME, MCE, and MMI with raw features on the three test
    sets of the Aurora-2 task using multi-condition training ......................... 161
5.6 Performance of SME, MCE, and MMI with MVN-processed features on
    Aurora-2 task. ........................................................................... 162
5.7 Performance of SME and MCE with MVN-processed features on the three
    test sets of the Aurora-2 task using multi-condition training. ................ 162
5.8 Performance of SME with MVN-processed and HLDA-transformed features on Aurora-2 task. ........................................... 164
5.9 Performance of SME with SME-processed features on Aurora-2 task. . . 165
5.10 Performance of SME with raw MFCC features on Aurora-3 task . . . . . 167
5.11 Performance of SME with MVN-processed MFCC features on Aurora-3 task. ................................................................. 167
# List of Abbreviation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>Amplitude Modulation</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Networks</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive Moving Average</td>
</tr>
<tr>
<td>ASR</td>
<td>Automatic Speech Recognition</td>
</tr>
<tr>
<td>CDCN</td>
<td>Codeword-Dependent Cepstral Normalization</td>
</tr>
<tr>
<td>CMN</td>
<td>Cepstral Mean Normalization</td>
</tr>
<tr>
<td>CMS</td>
<td>Cepstral Mean Subtraction</td>
</tr>
<tr>
<td>CMU</td>
<td>Carnegie Mellon University</td>
</tr>
<tr>
<td>CVN</td>
<td>Cepstral Variance Normalization</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DT</td>
<td>Discriminative Training</td>
</tr>
<tr>
<td>ECSS</td>
<td>Energy-Constrained Signal Subspace</td>
</tr>
<tr>
<td>EM</td>
<td>Expectation Maximization</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>GMM</td>
<td>Gaussian Mixture Model</td>
</tr>
<tr>
<td>GPD</td>
<td>Generalized Probabilistic Descent</td>
</tr>
<tr>
<td>HEQ</td>
<td>Histogram Equalization</td>
</tr>
<tr>
<td>HLDA</td>
<td>Heteroscedastic Linear Discriminative Analysis</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td>HTK</td>
<td>HMM Toolkit</td>
</tr>
<tr>
<td>IDCT</td>
<td>Inverse Discrete Cosine Transform</td>
</tr>
<tr>
<td>IDFT</td>
<td>Inverse Discrete Fourier Transform</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>IRATZ</td>
<td>Interpolated RATZ</td>
</tr>
<tr>
<td>JAC</td>
<td>Joint Compensation Of Additive And Convolutive Noises</td>
</tr>
<tr>
<td>KLT</td>
<td>Karhunen-Loève Transform</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear Discriminative Analysis</td>
</tr>
<tr>
<td>LLR</td>
<td>Log Likelihood Ratio</td>
</tr>
<tr>
<td>LME</td>
<td>Large Margin Estimation</td>
</tr>
<tr>
<td>LPC</td>
<td>Linear Predictive Coefficients</td>
</tr>
<tr>
<td>Acronym</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>MAP</td>
<td>Maximum a posteriori</td>
</tr>
<tr>
<td>MCE</td>
<td>Minimum Classification Error</td>
</tr>
<tr>
<td>MEMLIN</td>
<td>Multi-Environment Model-based Linear Normalization</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel-frequency Cepstral Coefficient</td>
</tr>
<tr>
<td>MFT</td>
<td>Missing Feature Theory</td>
</tr>
<tr>
<td>ML</td>
<td>Maximum Likelihood</td>
</tr>
<tr>
<td>MLLR</td>
<td>Maximum Likelihood Linear Regression</td>
</tr>
<tr>
<td>MMI</td>
<td>Maximum Mutual Information</td>
</tr>
<tr>
<td>MMSE</td>
<td>Minimum Mean Square Error</td>
</tr>
<tr>
<td>MPE</td>
<td>Minimum Phone Error</td>
</tr>
<tr>
<td>MSE</td>
<td>Modulation Spectrum Equalizer</td>
</tr>
<tr>
<td>MTF</td>
<td>Modulation Transfer Function</td>
</tr>
<tr>
<td>MWE</td>
<td>Minimum Word Error</td>
</tr>
<tr>
<td>MVN</td>
<td>Mean and Variance Normalization</td>
</tr>
<tr>
<td>MVA</td>
<td>Mean, Variance and ARMA filtering</td>
</tr>
<tr>
<td>p.d.f.</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>PLP</td>
<td>Perceptual Linear Prediction</td>
</tr>
<tr>
<td>PMC</td>
<td>Parallel Model Combination</td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
</tr>
<tr>
<td>QSVD</td>
<td>Quotient Singular Value Decomposition</td>
</tr>
<tr>
<td>RASTA</td>
<td>Relative Spectra</td>
</tr>
<tr>
<td>RATZ</td>
<td>multivaRiate gaAussian-based cepStral normalized</td>
</tr>
<tr>
<td>SAP</td>
<td>Signal Absence Probability</td>
</tr>
<tr>
<td>SME</td>
<td>Soft-Margin Estimation</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SPLICE</td>
<td>Stereo-based Piecewise Linear Compensation for Environment</td>
</tr>
<tr>
<td>STAR</td>
<td>STAstatistical Reestimation</td>
</tr>
<tr>
<td>STFT</td>
<td>Short-Time Fourier Transform</td>
</tr>
<tr>
<td>STSA</td>
<td>Short-Time Spectral Amplitude</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machines</td>
</tr>
<tr>
<td>TES</td>
<td>Temporal Smoothing</td>
</tr>
<tr>
<td>TSN</td>
<td>Temporal Structure Normalization</td>
</tr>
<tr>
<td>TSNseg</td>
<td>Segment-based Temporal Structure Normalization</td>
</tr>
<tr>
<td>VAD</td>
<td>Voice Activity Detection</td>
</tr>
<tr>
<td>VC dim</td>
<td>Vapnik-Chervonenkis dimension</td>
</tr>
<tr>
<td>VTS</td>
<td>Vector Taylor Series</td>
</tr>
<tr>
<td>WER</td>
<td>Word Error Rate</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this thesis, we will investigate the robustness issues in current automatic speech recognition (ASR) systems and propose algorithms to improve ASR performance in adverse environments.

The objective of ASR is to recognize human speech, such as words and sentences, using algorithms executed in a computer. ASR is essentially a multi-class sequential pattern recognition task. The features are usually a sequence of representative vectors that are extracted from speech signals and the classes are either words or subword units such as phonemes. Current ASR systems usually model the probability distribution of feature vectors using hidden Markov model (HMM) [1-5] and adopts the maximum a posteriori (MAP) decision rule for classification [6]. Good recognition accuracy can be achieved when the data used for system training and testing are similar to each other.

A major issue in current ASR systems is their robustness against noise distortions [7-9]. It is found that ASR systems trained from clean speech data usually fail when tested on noise-distorted speech data. The performance degradation is mainly due to the statistical mismatch between the noisy test speech features and the clean-trained acoustic model of the ASR system. It is necessary to address this problem to enable the deployment of ASR systems in real world problems.

Many methods have been proposed to address the feature-model mismatch, and can be classified into two approaches: feature compensation approach and model adaptation approach. The feature compensation approach modifies the noisy features towards the unobserved clean features during or after the feature extraction process. Such techniques include speech enhancement techniques that are originally designed to enhance speech
signal for human listening [10–14], feature compensation techniques that try to estimate clean features from noisy features [15–18], feature normalization techniques that normalize both clean and noisy features to a new space where the noise distortion is reduced [19–26], and temporal filters that reduce the dissimilarity of features in the modulation spectrum domain [27–34], etc. In contrast, the model adaptation approach adapts the clean-trained acoustic model to better represent the noisy features. Examples of this approach include parallel model composition (PMC) [35], maximum a posteriori adaptation (MAP) [36], maximum likelihood linear regression adaptation (MLLR) [37], STAtistical Reestimation (STAR) [38] technique, joint compensation of additive and convolutive noises (JAC) [39–41], and ensemble modelling [42, 43].

1.1 Contributions

In this thesis, we study two novel methods to improve the robustness of ASR addressing features and models. The first method is called the temporal structure normalization (TSN) filter. When speech signals are distorted by noise, the statistics of speech features, including probability density function (p.d.f.) and power spectral density function (PSD), are all corrupted. The TSN filter is designed to reduce the corruption in the PSD functions of the features, which represent the temporal structures of the feature trajectories. The design of the TSN filter is based on a criterion to normalize the PSD functions of the features to a group of reference PSD functions. For each utterance or speech segment, a group of filters are designed, one for each feature dimension. It is expected that by restoring temporal structures of features, the systematical mismatch between clean and noisy features can be reduced. Our experimental study confirmed the effectiveness of the TSN filter on both small and large vocabulary tasks. On average, the TSN filter performs better than other state-of-the-art temporal filters. Compared to other temporal filters, a key advantage of the TSN filter is its ability to adapt to changing environmental conditions of the speech signal. When applied in cascade with other normalization methods, such as HEQ, a more complete normalization scheme can be formed which normalizes both the p.d.f. and PSD functions of the features.

Our second method is to improve ASR robustness by improving the generalization capability of acoustic model. The principle is to enable the acoustic model to tolerate
the mismatch between the clean-trained acoustic model and the noisy testing data. We study the relationship between the robustness of acoustic model and the generalization capability of acoustic model. According to statistical learning theory [44], the generalization capability of a model can be improved by increasing the margin of the model, i.e. the desired minimum distance between any training sample to the decision boundary. In our study, the soft-margin estimation (SME) method [45, 46] is applied to estimate the acoustic model. The SME method was motivated by the concept of increasing margin for better generalization capability for speech recognition problem. It is used here to study the relationship between model generalization and robustness. For comparison, another two popular discriminative training (DT) methods, including the minimum classification error (MCE) [47, 48] and maximum mutual information (MMI) [49–52] are also studied. Our experimental results show that improving a model’s generalization capability improves the performance of speech recognition for both matched and mismatched training-testing cases. However, SME improves very little for highly mismatched scenarios. This is due to the fact that the training and testing data follow different distribution functions and therefore the assumption of SME and statistical learning theory is violated. To reduce this limitation, mean and variance normalization (MVN) is applied to the features prior to the SME training. After performing MVN, the global mean and variance of the training and testing data are identical and the mismatch between their distributions becomes smaller. Experimental result show that the reducing feature mismatch by MVN enables SME to perform better than using SME alone. Our results also show that SME generally performs better than MCE and MMI in noise robust tasks.

My works on TSN filter have been published in two journal papers, including 1 in IEEE Signal Processing Letters and 1 in IEEE Transactions on Audio, Speech and Language Processing; and two conference papers, including 1 in ICASSP’07 and 1 in Inter-Speech’07. My works on applying SME to robust ASR have been accepted for publication by one journal: IEEE Transactions on Audio, Speech and Language Processing and one conference: Automatic Speech Recognition and Understanding (ASRU’09).

1.2 Thesis Outline

This thesis is organized as follows:
Chapter 1. Introduction

In Chapter 2, we provide background information about the statistical automatic speech recognition, with focus on the extraction of speech features and the estimation of the parameters of the HMM-based acoustic model.

In Chapter 3, previous techniques for noise robust speech recognition are reviewed and compared. The connections between techniques are discussed.

In Chapter 4, we propose the temporal structure normalization filter for post-processing of speech features. The performance of TSN filter is compared with other popular temporal filters such as RASTA filter [27] and ARMA filter [28].

In Chapter 5, we study the approach of improving acoustic model’s generalization capability. The approach is also combined with feature domain methods such as mean and variance normalization (MVN).

Finally, we conclude in Chapter 6, where we summarize our contributions and provide possible future research directions.
Chapter 2

Introduction of Speech Recognition

In this chapter, we will first briefly review several approaches to perform speech recognition by machine, and then describe one approach in detail that is popularly adopted in state-of-the-art speech recognition systems.

There are at least three approaches to perform automatic speech recognition in the literature as summarized in [53], namely the acoustic-phonetic approach, the statistical pattern recognition approach, and the artificial intelligence approach. In the acoustic-phonetic approach, acoustic and phonetic knowledge are used to recognize speech words from a sentence. Typically, the first step is to extract speech features from the speech signal to be recognized. Examples of speech features are formants, pitch, whether the speech is voiced or unvoiced, and whether the voice is nasal or not, etc. Based on these features and the acoustic/phonetic knowledge about the target language, a phoneme lattice is generated for the speech signal to be recognized. The phoneme lattice is a uni-directional (left to right) network containing all possible phoneme sequences for the sentence. Given the phoneme lattice, higher level information, such as lexical and syntactic information, are then applied to determine the final word sequence. In practice, there are several difficulties to implement acoustic-phonetic approach for acceptable performance [53]. One difficulty is that as the acoustic-phonetic approach is a rule-based system, it requires extensive knowledge of the acoustic properties of phonetic units. Given the large variations of speech signal (e.g. due to speaker effect, environment, accent, etc), it is difficult to scale up the complexity of the task. In addition, the choice of the speech features and the design of the classifiers are usually ad-hoc and not optimal in well-defined sense.
Due to these reasons, the acoustic-phonetic approach has not reach the same level of performance as other approaches.

In the statistical pattern recognition approach, there are usually two phases, i.e. the training phase and the testing phase. In the training phase, statistical models are usually trained from a large amount of representative data to represent the speech units (such as phonemes) in the task scope. In the testing phase, the speech signal to be recognized is matched with models of all speech units and the model that produces the best match is chosen as the recognized speech. In the past two to three decades, the pattern recognition approach has been refined steadily and becomes the mainstream technology for automatic speech recognition due to several reasons, e.g. the use of hidden Markov models (HMM) [1–4] to model the dynamics of speech signal, the efficient implementation of model training in some optimal sense (e.g. maximum likelihood [1–4]), the efficient decoding of speech using the Viterbi algorithm [54], and the large amount of speech data for model training. However, one of the major issue of statistical pattern recognition approach is the robustness of the system against signal variations, such as those due to noise corruptions. It is this thesis’s interest to investigate ways to enhance statistical speech recognition’s robustness against noise corruptions.

In the artificial intelligence (AI) approach [55], artificial neural networks (ANN) are used to perform speech recognition. ANN is a network of neurons, each of which has similar function and working mechanism as the neurons in human brain. An ANN, such as multi-layer perceptron (MLP), is very good at modeling nonlinear and complex relationship between the input and desired output of the network. However, the temporal dynamics of speech signal is usually more difficult to be modelled by ANN than by HMM. Generally speaking, the AI approach has not yet reached the same level of performance as the pattern recognition approach which is generally based on HMM. However, the concept of ANNs has been applied in many aspects of speech recognition [56]. For example, MLP is successfully applied to extract long-term speech features for speech recognition [57]. The long-term features are used in a HMM-based speech recognition system.

The pattern classification approach is the prevailing choice of the state-of-the-art ASR systems. It is also adopted in this thesis. Therefore, in the remaining sections of this chapter, more details of this approach will be given.
2.1 Overview of Statistical Speech Recognition System

Two modules of an ASR system will be reviewed in detail, including the feature extraction module and the acoustic matching module. These two modules are especially important in dealing with robustness issue of ASR.

Speech recognition is a multi-class sequential pattern recognition problem. Let \( W = \{W_1, W_2, ..., W_V\} \) denote the vocabulary of the task, i.e. the set of words to be considered during recognition, and \( O = O_1, O_2, ..., O_N \) denote the test set of utterances. The objective of speech recognition is to correctly recognize each utterance \( O_i \) into the word sequence that is spoken in the utterance. To perform the recognition, two models are usually needed, i.e. acoustic model and language model. The acoustic model allows us to evaluate how the observation \( O_i \) is related to each word \( W_j \), while the language model provides the information about the probability of all possible word sequences that could be generated from words in \( W \). The information provided by the two models will be used to generate a search space that contains all possible word sequences and their prior probabilities, and a search algorithm is used to find out the most likely word sequence based on the observation \( O_i \). After recognition, confidence measuring module may reject the recognized words if the confidence measure on the recognized word sequence is too low.

In practice, speech signals are not directly used in speech recognition. Instead, some discriminative representation of speech signals are used and such representation is called speech features. The process of extracting speech features from speech signal is called feature extraction.

In the brief introduction of a typical speech recognition system above, there are six modules involved, i.e. feature extraction, acoustic model, language model, word lexicon (vocabulary), pattern classifier (search algorithm) and confidence measuring. The relationship of these six modules are shown in Fig. 2.1. In this section, we will describe the modules of speech recognition that are related to the topic of this thesis, i.e. the robustness of speech recognition against noise distortion. Our description focuses specifically on the feature extraction process and the acoustic modelling of features as they are
directly related to the topic. We will not describe language modelling as the probabilities of word sequences are not changed by noise distortion. Similarly, we also skip the search algorithm and the confidence measuring.

2.2 Feature Extraction

The first step of speech recognition is to pre-process the speech signal and extract discriminative speech features from it. There are two main objectives in feature extraction. The first objective is to represent the speech signal in a more compact form. The second objective is to find speech features that are both good at discriminating different speech classes, such as phonemes, and insensitive to factors irrelevant to speech recognition. Examples of irrelevant factors include speaker variations, accent differences, speaking rates, background noises, and transmission/microphone distortions, etc [7, 8, 59]. In this thesis, we are interested in reducing the sensitivity of speech features to noise and channel distortions.

Currently, the most popular speech features are the Mel-frequency cepstral coefficients (MFCC) and perceptual linear predictive (PLP) features [60]. We will describe the extraction of MFCC features as an example. For more detailed discussion about feature extraction techniques, please refer to the review paper [61].

There are usually 39 dimensions in a MFCC feature vector, including 13 static features: 12 static features plus an energy feature, and the first and second derivatives of
the static features. The procedures of MFCC feature extraction is summarized as follows (see Fig.2.2):

**DC offset removal and pre-emphasis** This step removes the DC offset of the speech signal and pre-emphasize the signal spectrum by approximately 20 dB per decade to flatten the spectrum of the speech signal. The pre-emphasis filter is used to offset the negative spectral slope of voiced speech signal to improve the efficiency of the spectral analysis [61].

**Framing** The human speech signal is slowly time varying and can be treated as a stationary random process when considered under a short time frame. Therefore, the speech signal is usually separated into small duration blocks, called frames, and the spectral analysis is performed on these frames. The neighbouring blocks are overlapped by 1/2 to 2/3 length of the frame and the frame shift is the frame length minus the frame overlap. The commonly used frame length and frame shift are 20-30 ms and 10 ms respectively for speech recognition task because the positions of the articulators do not change much in the period of frame length.

**Windowing** After being partitioned into frames, each frame is multiplied by a window function prior to the spectral analysis to reduce the effect of discontinuity introduced by the framing process. Commonly used windows include Hamming and Hanning windows, both of which attenuate the values of the samples at the beginning and end of each frame. If no window is used, the case can be treated as the rectangular window. Each window has its own pros and cons. Compared to rectangular window, the Hamming and Hanning windows decrease the frequency resolution of the spectral analysis while reducing the sidelobe level of the window transfer function [62].

**Spectral estimation** The spectral coefficients of the speech frames are estimated using the fast Fourier transform (FFT) algorithm for MFCC. These coefficients are complex numbers containing both magnitude and phase information. For speech recognition tasks, the phase information is usually discarded and only the magnitude of the spectral coefficients are extracted. It is also common to use the power
Figure 2.2: The common procedures of feature extraction (MFCC).
of the spectral coefficients. Besides FFT, there is another spectral estimation technique called linear predictive coding (LPC) analysis which is used to extract the LPC cepstral coefficients. One difference between the LPC spectral analysis and FFT spectral analysis is that the LPC spectrum is a parametric estimate of the smoothed spectral envelope, while the FFT spectrum tends to provide more details of the spectrum of the speech frame.

**Mel filterbank** The spectrum of speech signal is then filtered by a group of triangle bandpass filters that simulate the characteristics of human’s ear. These windows are called the Mel windows and the filtering process is called Mel filtering. The Mel filtering is to model the human auditory system that perceives sound in a nonlinear frequency binning [63]. For example, the musical pitch described in octaves and semitones is basically proportional to the logarithm of frequency. The ears analyze the spectrum of the sound in groups according to a series of overlapped critical bands. The critical bands are distributed in a way that the frequency resolution is high in the low frequency region and low in the high frequency region.

There are several ways to distribute the critical bands and Mel frequency scale is one of them (See Fig. 2.3) [61]. The bandwidth of the window is narrow at low frequencies and gradually increases for higher frequencies. The edge of the window is arranged so that it coincides with the center of the neighbour window. To decide the location of the Mel frequency of the center of the windows, the Mel frequencies for minimum and maximum linear frequency are first calculated using

\[
    f_{\text{Mel}} = 2595 \times \log(1 + f/700) \tag{2.1}
\]

where \( f_{\text{Mel}} \) is the Mel frequency for the linear frequency \( f \). The windows are evenly distributed in Mel frequency, and the center frequencies of the windows, when converted back to linear frequency, are not linear.

**Natural logarithm** While the Mel filtering approximates the nonlinear characteristics of human auditory system in frequency, the natural logarithm deals with the loudness nonlinearity. It approximates the relationship between a human’s perception of loudness and the sound intensity [64]. Besides this, natural logarithm converts
the multiplication relationship between parameters into addition relationship [65]. For example, the convolutional distortions, such as the filtering effect of microphone and channel, and the multiplication in frequency domain, such as the amplification of soft sound, become simple additions after the logarithm. Hence they can be easily removed by subtracting the mean of the coefficients. This technique is called cepstral mean subtraction/normalization [19]. Another purpose of the natural logarithm is to reduce the dynamic range of the speech coefficients. Besides the natural logarithm, the $n^{th}$ root compression has also been used to reduce the dynamic range of features [60].

**Discrete cosine transform** The DCT is applied on the log Mel filterbank coefficients to generate the cepstral coefficients, and this process is modified Homomorphic processing [66]. Only the lower order coefficients (usually the lowest 12 or 13 coefficients) are used for speech recognition, hence a dimension reduction is achieved. Another benefit of the DCT is that the generated cepstral coefficients are less correlated than the log Mel filterbank coefficients. Therefore, it is possible to use a diagonal matrix for the covariance matrix of the Gaussian in the HMM acoustical model to significantly reduces the number of parameters in the acoustical model.

**Log energy calculation** In addition to the normal MFCC features, the energy of the speech frame is also used as a feature. The log energy, called logE, is calculated directly from the time-domain signal of a frame. Sometimes, it is replaced by c0,
the 0th component of the MFCC feature, which is the sum of the log Mel filterbank coefficients.

**Derivatives and accelerations calculation** The trend of speech signals in time, i.e. temporal information, is lost in the frame-by-frame analysis. To recover the temporal information, time derivatives (the first delta) and accelerations (second delta) are used [61]. For speaker independent speech recognition system, the derivatives and accelerations are especially important [61]. Although the location of formants in speech varies from person to person, the temporal trend of the formant are quite constant among different speakers.

There are several ways to approximate the first order derivative of the cepstral coefficients. For example, if we denote a cepstral coefficient as \( x(t) \) where \( t \) is the frame number, its derivative can be calculated as [67]

\[
\dot{x}(t) \equiv \frac{d}{dt}x(t) \approx \frac{\sum_{m=-M}^{M} mx(t + m)}{\sum_{m=-M}^{M} m^2} \tag{2.2}
\]

where \( 2M + 1 \) is the number of frames considered in the evaluation. The same formula can be applied to the first order derivative to produce the second order derivative. These derived features are simply concatenated to the original cepstral features to form the final feature vector. As we will discuss later, the delta features can be seen as band-pass filtered versions of the static features.

### 2.3 Acoustic Model Training

The acoustic model captures speech feature statistics in a parametric way. As temporal evolution of feature statistics is vital in discriminating different speech classes, acoustic models in speech recognition are usually built on hidden Markov models (HMM) [1–5]. In this section, we will describe HMM and its application in speech recognition, followed by the training of HMM parameters and a brief introduction to the decision rule used in speech recognition.
Figure 2.4: An illustration of HMM with three true states. The start and stop states are
dummy ones which do not model the signal. The different parts of the speech segment
can be modeled by different states.

2.3.1 Hidden Markov Models

The HMM is designed to capture time varying signal statistics and has been success-
fully applied to capture speech dynamics [1–5]. A general HMM consists of several
inter-connected states and the transition between states are governed by state transition
probabilities. The HMM used in speech recognition is a simplified version of the general
model: the first-order left-to-right HMM. In a first-order HMM, the occurrence of a state
is only related to its previous state. By left-to-right, it means that the state transitions
are usually constrained to be from left to right or self repetition. Such a simplified HMM
is illustrated in Fig. 2.4. In the figure, a speech segment is modeled by an HMM. The
three sections of the speech signal could be modeled by the three states of the HMM,
and the temporal correlation of speech frames is weakly modeled by the state transition
probabilities. The feature p.d.f. in each state is usually GMM.

In the acoustic model, an HMM can be built for each word to represent the unique
characteristics of the word. In large vocabulary speech recognition tasks, there are typ-
ically thousands of unique words and it is not feasible to build an HMM for each word.
In such case, phonemes can be used as the basic units of speech as there are typically
only dozens of phonemes in a language. The phonemes can be categorized into context-
-independent phonemes (monophone), such as the 40 phonemes used in English, and
context-dependent phonemes, such as triphone. An example is given here to better illustrate the phonemes [68]. The English word “composite” can be represented as a sequence of monophones: “k ah m p aa z ah t”. It can also been represented as a sequence of triphones: “k+ah k-aa+m aa-m+p m-p+aa p-aa+z aa-z+ah z-ah+t ah-t”. For each triphone, there is a center phone, a left context phone (before -) and a right context phone (after +). Triphones provide more accurate modelling of speech pronunciation as phones may be pronounced differently in different contexts. Note that “k+ah” and “ah-t” are biphones as they are at the word boundaries. If necessary, more context can be added to form even longer phonemes, such as quinphones where two context phones are used in both front and back. Generally speaking, longer contexts provide better modeling of speech. However, as the number of potential phonemes increase exponentially with the length of context, much more data is required to robustly train the longer context-dependent phoneme models. In this thesis, we will use word models for small tasks such as Aurora-2 [69] and Aurora-3 [70] and triphone models for large task such as Aurora-4 [71].

The models of larger speech units can be built from basic phoneme models. By concatenating the HMMs of phonemes, we can build an HMM for any word. Similarly, by concatenating the HMMs for words, we can obtain an HMM for any possible word sequence. The concatenation of phoneme HMMs follows the definitions in the dictionary, while the prior probability of word sequences are determined by the language model. During speech recognition, the recognizer builds a word-level network as shown in Fig. 2.5 and searches for the path that produces the highest probability on the observed feature vector sequences.

2.3.2 Model Training Methods

There are several criteria for training HMMs, including the classic maximum likelihood criterion and a group of newer criteria called discriminative trianing (DT). The major difference between DT and ML estimation is that: ML estimation only focus on correctly learning the distribution of the speech features with no direct link to the final speech recognition task, while DT criteria update model parameters to reduce the training error in direct or indirect ways.
Figure 2.5: Illustration of the building of recognition network. At the top level is a word sequence. Each word sequence is a sequence of phonemes and followed by a word end node. Each phoneme is represented by an HMM. Adapted from [67].

There are several popular DT methods, including the maximum mutual information estimation (MMI) [49–52], minimum word/phone error estimation (MWE/MPE) [51, 72], minimum classification error estimation (MCE) [47, 48], and large/soft margin estimation (LME [73–75], SME [45]). A common property of these methods is that the model is trained to reduce some cost functions directly/indirectly related to the performance measure of speech recognition task, typically the word/phone error rate. Therefore, when the testing environment is similar to that of training, these DT methods have been shown to improve the recognition accuracy significantly.

The reason to review DT methods here is that they have the potential to improve the separation of acoustic classes and hence make the acoustic model more general. A more general model may be able to perform better on mismatched training-testing scenarios. Hence, DT methods could be applied to improve the robustness of speech recognition.

In this section, we will first review the ML estimation and MAP decision rule, and then review the DT methods and compare them with the ML estimation.

### 2.3.2.1 Maximum Likelihood Training

The parameters of acoustic model (HMM) are usually estimated using the maximum likelihood (ML) criterion in speech recognition [4]. In ML, acoustic model parameters
are estimated to maximize the likelihood of the training data given their correct word sequence (i.e. correct sequence of word HMMs):

\[
\hat{\Lambda} = \arg \max_{\Lambda} \frac{1}{N} \sum_{i=1}^{N} \log P_{\Lambda}(O_i|S_i)
\]  

(2.3)

where \(\hat{\Lambda}\) is the estimate of the acoustic model parameters, \(N\) is the number of training utterances, \(O_i\) and \(S_i\) are the observed feature vectors and the correct label of the \(i^{th}\) training utterance, respectively. Note that here \(O_i\) refers to speech features in general rather than any specific features such as MFCC. Label is a sequence of words or phonemes. \(P_{\Lambda}(O_i|S_i)\) is the acoustic model likelihood.

The optimization of (2.3) does not have a closed form solution and needs to be implemented iteratively. In speech recognition, the expectation maximization (EM) algorithm is usually used [76]. The EM algorithm guarantees that the likelihood of the training data on the model of current iteration is larger than or equal to the likelihood on the model of previous iteration. For details of HMM training, please refer to [4].

During recognition, an utterance is recognized as the word sequence corresponding to the most likely model sequence with the highest \textit{a posteriori} probability:

\[
\hat{S}_j = \arg \max_{S_j \in \Psi} P_{\Lambda}(S_j|O_j) \\
= \arg \max_{S_j \in \Psi} \frac{P_{\Lambda}(O_j|S_j)P(S_j)}{P_{\Lambda}(O_j)} \\
= \arg \max_{S_j \in \Psi} P_{\Lambda}(O_j|S_j)P(S_j)
\]  

(2.4)

where \(\Psi\) is the set that contains all possible word sequences, and \(P(S_j)\) is the probability of specific sequence \(S_j\). As \(P_{\Lambda}(O_j)\) is a constant given a set of \(\Lambda\), it can be ignored in the maximization. From the above equation, the final recognition result depends on both the prior probability of word sequence \(P(S_j)\) (given by language model) and the likelihood of feature vectors given the word sequences \(S_j\) (given by acoustic model). This decision rule follows the maximum \textit{a posteriori} (MAP) criterion. As the acoustic model is just an approximation of the true feature distribution, the decision rule is called \textit{plug-in} MAP decision rule [6].

The performance of plug-in MAP decision rule in (2.4) is dependent on how well the acoustic model \(\hat{\Lambda}_{ML}\) represents the features. If the distribution learnt by ML is the
true distribution of the features, the plug-in MAP decision will be Bayesian optimal
decision that gives the lowest word error rate (WER). In practice, there are several
difficulties in obtaining the true probability distribution of features [4]. One difficulty
is that we don’t know the true parametric form of the feature distribution. In current
ASR systems, HMM is used due to its good mathematical properties and reasonable
performance. However, HMM may not be the true parametric forms of the feature
distribution. Another difficulty is the lack of sufficient data for the robust estimation of
feature distribution. To obtain a close approximation of the true feature distribution,
we need a large amount of representative training data. In practice, we usually don’t
have enough representative training data. For example, the population of human speech
includes speech of all speakers uttered in all emotional states of the speakers under all
possible environmental/channel distortions. However, the data used to train an ASR
system is often limited. Typically, only some utterances spoken by hundreds of speakers
in normal emotional state under relatively small environmental/channel distortions are
used for training. Due to these two difficulties, the distribution learnt by ML is usually
different from the true feature distribution, and hence the plug-in MAP decision rule
would be suboptimal.

To improve the recognition performance of the acoustic model with the above dis-
cussed problems, DT methods estimate the model parameters by minimizing some cost
functions that are directly or indirectly related to the word/phone error rate of the train-
ing data. In this way, the model training process is more relevant to the objective of
speech recognition. In the following sections, we will review these DT methods.

2.3.2.2 Maximum Mutual Information and Derivatives

In MMI [49–52], the parameters of acoustic model are estimated as follows:

\[
\hat{\lambda}_{\text{MMI}} = \arg \max_{\lambda} \frac{1}{N} \sum_{i=1}^{N} \log \frac{P_\lambda(O_i|S_i)P(S_i)}{\sum_{S_i} P_\lambda(O_i|S_i)P(S_i)}
= \arg \max_{\lambda} \frac{1}{N} \sum_{i=1}^{N} \log P_\lambda(S_i|O_i)
\]  

(2.5)
where $\hat{S}_i$ is theoretically all possible labels for the $i^{th}$ training utterances. Rather than maximizing the likelihood of training observations given their correct transcriptions, MMI maximizes the posterior probability of the correct label $S_i$ when the observations $O_i$ are observed. By doing this, the system not only uses the correct transcriptions as “positive” learning objective, it also uses the wrong transcriptions as “negative” learning objectives. In other words, the MMI tries to maximize the likelihoods of training data on their correct models while minimizing their likelihoods on wrong models, whereas ML only maximize the likelihoods on correct models.

In practice, not all possible transcriptions are taken into account in MMI training, as this is neither possible nor necessary. Instead, only a set of competing transcriptions with the highest likelihoods are used, and these competing transcriptions are called the N-best transcriptions. In small vocabulary tasks such as Aurora-2 [69] whose dictionary contains only 11 English digits, several competing strings are usually used. In large vocabulary tasks such as Aurora-4 [71] which have thousands of triphones, the information about competing transcriptions are contained in lattice. The size of lattice is controlled to allow proper level of details about the confusion patterns of the acoustic model regarding to the training data. In implementation, there is a problem with using multiple competing transcriptions: as the likelihoods of some competing transcriptions may be too small compared to the dominant transcription, they virtually don’t contribute to the objective function and updating of model parameters. To solve this problem, a scaling factor is introduced to MMI’s objective function as follows:

$$
\hat{\Lambda}_{\text{MMI}} = \arg \max_{\Lambda} \frac{1}{N} \sum_{i=1}^{N} \log \frac{P_A(O_i|S_i)^\kappa P(S_i)^\kappa}{\sum_{\hat{S}_i} P_A(O_i|\hat{S}_i)^\kappa P(\hat{S}_i)^\kappa}
$$

(2.6)

where $\kappa$ is the scaling factor and usually $0 < \kappa < 1$. By using a scaling factor $\kappa$ that is less than one, the difference between the most dominant transcription and the less dominant transcriptions will be smaller. Usually, the smaller the $\kappa$ is, the more competing transcriptions can be taken in to account. As we will discuss in Chapter 5, the scaling factor $\kappa$ is important for the generalizing ability of the acoustic model trained by MMI.

MWE and MPE [51, 72] are extensions of MMI. In MPE, the model parameters are estimated as:
\[
\hat{\Lambda}_{\text{MPE}} = \arg\max_{\Lambda} \frac{1}{N} \sum_{i=1}^{N} \log \frac{\sum_{\hat{S}_i} P_{\Lambda}(O_i|\hat{S}_i)P(\hat{S}_i)\text{RawPhoneAccuracy}(\hat{S}_i)}{\sum_{\hat{S}_i} P_{\Lambda}(O_i|\hat{S}_i)P(\hat{S}_i)}
\]

where \(\text{RawPhoneAccuracy}(\hat{S}_i)\) is the raw phone accuracy of the label \(\hat{S}_i\). By comparing (2.7) and (2.5), MPE can be seen as an soft-decision extension of MMI. If \(\text{RawPhoneAccuracy}(\hat{S}_i)\) is defined as follows:

\[
\text{RawPhoneAccuracy}(\hat{S}_i) = \begin{cases} 
1, & \hat{S}_i = S_i; \\
0, & \text{Otherwise}.
\end{cases}
\]

MPE degenerates to MMI. MWE is similar to MPE except that the RawPhoneAccuracy is replaced by RawWordAccuracy.

### 2.3.2.3 Minimum Classification Error

Another DT method is minimum classification error (MCE) [47, 48]. In MCE, the model parameters are estimated as:

\[
\hat{\Lambda}_{\text{MCE}} = \arg\min_{\Lambda} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1 + \exp(-\gamma d_i(O_i, \Lambda) + \theta)}
\]

where \(d_i(O_i, \Lambda)\) is a misclassification measure of the \(i^{th}\) utterance given acoustic model \(\Lambda\). \(\gamma\) and \(\theta\) are the parameters of the sigmoid function that makes the cost function continuous and differentiable. Hence, MCE directly includes the misclassification measure into the cost function. The misclassification measure suggested in [48] is defined as follows:

\[
d_i(O_i, \Lambda) = -g_i(O_i, \Lambda) + \log \left[ \frac{1}{M-1} \sum_{j,j\neq i} \exp[g_j(O_i, \Lambda)\eta] \right]^{1/\eta}
\]

where \(g_i(O_i, \Lambda)\) can be \(g_i(O_i, \Lambda) = P(O_i|S_i)\) or any other reasonable functions, \(M\) is the total number of transcriptions taken into consideration (including the correct one) and \(\eta\) is a scaling factor whose function is similar to \(\kappa\) in MMI. Note that \(g_i(O_i, \Lambda)\) is a quantity obtained from correct transcriptions, while \(g_j(O_i, \Lambda), j \neq i\) are from competing transcriptions. the misclassification measure in (2.10) is a difference between the quantity from the correct transcription and the quantities from the competing transcriptions.
When $\eta$ approaches $\infty$, the competing transcription with the highest likelihood will dominate the second term on the right hand side of (2.10).

When $\eta$ is smaller, more transcriptions will have significant contribution to (2.10). The misclassification measure is an approximation of the classification error, e.g. $d_i(O_i, \Lambda) \leq 0$ roughly implies correctly classified training utterance and vice versa.

### 2.3.2.4 Large Margin Estimation

Recently, a margin-based DT method, large margin estimation (LME) [73–75], is applied to speech recognition. The main objective of LME is to improve the generalization capability of acoustic model by improving the margin of the model.

It is well known that when a model is too complex relative to the amount of training data, the model may be overfitted to the training data and will not perform well on unseen test data. As the statistical mismatch between the training and testing data always exists, it is desirable to improve the generalization capability of the acoustic model. From statistical learning theory, the generalization risk of a model is a monotonically increasing function to its VC dimension which measures the model’s complexity. Although the VC dimension itself is usually difficult to compute, it is bounded by a decreasing function of margin [44]. Therefore, the LME reduces the generalization risk by maximizing the margin between training samples and the decision boundary.

In LME [73, 74], the acoustic model parameters are estimated as follows:

$$\hat{\Lambda}_{\text{LME}} = \arg \max_{\Lambda} \min_{O_i \in S} d(O_i, \Lambda)$$  \hspace{1cm} (2.11)

where $d(O_i, \Lambda)$ the separation measure and $S$ is called the support vector set. The separation measure is defined as the difference between the correct transcription and the closest competing transcription:

$$d(O_i, \Lambda) = P_\Lambda(O_i|S_i)P(S_i) - \max_{\hat{S}_i \neq S_i} P_\Lambda(O_i|\hat{S}_i)P(\hat{S}_i)$$  \hspace{1cm} (2.12)

Note that the language model probability is included in the separation measure of LME. The support vector set is defined as follows:

$$S = \{ O_i | 0 \leq d(O_i, \Lambda) \leq \rho \}$$  \hspace{1cm} (2.13)
Note that $\mathcal{S}$ only includes those correctly classified training utterances. In a later extension of LME, called soft-LME [75], the wrongly classified training utterances are also included for the estimation of the acoustic model parameters.

### 2.3.2.5 Soft-Margin Estimation

Another margin-based model estimation method is the soft-margin estimation (SME), which also aims to improve the model’s generalization capability by increasing the margin.

In SME, the expected risk of the acoustic model is approximated, and the model parameters are estimated to minimize the expected risk as follows:

$$
\hat{\Lambda}_{\text{SME}} = \arg\min_{\Lambda} \left( \frac{\lambda}{\rho} + \frac{1}{N} \sum_{i=1}^{N} (\rho - d(O_i, \Lambda) I(O_i \in U)) \right)
$$

(2.14)

where $\rho$ is the desired margin, i.e. the desired minimum distance between any training sample and the decision boundary, $\lambda$ is a tradeoff variable, $d(O_i, \Lambda)$ is the separation measure of $O_i$ given model $\Lambda$, $I(O_i \in U)$ is an utterance selection function and $U$ is the set containing all the utterances in which $d(O_i, \Lambda)$ is less than the margin. The first item of the objective function, $\frac{\lambda}{\rho}$, approximates generalization risk, i.e. the risk of testing the acoustic model on unseen data. It is inversely proportional to the margin value. To make the model more general to tolerate training-testing mismatch, it is desirable to have a larger margin value. The second item of the objective function represents the empirical risk, i.e. the cost due to training data. The separation measure usually represents how well the correct model of $O_i$ is separated from the competing models. If the separation is not large enough, i.e. the separation measure is less than the margin, a loss is generated that equals to $\rho - d(O_i, \Lambda)$. From this definition, it is obvious that the empirical risk is proportional to the value of margin. The parameters of the acoustic model are determined by balancing these two conflicting requirements and the variable $\lambda$ is used to control the relative weight of them.

One common separation measure used in SME is the average log likelihood ratio (LLR) of frames with confusion [45]:

$$
d(O_i, \Lambda) = \frac{1}{n_i} \sum_{j \in F_i} \log \frac{P_{\Lambda}(O_{ij} | S_i)}{P_{\Lambda}(O_{ij} | \tilde{S}_i)}
$$

(2.15)
where $O_{ij}$ is the $j^{th}$ feature vector of the $i^{th}$ utterance; $S_i$ and $\hat{S}_i$ represent the correct and the closest competing alignments of $O_i$, respectively; $F_i$ is the set of frames in $O_i$ with confusion, i.e. whose state identities in the correct and competing alignments are different; and $n_i$ is the number of frames in $F_i$. The utterance selection function is defined as

$$I(O_i \in U) = \begin{cases} 
1, & \rho > d(O_i, \Lambda); \\
0, & \text{Otherwise.}
\end{cases} \quad (2.16)$$

In practice, it can be approximated by a sigmoid function to make the objective function smooth and differentiable. For example:

$$I(O_i \in U) = \frac{1}{1 + \exp(-\gamma(\rho - d_i(O_i, \Lambda)))} \quad (2.17)$$

### 2.4 Summary

In this chapter, we have reviewed the fundamental aspects of speech recognition that are related to the robustness issue. The modules of ASR discussed in details include the feature extraction and acoustic modelling. In next chapter, we will review the existing techniques that operate in either the feature domain or acoustic model domain to improve ASR robustness.
Chapter 3

Review of Robust Speech Recognition Techniques

In this chapter, a review of robust speech recognition techniques is provided. This chapter will cover most important techniques that have been proposed in the past two decades. An overview of the robustness techniques will be given first and these techniques will be then grouped into two categories, i.e. the feature domain techniques and model domain techniques. Then the techniques in each categories and sub-categories will be reviewed individually.

As the review presented in this chapter is quite extensive and a lot of techniques will be covered, some of these techniques may not be closely related to the contributions of this thesis. To make it easier to appreciate the relationship between our contributions to the existing techniques, it is necessary here to point out these techniques that are closely related to our study. The first contribution of this thesis is the temporal structure normalization filter (Chapter 4). The related techniques include temporal filters (Section 3.3.4) and feature normalization techniques (Section 3.3.3). The second contribution of this thesis is the use of margin-based model training technique (Chapter 5). The related literature is the DT methods reviewed in Section 2.3.2.

3.1 Overview of Noise Robust Speech Recognition

3.1.1 Mismatches and Robust Approaches

When speech signals are corrupted by noise, the statistics of speech features extracted from the speech signals are also distorted in various domains, e.g. the probability dis-
Figure 3.1: Distortions in the feature and model spaces.

tribution of features and the PSD functions of feature trajectories. In general, the noise effect in speech recognition can be represented by Fig. 3.1. In the figure, $X$ and $Y$ represent clean and noisy features respectively; $M_x$ and $M_y$ represent the acoustic model trained from $X$ and $Y$ respectively. If noisy feature $Y$ is tested on clean-trained model $M_x$, there will be mismatch between feature and model and the recognition performance will be degraded.

There are generally three ways to reduce the mismatch between feature and model. The first way is to use noisy data to train the acoustic model, and this is usually called multi-condition/multi-style training [77]. This method is very effective, e.g. the average word recognition accuracy on noisy test data can be improved from about 60% to about 87% on connected digital string Aurora-2 task [69] by adding noisy speech data in the training data set. However, there are three major problems with multi-condition training. One is that when the signal condition of the test data is different from that of the training data, such as different corrupting noise types and signal-to-noise ratios (SNR), there is still mismatch between the training and testing data. The second problem is that it is usually expensive to obtain enough representative noisy data that can cover a wide range of noise types and SNR levels. In addition, if the training data include speech data with very diverse characteristics, the model obtained from the training data will not be “sharp” enough to discriminate various sound classes. This is demonstrated by the degraded recognition accuracy on clean test data when multi-condition/multi-style training is used.
Figure 3.2: Three approaches to reduce the effects of noise distortions.

The second way to reduce feature-model mismatch is to compensate the features such that features from all environment conditions become consistent with each other and are still discriminative. These methods are collectively called feature domain methods in this thesis, and are illustrated in Fig. 3.2(a)(b). In Fig. 3.2(a), clean features are estimated from observed noisy features. Methods belonging to this approach include speech enhancement methods [10–14], feature compensation methods [15–18] and temporal filters [27–34], etc. In Fig. 3.2(b), instead of estimating clean features, feature normalization methods [19–26] normalize the statistics of both clean and noisy features to a group of reference statistics to reduce systematic distortion of feature statistics. For example, the mean, variance and histogram of features can be normalized. After normalization, the mismatch between the normalized clean features $X^*$ and normalized noisy features $Y^*$ are expected to be smaller than that between the un-normalized noisy and clean features.

The feature domain methods can also be grouped according to the domain they operate in, e.g., speech enhancement methods are applied in time or frequency domain, feature compensation and normalization methods usually operates in log filterbank or cepstral domain, and temporal filters are used to reduce mismatch in the modulation spectrum domain.

The third way to reduce feature-model mismatch is to adapt the acoustic model to make it better represent the noisy test data [35–43]. This concept is illustrated in Fig. 3.2(c), where the noisy model $M_y$ is estimated based on the observed noisy feature $Y$ and clean-trained acoustic model $M_x$. By adapting the model parameters to represent
noisy data, the decision boundary of the model will be more suitable to classify noisy data and performance can be improved. Based on whether environment model described in section 3.2 is used, model adaptation methods can be classified as either data-driven adaptation [36, 37, 43, 78–80] or environment-model-based adaptation [35, 39].

Although it is quite straightforward to reduce feature-model mismatch to improve robustness of speech recognition, it is not the only way. The reason that feature-model mismatch causes performance degradation is that the noisy features’ log likelihood score deviates from that of clean features’. By reducing feature-model mismatch, the deviation could be reduced and performance could be improved. However, another way to alleviate the noise problem is to make the acoustic model more tolerable to such deviations, i.e. make the model more general. For example, discriminative training (DT) methods have the potential power to increase the generalization capability of the acoustic model [46]. Therefore, we have also reviewed DT methods in the previous chapter.

In the following sections, we will review three groups of techniques, i.e. feature domain methods, model adaptation methods. The review presented here is not a complete study of all noise robust speech recognition techniques proposed in the past two decades. The purpose of the review is to provide a brief summary of existing techniques and their relationships and to clarify the position and contribution of this thesis in the noise robust speech recognition field.

3.1.2 About Notations Used in This Thesis

In this thesis, we will cover a wide range of topics in ASR, e.g. the robustness issues in ASR and the discriminative training of models. Commonly used notations may be
quite different in different fields. For example, $x$ and $y$ are usually used to represent clean and corrupted speech feature vectors, respectively, in robustness papers, while $O$ is usually used to represent observed feature vectors in model training literature. We will try our best to keep the notations in this thesis as consistent as possible. If there is any inconsistency, the meaning of each symbol should be clear from the context.

As many methods process speech signal in a frame-by-frame manner, the frame index $t$ is usually ignored for simplicity unless it is explicitly specified for accurate description of methods.

### 3.2 Noise Effect in Filterbank and Cepstral Domains

Before we discuss robustness techniques, let’s first review our knowledge about the effect of noise on speech recognition. Specifically, we will describe the noise effect in the filterbank and cepstral domains based on a popular environment model that characterizes the physical relationship between the speech and noise. As we will show in the following sections, the noise robust techniques that use an environment model are called model-based techniques while others without using an environment model are called data-driven techniques. In this section, we will use a commonly used environment model in the noise robust speech recognition field [38, 81].

There are two common types of noise that may affect the speech signal, they are additive noises and convolutional noises. Example of additive noises are background noise, traffic noise, etc., and convolutional noise may be transmission channel distortions, microphone filtering, room reverberation, etc. The environment model we are going to discuss is shown in Fig. 3.4, where the clean speech signal is distorted by the channel first, then corrupted by the additive noise further [38, 81].

We now derive the mathematical representation of the environment model in the time and frequency domains. Let $x(n)$, $n(n)$ and $y(n)$ represent the digital clean speech, additive noise and degraded speech in time domain respectively where $n$ is the time sample index, and let $h(n)$ represent the channel impulse response. The environment model in time domain is:

$$y(n) = x(n) * h(n) + n(n)$$

(3.1)
where $*$ represents convolution. Note that we don’t consider the framing and windowing in the feature extraction process for simplicity. We can consider the signal in (3.1) as a frame. Applying the discrete Fourier transform, the model in frequency domain for a single frame becomes

$$Y(k) = X(k)H(k) + N(k)$$  \hspace{1cm} (3.2)

where $k = 1, ..., K$ is Fourier coefficient index and $K$ is the number of Fourier coefficients, $Y(k)$, $X(k)$, $N(k)$ and $H(k)$ are the Fourier transform coefficients of $y(n)$, $x(n)$, $n(n)$ and $h(n)$ in current frame respectively. The power spectral density of noisy speech signal is found using:

$$|Y(k)|^2 = Y(k)Y^*(k)$$

$$= (X(k)H(k) + N(k))(X^*(k)H^*(k) + N^*(k))$$

$$= X(k)X^*(k)H(k)H^*(k) + X(k)H(k)N^*(k) + X(k)^*H^*(k)N(k) + N(k)N^*(k)$$

$$= |X(k)|^2|H(k)|^2 + |N(k)|^2 + X(k)H(k)N^*(k) + (X(k)^*H(k)N^*(k))^*$$  \hspace{1cm} (3.3)

By using the property that the product of two complex number equals to the product of their magnitudes and the cosine of the angle between them, i.e., $ab = |a||b|\cos(\alpha)$, the equation (3.3) can be rewritten as

$$|Y(k)|^2 = |X(k)|^2|H(k)|^2 + |N(k)|^2 + 2|X(k)||H(k)||N(k)|\cos\theta_k$$  \hspace{1cm} (3.4)

$$\cos\theta_k = \frac{X(k)H(k)N^*(k)}{|X(k)||H(k)||N(k)|} = \frac{X^*(k)H^*(k)N(k)}{|X(k)||H(k)||N(k)|}$$  \hspace{1cm} (3.5)

where $\theta_k$ denotes the random angle between the two complex variables $N^*(k)$ and $X(k)H(k)$, and contains the phase information of the Fourier transform coefficients. Depending on whether the phase term is included, there are two versions of environment model. We will introduce them separately in the following subsections.
3.2.1 The phase-insensitive model

If the phase term is ignored in equation (3.3), the power spectral density of \( Y(k) \) is simplified to

\[
|Y(k)|^2 \approx |X(k)|^2 |H(k)|^2 + |N(k)|^2
\]  

(3.6)

Note that this simplification is justified by the fact that \( E[X(k)H(k)N^*(k)] = 0 \), because the noise and speech are assumed to be independent and that the noise has zero mean. This simplification causes the loss of phase information as the equation (3.6) only holds true in an expectation sense. After obtaining the power spectrum, the \( L \) Mel filterbanks coefficients are computed (this part of derivation follows that in [17]). Let \( w^l_k, k = 1, \ldots, K/2 \) denote the Mel window weights for the \( l^{th} \) filterbank, where \( \sum_{k=1}^{K/2} w^l_k = 1 \). During the evaluation, half of the linear power spectrum are discarded due to its symmetrical structure. The calculation of the filterbank coefficients is as follows

\[
|\tilde{Y}^l|^2 = \sum_{k=1}^{K/2} w^l_k |Y(k)|^2, \quad l = 1, \ldots, L
\]

(3.7)

\[
= \sum_{k=1}^{K/2} w^l_k (|X(k)|^2 |H(k)|^2 + |N(k)|^2)
\]

\[
= |\tilde{X}^l|^2 |\tilde{H}^l|^2 + |\tilde{N}^l|^2
\]

where

\[
|\tilde{X}^l|^2 = \sum_{k=1}^{K/2} w^l_k |X(k)|^2
\]  

(3.8)

\[
|\tilde{N}^l|^2 = \sum_{k=1}^{K/2} w^l_k |N(k)|^2
\]  

(3.9)

\[
|\tilde{H}^l|^2 = \frac{\sum_{k=1}^{K/2} w^l_k (|X(k)|^2 |H(k)|^2)}{|\tilde{X}^l|^2}
\]  

(3.10)

Note that it is also possible to calculate the filterbank coefficients from magnitude spectrum rather than power spectrum. To examine the effect on the cepstral domain, the natural logarithm and DCT are applied on the filterbank coefficients:

\[
\text{DCT}(\ln |\tilde{Y}^l|^2) = \text{DCT}(\ln(|\tilde{X}^l|^2 |\tilde{H}^l|^2 + |\tilde{N}^l|^2)), \quad l = 1, \ldots, L
\]  

(3.11)
where DCT represents the discrete cosine transform. To represent equation (3.11) in a simpler manner, the following definitions are introduced

\[ x = \text{DCT}(|\tilde{X}_l|^2) \]  
\[ n = \text{DCT}(|\tilde{N}_l|^2) \]  
\[ y = \text{DCT}(|\tilde{Y}_l|^2) \]  
\[ h = \text{DCT}(|\tilde{H}_l|^2) \]  

Equation (3.11) can be rewritten in a simpler manner after the following algebra transformation

\[
y = \text{DCT} \left\{ \ln \left[ |\tilde{X}_l|^2 |\tilde{H}_l|^2 \left( 1 + \frac{|\tilde{N}_l|^2}{|\tilde{X}_l|^2 |\tilde{H}_l|^2} \right) \right] \right\} \\
= \text{DCT} \{|\tilde{X}_l|^2\} + \text{DCT} \{|\tilde{N}_l|^2\} + \text{DCT} \left\{ \ln \left( 1 + \frac{|\tilde{N}_l|^2}{|\tilde{X}_l|^2 |\tilde{H}_l|^2} \right) \right\} \\
= x + h + \text{DCT} \{\ln(1 + \exp(\text{IDCT}[n - h - x]))\} \]  

From (3.16), the distortion in the noisy feature \( y \) include \( h \) and a complex nonlinear term that are jointly determined by the clean speech, the additive noise and the channel distortion.

### 3.2.2 The phase-sensitive model

If the phase term is included in equation (3.3), there will be one phase-related term in the Mel filterbank representation of noisy speech signal (this part of derivation follows that in [17]). The equation (3.11) is expanded as

\[
\ln |\tilde{Y}_l|^2 = \ln(|\tilde{X}_l|^2 |\tilde{H}_l|^2 + |\tilde{N}_l|^2 + 2\alpha_l |\tilde{X}_l| \tilde{H}_l |\tilde{N}_l|), \quad l = 1, ..., L
\]

where the phase term \( \alpha_l \) is defined as

\[
\alpha_l \equiv \frac{\sum_{k=1}^{K/2} w_k^2 |X(k)||H(k)||N(k)| \cos \theta_k}{|\tilde{X}_l||\tilde{H}_l||\tilde{N}_l|}
\]

After applying the natural logarithm and DCT, and following the same definition as in equation (3.12-3.15), the model in cepstral domain is represented as:

\[
y = x + h + \log[1 + \exp(n - x - h) + 2\alpha \exp(\frac{n - x - h}{2})] \]
\[
\alpha = \frac{\exp(y - x - h) - \exp(n - x - h) - 1}{2 \exp(\frac{n - x - h}{2})}
\]

47
where \( \alpha = [\alpha^1, \alpha^2, ..., \alpha^t]^T \) is a vector and \( \bullet \) denotes the element-wise multiplication. This model is adopted in [17].

### 3.2.3 Summary

From the environment model described in this section, the noise effect in log filterbank and cepstral domains are highly nonlinear. In [38], the distribution of noise-corrupted log power spectra coefficients are studied using Monte Carlo methods [82]. It is found if speech is modeled as a Gaussian distribution, that the mean and variance of the Gaussians will be changed by noise. With mismatched mean and variance, the decision boundary of the acoustic model trained from clean features will not classify the distorted noisy features well. In the next two sections, we will review the feature compensation methods that compensate the noisy features, and the model adaptation methods that adapt the model parameters and the decision boundary.

### 3.3 Feature Domain Techniques

Feature domain techniques aim to make speech features more consistent under different kinds of noise distortions and signal to noise ratios (SNR). The ultimate objective of feature domain techniques is to make the clean and noisy features identical to each other while preserving the discriminative power of the features. In this section, we will review four groups of feature domain techniques, including speech enhancement, feature compensation, feature normalization, and temporal filters.

#### 3.3.1 Speech Enhancement

Speech enhancement techniques were originally designed for enhancing noise-corrupted speech signals for human listening. The objective of human listening is quite different from that of automatic speech recognition. For human listening, the objective is to improve the quality and intelligibility of speech signals, while for ASR the objective is to reduce the difference between the clean and noisy speech features. Despite this difference, it is reasonable to assume that the good quality and intelligibility of speech signal usually leads to small mismatch between clean and noisy speech features. Therefore, many speech
enhancement techniques have been modified and applied in the feature extraction process of speech recognition systems with some success. Popular speech enhancement techniques include spectral subtraction \[10, 83\], the minimum mean square error (MMSE) short-time spectral amplitude (STSA) estimator \[13, 14, 84\], and the signal subspace-based techniques \[85, 86\].

### 3.3.1.1 Spectral Subtraction

Spectral subtraction is a simple yet effective way to reduce additive noise’s effects in speech signal. Spectral subtraction estimates the clean speech spectrum by subtracting the estimated additive noise spectrum from the noisy speech spectrum \[10, 11\]:

\[
|X(k)|^2 = |Y(k)|^2 - |N(k)|^2
\]  

(3.21)

where \(|X(k)|^2\) is the estimated clean speech spectrum, \(|Y(k)|^2\) is the observed noisy speech spectrum, \(|N(k)|^2\) is the estimated noise spectrum, and \(k\) is the frequency bin index. Spectral subtraction is motivated by the fact that the noise corruption is additive in the power spectrum domain in expected sense if the noise has zero mean in time domain and is assumed to be independent from the speech, i.e.

\[
E[|Y(k)|^2] = E[|X(k) + N(k)|^2] = E[|X(k)|^2] + E[|N(k)|^2]
\]  

(3.22)

where \(E[\cdot]\) denotes expected value.

The performance of spectral subtraction is directly affected by the accuracy of noise estimation, which is a very difficult task by itself. In some cases, the noise estimate \(|N(k)|^2\) is even larger than the noisy speech \(|Y(k)|^2\), hence the estimated clean spectrum will be negative. When this happens, the value of clean spectrum is usually set to zero. This simple solution results in nonnatural spectral vectors and causes “musical noise” phenomenon, which is annoying for human listening and degrades speech recognition performance. There are several kinds of spectral subtraction, and they mainly differ in the way to handle the “musical noise”. For example, in \[87\], over subtraction and spectral floor are used to provide a tradeoff between the “musical noise” and residual noise level. In \[88\], it was proposed to apply the masking properties of ear to determine the amount of subtraction. The basic idea is that for those noise components that can not be heard
by a human ear, it is not subtracted, so there is a smaller amount of subtraction and therefore smaller degree of distortion.

Despite its application in enhancing speech for hearing, spectral subtraction has also been used to preprocess noisy speech in the feature extraction process of speech recognition systems [8]. In [12], a nonlinear spectral subtraction was proposed for robust speech recognition in car environment and shown to outperform standard spectral subtraction. The limitation of spectral subtraction is that it aims to reduce noise distortion in the signal domain and has no direct relationship with the final speech recognition task, i.e. to achieve high recognition accuracy. In addition, spectral subtraction’s performance depends heavily on the accuracy of noise estimation which is difficult especially when the noise is non-stationary and has similar characteristics as speech signal, such as babble noise.

3.3.1.2 MMSE Spectral Magnitude Estimator

Besides spectral subtraction, another more advanced speech enhancement technique has been proposed, i.e. the optimal estimator of speech’s short-time spectral amplitude (STSA) in the minimum mean square error (MMSE) sense [13, 14]. In the MMSE STSA estimator, the phase and amplitude of spectral components of clean speech signal and noise are assumed to be independent Gaussian variables. With this assumption, the distribution of the spectral component of noisy speech signal follows the Rayleigh’s distribution. The MMSE estimate of the clean spectral amplitude is then derived based on these models, and the resulting solution of the estimator is a function of the a priori SNR and the a posteriori SNR of the speech signal. The a priori SNR is the expected SNR before the current frame is observed. It is critical for the performance of MMSE STSA estimator and can be estimated using either ML estimation or a “decision-directed” method. The a posteriori SNR is the instantaneous SNR of the current frame.

Several methods have been proposed to improve the accuracy of the estimate of the a priori SNR. Some researchers focused on improving the average weighting parameter \( \alpha \) of the decision directed method, which controls the speed of adaptation. Soon and Koh [89] proposed to estimate \( \alpha \) from the changing speed of frame energy. This idea is further extended in [90] by using a frequency-dependent MMSE estimator of \( \alpha \). Besides
the estimation of $\alpha$, to incorporate more information, Israel [91] proposed a noncausal a priori SNR estimator that employs both past and future frames for better estimation. Another approach by Hu and Loizou [92] reduces the variance of the a priori SNR estimate indirectly by reducing the variance of noise estimate.

Another important characteristics of the MMSE STSA estimator is that it incorporates a signal absence probability (SAP), which was first introduced by McAulay and Malpass [83]. With SAP, the estimate of a clean speech becomes:

$$\hat{X}(k) = P(\text{speech present}|Y(k))\hat{X}_{MMSE}(k)$$  \hspace{1cm} (3.23)

where $P(\text{speech present}|Y(k))$ is the posterior probability of speech presence at the $k^{th}$ frequency bin when noisy spectral coefficient $Y(k)$ is observed, and $\hat{X}_{MMSE}(k)$ is the MMSE estimation of clean spectral coefficients.

Later, the MMSE estimator of STSA is extended to log spectral domain [14] to simulate the nonlinear compression of the human auditory system. It was reported that the log MMSE STSA estimator yielded better performance than the original estimator in [13].

The MMSE STSA have been successfully applied to noisy speech recognition tasks, e.g. in [93, 94]. Although it is more advanced than spectral subtraction, it also suffers the limitation of spectral subtraction, i.e. lack of direction relationship to the final speech recognition task and dependency on noise estimation performance.

### 3.3.1.3 Subspace-based Techniques

Another popular speech enhancement technique is the signal subspace method, which is motivated by the fact that noisy speech signal can usually be decomposed into two subspaces: the signal plus noise subspace and the noise only subspace. During the enhancement process, the noise only subspace can be removed completely and the clean speech signal can be estimated from the signal plus noise subspace.

There are two methods to decompose the noisy signal into the two subspaces, namely the singular value decomposition (SVD) method and the Karhunen-Loève transform (KLT). In the SVD-based method proposed by Dendrinos et al [85], the clean signal is reconstructed from the singular vectors corresponding to the largest singular values.
It is believed that the singular vectors corresponding to the largest singular values contain speech information, while the singular vectors corresponding to the smallest singular values contain noise information. This approach provides large SNR gains for speech corrupted by white noise. In the Quotient SVD-based approach proposed by Jensen et al [95], the previous approach is extended to suppress colored noise. However, QSVD was found to be computationally expensive and provided no method for shaping or controlling the residual noise.

Many approaches also use KLT to decompose noisy signal. In Ephraim and Van Trees’s method [96], the estimator minimizes speech distortion subject to a given residual noise level constraint. In this way, a mechanism is provided to adjust the tradeoff between the signal distortion and the residual noise level. Huang and Zhao [97] extended the method of Ephraim and Van Trees by proposing an energy-constrained signal subspace method (ECSS). The idea was to match the short-time energy of enhanced speech signal to the unbiased estimated of the clean speech. They declared that this method recovered the low-energy segments in continuous speech effectively. Rezayee and Gazor [98] proposed an algorithm to reduce colored noise by diagonalizing the noise correlation matrix using the estimated eigenvalues of the clean speech and nulling any off-diagonal elements. Mittal and Phamdo [86] extended Ephraim and Van Trees’s method to colored noise by providing proper noise shaping for colored noise without pre-whitening.

One important assumption of signal subspace approach is that the largest singular values or eigenvalues are from speech and the smallest values are from noise. However, in very noisy cases such as SNR=-5dB, the noise power may be higher than the signal power and this assumption may not hold any more.

Several subspace-based method have been evaluated on noisy speech recognition task in [99]. Significant improvement of performance was reported on speech corrupted by white and colored noises.

### 3.3.2 Feature Compensation

While speech enhancement techniques try to recover the time domain speech signal for human hearing, feature compensation methods aim at recovering clean speech coefficients from noisy speech coefficients during the feature extraction process of speech recognition
without generating corrected speech signal in time domain. There are another two major
differences between these two groups of methods. One difference is that speech enhance-
ment techniques usually operate in time domain, frequency domain or log frequency
domain, while feature compensation methods usually work in the log filterbank domain
or cepstral domain. Another difference is that feature compensation methods are solely
designed for noisy speech recognition tasks, while speech enhancement methods are origi-
naUy proposed to improve speech signal for human listening.

Feature compensation methods can be classified into two groups based on whether
they use the environment model described in section 3.2: i.e. model-based approach and
data-driven approach. We will review these two groups of feature compensation methods
separately in this section.

3.3.2.1 Model-based Feature Compensation

An early model-based feature compensation technique is called the code-dependent cepstral
normalization (CDCN) [81]. In CDCN, the phase-insensitive environment model of
(3.16) is adopted. The clean cepstral vectors \( \mathbf{x} \) are modeled by a Gaussian mixture model
(GMM). The CDCN estimates the clean cepstral vector from the noisy observations in
the MMSE sense, and the closed-form solution of the MMSE estimator of \( \mathbf{x} \) is:

\[
\hat{\mathbf{x}} = \mathbf{y} - \hat{\mathbf{h}} - \sum_{i=1}^{M} p(i|\mathbf{y})\hat{\mathbf{r}}(i)
\]  

(3.24)

where \( \hat{\mathbf{x}} \) and \( \hat{\mathbf{h}} \) are the estimate of the clean cepstral vector and channel distortion,
respectively (see definition in (3.12-3.15)), \( \mathbf{y} \) is the noisy cepstral vector, \( p(i|\mathbf{y}) \) is the
posterior probability of the \( i^{th} \) mixture in the GMM after \( \mathbf{y} \) is observed, \( M \) is the num-
ber of mixtures, and \( \hat{\mathbf{r}}(i) \) is the codeword-dependent correction vectors that need to be
estimated. Note that the GMM of the clean cepstral vector \( \mathbf{x} \) are first adapted to noisy
GMM using estimated noise and channel distortions and environment model, then \( p(i|\mathbf{y}) \)
for all mixtures are calculated. By comparing the environment model of (3.16) and the
MMSE estimator, we note that the corruption term \( \text{DCT}\{\ln(1 + \exp(\text{IDCT}[\mathbf{n} - \mathbf{h} - \mathbf{x}]))\} \)
in the environment model is represented by a weighted sum of the codeword dependent
correction vectors.
The noise and channel distortions are estimated using the ML criterion as there is usually no prior information about them available. The distortions are assumed to be constant during the analysis duration, e.g. an utterance. By assuming speech frames to be independent from each other, the log likelihood of the training data is:

\[
\log p(Y|\mathbf{n}, \mathbf{h}) = \sum_{t=1}^{T} \log p(y_t|\mathbf{n}, \mathbf{h})
\]

(3.25)

where \( Y = y_1, ..., y_T \) is the feature vector sequence of an utterance, \( \mathbf{n} \) is the noise distortion and \( T \) is the number of frames in the utterance to be processed. In [81], the distribution \( p(y|\mathbf{n}, \mathbf{h}) \) is obtained by using the environment model and the distribution of \( \mathbf{x} \) with some assumptions. The optimization is implemented using the expectation maximization (EM) algorithm [76].

Another model-based feature compensation method is proposed by Deng et. al. in [17, 18, 100]. A major difference between Deng’s estimator and CDCN is that the phase sensitive environment model of equation (3.19) is used in Deng’s estimator for more accurate modelling of speech-noise relationship. Besides, Deng’s estimator operates in the log Mel filterbank domain, while CDCN is a cepstral coefficients estimator. Similar to CDCN, there are two major part in Deng’s estimator, i.e. the MMSE estimation of the clean speech feature vector based on the adopted environment model and the prior probability distribution of clean speech, and the estimation of noise distortion.

The GMM is also used for clean feature vector modelling in Deng’s estimator. The phase factor \( \alpha \) in (3.19) is also modeled as zero-mean Gaussian distributed. With these prior distribution of clean speech, phase, and the phase-sensitive environment model, the MMSE estimator of clean feature vector can be obtained. However, the estimator is too complex and needs to be simplified by using the second-order Taylor series expansion. Channel distortion is ignored and a sequential noise estimation [101] is used to track additive noise. In addition, the assumption of stationary noise in CDCN is removed in Deng’s estimator.

The recognition performance of the phase-sensitive MMSE estimator [17] is found to be better than that of the phase-insensitive MMSE estimator in [102] with 54% error rate reduction. This shows that the incorporation of the phase information benefits the
feature compensation process by including relevant information. If the phase factor is set to zero, the phase-sensitive MMSE estimator degenerates to spectral subtraction.

Later, the phase-sensitive MMSE estimator is expanded to include the first order derivatives of the speech features in the log Mel filterbank domain [18], due to the assumption that the strong dynamic property of speech features are important for the enhancement of the features. The static and dynamic features are assumed to follow a GMM distribution and be independent from each other. Then the noisy speech feature distribution function is derived and the clean speech features are estimated using the MMSE criterion. The recognition accuracy on Aurora-2 shows that the incorporation of dynamic features leads to better performance. Furthermore, the enhanced spectrogram from system with the use of dynamic features is smoother than that from system without using dynamic features. The trend information in time introduced by dynamic features is orthogonal to the information of static features and therefore provides better enhancement.

The work of Deng et. al. was further expanded by incorporating a feature compensation uncertainty [100] in the decoding process. The feature compensation uncertainty accounts for the deviation of the enhancement feature from the clean feature, i.e. the variance of the feature estimator. To better decode the noisy speech, this uncertainty should be taken into account in the decoding process. One way to do this is to integrate the acoustic score over this uncertainty space, i.e. over all possible clean feature values. One issue for incorporation of the uncertainty is how to efficiently calculate the integration. The integration is effectively the same as adding the variance of the feature estimator (the uncertainty) to the Gaussian’s of the HMM states if the feature estimation error is assumed to be zero-mean Gaussian distribution [100]. Another issue is how to effectively estimate the feature estimator’s variance. In [100], analytical solutions are derived by making use of the phase-sensitive environment model.

3.3.2.2 Data-driven Feature Compensation

The simplest data-driven feature compensation is the cepstral mean normalization (CMN) [19]. In CMN, the features are compensated simply by:

\[ \hat{x}_t = y_t - r \]  

(3.26)
where $\hat{x}_t$ and $y_t$ are the estimated clean feature vector and noisy feature vector for the $t^{th}$ frame, respectively, and $r$ is the correction term that is the mean of the features, usually obtained by averaging the feature vectors over an utterance. The mean of features is in fact the optimal estimate of the correction term $r$ in the MMSE sense if only a single correction vector is allowed [103].

The operation of CMN to compensate all feature vectors by a single fixed correction vector is too limiting. The use of a single vector $r$ can only compensate for convolutional noise in the feature domain. In [38, 78], a method called multivariate Gaussian-based cepstral normalization (RATZ) is proposed to use multiple correction vectors. In RATZ, the clean feature space is modeled by a GMM. The distribution of the noisy speech is also assumed to be GMM. It is observed that in the log Mel filterbank and cepstral domain, the effect of noise on the distribution of speech signal is that the mean is shifted and the variance is either decreased or increased depending on the SNR. Therefore, the noisy GMM can be approximated by adding a correcting term to the mean and variances of the clean GMM. Let the distribution of the cepstral vectors of the clean and noisy speech be GMM with the same number of mixtures

$$p(x) = \sum_{i=1}^{M} P(i) \mathcal{N}_x(\mu_x^i, \Sigma_x^i)$$

$$p(y) = \sum_{i=1}^{M} P(i) \mathcal{N}_y(\mu_y^i, \Sigma_y^i)$$

(3.27)  

(3.28)

The noisy distribution function can be approximated by adding correction terms to the clean mean and covariance parameters

$$\mu_y^i = \mu_x^i + r^i, i = 1, ..., M$$

$$\Sigma_y^i = \Sigma_x^i + R^i, i = 1, ..., M$$

(3.29)  

(3.30)

The correction terms are estimated based on the maximization of the likelihood for the noisy observation. As there is no closed-form solution for the correction terms, the EM algorithm is applied again. After the $r^i$ and $R^i$ are obtained, the RATZ estimates the clean cepstral vector using the MMSE criterion as follows:

$$\hat{x} = y - \sum_{i=1}^{M} p(i|y, \mu_y^i, \Sigma_y^i) r^i$$

(3.31)
which is a weighted sum of mean correction vectors $\mathbf{r}^i$. As we will discuss later, the correction vectors $\mathbf{r}^i$ and $\mathbf{R}^i$ can also be used to adjust the parameters of acoustic models for better match between the model and noisy data. Note that the estimated noisy mean and variance are used to evaluate the posterior of mixtures.

In the RATZ method, the selection of correct Gaussian mixture using $p(i|y, \mu_y^i, \Sigma_y^i)$ may not be accurate as the distribution of $y$ is estimated from limited testing data. The problem can be reduced by another data-driven feature compensation method, the stereo-based piecewise linear compensation for environment (SPLICE) [104–107].

In SPLICE, stereo data is used, i.e. two data streams, one clean stream and one noisy stream. Both the noisy and clean feature distributions are modeled as GMM, and it is assumed that there is a one-to-one correspondence between the mixtures in the clean and noisy GMMs. Furthermore, for any pair of Gaussians, the mean vector of clean speech is assumed to be just the summation of the corresponding noisy mean vector plus a correction vector associated this Gaussian pair. With these assumptions, the MMSE estimate of clean feature vector based on noisy feature vector is shown to be

$$\hat{x} = y - \sum_{i=1}^{M} p(i|y)\mathbf{r}_i,$$  \hspace{1cm} (3.32)

which is similar to the MMSE estimator of RATZ. The correction vectors $\mathbf{r}_i$ are also trained from the stereo data. The maximum likelihood criterion is used to find the optimal correction vectors that corrects the noisy feature vectors to their corresponding clean feature vectors. Note that for SPLICE to work well, the noisy training data should be distorted by similar environmental distortions, e.g. same noise type and SNR level. In another word, SPLICE can only compensate for a single type of environment.

SPLICE can only work with noisy test features that are similar to the noisy training features. If there is mismatch between training and testing data, performance degrades. This constraint can be relaxed by using multi-environments in the SPLICE framework. During the training, the training data are classified into several environments, e.g. according to their noise type and SNR level. For each environment, a noisy GMM and correction vectors are trained using SPLICE. During testing, the environment of the test data are first estimated and the most similar training environment is used to compensate the test data. The environment selection is based on the likelihood of the test data on
various environments. The selection is also smoothed over time. The estimated clean vector can also be a weighted sum of the estimation obtained from all environments and the weights can be the posterior probability of the environments. Similar environment partition strategy is also used in interpolated RATZ (IRATZ) [38].

A recent data-driven feature compensation method that also makes use of environment partition is the multi-environment model-based linear normalization (MEMLIN) [103]. Similar to SPLICE, the noisy training data are partitioned into several environments and each environment is modeled by a GMM. Unlike SPLICE, MEMLIN also models clean speech directly using a GMM, and a correction vector is trained for any pair of the clean Gaussian and noisy Gaussian. The MEMLIN is shown to outperform IRATZ and multi-environment SPLICE [103].

Another data-driven feature compensation method is called the stochastic matching [79, 108, 109]. In stochastic matching, noisy feature vector is mapped back to clean feature vector by

$$
\hat{x} = F_\nu(y) \quad (3.33)
$$

where $F_\nu(\cdot)$ is the mapping function and $\nu$ is the parameters of the mapping function. In [79], the mapping function is simply

$$
x_t = y_t - r_t \quad (3.34)
$$

where $t$ is the frame index and $r_t$ can either be a constant over an utterance or time-varying. This formulation is actually quite similar to that of previously reviewed methods, such as RATZ and SPLICE. The major difference between stochastic matching and other methods is that the $r_t$ is estimated by maximizing the likelihood of noisy observations as follows:

$$
\hat{\nu} = \arg \max_\nu p(Y|\nu, A_x) \\
= \arg \max_\nu \sum_s \sum_C p(Y, S, C|\nu, A_x) \quad (3.35)
$$

where $A_x$ is the HMM-based acoustic model trained from clean features, $Y$ is the collection of feature vector for an utterance, $S$ and $C$ are all possible state sequences and mixture sequences of $Y$. The optimization could be solved by EM algorithm.
Chapter 3. Review of Robust Speech Recognition Techniques

The stochastic matching approach is also used for model adaptation. We will review this when we review model adaptation methods. The linear feature mapping function in [79] is generalized to nonlinear function by using an artificial neural networks (ANN) in [108]. The training is implemented using the generalized EM algorithm and modest improvement over linear mapping function is obtained. In [109], an SNR-incremental stochastic matching (SISM) is proposed to improve the selection of initial condition of EM algorithm that affects the final performance.

Another group of data-driven feature compensation methods make use of the missing feature theory (MFT) and is quite different from the above mentioned methods [110, 111]. The basic idea of MFT is that the noise distortion affects the time-frequency representation of speech signal differently and we should rely on those less affected speech information. Consider speech spectrum as an example. The locations with high speech energy such as harmonic peaks are less distorted by noise than those locations with low speech energy such as harmonic valleys. The less affected locations are seen as reliable and the more affected are seen as unreliable or missing features. There are two ways to apply the MFT to speech recognition. One approach is to reconstruct the unreliable locations, i.e. the missing features, before the recognition takes place. The other approach is to modify the speech recognizer to not consider these missing features during the recognition process. We will review the first approach here.

Two methods have been proposed to reconstruct the unreliable log Mel filterbank coefficients prior to MFCC generation [15, 112]. The first algorithm, called the correlation-based reconstruction, estimates the filterbank coefficients of the missing components using the relationship between the missing components and the reliable neighbour components. The second algorithm, called the cluster-based reconstruction method, uses a Gaussian mixture model (GMM) to model the distribution of the log Mel filterbank coefficient vectors and reconstructs the missing components’ filterbank coefficients using MAP criterion.

Another feature reconstruction method which operates during the recognition process is proposed by Cooke [111]. It is called the state-dependent imputation. It estimates the missing input features as the mean of the state-dependent distribution of the missing coefficients conditioned on the reliable coefficients during the recognition process. The
performance of the imputation algorithm was improved in [113, 114] by estimating the cepstral domain coefficients directly from the log Mel filterbank coefficients through a nonnegative least square approach.

### 3.3.3 Feature Normalization

Unlike speech enhancement and feature compensation methods that aims to recover the clean speech coefficients, the feature normalization methods normalize the speech coefficients, usually cepstral coefficients, to a new space where the noise distortion is reduced. It should be mentioned that both the compensation and normalization methods modify feature vectors and thus the difference between them is not very clear, however, feature normalization methods usually modify certain statistics of features, e.g. global means and variances, to some reference values which are usually obtained from clean speech or simply pre-defined values. A rationale of doing so is that the statistics of speech features are changed when speech signal is distorted by noise. By normalizing the statistics of the speech features, it is expected that some systematic distortion caused by noise will be reduced. In this section, we will review major feature normalization methods.

A simple and effective feature normalization method is the cepstral mean normalization (CMN, also called cepstral mean subtraction, CMS) [19–21]. Note that CMN is already introduced as a data-driven feature compensation method in section 3.3.2.2. However, it can also be treated as a feature normalization method. The working of CMN is very simple: it subtracts the features’ mean values from the features. After subtraction, all the feature dimensions will have a zero mean. CMN is known to be able to reduce convolutional noises, such as microphone mismatch and linear transmission channels distortion. This is because convolutional noises becomes multiplicative in the frequency domain and additive in the log filterbank and cepstral domain. If the convolutional noise is fixed, it causes a constant shift in the log filterbank and cepstral domain. Therefore, by subtracting the mean from the feature for both clean and noisy speech, the convolutional noise can be removed in theory. The basic CMN [19, 20] estimates the sample mean vector of the cepstral vectors of a sentence and then subtract this mean vector from every cepstral vector of the sentence. Later, an augmented cepstral normalization method [21]
estimates the mean vectors for the silence and speech segments of the sentence separately and achieved better results. Instead of using a hard decision on whether a frame is silence or speech, one improvement suggests the use of the a posteriori probability of the frame of being silence \( p(n) \), which is similar to the speech absence probability used in the MMSE STSA. The final mean vector is the weighted sum of the silence mean and speech mean, with the weights be \( p(n) \) and \( 1 - p(n) \) respectively. In another study [115], CMN is also used together with microphone array and is called position-dependent CMN. The speaker’s position is first estimated by the microphone array, then a pre-trained feature mean for the location is used to perform CMN. In general, the advantage of CMN is its simplicity, low computational cost and easy to be implemented. However, its performance is limited as it uses very few items of prior information about speech and noise, and the use of a single compensation vector provides very little flexibility.

Besides mean normalization, the cepstral variance normalization (CVN) [22] normalizes the variances of features to unity. It is well known that noise distortion can change the variance of speech features [38]. At different SNR levels, the variance of features may be very different. The CVN is similar to a dynamic gain control. It normalizes the total power of feature trajectories to reduce the difference among features of different environmental conditions. In practice, CMN and CVN is normally used in cascade and called the mean and variance normalization (MVN).

In [22], segmental implementation of MVN is proposed. It was found that by limiting the analysis length of MVN to a suitable duration, e.g. several seconds, better performance could be obtained. One reason for the improved performance is that the power of feature trajectories may vary within a long utterance. By using a shorter analysis segment length, more uniform power of feature trajectories can be achieved throughout the whole sentence. The optimal segment length may be database-dependent. It is noted that when there are very long silence segments in the data, the segment length needs to be big such that one MVN analysis segment will cover some speech frames. If the whole MVN segment consists of only silence/noise frames, the noise will most likely be amplified and speech recognition performance will be affected.

While CMN and CVN normalizes the first and second moments of features respectively, histogram equalization (HEQ) [23–25] normalizes the histogram of the features,
i.e. the probability density function (p.d.f.) of the features. Originally used in image processing to automatically balance the contrast of images, HEQ is a technique that can change the histogram of any random variable to match any other desired histogram. For a random variable $x$ with known cumulative distribution function (cdf) $C_x$, we can change its cdf to $C_y$ by performing

$$y = C_y^{-1}(C_x(x)) \quad (3.36)$$

where $y$ is the transformed version of $x$. In speech recognition systems, HEQ can be applied to normalize the distribution of speech features as follows. A reference histogram is first learnt from the training data of acoustic model. Then the histogram of the test features are normalized towards this reference histogram. The process is performed on a dimension-dependent and utterance-wise basis. Besides histogram of training data, we can also use common probability distributions, such as Gaussian distribution, as the reference. Usually, both the training and testing features are processed by HEQ.

HEQ can be seen as a generalization of CMN and CVN, since when the histogram (p.d.f.) of features are normalized, all moments should be normalized. From another viewpoint, CMN and CVN provide a linear transformation of the features, while HEQ is able to transform the features nonlinearly.

The CMN, CVN/MVN, and HEQ all have two assumptions, i.e. the assumption that the noise distortion does not change the order statistics of feature trajectories within an utterances or segment; and global statistics of an utterance match that of the whole training set. The order statistics of a feature trajectory refers to which element of the trajectory has the highest value, the second highest value, and so on. In fact, noise distortion usually breaks this order. Hence, even if we can normalize the histogram of the noisy trajectory to the histogram of corresponding true clean trajectory, the normalized trajectory won’t be the same as the clean trajectory. Besides this assumption, the assumption about matched statistics is also violated in real situations. In training data, we have a balanced proportion of all the phonemes. However, during testing, as there are very limited number of phonemes in an utterance, the phoneme composition of an utterance may be quite different from that of the training set. Hence, it is coarse to normalize the histogram of just one utterance to the global histogram of the entire training
set which usually consists of thousands of utterances. There is another simple example to demonstrate the drawbacks of CMN, CVN, and HEQ. Suppose there is an utterance with several words. If several silence frames are appended in both the front and end of the utterance, another utterance is obtained. Although the acoustic content of these two utterances are exactly the same, their normalized versions by CMN, CVN/MVN, and HEQ will be different due to the different proportions of silence frames in the two utterances.

The violated assumptions of CMN, CVN, and HEQ are alleviated by the use of cluster-based normalization techniques. As we already reviewed previously, in [21] a two-class CMN is used, one class for speech frames and one class for silence frames. In [26], a more general solution is proposed, i.e. the class-based HEQ. In this method, the clean training feature vectors are first clustered, then a reference histogram is estimated for each cluster. During recognition, the noisy feature vectors are first classified into clusters, and then the conventional HEQ is performed for each cluster independently. With class-based HEQ, it is possible that the order statistics of the normalized feature vectors will be different from that of the original vectors. Although this does not guarantee that the order statistics of the normalized features will be more like the corresponding clean features, cluster-based HEQ is shown to outperform traditional HEQ significantly in [26] at all SNR levels other than the clean case.

3.3.4 Temporal Filters

Filtering of feature trajectories is also a popular approach to improve the robustness of speech recognition against noise corruption. Typically, the filtering is applied to the trajectories of log filterbank coefficients or cepstral coefficients, which are treated as time domain signals. The filters are usually called temporal filters. The most significant difference between temporal filters and previous feature domain methods are that temporal filters modify the correlation of features, i.e. second order statistics of features or modulation spectrum, while previous methods modify the probability distribution of features, i.e. first order statistics of features.

A common temporal filtering technique is the extraction of delta and acceleration features [116]. The delta and acceleration features are generated using (2.2), which can
Figure 3.5: Comparison of the magnitude responses of common temporal filters.

be seen as a finite impulse response (FIR) filter. The magnitude of delta filter is shown in Fig. 3.5. It is observed that the delta filter is a band-pass filter with the center of passband near 15Hz modulation frequency. Delta and acceleration features are usually appended to the static features and they are shown to improve the performance of speech recognition significantly.

Another method, the CMN, can also be treated as a temporal filter. Strictly speaking, the magnitude response of CMN is time-varying and can only be roughly seen as a high-pass filter. CMN eliminates the very low frequency components of feature trajectories that could be caused by but not limited to channel distortions.

The first commonly used temporal filter specifically designed to reduce the effect of channel distortion and additive noise is the RASTA filter (relative spectra) [27]. The RASTA filter is an IIR filter whose transfer function is defined as:

\[ H_{RASTA}(z) = 0.1 z^4 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 -pz^{-1}} \]  

(3.37)

where \( p \) is a parameter controlling the cut-off frequency of the high-pass portion of the filter. Typically, \( p \) is set to either 0.98 or 0.94 [27]. The magnitude response of RASTA
is shown in Fig. 3.5 with \( p = 0.94 \). It is a bandpass filter that removes the very low frequency and high frequency components of feature trajectories. The design agrees with research findings that speech modulation frequency of 1-16Hz is most important for both human and automatic speech recognition [117–122]. RASTA and CMN are both able to reduce channel distortions, and they can be used in concatenation to produce better results.

Another well-known temporal filter designed to reduce feature variation is the autoregressive moving average (ARMA) filter used in the MVA processing [28]. The ARMA filter is defined as:

\[
y(t) = \frac{y(t - M) + y(t - M + 1) + \ldots + y(t - 1) + x(t) + \ldots + x(t + M)}{2M + 1}
\]  

(3.38)

where \( x(t) \) and \( y(t) \) are the input and output of the filter, \( M \) is the order of the filter. The transfer function is:

\[
H_{ARMA}(z) = \frac{z^M}{2M + 1} \frac{1 + z^{-1} + \ldots + z^{-M}}{1 - (z^{-1} + z^{-2} + \ldots + z^{-M})/(2M + 1)}
\]  

(3.39)

As shown in Fig. 3.5, the larger the filter order \( M \), the lower the cut-off frequency of the ARMA filter. The optimal value of \( M \) is usually dependent on the task.

Besides the empirically designed temporal filters such as RASTA and ARMA, some researchers also propose to use data-driven methods for filter design [29–34, 123]. The filters are usually designed from some training data which can be both clean and noisy. Typically, discriminative criteria are used to guide the filter design, e.g. linear discriminative analysis (LDA) and minimum classification error (MCE). Besides, principle component analysis has also been used. The filter parameters are estimated by optimizing the objective function of these criteria. The resulting filter are mostly low-pass or band-pass, similar to RASTA and ARMA filters.

Most of current temporal filters are fixed after being designed. They are not able to track the changes of signal condition during speech recognition. We will discuss one of our novel contributions in this thesis, the temporal structure normalization filter, that is able to track environment condition in the next chapter.
3.4 Model Adaptation Techniques

In contrast to feature domain methods that aim at making features more consistent in various environmental conditions, the model adaptation methods adapt acoustic model to make it better fit to the noisy acoustic environment. In this section, several model adaptation techniques will be reviewed and compared. Based on whether environment model is used, these methods can be grouped into data-driven-based adaptation and environment-model-based adaptation.

3.4.1 Data-driven-based Adaptation

3.4.1.1 STAR

The STAR algorithm of Moreno [38, 78] is closely related to the RATZ feature compensation algorithm described in section 3.3.2.2. As we will find more such example later, feature compensation methods usually have a model adaptation counterpart. The basic concept of STAR is similar to that of RATZ. However, unlike RATZ which uses a separate GMM for the prior distribution of clean speech, STAR utilizes the HMM in the CMU SPHINX-II speech recognition engine directly. The SPHINX-II speech recognition engine uses discrete HMM acoustic models and all the states share a pool of 256 Gaussians. STAR estimates the correcting terms, $\mu_k$ and $\Sigma_k$, for the 256 Gaussians using the same way as RATZ, and then compensates the clean mean and variance vectors to approximate the noisy speech distribution. As these Gaussians are shared by all HMM models, once they are compensated, all the HMM states are adapted.

3.4.1.2 Stochastic Mapping

In section 3.3.2.2, we reviewed the stochastic matching method that compensate the noisy feature vectors. The same method can also be used to compensate the acoustic model’s parameters [79, 108, 109]. In stochastic matching, it is assumed that the matched acoustic model can be adapted from the clean-trained acoustic model by

$$\Lambda_y = G_\eta(\Lambda_x)$$  \hspace{1cm} (3.40)
where $G_n(\cdot)$ is the transformation function and $\eta$ is its parameters. In [79], the transformation function is assumed to be:

\begin{align}
\mu_y &= \mu_x + \mu_b \\
\Sigma_y &= \Sigma_x + \Sigma_b
\end{align}

where $\mu_b$ and $\Sigma_b$ are the correction mean and variance and they are estimated for every Gaussian in the acoustic model. Note that in RATZ and STAR, the same type of correction vectors are assumed to estimate noisy speech distribution from clean distribution. The difference is that in stochastic matching, the model is HMM rather than GMM. Similar to feature space stochastic matching, the correction vectors are estimated by maximizing the likelihood of noisy utterance

$$
\hat{\eta} = \arg \max_{\eta} p(Y|\eta, \Lambda_x)
= \arg \max_{\eta} \sum_{S} \sum_{C} p(Y, S, C|\eta, \Lambda_x)
$$

where $\Lambda_x$ is the HMM-based acoustic model trained from clean features, $Y$ is the collection of feature vector for an utterance, $S$ and $C$ are all possible state sequence and mixture sequences of $Y$. The optimization could be solve by EM algorithm. The stochastic matching is improved later by using nonlinear mapping function in [108] and by an SNR-incremental stochastic matching in [109].

### 3.4.1.3 MAP and MLLR

Another two model adaptation methods, the maximum likelihood linear regression (MLLR) [37] and MAP [36, 80], are originally designed for adapting speaker independent acoustic models to a specific test speaker. Due to the similarity between the speaker adaptation and environment adaptation, they are also used for noise robust speech recognition.

The MAP approach adapts the acoustic model by optimally using the prior information in the clean trained acoustical model and the posterior information in the noisy observations. The observations are recognized by speech recognition, and only those with high acoustic likelihood score are used for adaptation. The Bayesian adaptation framework used in the MAP approach enables the optimal use of the noisy observations
in model adaptation. When the adaptation data are few and the posterior information is weak compared to the prior HMM acoustical models, the models are not adapted much. As there are more and more adaptation data, the models becomes asymptotically equivalent to the ML estimate from noisy observations, which provides optimal decision rule on the test data. However, this adaptation process is quite slow, as only the model parameters directly related to the adaptation data are adapted. In real applications, the adaptation data are few and hence it is necessary to reduce the number of model parameters needed to be adapted.

To achieve good adaptation performance, MLLR [37] uses the parameter sharing strategy, i.e., the similar models are tied together and their parameters are adapted together. The degree of model tying is high if the available amount of adaptation data is low and vice versa. For very few data, a global transform strategy maybe used. The basic MLLR adapts the mean vectors of the Gaussian by multiplying it with a transform matrix, which is obtained using maximum likelihood criterion and EM algorithm. The models tied together share the same transformation matrix. The advantage of the MLLR is its ability to provide good adaptation even if data are few. However, MLLR has poor asymptotic properties, which leads to the fast saturation of performance gain with increased data. Usually, the MLLR outperforms the MAP if the adaptation data are few, but MAP adapts the models better when there are a lot of data. The combination of the two yields best performance.

3.4.1.4 Ensemble Modelling

In practice, there is often very little or no data for supervised adaptation. It is thus important to reduce the number of free parameters that need to be estimated during adaptation. In the eigenvoice-based speaker adaptation method [124], the number of free parameters is reduced to about 10 such that these parameters can be estimated from limited data robustly. The ensemble modelling can be seen as a generalization of the eigenvoice approach for the environment-adaptation problem [42, 43].

In the ensemble modelling approach, an ensemble of acoustic models are trained using speech data of various environment conditions, e.g. different noise and SNR combinations. After obtaining $P$ acoustic models, the mean vectors of Gaussians in each model
are concatenated to form a supervector and there are totally $P$ supervectors, one supervector for one acoustic model. Each supervector has $M \times D$ dimensions, where $M$ is the number of Gaussian mixtures in an acoustic model and $D$ is the feature dimension. The idea is to estimate a supervector from these $P$ supervectors based on the noisy observations. The estimated supervector can then be used to construct an acoustic model for speech recognition.

In its most general form, the test supervector can be estimated as follows:

$$\hat{s}_{test} = \sum_{p=1}^{P} A_p s_p + b \quad (3.44)$$

where $\hat{s}_{test}$ is the estimated supervector that is supposed to be matched with the noisy test data, $s_p$ and $A_p$ is the $p^{th}$ supervector and transformation matrix, and $b$ is a correction vector. The transformation matrices $A_p$, $p = 1, ..., P$ and the correction vector $b$ can be estimated using the maximum likelihood criterion and the noisy test data. Note that, if there is only one model in the ensemble, i.e. $P = 1$, the approach degenerates to MLLR. A problem with this general form is that there are too many parameters and it is very difficult if not impossible to estimate these parameters robustly.

A practical form of ensemble modelling approach [42, 43] is as follows:

$$\hat{s}_{test} = \sum_{p=1}^{P} w_p s_p \quad (3.45)$$

where $w_p$ is a scalar weight rather than a transformation matrix. In addition, the correction vector is removed. In this formulation, the number of free parameters is $P$. To further reduce complexity, $K$ eigenvectors ($K \leq P$) can be obtained by PCA and used to replace $s_p$ in (3.45). In this case, there are only $K$ free parameters. It is reported that the ensemble modelling approach outperforms MLLR significantly on Aurora-2 task.

Two extensions of ensemble modelling were reported in [42]. The first extension is to use a tree-based clustering of the environments. During testing, the environment is first estimated, and only the supervector of the selected environment are used for adaptation. The second extension is to use minimum classification error (MCE) criterion rather than ML [47, 48] to obtain discriminative supervectors.
3.4.2 Environment Model-based Adaptation

Another group of model adaptation methods uses the information of the environment model. We will review two popular such methods, the parallel model combination (PMC) [35] and Joint compensation of Additive and Convolutional noise (JAC) [39].

3.4.2.1 PMC

Gales and Young [35] proposed the PMC approach, which synthesizes a noisy acoustic model using clean acoustic model and a noise model. In PMC, the noise is represented by a single or multi-state HMM depending on whether the noise is stationary. During the adaptation process, the noise model can be trained from the frames of silence segments in the testing utterances. Both the clean acoustic model and noise model are trained from cepstral features, however, the noisy acoustic model is synthesized in linear spectral domain, i.e. the filterbank domain before natural logarithm and DCT. Therefore, it is necessary to convert the mean and variance vectors of the clean and noise models back to the linear spectral domain first. After the noisy acoustic model is obtained in the linear spectral domain, it is then converted to the cepstral domain and used for speech recognition. The parameters of the clean acoustic model is compensated by adding the parameters of the noise model in the linear spectral domain. Specifically, for each clean and noise state pair, the mean vectors and covariance matrices of the two model are combined using the following formulae

\begin{align}
\hat{\mu} &= g\mu + \hat{\mu} \\
\hat{\Sigma} &= g^2\Sigma + \hat{\Sigma}
\end{align}

(3.46) (3.47)

where \((\hat{\mu}, \hat{\Sigma})\), \((\mu, \Sigma)\) are the noisy and clean speech model parameters, \((\hat{\mu}, \hat{\Sigma})\) are the noise model parameters, all in linear spectral domain. The gain matching term \(g\) is used, as the relative strengths of the speech and the noise in the testing environment may be different from these of the training environment, and it is estimated as

\[ g = \frac{E_{ns} - E_n}{E_s} \]

(3.48)

where \(E_s\), \(E_{ns}\) and \(E_n\) are the average energy of the clean speech, noisy speech and background noise respectively.
Experiment results in [35] shows that PMC significantly improves recognition accuracy on an isolated digit task. Furthermore, the use of multi-state HMM (2-4 states) for non-stationary noise produces much better results than using single state HMM.

Later, another technique [125] also used the concept of HMM decomposition and it considered both additive noise and convolutional reverberation at the same time.

3.4.2.2 JAC

JAC [39] is another model adaptation method that uses an environment model. Similar to PMC, JAC also transforms the acoustic model’s parameters back, but to log filterbank domain rather than linear spectral domain. Furthermore, JAC deals with both additive and convolutive noises, while original PMC only compensate for additive noise. In JAC, the phase-insensitive environment model is used. JAC first estimates the additive and convolutive noises from the current noisy test utterance using an EM algorithm. With these noise estimates, JAC adapts acoustic model’s parameter in the log filterbank domain, and then converts the parameters back to cepstral domain. Note that the adapted model is used to decode the test utterance, and the output alignment information is used to obtain a better estimation of the channel distortion, which is used as the initial value for the channel distortion for next test utterance. The additive noise estimate of current utterance is not similarly used in next test utterance as additive noise is assumed to be highly non-stationary.

There are two extensions to JAC [40,41]. In the first extension [40], vector Taylor series (VTS) is used to linearize the nonlinear distortion in the environment model. Another improvement is that the adaptation is now carried out in the cepstral domain directly and there is no need to convert the model parameters to log filterbank domain. In addition, the extended JAC also adapts the Gaussian variances of the acoustic model, which are not adapted in original JAC. The second extension [41] is an improvement over the first extension. The major improvement is that the phase-insensitive environment model is replaced by the phase-sensitive model for more accurate modelling of the relationship between noises and speech. Experimental results showed that the use of a phase-sensitive model significantly improves recognition performance on the Aurora-2 task, where the data follows the phase-sensitive model well.
3.5 Summary

In this chapter, we reviewed previous noise robust speech recognition techniques and discriminative model training methods. Previous robustness methods mainly focus on reducing the mismatch between the clean-trained acoustic model and the noisy test features. The mismatch can be reduced in the feature domain or in the model domain or both, and both approaches are effective. In the next two chapters, we are going to introduce our proposed methods. In Chapter 4, the proposed temporal structure normalization (TSN) filter will be described. The TSN filter is designed to reduce the feature-model mismatch by reducing the feature distortion in the modulation spectrum domain. In Chapter 5, the margin-based model training method SME is applied to obtain better robustness of the acoustic model by improving its generalization capability.
Chapter 4

Normalizing the Temporal Structure of Feature Trajectories

In the previous chapter, we have reviewed some of the current robustness techniques. Among them, there is one group of techniques that are of great interest to us, i.e. the techniques that utilize the temporal information of speech feature trajectories. For example, the temporal filters [27–34], have been used to improve the temporal characteristics of feature trajectories for better robustness. In this chapter, we will propose a novel temporal filter design method that normalizes the temporal structure of feature trajectories.

The temporal information of feature trajectories is an important element in human speech recognition. In the feature extraction stage of current ASR systems, a speech signal is divided into overlapping frames and a feature vector is extracted from each frame. As the statistics of the speech signal is slowly varying and the frames are overlapped, the characteristics of neighbouring frames are similar and hence the extracted feature vectors are highly correlated. This gradual evolution of feature vectors with time, i.e. temporal information of speech, plays an important role in discriminating different speech sounds. In fact, researchers have showed that the temporal information in speech signals in some cases may even be more important than the spectral information which represents the short-term structure of speech within one frame. For example, in [126], Shannon et. al. conducted an experiment where the spectral information of speech signal is removed and human listeners could still recognize different consonants and vowels accurately based mainly on temporal information. Although the finding of Shannon et. al. may not be
true for other languages, it shows the importance of temporal information for speech recognition by human.

Although current ASR systems take a very different approach from the human speech recognition, whose working mechanism is still unclear, it is reasonable to argue that the temporal information of feature trajectories is also important for ASR. Current mainstream statistical ASR systems usually use hidden Markov model (HMM) to capture the temporal structure of speech features for better speech recognition [1–5]. However, the transition probabilities from states to states only loosely model the dynamics of speech. Besides HMM, dynamic features, such as delta and acceleration features [116], are also widely used to provide additional temporal information. Dynamic features are bandpass filtered versions of the speech features and they capture temporal information up to about 100ms duration and their use as features improves speech recognition performance significantly. Other techniques utilizing temporal structure have also been proposed, for example, the temporal-derived features [57], the modulation spectrogram features [127], and the temporal filtering of feature trajectories [27–34].

In this chapter, we will study a novel temporal filter design method, called the temporal structure normalization (TSN) filter, for robust speech recognition. The temporal structure of features are represented by the power spectral density (PSD) function of the features, i.e. the modulation spectra of the speech signal. Our objective is to design a temporal filter that normalizes the noise-corrupted modulation spectra of speech to a group of reference modulation spectra that represents the temporal information of clean speech. Compared to other temporal filters [27–34], an advantage of the TSN filter is that it is able to automatically adapt to environment distortions in the speech signal and provides customized filtering. The work to be presented in this chapter has been published in [128–131].

4.1 Noise Effect in Modulation Spectra Domain

To better exploit temporal information for speech recognition, it is necessary to examine the temporal information systematically, especially under noise distortions. In this section, we will carry out such an examination. We first introduce the term modulation spectrum, which can be interpreted as the spectrum of the feature trajectory, we
then review published work about modulation spectrum and carry out examination on how modulation spectrum would be affected in noisy environments. These observation motivates us to propose the temporal structure normalization filter.

### 4.1.1 Definition of Modulation Spectra

Before introducing the definition of modulation spectrum, let’s first look at an example of speech signal in Fig. 4.1. The speech signal is one clean utterance, with its contents being English digits “Zero One One”. Here, an example of amplitude modulation existing in speech signal will be shown, especially during vowels. The waveform of the utterance is shown in subfigure (a). Before a meaningful amplitude demodulation is made to the signal, it is necessary to apply bandpass filtering to the signal so that the filtered signal can be roughly treated as a narrowband signal. As suggested in [132], a Gabor bandpass filter with passband centered at 500Hz (see subfigure (b)) is used to filter the signal. The filtered signal and its amplitude modulation signal are shown in subfigure (c). The amplitude demodulation is carried out by using DESA-1 method of [132]. From the magnified view of the signal in subfigure (d), we can clearly see the low frequency amplitude modulation signal and the high frequency carrier signal, just like those in a amplitude modulation (AM) communication system.

After examining the previous example, we now introduce the modulation spectra. The term modulation spectra was introduced by Houtgast and Steeneken [117, 118] when they proposed the modulation transfer function (MTF) to measure speech intelligibility in the field of room acoustics. The meaning of modulation in speech signal is similar to that of AM [133] in communication systems. In an AM system, a low frequency information-bearing signal is used to modulate the amplitude of a high frequency signal to transmit the information through the medium. Speech signal can also be approximately viewed as an AM system: the very low frequency inaudible message-bearing waves are used to modulate the amplitudes of the audible sound waves [134]. Since speech is wide band signal, it is more appropriate to analyze the amplitude modulation for individual frequency bands. In each frequency band, the speech signal can be considered as narrow band and its energy envelope can be seen as the amplitude modulating signal of the band. The modulation signal’s PSD function (the modulation spectrum) usually varies
Figure 4.1: Illustration of the amplitude modulation property of speech signal. The subfigures are: (a) An English utterance “Zero One One” from the Aurora-2 database [69]. The sampling rate of the signal is 8000Hz; (b) The magnitude response of the Gabor band-pass filter used to filter the speech signal. The center frequency of the filter is at 500Hz and length of filter is 201 taps. See [132] for design of the Gabor filter; (c) The band-pass filtered signal and estimated amplitude modulation signal estimated by using DESA-1 method of [132]; (d) Magnified view of the 0.05-0.15 second interval of the filtered signal and amplitude modulation signal.
from band to band. The collection of the modulation spectra from all frequency bands forms the joint acoustic-modulation frequency representation of the speech signal [135]. The “acoustic frequency” here refers to the “conventional Fourier decomposition of the signal” [135] and the “modulation frequency” is for the modulating signal.

In the context of feature extraction for speech recognition, a convenient way of calculating the modulation spectra is described as follows (see Fig. 4.2). For a time domain speech signal $x(n)$, its short-time power spectral density (i.e. the spectrogram, shown in the left panel of Fig. 4.2) is defined as:

$$|X(t, f)|^2 = |\text{STFT}[x(n)]|^2$$  \hspace{1cm} (4.1)

where $n$, $t$ and $f$ are the sample index, frame index and acoustic frequency index, respectively. The STFT[$\cdot$] denotes the short-time Fourier transform (STFT) and $|\cdot|$ denotes the magnitude. In speech recognition, the frame length is normally 25ms and the frame shift is usually 10ms. With such setting, there are 100 frames of speech samples per second. The trajectory of spectrogram coefficients in a single acoustic frequency bin represents
the energy envelop of the speech signal in the acoustic frequency band centered at the bin (see the slices of the spectrogram in Fig. 4.2). Therefore, the trajectory of a single frequency bin can be interpreted as the down-sampled modulating signal of speech (i.e. from sampling rate of speech signal, e.g. 8000Hz, to frame rate of STFT, e.g. 100Hz) and its PSD function is the modulation spectrum for the acoustic frequency bin. The collection of the PSD functions for all the bins form the modulation spectra of the speech signal. While the maximum acoustic frequency is determined by the sampling frequency of the speech signal, the maximum modulation frequency is related to the frame rate of the STFT, which is the sampling rate of the feature trajectories. With a typical setup, the frame rate is 100Hz and the maximum modulation frequency is 50Hz. An example of speech modulation spectra and the corresponding spectrogram are shown in Fig. 4.3.

In this thesis, the definition of modulation spectra is loosely extended from spectrogram to feature coefficients, for example, to describe log filterbank coefficients and cepstral coefficients. In this way, the modulation spectra derived from feature coeffi-
Figure 4.4: Comparison of the count histogram of speech syllables and modulation spectrum. The frequency histogram for 2925 syllables is obtained from a portion of the Switchboard corpus. The modulation spectrum is obtained from a two minutes speech signal from a single speaker. The modulation spectrum is obtained from the 1-2kHz acoustic frequency band of speech signal. Reprinted from [136].

coefficients are not the joint acoustic-modulation frequency representation of speech signal, but rather the temporal characteristics of the feature coefficient trajectories. In the following sections, we will examine the characteristics of modulation spectra generated from different coefficient trajectories.

4.1.2 Importance of Modulation Spectra for ASR

The modulation spectra is determined by the fluctuation of speech power in acoustic frequency bands. This fluctuation is affected by the organizational structure of the speech, such as syllables. For example, in English, the most common syllables are consonant-vowel (CV), CVC, VC, etc. As vowels normally have larger energy than consonants and there may be short silence between syllables, the energy of speech is usually smaller between syllables than within syllables. Hence, the fluctuating rate of speech energy is related to the syllable rate. In [136], Greenberg et. al. showed that the modulation spectrum has similar shape to the histogram of the reciprocal of syllable durations (see Fig. 4.4). This finding indicates that the modulation spectrum is closely related
to the length and frequency of syllables. The peak of the modulation spectrum is 4Hz, which corresponds to the most common syllable length of 0.25 second. The energy of the modulation spectrum concentrates in the 1-15Hz modulation frequency range, roughly corresponding to lengths from 67ms to 1000ms, and this is the length of common syllables. Besides, the average modulation spectra of speech signals are found to be similar for different languages [137].

As the objective of speech recognition is to recognize the words embedded in speech signal, the message bearing modulation signal should play a big role in speech recognition. The modulation signal of different acoustical frequency bands are the temporal information we are interested here. In Shannon et. al.’s experiments [126], these modulation signal is shown to be important for human speech recognition. In addition, different modulation frequency components of the modulation signal are not equally important for speech recognition. Previous research works suggest that the 1-16Hz range of the modulation frequency is most important for speech intelligibility, both for human comprehension and automatic speech recognition [117–122]. This agrees with Greenberg et. al.’s finding [136] shown in Fig. 4.4. The 0-1Hz and 16-50Hz modulation frequency components are less likely to be produced by human beings and more likely due to factors irrelevant to speech recognition, such as noise distortion.

4.1.3 Modulation Spectra of Clean Speech and Noises

When a speech signal is corrupted by noise, its modulation spectra are also affected. To analyze the interactions between speech and noise, we will discuss the characteristics of modulation spectra for speech signals and noises individually in this section and discuss how noise affects the characteristics of speech modulation spectra in the next section.

The modulation spectra of speech signals are known to be affected by reverberations and additive noises. For example, the reverberation usually shifts the peak of the modulation spectra from 4Hz to a lower modulation frequency due to the summation of signals from multiple transmission paths at the receiver [118]. When speech is corrupted by additive noise, it is intuitive that the modulation spectra will also be distorted. Besides reverberation and noise, the characteristics of modulation spectra may also depend
Figure 4.5: Comparison between speech and noise: their feature trajectories and modulation spectra. The subfigures are: (a) an example of babble noise (filterbank coefficients trajectory of the 8th Mel filterbank, same for other noises and clean speech). The 8th Mel filterbank is centered at 656.25Hz with a bandwidth of about 280Hz; (b) an example of car noise; (c) an example of subway noise; (d) an example of clean speech; All these trajectories are processed by mean and variance normalization (MVN) [22] to remove the DC offset and to normalize the power of the trajectories; (e) long-term average modulation spectra of noises and clean speech, all obtained from the 8th filterbank coefficients. The Yule-Walker method [138] is used to estimate the modulation spectra from feature trajectories. The order of the AR model used in the Yule-Walker method is 6 for proper degree of details in the modulation spectra. The modulation spectra are represented in decibel where 0dB indicates a power of 1 and -10dB indicates 0.1.
on other factors, such as the spoken content of the utterance, the speaker’s characteristics, etc. In this section, our interest is to examine environmental factors such as the corrupting noise and the SNR level that could affect the modulation spectra.

Before we examine the noisy speech’s modulation spectra (noisy modulation spectra), let’s first examine the modulation spectra of noises (noise modulation spectra) and the modulation spectra of clean speech (clean modulation spectra) separately. As speech and noise have very different acoustic natures, their modulation spectra will be different. We choose three kinds of noise from the Aurora-2 database [69] as examples: babble noise, car noise and subway noise. All these noises are recorded in real world environments. Examples of these noise filterbank coefficients trajectories are shown in Fig. 4.5(a)-(c) and an example trajectory of clean speech is shown in subfigure (d). Both the noise and speech data are taken from the Aurora-2 database. These trajectories are normalized by MVN to have zero mean and a variance of one. It is observed that the noise trajectories are much less smooth than the speech trajectory. While the noise trajectories are quite random, the speech trajectory has a series of pulses, which are the result of high energy syllables.

The difference between noise and speech can also been observed in the modulation spectrum domain. The average modulation spectra of these three noises are shown in Fig. 4.5(e), together with the average modulation spectrum of the clean speech. The average noise modulation spectrum of each noise is estimated from a 10 seconds long noise segment. The average clean modulation spectra are estimated from 200 utterances (13 male speakers and 13 female speakers), and the total length of these utterances is 340 seconds. Similar averaging methods have also been used to analyze speech in [139] and to obtain smooth MTF in [140]. As the modulation spectra are averaged over 200 utterances, the effects of the spoken content and speaker are reduced and the observation shows the common trend of modulation spectra of clean speech and noises. From the figure, it is clear that the modulation spectra of noises are flatter than that of speech. Similar observations have been reported in [121]. The observation is common for modulation spectra generated from other filterbank trajectories. In addition, similar observations are also found for log filterbank trajectories (Fig. 4.6(a)) and cepstral coefficient trajectories (Fig. 4.6(b)).
Figure 4.6: Comparison of the modulation spectra of speech and noise. The subfigures are: (a) long-term average modulation spectra of noises and clean speech, all obtained from the $8^{th}$ log filterbank coefficients; (b) long-term average modulation spectra of noises and clean speech, all obtained from the $8^{th}$ MFCC coefficients. The settings to obtain these modulation spectra are the same as that of Fig. 4.5, except that different feature trajectories are used.
An inquiry into the production process will help us understand the difference between the modulation spectra of human speech and noises. As the human speech production process is limited by the anatomical prerequisite of the human articulatory apparatus [141], the resulting sound patterns, their energy and changing rate in an utterance are bounded by the physical laws. As a result, in the modulation spectra, the power is concentrated in the low modulation frequencies. Unlike speech signals, background noises usually have less constraints and are more volatile. Therefore, the noises’ power spreads more evenly over the whole modulation spectrum from 0 to 50Hz. As will be shown in the next section, the different characteristics between the speech and noise’s modulation spectra will determine the shape of the noisy modulation spectra.

4.1.4 Modulation Spectra of Noisy Speech

The modulation spectra of noisy speech are jointly determined by clean speech and noise. To better understand the relationship among them, we derive the mathematical representation of the noisy modulation spectra generated from the filterbank coefficients in the Appendix. The corrupting noise is assumed to be additive and independent from the speech signal. From the derivation, the noisy modulation spectrum of any filterbank can be represented as

\[ P_y = P_x + P_n + Q \]

(4.2)

where \( P_x \) and \( P_n \) are the modulation spectra of clean speech and the corrupting noise for the same filterbank, respectively; \( Q \) represents the phase item (see eq. (8) in Appendix). From (4.2), the noisy modulation spectrum are the sum of the clean modulation spectrum, the noise modulation spectrum and the phase item. Our experimental studies have shown that the phase item is usually less significant than the other two items. In other words, the characteristics of the noisy modulation spectrum are mainly determined by the modulation spectra of clean speech and noise.

As speech and noise have different modulation spectra characteristics as shown in section 4.1.3, the mixing of the speech and noise in (4.2) will inevitably change the shape of the noisy modulation spectra. In Fig. 4.7(a)-(d), some example feature trajectories of speech at different SNR levels are shown. The four trajectories are from the same utterance and the only difference among them is the SNR level. The corrupting noise
Figure 4.7: Speech feature trajectories and modulation spectra in different SNR levels. The car noise from the Aurora-2 database [69] is used to corrupt the clean speech to produce the noisy speech. The subfigures are: (a) an example filterbank coefficient trajectory of clean speech (the 8th Mel filterbank, the same for other subfigures); (b) an example filterbank coefficient trajectory of noisy speech (SNR=15dB); (c) an example filterbank coefficient trajectory of noisy speech (SNR=5dB); (b) an example filterbank coefficient trajectory of noisy speech (SNR=-5dB); All these trajectories are processed by mean and variance normalization (MVN) to remove the DC offset and to normalize the power of the trajectories; (e) long-term average modulation spectra of clean and noisy speech, all obtained from the 8th filterbank coefficients. The settings to obtain these modulation spectra are the same as that of Fig. 4.5.
here is car noise from the Aurora-2 database [69]. It is observed that as the SNR level decreases, the speech filterbank coefficient trajectory becomes more corrupted and less smooth. The pulse-like structure of the clean trajectory also becomes less obvious in noisy trajectories. The average noisy modulation spectra (averaged over 100 utterances) of filterbank coefficients for different SNR levels are shown in Fig. 4.7(e). It is observed that the three noisy modulation spectra have higher power than the clean modulation spectrum in the high modulation frequency range, i.e. they are flatter than the clean modulation spectrum. In addition, the flatness of the noisy modulation spectra is inversely proportional to the SNR level. At -5dB, the average speech modulation spectrum is similar to the modulation spectrum of car noise as noise is more dominant than speech in this SNR level.

The observations about the noisy modulation spectra are also true for log filterbank coefficients and cepstral coefficients. To generate the cepstral coefficients for speech recognition, the filterbank coefficients usually undergo two more processing stages: the dynamic range compression and the DCT. The dynamic range compression reduces the dynamic range of the filterbank coefficients, e.g. the logarithm compression and the $n^{th}$ root compression [24]. The DCT reduces the feature vector size and decorrelates the feature dimensions so that the covariance matrix of cepstral features can be treated as diagonal [142]. Due to the complexity of these operations, it is difficult to analyze the modulation spectra derived from log filterbank coefficients and cepstral coefficients mathematically. In order to study the characteristics of the noisy modulation spectra in these two domains, we use experimental analysis. In Fig. 4.8(a),(b), the average noisy modulation spectra generated from log filterbank coefficients and cepstral coefficients are shown. From the figures, we find that the logarithm compression and the DCT process do not change the observations we concluded for the filterbank coefficients. Like in the filterbank domain in Fig. 4.7(e), the noisy modulation spectra in the log filterbank and cepstral domains are generally flatter than the corresponding clean modulation spectra. In addition, the flatness of the noisy modulation spectra are also inversely proportional to the SNR level.
Figure 4.8: Speech modulation spectra in different SNR levels, from log filterbank coefficients and MFCC features. The subfigures are: (a) long-term average modulation spectra of clean and noisy speech, all obtained from the 8th log filterbank coefficients; (b) long-term average modulation spectra of clean and noisy speech, all obtained from the 8th MFCC coefficients. The settings to obtain these modulation spectra are the same as that of Fig. 4.5.
Figure 4.9: Illustration of the TSN framework. The framework is divided into three steps: (a) the offline training of the reference PSD functions; (b) the designing of the TSN filters for current utterance, one filter for each feature trajectory; (c) the filtering of the feature trajectories on an utterance-by-utterance or segment-by-segment basis.

4.1.5 Summary

In this section, we showed that the modulation spectra of noisy speech are different from that of clean speech due to noise corruption (see Fig. 4.7-4.8). In general, speech power concentrates in the low modulation frequencies more for the cleaner signal than for the noisier signal. This leads to different shapes of the modulation spectra in different SNR levels. These observations show a need to correct the modulation spectra of noisy speech and we will discuss our method next.

4.2 Temporal Structure Normalization

4.2.1 Overview of TSN Framework

To reduce the temporal differences between the clean and noisy features, such as the smoothness of the feature trajectory, we propose the temporal structure normalization (TSN) filter. The magnitude response of the TSN filter is designed from the PSD functions of the feature trajectories, i.e. the modulation spectra, since they represent the
temporal characteristics of the feature trajectories. Specifically, the magnitude response is designed to modify the feature trajectory’s PSD function towards a reference PSD function. In this way, the temporal characteristics of the features are normalized. In the following texts, we will use the term PSD function to denote modulation spectrum.

The TSN framework consists of three steps as shown in Fig. 4.9. The first step involves the training of reference PSD functions from a group of clean utterances (see Fig. 4.9(a)). The reference PSD functions are used to represent the common temporal characteristics of the clean features. In the second step, temporal filters are designed for the utterance to be processed, one filter for each feature trajectory (e.g. 39 filters for 39 MFCC features, see Fig. 4.9(b)). The final step is to filter the feature trajectories (see Fig. 4.9(c)). While the reference training step is performed offline, the filter design and filtering steps are performed on an utterance-by-utterance or segment-by-segment basis.

In both the training and processing steps, the N feature trajectories are preprocessed individually by either the mean and variance normalization (MVN) or HEQ before the PSD estimation. The purpose of the preprocessing is to reduce the feature variations and therefore make the features more suitable for temporal filtering. Such preprocessing is common when temporal filters are used to improve features’ robustness for speech recognition [28,143]. In the next two sections, we will describe the filter design process and the training of the reference PSD functions.

4.2.2 Filter Design

The filter design process is summarized in Table 4.1. The design process is divided into two parts: the first part estimates the desired magnitude response of the filter with optional modification (step 1-3 in Table 4.1) and the second part designs the time domain finite impulse response (FIR) filter that implements the desired magnitude response (step 4-7 in Table 4.1). The same filter design process is performed individually for each feature trajectory. The following section describes the design using one of the feature trajectory as an example.

Step 1. Estimate the PSD function of the feature trajectory to be processed

The PSD function of the feature trajectory can be obtained by either non-parametric methods, such as the fast Fourier transform (FFT), or parametric methods, e.g. the Yule-
Walker method [138]. In the design of TSN filter, we choose the Yule-Walker method due to its simplicity and smooth spectrum estimation. As we will discuss later, smooth PSD function is desired for designing the TSN filter, as it is neither necessary, nor realistic to use the details in the PSD function produced by FFT.

**Step 2. Find the desired magnitude response of the filter**

Let \( P_{\text{ref}}(k, j) \) and \( P_{\text{test}}(k, j) \) denote the reference PSD function and the test PSD function of the \( j^{th} \) feature trajectory, respectively, and \( k \) is the modulation frequency index. To simplify the notation, we will drop the feature index \( j \) in the following discussion. The reference \( P_{\text{ref}}(k) \) is trained from clean utterances during the training step (Fig. 4.9(a)) and the \( P_{\text{test}}(k) \) is estimated from the current feature trajectory to be processed. To normalize \( P_{\text{test}}(k) \) to \( P_{\text{ref}}(k) \), the filter’s magnitude response should be:

\[
|H(k)| = \sqrt{P_{\text{ref}}(k)/P_{\text{test}}(k)} \tag{4.3}
\]

Note that this magnitude response is in fact the square-root of the response of the Wiener filter [144].

**Step 3. Optional modification to the desired magnitude**

In step 3 of Table 4.1, we include the option to modify the desired magnitude response to improve performance, e.g., by combining the TSN filter and another filter.

The combination of temporal filters may result in better performance if the filters are complementary to each other, for example, the cascade of RASTA and ARMA filters in [28] and the combination of TSN and ARMA filters in [130] both produce better results than any single filter alone for the Aurora-2 task. To take advantage of other temporal filters, we can simply multiply their magnitude responses with that of the TSN filter:

\[
|H'(k)| = |H(k)||G(k)| \tag{4.4}
\]

where \( |H'(k)| \) is the combined magnitude response, and the \( |H(k)| \) and \( |G(k)| \) are the magnitude responses of the TSN filter and other filters, respectively. \( |G(k)| \) can be any magnitude response. As the filter length used for feature processing cannot be too long,
Table 4.1: Summary of TSN filter design procedures

For the $j^{th}$ feature trajectory of current utterance,

1) Estimate the PSD of the feature trajectory: $P_{\text{test}}(k, j)$.

2) Find the desired magnitude response of the filter using
   \begin{equation}
   |H(k, j)| = \sqrt{P_{\text{ref}}(k, j)/P_{\text{test}}(k, j)}.
   \end{equation}

3) Optional modification of the filter response using (4.4).

4) Find the filter’s weights using the inverse discrete Fourier transform (IDFT). $w(\tau, j) = \text{IDFT}(|H(k, j)|)$.

5) Extract the central taps of $w(\tau, j)$ to form $w'(\tau, j)$.

6) Apply Hanning window on $w'(\tau, j)$ to reduce the truncation effect.

7) Normalize the sum of the weights $w'(\tau, j)$ to one.

the flexibility of the magnitude response of the FIR filter will be limited. Hence, $|G(k)|$
need to be realizable by a FIR filter with a short filter length.

As we will show in section 4.5.6, the desired magnitude response can also be modified to study the relative importance of TSN filter’s magnitude response at different modulation frequencies.

**Step 4. Find the filter weights**

To retain the phases of the feature trajectories and prevent relative phase shift among
the trajectories, finite impulse response (FIR) filters are used due to their linear phase response. The filter structure is non-causal and the filter weights are symmetrical w.r.t.
the central tap. Using the same filter length for all feature trajectories, the resulting filters will have the same group delay for all feature trajectories. The FIR filter weights can be found from $|H(k)|$ by using the Windowing method [144], i.e. taking the inverse discrete Fourier transform (IDFT) of the desired magnitude response. It is noted that the input to IDFT should be two-sided magnitude response, which is a symmetric function representing the gain of the filter from -50Hz to 50Hz (assuming the frame rate be 100Hz).

91
The two-sided magnitude response can be obtained from one-sided magnitude response.

**Step 5. Truncate the filter weights**

The filter weights from IDFT need to be rearranged to a symmetric function, with the central weights normally have the most significant values. The total number of weights is usually hundreds. Filters of this length are not suitable for feature filtering as the transition duration of the filter is half of the filter length. Longer transition duration means more frames at the both ends of an utterance cannot be filtered properly and this may introduce non-uniform temporal structure in the filtered feature trajectories. Therefore, it is suitable to use a short filter, e.g. with less than 50 taps. Therefore, only the central weights are extracted. The number of weights to extract is the final filter length and need to be determined experimentally. Usually, the longer the filter length, the closer is the realized magnitude response to the original magnitude response.

**Step 6 and 7. Windowing and normalization of weights**

The truncation of weights in the previous step can be seen as multiplying the weight vector with a rectangular window function that has value 0 for those discarded weights and value 1 for other weights. In the modulation spectrum domain, the truncation has the same effect as convolving the magnitude responses of the filter and the rectangular window. Hence, the final filter response will be distorted by the truncation. Besides using a rectangular window, we can also apply other windows on the extracted weights, such as Hanning window, that has a lower side lobe in its magnitude response than the rectangular window. With a lower side lobe, spectral leakage can be reduced. On the other hand, using the Hanning window will reduce the frequency resolution of the final filter response, as the Hanning window has a wider main lobe than the rectangular window. However, this is not a problem as TSN does not require very high frequency resolution as we will discuss in the following section.

To preserve the scale of the original feature trajectory during the filtering process, the sum of the weights are normalized to one. Although this will cause imperfect normalization of features’ PSD functions, it is found that by fixing the filter gain to be one at 0Hz, speech recognition performance could be improved slightly.
4.2.2.1 Normalizing the trend of the PSD function only

Suppose the filter with response $|H(k)|$ in (4.3) can be implemented ideally, the filtered feature trajectory will have the following PSD function:

$$P_{\text{norm}}(k) = |H(k)|^2 P_{\text{test}}(k) = P_{\text{ref}}(k)$$  \hspace{1cm} (4.5)

For the speech recognition task, the normalization, however, needs not be carried out exactly. In fact, it is impossible to achieve ideal normalization. As the modulation spectrum carries information about the spoken content, the exact normalization of the modulation spectrum to the reference will render the feature not discriminative. We may only normalize the overall shape of the modulation spectrum which is related to the recording environment, while leaving the details of the spectrum intact. In the filter design process, several steps are designed to avoid extreme normalization. The simplest approach is to use a smooth $P_{\text{ref}}$ and $P_{\text{test}}$ in the filter design process. To obtain smooth PSD functions, the autoregressive (AR) model-based Yule-Walker method [138] is used for PSD estimation instead of the Fourier transform. The smoothness of the PSD function is controlled by the order of the AR model and a small order of 6 is found to be sufficient. A comparison of FFT-based and Yule-Walker-based PSD functions are shown in Fig. 4.10(a). Another step to avoid extreme normalization is to use a short filter. As shown in the step 5 and 6 of Table 4.1, the filter’s weights are truncated and applied with a Hanning window. The truncation and windowing have smoothing effect on the realized filter’s magnitude response. Due to these operations, the realized filter’s magnitude response will be smoother than the ideal one in (4.3) and only the global trend of $P_{\text{test}}(k)$ is normalized. In Fig. 4.10(b), an example of the desired magnitude response and realized magnitude response are shown. The realized response is slightly smoother than the desired response. Note that, the scale of the responses have different scales due to the weight normalization.

4.2.3 Training of Reference PSD Functions

Similar to the role of reference histogram in the HEQ techniques [24–26, 145], the reference PSD functions act as the reference for the normalization of the PSD functions of all feature
Figure 4.10: Illustration of normalizing the trend of the PSD function only. In subfigure (a), the reference PSD function and two versions of test PSD functions of an utterance are shown, all for the 8th MFCC feature. The Yule-Walker method’s order is 6 and the FFT size is 512. In subfigure (b), the desired filter magnitude response (obtained from the Yule-Walker test PSD function in (a)) and the final realized magnitude response are compared.
trajectories, including those of the clean utterances. Intuitively, to fulfill their role, the reference PSD functions should be able to represent the general temporal structure of clean feature trajectories.

In our study, we observe significant variability in the modulation spectra of clean speech. Fig. 4.11 demonstrates this variability. Subfigure (a) shows ten instances of clean modulation spectrum and a mean modulation spectrum averaged over 1000 instances, all from the 2\textsuperscript{nd} MFCC feature trajectory. It is observed that the instances can be very different from that of the mean. The variability is further demonstrated in subfigure (b), where the mean and standard deviation of the clean modulation spectrum are shown. Significant deviation is observed in low modulation frequency range.

The variability of clean modulation spectrum is reasonable. Previously we showed Green et. al.’s findings [136] in Fig. 4.4 that the average modulation spectrum of speech is strongly related to length and frequency of syllables. As different utterances have different composition of syllables, the modulation spectrum of a single utterance will exhibit high level of variability due to their different contents.

To reduce this high variability and extract the common characteristics of clean modulation spectrum, we train the reference PSD functions by averaging the modulation spectrum over a large number of clean utterances (see Fig. 4.9a). The averaging process reduces other factors affecting the PSD functions, such as spoken content and speaker’s characteristics. For each feature trajectory, the reference PSD function captures the feature’s unique temporal characteristics. The data used to train the reference PSD functions are usually the same as that used for the training of the acoustic model of the speech recognition system.

4.2.4 Comparison of TSN and other temporal filters

The most important property of the TSN filter is that its magnitude response is determined by the feature trajectories’s PSD function. Therefore, the TSN is able to adapt its filter response to each trajectory of every utterance. Some of previous temporal filters, such as the RASTA filter [27], the ARMA filter [28] in the MVA processing uses fixed filter responses once the filter is designed. Although the data-driven filter’s filter responses
Figure 4.11: Demonstration of the variability of clean modulation spectra. The subfigures are: (a) 10 examples of clean modulation spectra (thin lines) and mean modulation spectra (dashed bold line). All these modulation spectra are computed from the 2\textsuperscript{nd} MFCC feature trajectory; (b) mean and standard deviation of clean modulation spectra computed from 1000 instances.
are learnt from data [29–34, 123], these filter responses do not adapt to test data during recognition.

The TES [143] and modulation spectrum equalizer (MSE) [146] also normalize the temporal structure of feature trajectories. They share a similar motivation with TSN but all the three filters have been proposed independently from each other. In TES, the filter’s response is

$$H(z) = \frac{A(z)}{B(z)}$$  \hspace{1cm} (4.6)

where $A(z)$ is called the whitening filter and $B(z)$ is used to impose the desired autocorrelation structure on the feature trajectory. If we let $z = e^{-j\omega}$, we will have

$$|A(k)|^2 = \frac{1}{P_{test}(k)}$$ \hspace{1cm} (4.7)\n
$$|B(k)|^2 = \frac{1}{P_{ref}(k)}$$ \hspace{1cm} (4.8)

where $k$ is the discrete frequency bin index, $P_{test}$ and $P_{ref}$ are similarly defined as in (4.3). The magnitude of the transfer function in (4.6) will be the same as that of the TSN filter in (4.3). The difference between the TES and TSN filter is their phase responses. In TSN, linear phase response is achieved by using symmetrical FIR filters. In TES, the phase response won’t be linear as the filter is an ARMA filter.

The magnitude of the MSE filter is the same as that of the TSN filter. There are however two main differences between these two filters. In MSE, an IIR filter with four poles and one zero (same as RASTA filter) are used to approximate the desired magnitude response, while in TSN, FIR filter is used. Furthermore, in MSE, a group of equalizing filters are designed for each noise type, one filter for each feature trajectory, while in TSN, a group of filters are designed for each utterance. Therefore, TSN is a more flexible and general way to adapt the temporal structure of every utterance to the reference. In the next section, we propose the segment-based implementation of TSN that normalizes the temporal structure of every speech segment to the reference for better adaptability.

### 4.3 Segment-based Implementation

The TSN filter presented in Section 4.2 is implemented utterance-by-utterance, which leads to a large processing delay. When short processing delay is desired or environmental conditions change rapidly within one utterance, it is more desirable to process the
features in a segment-by-segment fashion. To achieve this, we propose a segment-based implementation of the TSN filter (we call it TSNseg for short) and discuss it in this section.

### 4.3.1 Implementation Scheme

The way of dividing an utterance into segments is similar to the framing scheme used in the STFT (see Fig. 4.12). The whole utterance is divided into multiple overlapping and equal-sized segments with the last segment usually longer. The shift between two neighbouring segments should be smaller than the segment length to ensure a gradual change of feature statistics between segments. For each segment, TSN filters are designed as if the segment is an utterance. The filter design process is the same as that of the utterance-based TSN describe in section 4.2. The TSN filters of a segment are used to process the frames near the center of the segment with exceptions for the first and last frames as shown in Fig. 4.12.

The clean utterances used for training the reference PSD functions should also be processed segmentally. The segment-based MVN and HEQ are implemented using the same segmenting scheme as the TSNseg. When segments of speech are used to train the reference PSD functions, each segment is treated as an utterance and the same PSD
training procedures are applied as the case where utterances are used for PSD training. This means that the PSD functions are averaged over segments rather than utterances to generate the reference PSD functions. The segment length and shift used during the reference PSD function training match that used for the TSNseg. The training of acoustic model remains the same.

4.3.2 Considerations

Two important parameters of the TSNseg are the segment length and the segment shift. A longer segment provides more frames for better estimation of the feature statistics such as the PSD function. However, a longer segment also leads to poorer adaptation to environmental changes such as a sudden increase/decrease in SNR level. The appropriate length of the segment is determined experimentally and may be environment/database dependent. The other parameter, the segment shift, controls the computational cost and the gradualness of the change of the feature statistics between neighbouring segments. Ideally, the segment shift should be one frame as in several existing segment-based normalization methods [145]. However, with a shift of one frame, the TSN filters need to be re-designed for every frame and the computation becomes too expensive. In our experimental results, we find that the performance of the TSNseg is not very sensitive to the segment shift. We report the choice of these two parameters in the next section.

4.4 Effect of Temporal Structure Normalization on Features

In this section, we will examine the effect of the TSN filter on the feature trajectories and their PSD functions.

4.4.1 TSN alone

We examine the effect of TSN filter alone here. Four example MFCC trajectories as shown in Fig. 4.13 are used for illustration. These trajectories are from the 1st MFCC feature of the same utterance but corrupted by car noise in different SNR levels. It is observed that the noisier trajectories are usually less smooth than the cleaner trajectories.
Figure 4.13: The original feature trajectories. The trajectories are taken from the first MFCC feature of an utterance corrupted by additive noise in different SNR levels. These trajectories are processed by mean and variance normalization (MVN).

Figure 4.14: Effect of TSN filter on feature trajectories. These trajectories are filtered version of the trajectories shown in Fig. 4.13 by TSN filter.
Figure 4.15: Effect of TSN filter on modulation spectra with/without weight normalization.
The TSN-filtered trajectories are shown in Fig. 4.14. It is observed that the clean trajectories is almost unchanged, while the noisy trajectories are smoothed. After the processing, the temporal characteristics of the trajectories, such as smoothness, become more similar to each other.

We also examine the effect of TSN filter in the modulation spectrum domain. In Fig. 4.15(a), the PSD functions of the original feature trajectories of Fig. 4.13 and the reference PSD function are shown. The noisy PSD functions are usually flatter than the clean PSD function. As the reference PSD function is averaged from hundreds of clean PSD functions, it will not be equal to the PSD function of a single clean trajectory. This is because there is significant variability in the PSD functions of feature trajectories due to factors other than SNR, such as the content of the utterance, as we have shown in Fig. 4.11. In the example studied here, the clean PSD function has lower power in high modulation frequency than the reference.

The PSD functions of the TSN-filtered trajectories are shown in Fig. 4.15(b). Note that the sum of weights of the TSN filter is not normalized for this demonstration. In addition, the first and last several frames are removed before the PSD functions are estimated, as these frames are not filtered by the TSN filter and have different temporal characteristics from other frames. From the figure, all the PSD functions are very similar to the reference function. This shows the effectiveness of the TSN filter in normalizing the PSD functions of the feature trajectories.

If the weights of the filters are normalized, the resulting PSD functions are shown in Fig. 4.15(c). From the figure, we found that the normalized PSD functions may not be close to the reference function, especially for low SNR cases, such as -5dB. This is because low SNR trajectories have high power in high modulation frequencies. Hence, the filter for low SNR trajectories are more low-pass and it filters out more power from high modulation frequency. As the original feature trajectories are MVN processed and have the same total power, the filtered low-SNR trajectories may have less total power than those of high-SNR trajectories. Despite the difference in total power, the relative weight of power in different modulation frequencies, i.e. the shape of the PSD function, are the same for both high and low SNR trajectories. This is shown in the figure that, the -5dB function is roughly a shifted version of other PSD functions.
Figure 4.16: Magnitude response of the ARMA filter with order 3.

Although the TSN filter without weight normalization provides closer normalized PSD functions to the reference function, we found that its performance in noise robust speech recognition is slightly poorer than the TSN filter with weight normalization. Therefore, the weights are always normalized in these speech recognition experiments.

4.4.2 TSN combined with ARMA

The ARMA filter [28] has been shown to be effective in improving feature robustness. In this section, we examine the effect of TSN combined with an ARMA filter. The desired magnitude response of the TSN filter and that of the ARMA filter are multiplied according to (4.4). The order of the ARMA filter is set to be 3 and the magnitude response of the ARMA filter is shown in Fig. 4.16. The ARMA filter is a low pass filter with high attenuation in the stop band. The cut-off frequency of the ARMA filter is about 9.2Hz.

The feature trajectories filtered by the combined filter of TSN and ARMA are shown in Fig. 4.17. Compare these trajectories with the original trajectories in Fig. 4.13 and the TSN-alone filtered trajectories in Fig. 4.14, we can observe that they are much smoother due to the effect of the ARMA filter and the temporal structures of the filtered trajectories are similar.
Figure 4.17: Effect of the combination of TSN and ARMA filters on feature trajectories. The original trajectories are shown in Fig. 4.13.
Figure 4.18: Effect of the combination of TSN and ARMA filters on modulation spectra of a test utterance.
The PSD functions of the filtered trajectories are shown in Fig. 4.18(b)(c). Due to the effect of the ARMA filter, the PSD functions of the filtered trajectories all have less power in high modulation frequencies than the reference. The shape of these PSD functions are similar to the magnitude response of the ARMA filter.

4.5 Speech Recognition Experiments

The TSN filter is evaluated on both the small vocabulary Aurora-2 task [69] and the large vocabulary Aurora-4 task [71]. The Aurora-2 and Aurora-4 data are generated by artificially adding recorded noises to clean speech. The noises are typical real life noises, such as subway train and babble noises. In Table 4.2-4.3, the details of these tasks and the corresponding recognizers’ settings are listed. These recognizers’ settings comply with mainstream settings in the noise robust speech recognition community. For both tasks, the acoustic models are trained with clean utterances.

Mel-frequency cepstral coefficients (MFCC) are used as the features for speech recognition. The MFCC features are extracted using the standard WI007 feature extraction program [69] delivered with the Aurora-2 task. The Fourier transform frame size is 25ms, or 200 samples at 8000Hz sampling frequency. The frame shift is 10ms, hence the frame rate is 100Hz. In total, 39 features, including the 13 static cepstral features and their delta and acceleration features, are used as raw features. If not otherwise stated, the cepstral energy feature c0 is used instead of the log energy for its good performance in MVN preprocessing [28]. The order M in (2.2) is set to 3 for generating first order differential features and 2 for generating second order differential features. The 39 raw feature trajectories are processed by the TSN framework and then used for system training and testing.

For the TSN filter, the Yule-Walker method is used to estimate the PSD functions of feature trajectories. The order of the AR model for PSD estimation is set to 6 to obtain proper level of details. The filter length of 33 taps is used for the evaluation if not otherwise stated. We will discuss the effect of the filter length later.
Table 4.2: The settings for the Aurora-2 task.

<table>
<thead>
<tr>
<th>Nature of the task</th>
<th>English connected digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling freq.</td>
<td>8000Hz</td>
</tr>
<tr>
<td>Avg. utterance length</td>
<td>≈ 1.8 seconds</td>
</tr>
<tr>
<td>Training data</td>
<td>8440 clean utterances</td>
</tr>
<tr>
<td></td>
<td>55 male speakers + 55 female speakers</td>
</tr>
<tr>
<td>Test data</td>
<td>3 test sets (A,B,C) / 10 noises or environments</td>
</tr>
<tr>
<td></td>
<td>52 male speakers + 52 female speakers</td>
</tr>
<tr>
<td>Set A</td>
<td>4 noises: subway, babble, car, exhibition</td>
</tr>
<tr>
<td>Set B</td>
<td>4 noises: restaurant, street, airport, station</td>
</tr>
<tr>
<td>Set C</td>
<td>2 noises + channel distortion: subway, street</td>
</tr>
<tr>
<td>For each environment</td>
<td>7 SNR levels: clean + 20dB to -5dB (5dB step)</td>
</tr>
<tr>
<td>For each SNR level</td>
<td>1001 test utterances</td>
</tr>
<tr>
<td>In total</td>
<td>70070 utterances (10×7×1001) containing 230181 words</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>11 words: “zero” to “nine” + “oh” (also represents 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Recognizer</th>
<th>HMM Toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acoustic model</td>
<td>16 states word model</td>
</tr>
<tr>
<td></td>
<td>3 Gaussian mixtures per state</td>
</tr>
<tr>
<td>Language Model</td>
<td>No language model</td>
</tr>
<tr>
<td>Test parameters</td>
<td>0 word insertion penalty</td>
</tr>
<tr>
<td></td>
<td>250 pruning threshold</td>
</tr>
</tbody>
</table>

4.5.1 Evaluation on the Aurora-2 Task

In the Aurora-2 task, the performance of the utterance-based TSN filter is compared with that of two other popular temporal filters, the RASTA filter [27] and the ARMA filter in the MVA processing [28]. The order of the ARMA filter is set to 3 and the pole value of the RASTA filter is 0.94 as suggested in [27]. Two techniques are used as the preprocessing units of the temporal filters: the MVN and the HEQ. The details of the performance of these preprocessing alone are shown in Table 4.8 (The table is at the end of this chapter).

The performance of the temporal filters are shown in Table 4.4. The results are averaged over the ten test cases of Aurora-2 and 5 SNR levels from 0 to 20dB. From the table, the performance of the TSN filter is similar to that of the ARMA filter and better than that of the RASTA filter. The details of TSN performance are shown in Table4.9.

While the ARMA filter and the TSN filter produce similar improvements on the
Table 4.3: The settings for the Aurora-4 task.

<table>
<thead>
<tr>
<th>Nature of the task</th>
<th>Large vocabulary continuous English speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling freq.</td>
<td>8000Hz</td>
</tr>
<tr>
<td>Avg. utterance length</td>
<td>≈ 7.6 seconds</td>
</tr>
<tr>
<td>Training data</td>
<td>7138 clean utterances</td>
</tr>
<tr>
<td></td>
<td>83 speakers and gender-balanced</td>
</tr>
<tr>
<td>Test data</td>
<td>14 test cases or environments</td>
</tr>
<tr>
<td></td>
<td>8 speakers and gender-balanced</td>
</tr>
<tr>
<td>case 1</td>
<td>clean test</td>
</tr>
<tr>
<td>case 2-7</td>
<td>6 noises: car, babble, restaurant, street, airport, train</td>
</tr>
<tr>
<td>case 8</td>
<td>Microphone mismatch</td>
</tr>
<tr>
<td>case 9-14</td>
<td>case 2-7 + microphone mismatch</td>
</tr>
<tr>
<td>For each case</td>
<td>166 utterances</td>
</tr>
<tr>
<td>In total</td>
<td>2324 utterances (166×14) containing 38010 words</td>
</tr>
<tr>
<td>Vocabulary size</td>
<td>4989 words, no out of vocabulary words</td>
</tr>
<tr>
<td>Recognizer</td>
<td>HMM Toolkit</td>
</tr>
<tr>
<td>Acoustic model</td>
<td>3 states triphone model</td>
</tr>
<tr>
<td></td>
<td>3150 tied states, 4 Gaussian mixtures per state</td>
</tr>
<tr>
<td>Language Model</td>
<td>Bigram model</td>
</tr>
<tr>
<td>Test parameters</td>
<td>-10 word insertion penalty</td>
</tr>
<tr>
<td></td>
<td>16 language model weight</td>
</tr>
<tr>
<td></td>
<td>250 pruning threshold</td>
</tr>
</tbody>
</table>
Table 4.4: Overall recognition accuracies (%) of TSN on the Aurora-2 task. The preprocessing is followed by the temporal filters. None denotes no temporal filtering.

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>Temporal Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
</tr>
<tr>
<td>MVN</td>
<td>78.49</td>
</tr>
<tr>
<td>HEQ</td>
<td>80.45</td>
</tr>
</tbody>
</table>

Aurora-2 task (see Table 4.4), they achieve this differently. The TSN filter adapts to changing environments. On average, the TSN filter applies more aggressive smoothing on noisier features. The ARMA filter on the other hand smooths all the feature trajectories identically. We now investigate if the TSN filter complements the fixed ARMA filter to improve the performance.

The effect of the ARMA filter is integrated into the TSN filter by multiplying the magnitude response of the two filters as shown in (4.4). With the combined magnitude response, the TSN filter not only normalizes the feature’s temporal structure, but also applies additional smoothing to the features. The performance of the TSN filter integrated with the effect of the ARMA filter is shown in Table 4.5. The ARMA filters with four different orders (1-4) are used. From the table, it is found that the integrated case TSN+ARMA is consistently better than both the ARMA filter and the TSN filter alone. This shows that the two filters are complementary for the Aurora-2 task. The detailed comparison of ARMA and TSN+ARMA is provided in Table 4.10 for order=3 case.

To ensure that the results of these tests are statistically significant, we carry out a hypothesis test as follows. The objective of the test is to decide whether two accuracies $p_1$ and $p_2$ are significantly different. We set the null hypothesis $H_0 : p_1 = p_2$ and the alternative hypothesis as $H_1 : p_1 \neq p_2$. If $H_0$ is accepted, the difference between $p_1$ and $p_2$ are not statistically significant and vice versa. The test statistic for the hypothesis test is shown as follows [147]:

$$z = \frac{\sqrt{N}(p_1 - p_2)}{\sqrt{p_1(1 - p_1)} + \sqrt{p_2(1 - p_2)}} \tag{4.9}$$

where $N$ is the number of words in the test and $N = 230181$ for Aurora-2 task (see Table 4.2). If the values of $p_1$ and $p_2$ are both around 85%, the difference between them must be larger than about 0.39% to reject $H_0$ with a confidence level of 99%,
Table 4.5: Recognition accuracies (%) of TSN filter integrated with the ARMA filter on the Aurora-2 task. MVN is used as the pre-processing. TSN+AMRA denotes the integrated case.

<table>
<thead>
<tr>
<th>ARMA Order</th>
<th>TSN</th>
<th>ARMA</th>
<th>TSN+ARMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84.44</td>
<td>82.23</td>
<td>85.15</td>
</tr>
<tr>
<td>2</td>
<td>84.44</td>
<td>83.65</td>
<td>85.67</td>
</tr>
<tr>
<td>3</td>
<td>84.44</td>
<td>84.59</td>
<td>86.41</td>
</tr>
<tr>
<td>4</td>
<td>84.44</td>
<td>84.09</td>
<td>86.01</td>
</tr>
</tbody>
</table>

i.e. the difference of any two accuracies in Table 4.4-4.5 must be larger than 0.39% to be statistically significant. From Table 4.5, the results of TSN+ARMA are always significantly better than that of ARMA and TSN alone.

4.5.2 Evaluation on the Aurora-4 Task

The TSN filter along with the RASTA and ARMA filters are also evaluated on the large vocabulary Aurora-4 task. Here, the best order of the ARMA filter is found to be one. The pole of the RASTA filter is 0.94.

The summary of performance of the temporal filters is shown in Table 4.6 (Details shown in Table 4.11 at the end of this chapter). The results are averaged over the 14 test cases of the task. From the table, all the temporal filters improve the performance over the baseline systems. The TSN filter yields better results as compared to both the RASTA and the ARMA filters. It is also observed that the results with HEQ preprocessing are consistently better than the results with the MVN preprocessing on the Aurora-4 task while it’s not so on the Aurora-2 task. This may be due to the fact that Aurora-4 has longer utterances so that the histograms of the features are better estimated for HEQ.

The statistical significance of the results in Table 4.6 is evaluated. The accuracies in Table 4.6 are in the range of 60%-70%, and the number of test words is 38010. By using the statistic in (4.9), the difference between two accuracies must be at least 0.96% to be significant with a confidence level of 95%. From Table 4.5, the results of TSN are always significantly better than that of ARMA and RASTA.

Comparison of the Aurora-2 and Aurora-4 tasks gives us some insights on the filtering of speech features for robust speech recognition. While feature smoothing can reduce the feature mismatches, it also leads to removal of useful speech information. The degree
Table 4.6: Overall recognition accuracies (%) of TSN on the Aurora-4 task. The preprocessing is followed by the temporal filters. None denotes no temporal filtering.

<table>
<thead>
<tr>
<th>Preprocessing</th>
<th>Temporal Filter</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
<td>ARMA</td>
</tr>
<tr>
<td>MVN</td>
<td>60.75</td>
<td>61.59</td>
</tr>
<tr>
<td>HEQ</td>
<td>64.84</td>
<td>65.17</td>
</tr>
</tbody>
</table>

of smoothing is a trade-off between these two factors and usually depends on the signal conditions such as the SNR level and the task complexity. Our experimental results show that the small vocabulary Aurora-2 task can be aggressively smoothed while the large vocabulary Aurora-4 task can only be mildly smoothed. This is why the ARMA filter uses a small order of one in the Aurora-4 task, i.e. less aggressive smoothing, while its order is three in the Aurora-2 task. Similarly, while the combination of the ARMA and TSN filters improves the recognition performance further by introducing additional smoothing to the TSN filter on the Aurora-2 task, such combination degrades the performance on the Aurora-4 task.

4.5.3 Evaluation on Different Features

The performance of the TSN filter may vary with different features. In this section, we evaluate the TSN filter on two other features, i.e. the MFCC features with log energy (MFCC+logE) and the PLP feature [60] on the Aurora-2 task. The log energy is computed by the WI007 feature extraction program [69] and the PLP features are generated using the Matlab implementation of PLP [148]. For the PLP, the 13 PLP static cepstral features and their delta and acceleration are used. The preprocessing used is MVN.

The performance of the TSN filter for different features on the Aurora-2 task is shown in Table 4.7. For comparison, the results of the MFCC features with c0 (MFCC+c0) is also shown. The ARMA filter here uses an order of three and the RASTA filter’s pole is 0.94. From the table, the improvements of the temporal filters for the MFCC+logE and PLP show similar range as that of the MFCC+c0. The TSN filter alone may not produce the best performance. However, the integration of the TSN with the ARMA filter (TSN+ARMA) always improves the performance.
Table 4.7: Recognition accuracies (%) of applying the TSN filter to different features on the Aurora-2 task. The MVN is used as the preprocessing unit for all results.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Temporal Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCC+c0</td>
<td>None ARMA RASTA TSN TSN+ARMA</td>
</tr>
<tr>
<td>MFCC+logE</td>
<td>78.49 84.59 81.84 84.44 86.41</td>
</tr>
<tr>
<td>PLP</td>
<td>78.44 84.07 81.24 83.51 85.10</td>
</tr>
</tbody>
</table>

Figure 4.19: The performance of segment-based TSN with different segment length on the Aurora-4 task. The features are the MFCC with c0 energy plus their delta and acceleration features.

4.5.4 Segment-based Implementation

The proposed segment-based implementation of the TSN filter (TSNseg) is evaluated on the Aurora-4 task as the utterances in Aurora-4 are long enough to test the segment-based implementation. For the TSNseg, the silence of the utterance should not be longer than the segment length, otherwise the preprocessing MVN/HEQ and the TSNseg will normalize the statistics of the silence to the reference statistics. Hence, voice activity detection (VAD) is used to find the speech segments and extra silence frames are removed
in the beginning and ending of each utterance before training and testing. Only 25 silence frames, or 250ms, are kept. The VAD is performed by decoding the training and testing data using the decoder of the HMM Toolkit (HTK), i.e. the HVite tool [67].

The performance of the TSNseg with MVN preprocessing and various segment length is shown in Fig. 4.19. From the experiments, we find that it is a good tradeoff to set the segment shift to be half of the segment length. For comparison, the utterance-based MVN preprocessing (MVN), the utterance-based TSN filter (TSN), and the segment-based MVN preprocessing (MVNseg) are also evaluated. All the configurations operate on the speech after VAD. From the figure, we have two observations. One is that the best segment length for the TSNseg is about 2.2s. The other is that although the TSNseg produces better results than the TSN, its improvement over the MVNseg preprocessing (about 2%) is similar to the improvement of the TSN over the MVN preprocessing (1.7%). This suggests that the gain of TSNseg over TSN is mainly due to its use of MVNseg preprocessing.

The main advantage of the TSNseg is the reduction of the processing delay. The average processing delay of the TSNseg is half of the segment length. If the segment length is chosen to be 2.2 seconds, the average delay will be 1.1 seconds, which is much lower than the utterance length of the Aurora-4 that can be more than 10 seconds. In addition, with the segment-based implementation, the TSN filter can adapt much faster to run-time changing environments, thus suitable for low-delay speech recognition deployments.

4.5.5 Effect of Filter Length

In the TSN filter design process, the weights used for filtering are only the central part of the available weights (step 5 of Table 4.1). The performance of the TSN filter is affected by the number of weights used, or the filter length. Since the TSN filter is implemented by FIR filter, the flexibility of the realized magnitude response of the TSN filter is proportional to the length of the filter. A long filter can model the desired magnitude response better, but also means higher computational cost and longer transition period for filtering. With a long transition period, many frames in the beginning and ending of each utterance cannot be filtered and this causes inconsistency in the temporal characteristics
Figure 4.20: The performance of two TSN configurations with different filter length. The features are the MFCC with c0 energy plus their delta and acceleration features, the preprocessing is MVN and the task is Aurora-2.

within a feature trajectory. Due to these considerations, a short filter is preferred if it does not degrade the performance too much compared to a long filter.

In Fig. 4.20, the performance of the utterance-based TSN filter on the Aurora-2 task with MFCC features and different filter lengths is shown. Both the basic TSN filter (TSN alone) and the combination of the TSN filter with the ARMA filter (TSN+ARMA) are studied. From the figure, we found that although the optimal filter lengths for different configurations are different, the filter length of about 33 taps is a good choice. With 33 taps, or 330ms (the shift between two neighbouring frames is 10ms), the TSN filter can exploit temporal information on the scale of common syllables.

4.5.6 Contribution of Modulation Frequency Bands

It is generally believed that the human hearing system mainly uses information of the 1-16Hz modulation frequency range to differentiate speech sounds [117–120,122]. In
Figure 4.21: Illustration of modifying the TSN filter’s magnitude response. The magnitude response of the TSN filter for a feature trajectory of an utterance (car noise, SNR=5dB) is shown. The edge frequency here is 20Hz. Original response denotes the TSN filter’s magnitude response before the modification. The gain of the modified response is similar to that of the original response in the 1-20Hz range, but approximately one in the 20-50Hz range. Note the imperfect transition of the modified response around 20Hz due to the short FIR filter used.

[121], the 1-16Hz modulation frequency range is also found to be important for ASR. Such concurrence suggests that the source of speech intelligibility is mainly from the 1-16Hz modulation frequency range. If this is true, the performance improvement of the TSN filter should mainly come from its magnitude response in the 1-16Hz range, as the other modulation frequencies are not important for speech intelligibility anyway. To verify whether this hypothesis is true, we modify the magnitude response of the TSN filter to disable its gain in the high modulation frequency range. Specifically, we set the filter’s gain to ONE for the frequencies higher than a specified frequency (we call it edge frequency). Fig. 4.21 shows an example of modifying the TSN filter’s magnitude response with edge frequency=20Hz. By studying the TSN’s performance with different edge frequencies, we are able to appreciate which part of the magnitude response contributes
Chapter 4. Normalizing the Temporal Structure of Feature Trajectories

Figure 4.22: The performance of the TSN filter with different filter edge frequency. The preprocessing is MVN.

the most to the performance improvement.

In Fig. 4.22, the performance of the utterance-based TSN filter with different edge frequencies is shown. When the edge frequency is 50Hz, there is no modification to the gain of the TSN filter. From 50Hz downwards, the gain of the filter is set to one in more and more modulation frequencies and hence the performance of the filter decreases almost monotonically. When the edge frequency is 20Hz, the filter is disabled in the 20-50Hz range and the performance only drops about 1% from 84.4% to 83.4%. However, when the filter is also disabled in the 5-20Hz range, the accuracy drops significantly to below 79%. This observation suggests that the low modulation frequency range (<20Hz) of the magnitude response contributes much more to the improvement of the TSN filter than the high modulation frequency range (>20Hz).
4.6 Summary

In this chapter, we studied the temporal structure normalization (TSN) filter for speech recognition in noisy environments. The TSN filter is motivated by the observations that most noise changes the speech modulation spectra. Hence, the TSN filter aims to normalize the temporal structure of feature trajectories (represented by modulation spectra) to reduce the mismatch between clean and noisy features. It is expected that when the noise modulation spectra are significantly different from that of clean speech, the TSN filter will help to correct the temporal structure of the noisy speech features. As the information of the modulation spectra of the test utterances are taken into account during the filter design process, the TSN filter is able to adapt to changing environments. Generally, the TSN filter either performs comparably with or outperforms other state-of-the-art temporal filters on the two popular benchmark tasks: the Aurora-2 and Aurora-4.

The TSN filter is also a normalizing technique. From statistical point of view, the TSN normalizes the correlation of features over different frames, while other normalization technique such as HEQ normalizes the probability distributions of the features. Therefore, TSN filter can be seen as presenting a new way to address the feature-model mismatch problem.

The TSN filter can be easily integrated into common speech recognition systems. The reference modulation spectra are estimated using the same training data that we use for acoustic modelling. The feature normalization takes place right after the feature extraction before acoustic training/decoding processes. It would be an interesting future work to study the interaction between feature normalization and acoustic modelling in speech recognition applications.
Table 4.8: Details of baseline performance on the Aurora-2 task.

<table>
<thead>
<tr>
<th>SNR</th>
<th>A</th>
<th>With MVN Processing</th>
<th>Avg.</th>
<th>B</th>
<th>C</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>20dB</td>
<td>96.50</td>
<td>98.04</td>
<td>97.82</td>
<td>97.07</td>
<td>97.36</td>
<td>97.88</td>
</tr>
<tr>
<td>15dB</td>
<td>93.43</td>
<td>95.95</td>
<td>95.50</td>
<td>93.61</td>
<td>94.62</td>
<td>96.35</td>
</tr>
<tr>
<td>10dB</td>
<td>86.80</td>
<td>89.72</td>
<td>89.14</td>
<td>85.50</td>
<td>87.79</td>
<td>90.14</td>
</tr>
<tr>
<td>5dB</td>
<td>70.49</td>
<td>71.43</td>
<td>70.98</td>
<td>69.21</td>
<td>70.53</td>
<td>74.06</td>
</tr>
<tr>
<td>0dB</td>
<td>40.41</td>
<td>37.18</td>
<td>38.35</td>
<td>41.01</td>
<td>39.24</td>
<td>44.83</td>
</tr>
<tr>
<td>-5dB</td>
<td>15.66</td>
<td>14.12</td>
<td>15.60</td>
<td>18.88</td>
<td>16.07</td>
<td>17.81</td>
</tr>
<tr>
<td>Avg.</td>
<td>77.53</td>
<td>78.46</td>
<td>78.36</td>
<td>77.28</td>
<td>77.91</td>
<td>80.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>A</th>
<th>With HEQ Processing</th>
<th>Avg.</th>
<th>B</th>
<th>C</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clean</td>
<td>99.11</td>
<td>99.06</td>
<td>99.05</td>
<td>99.11</td>
<td>99.08</td>
<td>99.11</td>
</tr>
<tr>
<td>20dB</td>
<td>97.02</td>
<td>98.16</td>
<td>98.42</td>
<td>96.95</td>
<td>97.64</td>
<td>98.50</td>
</tr>
<tr>
<td>15dB</td>
<td>94.17</td>
<td>96.10</td>
<td>96.72</td>
<td>93.74</td>
<td>95.18</td>
<td>96.62</td>
</tr>
<tr>
<td>10dB</td>
<td>87.60</td>
<td>90.63</td>
<td>91.47</td>
<td>85.87</td>
<td>88.89</td>
<td>91.89</td>
</tr>
<tr>
<td>5dB</td>
<td>71.97</td>
<td>75.15</td>
<td>75.96</td>
<td>70.66</td>
<td>73.44</td>
<td>76.48</td>
</tr>
<tr>
<td>0dB</td>
<td>43.81</td>
<td>41.23</td>
<td>43.33</td>
<td>43.75</td>
<td>43.03</td>
<td>47.84</td>
</tr>
<tr>
<td>-5dB</td>
<td>17.75</td>
<td>15.30</td>
<td>15.06</td>
<td>18.11</td>
<td>16.56</td>
<td>18.88</td>
</tr>
<tr>
<td>Avg.</td>
<td>78.91</td>
<td>80.25</td>
<td>81.18</td>
<td>78.19</td>
<td>79.64</td>
<td>82.27</td>
</tr>
</tbody>
</table>
Table 4.9: Details of TSN performance on the Aurora-2 task.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MVN Processing</td>
<td>followed by TSN</td>
<td>filtering</td>
<td></td>
</tr>
<tr>
<td>Clean</td>
<td>99.20</td>
<td>98.91</td>
<td>98.87</td>
<td>99.20</td>
</tr>
<tr>
<td>20dB</td>
<td>96.99</td>
<td>97.82</td>
<td>98.03</td>
<td>97.22</td>
</tr>
<tr>
<td>15dB</td>
<td>94.44</td>
<td>96.37</td>
<td>96.69</td>
<td>94.91</td>
</tr>
<tr>
<td>10dB</td>
<td>88.64</td>
<td>91.87</td>
<td>93.17</td>
<td>88.89</td>
</tr>
<tr>
<td>5dB</td>
<td>77.31</td>
<td>81.02</td>
<td>83.69</td>
<td>77.69</td>
</tr>
<tr>
<td>0dB</td>
<td>54.41</td>
<td>54.38</td>
<td>62.45</td>
<td>55.66</td>
</tr>
<tr>
<td>-5dB</td>
<td>24.69</td>
<td>22.16</td>
<td>31.85</td>
<td>29.81</td>
</tr>
<tr>
<td>Avg.</td>
<td>82.36</td>
<td>84.29</td>
<td>86.81</td>
<td>82.87</td>
</tr>
</tbody>
</table>

|       | HEQ processing     | followed by TSN    | filtering          |      |
| Clean | 99.11              | 98.88              | 99.02              | 99.03   | 99.11 | 98.88    | 99.02  | 99.11   | 99.03   | 99.08 | 99.00    | 99.04    | 99.03 |
| 20dB  | 97.27              | 97.91              | 98.69              | 97.22   | 97.77 | 98.25    | 97.49  | 98.54   | 98.30   | 98.15 | 97.33    | 97.61    | 97.47 | 97.86  |
| 15dB  | 94.60              | 96.22              | 97.49              | 95.46   | 95.92 | 96.93    | 96.37  | 97.26   | 96.58   | 96.79 | 94.90    | 96.61    | 95.76 | 96.24  |
| 10dB  | 89.71              | 93.11              | 94.01              | 89.66   | 91.62 | 93.03    | 92.35  | 94.51   | 93.71   | 93.40 | 89.87    | 92.41    | 91.14 | 92.24  |
| 5dB   | 78.91              | 82.01              | 85.96              | 78.28   | 81.07 | 83.17    | 82.53  | 85.36   | 84.88   | 83.99 | 78.78    | 81.65    | 80.22 | 82.06  |
| 0dB   | 54.44              | 54.20              | 60.93              | 55.01   | 56.15 | 58.43    | 57.89  | 62.24   | 60.69   | 59.81 | 53.91    | 58.01    | 55.96 | 57.58  |
| Avg.  | 82.99              | 84.69              | 87.24              | 83.13   | 84.51 | 85.96    | 85.33  | 87.58   | 86.83   | 86.43 | 82.96    | 85.26    | 84.11 | 85.20  |
Table 4.10: Details of TSN and ARMA filter performance on the Aurora-2 task. In the first half of the table, the performance of ARMA filter with order=3 is shown. In the second half of the table, the performance of TSN filter combined with ARMA filter (order=3), i.e. TSN+ARMA, is shown.

<table>
<thead>
<tr>
<th>SNR</th>
<th>MVN Processing followed by ARMA filtering</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>20dB</td>
<td>97.70</td>
<td>97.88</td>
</tr>
<tr>
<td>15dB</td>
<td>95.21</td>
<td>96.16</td>
</tr>
<tr>
<td>10dB</td>
<td>89.59</td>
<td>91.23</td>
</tr>
<tr>
<td>5dB</td>
<td>79.83</td>
<td>78.72</td>
</tr>
<tr>
<td>0dB</td>
<td>58.24</td>
<td>50.36</td>
</tr>
<tr>
<td>-5dB</td>
<td>26.47</td>
<td>19.92</td>
</tr>
<tr>
<td>Avg.</td>
<td>84.11</td>
<td>82.87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>MVN Processing followed by TSN+ARMA filtering</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>20dB</td>
<td>97.11</td>
<td>98.04</td>
</tr>
<tr>
<td>15dB</td>
<td>95.61</td>
<td>96.70</td>
</tr>
<tr>
<td>10dB</td>
<td>90.05</td>
<td>92.71</td>
</tr>
<tr>
<td>5dB</td>
<td>81.39</td>
<td>83.07</td>
</tr>
<tr>
<td>0dB</td>
<td>60.88</td>
<td>57.89</td>
</tr>
<tr>
<td>-5dB</td>
<td>31.99</td>
<td>25.88</td>
</tr>
<tr>
<td>Avg.</td>
<td>85.01</td>
<td>85.68</td>
</tr>
</tbody>
</table>
### Table 4.11: Details of performance on the Aurora-4 task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Without Microphone Mismatch</th>
<th>Noisy</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>Car</td>
<td>Babble</td>
</tr>
<tr>
<td>MVN</td>
<td>86.41</td>
<td>74.70</td>
<td>61.69</td>
</tr>
<tr>
<td>MVN+ARMA</td>
<td>84.86</td>
<td>72.89</td>
<td>62.32</td>
</tr>
<tr>
<td>MVN+RASTA</td>
<td>84.24</td>
<td>75.84</td>
<td>63.20</td>
</tr>
<tr>
<td>MVN+TSN</td>
<td>86.45</td>
<td>75.36</td>
<td>65.08</td>
</tr>
<tr>
<td>HEQ</td>
<td>87.18</td>
<td>78.71</td>
<td>65.93</td>
</tr>
<tr>
<td>HEQ+ARMA</td>
<td>85.50</td>
<td>79.05</td>
<td>65.90</td>
</tr>
<tr>
<td>HEQ+RASTA</td>
<td>85.01</td>
<td>78.64</td>
<td>67.29</td>
</tr>
<tr>
<td>HEQ+TSN</td>
<td>87.77</td>
<td>78.71</td>
<td>69.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>With Microphone Mismatch</th>
<th>Noisy</th>
<th>Total Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clean</td>
<td>Car</td>
<td>Babble</td>
</tr>
<tr>
<td>MVN</td>
<td>77.68</td>
<td>64.83</td>
<td>50.98</td>
</tr>
<tr>
<td>MVN+ARMA</td>
<td>77.50</td>
<td>66.96</td>
<td>54.62</td>
</tr>
<tr>
<td>MVN+RASTA</td>
<td>77.20</td>
<td>66.74</td>
<td>54.11</td>
</tr>
<tr>
<td>MVN+TSN</td>
<td>77.57</td>
<td>66.67</td>
<td>56.24</td>
</tr>
<tr>
<td>HEQ</td>
<td>80.26</td>
<td>69.47</td>
<td>56.57</td>
</tr>
<tr>
<td>HEQ+ARMA</td>
<td>77.76</td>
<td>71.21</td>
<td>58.32</td>
</tr>
<tr>
<td>HEQ+RASTA</td>
<td>77.86</td>
<td>70.31</td>
<td>58.82</td>
</tr>
<tr>
<td>HEQ+TSN</td>
<td>80.77</td>
<td>71.45</td>
<td>60.22</td>
</tr>
</tbody>
</table>
Chapter 5

Improving Acoustic Model
Generalization Capability

Traditional techniques improve the robustness of speech recognition by reducing the mismatch between training and testing conditions directly, which is in turn achieved by compensating/normalizing features or/and adapting the acoustic model. Although reducing mismatch is a very natural and effective way of dealing with the robustness problem, it may not be the only way. In this chapter, we study a new and complementary approach to improve the robustness, i.e. by improving the generalization capability of the acoustic model.

5.1 Introduction

Generalization capability of a model is a good indicator of how well the model will perform on unseen test data. From statistical learning theory [44], a big margin usually result in better generalization capability, where the margin refers to some measure of separation between competing classes. The rationale is that when the models of different classes are more separated in the feature space (i.e. with big margin), there will be big buffer zone between competing classes and the models will be able to tolerate some degree of mismatch between training and testing data (i.e. good generalization capability). In this chapter, we explore the generalization capability of acoustic model to improve speech recognition robustness. A major difference between generalization capability and robustness is that generalization capability refers to the model’s ability to perform well
on unseen but similar data as the training data (i.e. training and testing data follow the same distribution), while robustness refers to whether the model is able to perform well on unseen and mismatched testing data (i.e. training and testing data follow different distributions). Due to this difference, it is not guaranteed that good generalization capability will always result in better robustness.

Discriminative training (DT) methods are used to improve the generalization capability of acoustic model in this study. DT methods are used as an alternative approach of the maximum likelihood (ML) method to train the hidden Markov model (HMM) based acoustic model. Generally speaking, they estimate the model parameters to reduce the empirical error (i.e. training error). Popular DT methods include minimum classification error estimation (MCE) [47, 48], maximum mutual information estimation (MMI) [49, 50], and minimum phone/word error estimation (MPE/MWE) [51]. Recently, margin-based training methods have also been applied to train acoustic models to improve generalization more explicitly, e.g. large margin hidden Markov model (LMHMM) [149, 150], large margin estimation (LME) [73], and soft-margin estimation (SME) [45]. These DT methods have also been applied to improve speech recognition robustness and shown to be effective to different extents [46, 151–153]. In [46], model robustness is shown to be correlated to margin size, an important indicator of model’s generalization capability.

In this study, we will re-study the DT methods for improving robustness of speech recognition from the perspective of generalization capability. Specifically, we will examine the relationship between the margin of the acoustic model and the speech recognition robustness. Three DT methods will be used in our study, including two classical methods MCE and MMI, and a relatively new method SME. It is worth noting that the framework of improving model generalization for better robustness is not limited to these techniques. For example, MPE [72] and LME [73–75] (see 2.3.2.4) could also be used to improve acoustic model’s generalization capability.

As it is theoretically not possible to guarantee that good generalization capability will lead to better robustness as due to the fact that the training-testing mismatch can be arbitrarily large, the objective of this study is to examine the practical usefulness of improving generalization capability by DT methods for improving robustness. We hope that this study will provide a useful reference for similar study in the future.

The major contribution of this chapter are summarized as follows:
• The noise effects on log likelihood scores of noisy features are analyzed, both using artificial data and real speech data. The study of noise effects motivated the use of DT methods to improve generalization capability of model for better robustness. In addition, theoretical comparison of SME, MCE, and MMI from a perspective of improving margin is also provided.

• The effect DT methods on speech separation measures is studied and compared. From the study, it is clearer how the margin of acoustic model on training data is increased by DT methods, and how the improved margin translates into better separation measure on noisy test data.

• The combination of SME with other robustness techniques are studied, including MVN, TSN, and HLDA. In noisy speech recognition problem, as the training and testing are from different distributions, the assumption of statistical learning theory about the same distribution of training and testing data are violated. To ease this violation, it may be useful to reduce the mismatch in feature domain first. Our study shows that MVN is especially useful when combined with DT methods. This suggests that it is an simple and effective way to combine feature domain robustness methods with SME method to further improve the robustness of ASR.

• The SME and MVN+SME have been evaluated on more realistic Aurora-3 task. The noisy data in Aurora-2 task are artificially synthesized by adding recorded noises to clean speech signals. The data in Aurora-3 task, on the other hand, are recorded in real noisy car environments. Our experimental results show that SME and MVN+SME are also effective in such realistic noisy environments.

5.2 Noise Effect in Log Likelihood Domain

In previous chapters, we have discussed how noise distorts speech in both feature and modulation spectrum domains. In this section, we will investigate the noise effect on the log likelihood scores of speech features. Let $Y$ and $X$ be the noisy and clean features, respectively, and $\Lambda = \lambda_1, ..., \lambda_C$ be the set of $C$ acoustic models trained from clean features. Note that each acoustic model corresponds to one class in the pattern classification
problem. The log likelihoods of the noisy features evaluated on the classes of the clean-trained acoustic model, \( \log p(Y|\lambda_j) \), will deviate from that of clean features, \( \log p(X|\lambda_j) \), for \( j = 1, ..., C \). As the decision boundary of \( \Lambda \) is only suitable for clean features, the deviation when noisy features are used instead causes degradation in recognition accuracy.

As shown in the plug-in MAP decision rule of (2.4), speech recognition is based on multiplication of likelihood of speech features on speech classes \( P_A(O_j|S_j) \) and the prior probability of the speech classes \( P(S_j) \). As the language model that models \( P(S_j) \) are not affected by noise distortions, the likelihood \( P_A(O_j|S_j) \) is the key to understand noise effect in the decision process of speech recognition.

In this section, we will study the effect of noise distortions in the log likelihood domain. Our study led us to conclude that it is possible to achieve better robustness by making the acoustic model more general. The generalization of the model can be improved by moving training samples away from the decision boundary in the log likelihood domain.

### 5.2.1 A Two-Class Example

Speech recognition is a multi-class sequential pattern recognition problem. The temporal dynamics of speech and the use of HMM make the direct analysis of noise effect on log likelihoods very difficult. In this section, we will first use a two-class, single feature vector based pattern classification problem as an example to demonstrate noise effect, and then examine noise effect in real speech recognition experimentally in the following section.

Let there be two classes “A” and “B” as shown in Fig. 5.1(a). Let there be \( N \) training samples, each has two elements \( \{X_i, C_i\} \), where \( X_i \) is a feature vector of \( D \) dimensions, \( C_i \) is the correct class of \( X_i \) and \( C_i \in \{A, B\} \). A common way to build a classifier for this problem is to first estimate the probability density function (p.d.f.) of each class and then use the maximum \textit{a posteriori} (MAP) decision rule to classify the testing samples. Let \( \lambda_A \) and \( \lambda_B \) denote the parameters of the p.d.f. of class \( A \) and \( B \), respectively. The classification decision for a feature vector \( X_i \) is made as follows:

\[
\hat{C}_i = \arg \max_j p(\lambda_j|X_i), \; j \in \{A, B\} \\
= \arg \max_j p(X_i|\lambda_j)p(\lambda_j)
\]

(5.1)
Figure 5.1: Illustration of the two-class classification problem in log likelihood domain: (a) the decision boundary of model is able to correct classify training samples; (b) the deviation caused by noise distortion may move the samples across the decision boundary.
where $p(\lambda_j)$ is the \textit{a priori} probability of class $j$, i.e. our prior knowledge about class $j$, and $p(\lambda_j|X_i)$ is the \textit{a posteriori} probability of class $j$ after $X_i$ is observed. Note that we use $p(\lambda_j)$ instead of $p(j)$ as there is a one-to-one correspondence between the class labels and the models. In speech recognition, the prior knowledge about classes, such as words, are represented by language model. As we are only interested in the noise effect in acoustic modelling, it is reasonable to ignore the language model in our analysis. We assume the two classes have equal \textit{a priori} probability and (5.1) can then be rewritten as follows:

$$\hat{C}_i = \arg \max_j p(X_i|\lambda_j)$$  \hspace{1cm} (5.2)

and this is the maximum likelihood (ML) decision rule. The ML decision rule is illustrated in Fig. 5.1(a). Due to our assumptions, the decision boundary is a straight line $\log p(X_i|\lambda_A) = \log p(X_i|\lambda_B)$. The $x$-axis and $y$-axis represent the likelihoods of training samples evaluated on class $A$ and $B$ models, respectively. In the log likelihood domain, if a training sample is far from the decision boundary, the sample is well separated by the model. From Fig. 5.1(a), the distance between a sample of class $A$ to the decision boundary is

$$d(X_i|\Lambda) = \frac{\sqrt{2}}{2} [\log(p(X_i|\lambda_A)) - \log(p(X_i|\lambda_B))]$$

$$= \frac{\sqrt{2}}{2} d^{LR}(X_i|\Lambda)$$  \hspace{1cm} (5.3)

where $d^{LR}(X_i|\Lambda)$ is the log likelihood ratio (LLR) of $X_i$ on model $\Lambda = \{\lambda_A, \lambda_B\}$. If $X_i$ is from class $B$, $d^{LR}(X_i|\Lambda) = \log(p(X_i|\lambda_B)) - \log(p(X_i|\lambda_A))$. The LLR here serves as a measure of separation, i.e. how well a training sample is separated from the decision boundary by the model.

The model $\Lambda$ is like a transformation, which transforms the D-dimension feature vector into two-dimension vector whose elements are the coordinates of the sample in the log likelihood domain. Usually the transformation parameters such as Gaussian means and variances are trained to project the training data into the correct side of the decision boundary. The classification decision is made based on the Euclidian distance of samples to the decision boundary. If $d(X_i|\Lambda) > 0$, $X_i$ is correctly classified and vice versa. If test
features have similar probability distribution as that of the training features, they can also be projected correctly. However, this is not true if the test features have different probability distribution, e.g. due to noise corruption.

When a signal is corrupted by noise, the features extracted from the signal will also be distorted. If we assume the distortion is additive and independent from clean features in the feature domain, the noisy features can be represented as:

\[ Y_i = X_i + N_i \]  \hspace{1cm} (5.4)

where \( Y_i \) is the corrupted feature vector and \( N_i \) is the distortion in feature domain. The distortion \( N_i \) will modify the values of \( \log p(Y_i|\lambda_A) \) and \( \log p(Y_i|\lambda_B) \), i.e. the two coordinates of \( Y_i \) in the log likelihood domain. Therefore, \( Y_i \) will wander off \( X_i \) and the distance between them is governed by a probability distribution. This is illustrated in Fig. 5.1(b), where we show three possible deviations of noisy sample from a clean sample. Although the clean sample is on the correct side of the decision boundary, its noisy version may cross the boundary and be wrongly classified. It is reasonable to expect that the lower the signal-to-noise ratio in the signal domain, the higher the variance of \( N_i \) in the feature domain, and possibly the larger deviation of \( Y_i \) from \( X_i \) in the log likelihood domain. With larger deviation, the test samples is more likely to be projected into the wrong side of the decision boundary and wrongly classified. In the next section, we will study the distribution of the deviations of log likelihood scores using a simplified two-class problem.

### 5.2.2 Investigation of Noise Effect Using Monte Carlo Methods

The distortion in feature domain is translated into the deviation of log likelihood values. In this section, we will investigate how the log likelihood will be distorted in the two-class example. Although the example to be presented is simplified for mathematical simplicity, it still gives us some intuitive ideas of how noise may affect log likelihood scores.

We first study the distribution of the deviation for each axis. The deviation is defined as follows:

\[ \Delta(Y_i|\lambda, X_i) = \log p(Y_i|\lambda) - \log p(X_i|\lambda) \]  \hspace{1cm} (5.5)
where $\lambda$ could be either $\lambda_A$ or $\lambda_B$ in this example. In our example, $\Delta(Y_i|\lambda_A, X_i)$ is the deviation in x-axis and $\Delta(Y_i|\lambda_B, X_i)$ is the deviation in y-axis. We assume that the feature distributions of both class A and B are Gaussian: $\mathcal{N}(\mu_A, \Sigma_A)$ and $\mathcal{N}(\mu_B, \Sigma_B)$, where $\mu_A$ and $\mu_B$ are the mean vectors, and $\Sigma_A$ and $\Sigma_B$ are the covariance matrices of the distributions. Suppose there is a sample $X_i$ belonging to class A, the deviation of its noisy version, $Y_i$, in the x-axis is:

$$
\Delta(Y_i|\lambda_A, X_i) = \log\mathcal{N}(Y_i; \mu_A, \Sigma_A) - \log\mathcal{N}(X_i; \mu_A, \Sigma_A)
$$

$$
= \log \left\{ \frac{1}{(2\pi)^{D/2}|\Sigma_A|^{1/2}} \exp \left( -\frac{1}{2} (X_i + N_i - \mu_A)^T \Sigma_A^{-1} (X_i + N_i - \mu_A) \right) \right\}
$$

$$
- \log \left\{ \frac{1}{(2\pi)^{D/2}|\Sigma_A|^{1/2}} \exp \left( -\frac{1}{2} (X_i - \mu_A)^T \Sigma_A^{-1} (X_i - \mu_A) \right) \right\}
$$

$$
= -\frac{1}{2} [(X_i + N_i - \mu_A)^T \Sigma_A^{-1} (X_i + N_i - \mu_A) - (X_i - \mu_A)^T \Sigma_A^{-1} (X_i - \mu_A)]
$$

$$
= -\frac{1}{2} N_i^T \Sigma_A^{-1} (N_i + 2X_i)
$$

$$
= -\frac{1}{2} N_i^T \Sigma_A^{-1} (N_i + 2\tilde{X}_i) \tag{5.6}
$$

where $\tilde{X}_i = X_i - \mu_A$ is the mean-normalized $X_i$. Similarly, we can also find out the deviation of $Y_i$ from $X_i$ in the y-axis:

$$
\Delta(Y_i|\lambda_B, X_i) = \log\mathcal{N}(Y_i; \mu_B, \Sigma_B) - \log\mathcal{N}(X_i; \mu_B, \Sigma_B)
$$

$$
= -\frac{1}{2} N_i^T \Sigma_B^{-1} (N_i + 2(\tilde{X}_i - \mu_B))
$$

$$
= -\frac{1}{2} N_i^T \Sigma_B^{-1} (N_i + 2\tilde{X}_i) - N_i^T \Sigma_B^{-1} (\mu_A - \mu_B) \tag{5.7}
$$

Note that as we assumed the true identity of $X_i$ to be $A$, the deviation of $X_i$ in x-axis is evaluated on its correct model, $\lambda_A$, and its deviation in y-axis is evaluated on the wrong model, $\lambda_B$. Comparing (5.6) and (5.7), we can see that when wrong model is used, there is an extra source of randomness $-N_i^T \Sigma_A^{-1} (\mu_A - \mu_B)$. The mean of $-N_i^T \Sigma_A^{-1} (\mu_A - \mu_B)$ can be shown to be zero if the noise vector $N_i$ has a zero mean.

We now use the Monte Carlo method [82] to study the distributions of deviations in (5.6) and (5.7) using simple settings. Let the feature vector dimension be 5, and the elements of the feature vector be independently and identically distributed (i.i.d.). We
Figure 5.2: Distribution of deviations in the log likelihood domain. Deviation of noisy feature $Y_i$ from clean feature $X_i$ in the log likelihood domain according to (5.6) and (5.7) are used. The subfigures are: (a) noise and speech features have equal variance; (b) noise variance is ten times of speech feature variance.

assume that the feature means for class $A$ and $B$ are 1 and -1, respectively, and the feature variances are both 1. The mean of the noise term $N_i$ in (5.4) is assumed to be zero, and the variance is assumed to be 1 or 10. The distributions of deviations of noisy feature $Y_i$ are shown in Fig. 5.2. Note that $Y_i$ belongs to class $A$ here. From the figure, we have two observations. First, the distributions of deviations in the two axes have the same mean but different variances according to our assumptions. The deviation in y-axis, i.e. evaluated on wrong model, has larger variance due to the extra term $-N_i^T \Sigma_A^{-1}(\mu_A - \mu_B)$ in (5.7). Due to the random deviations of values both axes, the noisy samples may be projected into the wrong side of the decision boundary. Second, when noise variance is increased, the means of the deviations in both axes becomes more negative, indicating larger deviations. Therefore, the deviation of the noisy samples from the clean samples will be larger in the log likelihood domain. In addition, the variances of the deviations become larger. The above observations are also true for samples of class $B$.

The deviations in the two axis are now examined jointly. Three scenarios are studied, and they differ in noise variance. Three noise variances are used: 0, 1, and 10. The
Figure 5.3: Illustration of noise effect in the two-class classification problem in the log likelihood domain. Noisy feature vectors are simulated using the relationship in (5.4). The simulated noisy feature vectors are shown in the log likelihood domain. The variance of elements of feature vector $X_i$ is 1. The variances of elements of noise distortion $N_i$ are: (a) 0; (b) 1; (c) 10.
simulated samples are shown in Fig. 5.3. In Fig. 5.3(a), it is observed that the clean samples are separated by the decision boundary well. When noise is added to the features, the samples are moved in the southwest direction. This is because the deviation in both axis are more likely to be negative than to be positive as we observed in Fig. 5.2. The noise also causes some samples to be mapped into the wrong side of the decision boundary and is more serious when SNR is poorer.

The study we observed showed that when there is additive noise in the feature domain, the log likelihood of noisy features will deviate from that of clean features. The deviation may cause the noisy features to be projected into the wrong side of the decision boundary and result in poorer separation between classes. With the help of observations in this section, we will study the noise effect in log likelihood domain for real speech recognition system.

5.2.3 Empirical Study of Noise Effect on Speech Recognition

The speech recognition task is much more complex than the two-class problem discussed in the previous sections. There are several major differences between the two and they are summarized as follows:

(i) Speech patterns are represented by a sequence of feature vectors rather than a single feature vector. To model the temporal dynamics of speech, more complicated HMM is used as the model rather than Gaussian or GMM.

(ii) In speech recognition, there are many classes rather than two classes. With \( N \) classes, the features are projected into \( N \) dimensional vectors in log likelihood domain rather than two dimensional vectors. It is hard to carry on the study unless we consider only the correct and the most competing classes.

(iii) The assumption that the noise term is additive and independent from the speech is also not true in real feature extraction of speech recognition systems, such as Mel-frequency cepstral coefficients. It is well known that in the cepstral domain, the relationship between noise and speech is highly nonlinear [38, 81].

132
Due to the above differences and other unlisted factors, it is mathematically difficult to examine the noise effect for real speech recognition system. Nevertheless, the two-class example in the previous sections provided some insights in noise’s effect. In this section, we will empirically study how noise affects the log likelihood of features.

A speech utterance is consisted of a sequence of words. If the words are decomposed into more basic phoneme units, the sentence can also be represented by a sequence of phonemes. Hence, there are several levels of structure in speech utterances. In the following subsections, we will investigate the noise’s effect on log likelihood scores in two different levels, i.e. the phoneme level and the sentence level.

### 5.2.3.1 Investigation on phoneme-level log likelihood scores

In this section, we will examine the noise’s effect on phonemes’ log likelihood scores. The most important objective here is to see how noise causes deviations in log likelihood domain and increases the confusions between competing classes. To achieve the objective, we will show the “movement” of phoneme samples caused by noise distortion in the log likelihood domain.

The investigation is carried out on the Aurora-2 database [69] as it has well-labeled clean and noisy data. A good property of the Aurora-2 database is that, every clean test utterance has 6 noisy versions, from 20dB to -5dB at 5dB step, and the seven versions of the signals are exactly synchronized at the speech signal level. Hence, we can study how a phoneme is “shifted” by noise in the log likelihood domain by examining the location of its clean and noisy versions. To find the location of a phoneme sample in the log likelihood domain, we need to compute its coordinates \((x_i, y_i)\) defined as follows:

\[
x_i = \frac{1}{N_i} \log P(O_i | \lambda_A) \tag{5.8}
\]

\[
y_i = \frac{1}{N_i} \log P(O_i | \lambda_B) \tag{5.9}
\]

where \(\lambda_A\) and \(\lambda_B\) is the models for phoneme class \(A\) and \(B\), respectively, \(N_i\) is the number of frames in the phoneme. Using two competing classes a time \((A \text{ and } B)\) will enable the projection of the phonemes into a 2D plane for examination. If we consider
more competing classes, the projected space will be high dimensional and not easy for observation. Hence, in our study, we will only consider two competing classes each time.

The training of acoustic model is described now. The acoustic model used here consists of 20 full-covariance Gaussian mixtures. Each Gaussian mixture models the feature distribution of one acoustic class, including the 19 English phonemes that occur in the database and the silence model. Note that Aurora-2 is a connected digit database and only 19 English phonemes present in the database. The 20 Gaussian mixtures are trained from the 8,440 clean utterances of the Aurora-2 task. With the frame-level alignment of the utterance, we use the frames belonging to each acoustic class to train the corresponding Gaussian mixture. The alignment is obtained from force-alignment by using a monophone and HMM-based acoustic model trained from the same 8,440 utterances. The HMM models are also used to obtain the true frame-level alignment of test data. As the test utterances are synchronized, the alignment of a test clean utterance is used as the ground truth of its noisier versions. The noisy test utterances are taken from the noise case 1 of test set A, and the corrupting noise is subway train noise recorded in real world environment.

The observations of phoneme movements are shown in Fig. 5.4-5.6. Each figure shows two phoneme-pairs, and how noise distortion causes the deviation of log likelihood scores and increases the confusion between the two phonemes in each pair. To observe the movements of samples clearly, only 20 samples are shown for each class. In Fig. 5.4(a), the pair sil (silence) - ao is shown, where class A is sil and class B is ao. The decision boundary is shown as the diagonal dotted line. The x-axis represents the log likelihood score of a phoneme evaluated on class A model, i.e. \( \log P(O_i | \lambda_A) \); and y-axis represents \( \log P(O_i | \lambda_B) \). Hence, it is expected that the class A samples should be below the decision boundary and the class B samples should be above the boundary. The notation of the samples are like this: \( \bigcirc \) and \( \square \) represent clean and noisy (SNR=5dB) versions of samples of class A, + and \( \times \) represent clean and noisy (SNR=5dB) versions of samples of class B. For each sample, a dotted line is used to connect its clean and noisy versions such that we can see its deviation easily. From Fig. 5.4(a), it is observed that, in overall, both class A (sil) and class B (ao) samples are moved toward the decision boundary. When there is no noise, the two classes are separated perfectly by the decision boundary. When
Figure 5.4: Noise effect on phoneme pairs (sil-ao) and (sil-f) in the log likelihood domain. In both subfigures, the $x$ and $y$ axes represent the log likelihood scores of the phoneme samples evaluated on two competing classes $A$ and $B$’s models, respectively. The decision boundary is shown as the diagonal dotted line. $\bigcirc$ and $\blacksquare$ represent clean and noisy (SNR=5dB) versions of samples of class $A$. $+$ and $\times$ represent clean and noisy (SNR=5dB) versions of samples of class $B$. For each sample, a dotted line is used to connect its clean and noisy versions such that we can see its deviation easily. The noise used here is subway train noise.
(a) Class A (eh) vs class B (ey), from Clean to SNR5

(b) Class A (v) vs class B (f), from Clean to SNR5

Figure 5.5: Noise effect on phoneme pairs (eh-ey) and (v-f) in the log likelihood domain. Other descriptions are the same as Fig. 5.4.
Figure 5.6: Noise effect on phoneme pairs (s-th) and (uw-v) in the log likelihood domain. Other descriptions are the same as Fig. 5.4.
noise is added, some samples are moved to the wrong side of the decision boundary by noise effect. This shows reduced separation between the two classes.

In Fig. 5.4(b), the sil-f class pair is shown. It is observed that the noise effect is quite different for class sil and class f. This is different from the observation in Fig.5.3 where the noise effect is similar for the two classes as the simulation data is artificially generated. After noise distortion, all sil samples are moved across the decision boundary, while no class f samples cross the boundary. In other words, if there are just two classes, all sil samples will be recognized as f sound. This is reasonable as the phoneme f is quite noise-like. The model for f represent the noisy silence frames better than the model for sil trained from clean data.

Similar investigation is shown in Fig. 5.5(a) for two vowels, Fig. 5.5(b) and Fig. 5.6(a) for two consonant pairs, and Fig. 5.6(b) for a vowel-consonant pair. Although the noise effect is complicated and different for each cases, there is a common observation among them, i.e. the noise distortion generally moves the samples toward the decision boundary and hence reduce the separation between competing classes.

5.2.3.2 Investigation on sentence-level log likelihood scores

We now study the noise effect on sentence-level log likelihood scores. As there are usually multiple phonemes in each sentence, it is difficult to define the competing classes A and B, and hence it is difficult to use the technique for analyzing phonemes to analyze sentences. Instead of showing the samples in the log likelihood domain as in Fig. 5.4-5.6, we show the histogram of separation measures of sentence samples. The histogram provides a view of the distribution of the separation measure. Note that each sample is a log likelihood score generated from one utterance. The calculation of separation measure is now described. For each utterance, we find the state-level alignment of the utterance using the correct transcription and the acoustic model trained from clean features. We also find the most competing alignment of the utterance using the clean model. The next step is to examine the frames with confusion, i.e. the frames that have different state identities in the correct and competing alignments. The separation measure is defined as the average log likelihood ratio (LLR) of those selected frames [45]:

\[ d(O_i, A) = \frac{1}{n_i} \sum_{j \in F_i} \log \left[ \frac{P_A(O_{ij} | S_i)}{P_A(O_{ij} | \tilde{S}_i)} \right] \]  

(5.11)
Figure 5.7: Histogram of separation measures for different SNR levels. Each histogram is obtained from 10,010 separation measure instances.

where $O_i$ is the sequence of feature vectors for the $i^{th}$ utterance; $S_i$ and $\hat{S}_i$ represent the correct and the closest competing alignments of $O_i$, respectively; $F_i$ is the set of frames in $O_i$ with confusion; and $n_i$ is the number of frames in $F_i$. The $d(O_i, \Lambda)$ is the separation measure we are studying.

We study the noise effect on separation measure using the Aurora-2 task [69]. The testing data of Aurora-2 are divided into seven groups according to SNR level, including clean testing data, 20dB to -5dB testing data with 5dB step. In each SNR level, there are 10,010 utterances, each corrupted by one of 10 types of noises. For each utterance, we obtain its separation measure as described in (5.11), using acoustic model trained from clean data and the ML estimation. The histogram of separation measures for different SNR levels are compared in Fig. 5.7. The part of histogram on the left side of the vertical line at 0 are for those wrongly classified utterances. From the figure, we observe that as SNR level decreases, the histogram of separation measures shift left and becomes sharper. This shows that in overall, the distances between test samples and the decision boundary are reduced by noise distortion and some utterances are moved to the wrong side of the decision boundary. With lower SNR level, there are higher distortion in the
feature domain, and possibly larger deviation of noisy samples from clean samples in the log likelihood domain, and hence the histogram of separation measures are shifted left further.

5.2.4 Summary

Theoretically speaking, noise corruption causes noisy features to deviate from corresponding clean features randomly in the log likelihood domain. Our study on real noisy speech data showed that such random deviation, at least for the deviation resulted from the speech and noise data we used, generally brings the phoneme samples closer to the decision boundary and hence increases the confusion between competing classes. As a result, when noisy features are tested on clean trained models, or more generally, whenever there is statistical mismatch between training and testing features, the separation measure of test data will decrease and recognition performance will degrade. To improve the robustness of speech recognition systems against noise distortion, it is necessary to make the recognition process less sensitive to noise effect in the log likelihood domain.

5.3 Generalization Capability and Margin

Traditionally, the noise effect is reduced by either feature compensation methods or model adaptation methods. In feature compensation methods, if we can obtain an accurate estimate of the clean features from the observed noisy features, the deviation of log likelihood will be reduced and better classification can be performed. In model adaptation methods, the model are adapted to approximate the model trained from the noisy test features. If the adapted model can represent the noisy test features well, the projection of feature vectors to log likelihood domain will also be correct and performance will be improved. Although both feature compensation and model adaptation are very important and effective ways of reducing noise effect, we are going to study another approach for the problem. We aim at improving the generalization capability of the acoustic model, i.e. the robustness of the projection of acoustic model. In this section, we will introduce the concept of improving the generalization capability of acoustic model. In next section, we will describe the DT methods used to achieve our objective.
By generalization capability, we refer to the ability of the model to generalize well to data that are not observed during model training. When generalization capability of acoustic model is improved, the speech recognition system is more likely to perform well on mismatched test data.

Statistical learning theory [44] provides us some insights about improving pattern classification systems’ generalization capability. In this theory, the expected risk of a system is formulated as

\[ R(\Lambda) \leq R_{emp}(\Lambda) + R_{gen}(\Lambda) \]  

(5.12)

where the empirical risk \( R_{emp}(\Lambda) \) is the system’s recognition error on training data and the generalization risk \( R_{gen}(\Lambda) \) is a regularization term proportional to model complexity. Expected risk refers to the recognition error of the system on all data in the problem scope, i.e. both clean and noisy speech data in the case of noisy speech recognition. Both empirical and generalization risks are related to model complexity. For example, a more complex model is able to fit better to training data to produce lower empirical risk, however, it also leads to higher generalization risk. Minimum expected risk is obtained when a good balance between these two risks is achieved.

According to statistical learning theory, the generalization risk is bounded by a function which is proportional to model complexity. For the bound to be true, some assumptions are required, e.g. the training and testing data are generated from the same identical and independent distribution. However, in noisy speech recognition, the assumption is not true, hence the bound does not exist. Fortunately, the lack of a bound does not prevent us from reducing the generalization risk for mismatched problems. In fact, even when the assumption is true and a bound exists, the bound is usually not very useful in practical classifier design due to difficulties in evaluating the bound. Instead, the reducing of generalization risk relies on another factor, the margin of the model.

The generalization risk can be reduced if margin is increased [44] as illustrated in Fig. 5.8. The margin is the desired minimum distance between training samples and the decision boundary. During margin-based model training, e.g. SME [45], the objective of the training is to pull those training samples that fall within the margin away from the decision boundary, and those samples already far from the decision boundary are ignored and do not contribute to model parameter estimation. After training, all or most training
Figure 5.8: Increasing margin to improve the generalization capability of the model. As shown in subfigure (b), the objective is to adjust the model parameters to pull the training samples out of the class boundaries defined by the margin. As a result, a buffer zone will be created around the decision boundary, hence the model will be more robust against the deviations caused by noise as shown in subfigure (a).
samples will be outside the margin, and a “buffer zone” is formed around the decision boundary with width equal to the margin in each side. With this “buffer zone”, if a test sample deviates from the training samples of its correct class but the distance between the test sample and its nearest training sample is less than the margin, correct decision can still be made. If a larger margin is used during training, the “buffer zone” will also be wider and therefore larger mismatch is allowed. As will be shown later, other DT methods, such as MMI and MCE, can also improve the margin, but in an implicit way.

Although the margin approach and DT methods are originally applied to matched training-testing problems, it may also be effective in dealing with deviation of log likelihood values caused by noise distortion. With increased margin, the tolerance regions between competing classes are larger and distortions in the log likelihood domain may be tolerated. However, it is not guaranteed that improving margin will improve the performance of speech recognition at mismatched training-testing scenarios. This is because the training-testing mismatch may be too big such that the margin becomes ineffective. Furthermore, as speech recognition is a multi-class problem, the confusion pattern (which sound class confuses with which sound class) may change after speech is distorted by noise. Hence, improving margin of model on training data not necessarily lead to significant improvement on mismatched testing data. Our experimental study will provide a clue on how much robustness can we obtain by improving acoustic model’s generalization capability.

To make the margin-based training more useful in mismatched scenarios, it is necessary to preprocess speech features to reduce the mismatch. For example, feature normalization techniques are good choices for this purpose. In this paper, we will use a simple feature normalization technique, mean and variance normalization (MVN) [22], to preprocess speech features.

5.4 Improving the Margin Using SME, MCE, and MMI

A large margin is the key to improve the model’s generalization capability according to statistical learning theory [44]. In our experiments, we use three DT methods to
maximize the margin, namely SME [45], MCE [47, 48], and MMI [49–52]. In this section, we will provide a brief description of these techniques. For detailed implementation and discussions about these three methods, please refer to their original publications.

In SME, the parameters of the acoustic model are estimated by minimizing an approximated expected risk as follows:

\[ L_{\text{SME}}(\rho, \Lambda) = \frac{\lambda}{\rho} + R_{\text{emp}}(\rho, \Lambda) \]  \hspace{1cm} (5.13)

where \( \Lambda \) is the set of acoustic model parameters, \( \rho \) is the soft margin, \( \frac{\lambda}{\rho} \) addresses the generalization risk, and \( R_{\text{emp}}(\rho, \Lambda) \) is the empirical risk as defined in (5.14). The variable \( \lambda \) is used to control the relative weights of the two items in (5.13). With a large \( \lambda \), the training process will focus on reducing the generalization term \( \lambda/\rho \) and the margin will be large, and vice versa. To obtain good performance, it is important to obtain a good balance of these two terms.

The empirical risk is defined as the averaged risk of training utterances:

\[ R_{\text{emp}}(\rho, \Lambda) = \frac{1}{N} \sum_{i=1}^{N} l_{\text{SME}}(O_i, \rho, \Lambda) \]  \hspace{1cm} (5.14)

where \( O_i, i = 1, \ldots, N \) are the training utterances. The contribution of the utterance \( O_i \) to the total empirical risk is defined as:

\[ l_{\text{SME}}(O_i, \rho, \Lambda) = \begin{cases} 
\rho - d_{\text{SME}}(O_i, \Lambda), & \text{if } \rho > d_{\text{SME}}(O_i, \Lambda); \\
0, & \text{otherwise}. 
\end{cases} \]  \hspace{1cm} (5.15)

where \( d_{\text{SME}}(O_i, \Lambda) \) is a separation measure of \( O_i \) on acoustic model \( \Lambda \). The separation measure usually represents how well the correct model is separated from competing models regarding \( O_i \), or how far \( O_i \) is from the decision boundary. If the separation measure is not large enough, i.e. it is less than the margin, a loss is generated that equals to \( \rho - d_{\text{SME}}(O_i, \Lambda) \). In SME, the frame-normalized log likelihood ratio (LLR) defined in (5.11) is used as the separation measure \( d_{\text{SME}}(O_i, \Lambda) \).

The minimization of the objective function is solved by using generalized probabilistic descent (GPD) iteratively [46]. In order to obtain a differentiable loss function, the utterance loss function in (5.15) is embedded into a sigmoidal function as follows:

\[ l_{\text{SME}}(O_i, \rho, \Lambda) = \frac{\rho - d_{\text{SME}}(O_i, \Lambda)}{1 + \exp(-\gamma(\rho - d_{\text{SME}}(O_i, \Lambda)))} \]  \hspace{1cm} (5.16)
where $\gamma$ is used to control the transition slope of the sigmoidal function. With the smoothed loss function, the parameters of the acoustic model and the margin $\rho$ can be jointly optimized iteratively:

$$\begin{align*}
\Lambda_{t+1} &= \Lambda_t - \eta_t \nabla L_{\text{SME}}(\rho, \Lambda) |_{\Lambda = \Lambda_t} \\
\rho_{t+1} &= \rho_t - \kappa_t \nabla L_{\text{SME}}(\rho, \Lambda) |_{\rho = \rho_t}
\end{align*}$$

(5.17)

where $\eta_t$ and $\kappa_t$ are the learning step size for acoustic model parameters and margin.

In MCE, the loss function is defined as:

$$L_{\text{MCE}}(\rho, \Lambda) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{1 + e^{-\gamma d_{\text{MCE}}(O_i, \Lambda) + \theta}}$$

(5.18)

which is the average approximated classification error of training utterances. Unlike SME, there is no approximation of generalization risk in the MCE loss function. Therefore, MCE only reduces empirical risk. In this study, the following separation measure is used in MCE:

$$d_{\text{MCE}}(O_i, \Lambda) = - \log P_{\Lambda}(O_i | S_i) + \log \left[ \frac{1}{M-1} \sum_{j \neq i} P_{\Lambda}(O_i | S_j) \right]^{1/\eta}$$

(5.19)

where $M$ is the number of state-level alignments considered in the training, including the correct one. Note that $d_{\text{MCE}}(O_i, \Lambda)$ is not normalized by the number of confusing frames $n_i$ while $d_{\text{SME}}(O_i, \Lambda)$ is normalized in (5.11).

In MMI, the loss function is represented as follows:

$$L_{\text{MMI}}(O_i, \Lambda) = \frac{1}{N} \sum_{i=1}^{N} \log \frac{P_{\Lambda}(O_i | S_i)^\kappa P(S_i)^\kappa}{\sum_{S_i} P_{\Lambda}(O_i | S_i)^\kappa P(S_i)^\kappa}$$

(5.20)

$$= \frac{1}{N} \sum_{i=1}^{N} \log \frac{P_{\Lambda}(O_i | S_i)^\kappa}{\sum_{S_i} P_{\Lambda}(O_i | S_i)^\kappa}$$

(5.21)

where $\kappa$ is the scaling factor and usually $0 < \kappa < 1$. Note that we won’t use language model in our study as the tasks used are small tasks. Hence, the language model scores can be removed in (5.21). By using a scaling factor $\kappa$ that is less than one, the difference between the most dominant transcription and the less dominant transcriptions will be smaller. Usually, the smaller the $\kappa$ is, the more competing transcriptions can be taken into account.

145
5.5  Comparison of SME, MCE, and MMI

Before we present experimental results, let’s analyze the DT methods to be studied. This will make it easier to understand the merits of each method.

In most discriminative training methods, both the correct and the competing transcriptions of the training data are used for the training of the acoustic model. The correct and competing transcriptions guide the model training process such that the training data’s likelihoods on the correct models are increased and their likelihoods on the wrong models are reduced. In this way, the DT methods are able to reduce the training error more directly than the ML estimation.

Another property of DT methods is that, the training utterances do not contribute to the model training process evenly. In the training set, there are usually at least thousands of utterances. Acoustic model is usually trained to optimize some objective functions that are defined on the whole training set. Therefore, every utterance in the training set will affect the final model. However, it is not true in DT that every utterance contributes equally to the final model as they do in the ML training (assume equal length for all utterances). In fact, DT methods normally assign different weights, or importance, to different utterances. How to assign the weights to each utterances is one of the most significant differences between DT methods. For example, as will be shown later, in MCE training, those relatively well recognized training utterances will be less influential than those utterances near the decision boundary. This is because the objective of MCE is to reduce empirical error, hence the focus of the model estimation process is to correct those wrongly classified utterances. In SME, however, even if an utterance is well recognized, as far as its separation measure is below a pre-defined threshold, i.e. the margin, it will still contribute to the model estimation significantly. The different ways of assigning importance/weight to training utterances will affect the model estimation process and the ability of the DT methods to improve the generalization capability of the acoustic model. Hence, in this section, we will compare the SME with two other popular DT methods, namely the MMI and the MCE. Our comparison will focus on the weight assignment of utterances and the linkage between the weight assignment and the generalization capability of estimated models. We will now use an example adopted from [51] for demonstration. Our analysis is an extension to the analysis presented in [51].
Assume there is a two-class speech recognition problem and the two classes are $A$ and $B$. Let there be an utterance $O_i$ in the training set whose true identity is class $A$. For simplicity, we assume that there is just one word in each utterance. In continuous speech recognition, although there are multiple words or phonemes in each utterance, the problem normally can be reduced to this simplified case. Let the joint acoustic-language model probabilities of utterance $O_i$ on class $A$ and $B$ be:

$$a = P_A(O_i | A) P(A)$$

(5.22)

$$b = P_A(O_i | B) P(B)$$

(5.23)

respectively. The separation measure is defined as

$$d = \log a - \log b$$

(5.24)

which is the log likelihood ratio. Utterances with large $d$ ($d \gg 0$) are well recognized, while utterances with $d < 0$ are wrongly recognized. Ignoring the language model probability, the larger the $d$, the better an utterance is discriminated by the acoustic model. The contribution of utterance $O_i$ to the loss functions of DT methods, i.e. the loss caused by $O_i$, can be represented as a function of the separation measure $d$. By substituting $d$ into the objective functions of the DT methods, i.e. (2.6) for MMI, (2.7) for MPE, (5.18) for MCE, and (5.13) for SME, we can obtain the contribution of $O_i$ to various loss functions as listed in the following.

$$L_{\text{MMI}}(O_i, \Lambda) = - \log \left( \frac{a^\kappa}{a^\kappa + b^\kappa} \right) = - \log \left( \frac{e^{\delta \kappa}}{e^{\delta \kappa} + 1} \right)$$

(5.25)

$$L_{\text{MCE}}(O_i, \Lambda) = \frac{b^\gamma e^{-\theta}}{a^\gamma + b^\gamma e^{-\theta}} = \frac{e^{-\theta}}{e^{d \gamma} + e^{-\theta}}$$

(5.26)

$$L_{\text{MPE}}(O_i, \Lambda) = 1 - \frac{a^\kappa}{a^\kappa + b^\kappa} = \frac{b^\kappa}{a^\kappa + b^\kappa} = \frac{1}{e^{d \kappa} + 1}$$

(5.27)

$$L_{\text{SME}}(O_i, \Lambda) = \frac{b^\gamma e^{\rho \gamma}}{a^\gamma + b^\gamma e^{\rho \gamma}} (\rho - \log a + \log b) = \frac{\rho - d}{1 + e^{-\gamma (\rho - d)}}$$

(5.28)

where $\epsilon$ is the natural number, $\kappa$, $\gamma$, $\theta$, and $\rho$ are the parameters of DT methods. Note that for MMI and MPE, the loss functions are the reverse of the objective functions. From the list, the contribution to MPE is the same as the contribution to MCE if $\theta = 0$ and $\gamma = \kappa$ [51]. The contributions (except for MPE) are plotted against $d$ in Fig. 5.9.

147
Figure 5.9: Contribution of a single training utterance $O_i$ to loss functions of DT methods. (the x-axis is the separation measure $d$). The parameters of each DT method are just for illustration purpose and may not be the common values used in practice. From the figure, we have following observations:

- The contribution of $O_i$ to MCE loss function is zero at $d = \infty$ and one at $d = -\infty$. At $d = -\frac{\theta}{\gamma}$, the contribution is 0.5.

- The contribution of $O_i$ to MMI loss function is also zero at $d = \infty$, but approaches $-dk$ as $d$ approach $-\infty$.

- The contribution of $O_i$ to SME loss function is also approaching zero when $d > \rho$ (in Fig. 5.9, $\rho = 10$), and approaches $\rho - d$ as $d$ becomes less than the margin. The margin in the figure is 10. Note that there is imperfect transition in the SME curve due to the sigmoid function.

As the loss functions of DT methods can be minimized using gradient-based methods such as Generalized Probabilistic Descent (GPD) [154], it is more useful to compare the
contribution of an utterance to the gradients of the loss functions of the DT methods. The contribution to gradient can be obtained by taking the partial differentiation of the contribution to loss functions w.r.t. the separation measure \( d \). The partial differentiation of MMI loss function w.r.t. acoustic model parameters is:

\[
\frac{\partial L_{\text{MMI}}(O_i, \Lambda)}{\partial \Lambda} = \frac{\partial}{\partial \Lambda} \left[ -\log \left( \frac{e^{d\kappa}}{e^{d\kappa} + 1} \right) \right] = \left( \frac{\kappa}{e^{d\kappa} + 1} - \frac{1}{e^{d\kappa} + 1} \right) \frac{\partial d}{\partial \Lambda} = W_{\text{MMI}}(O_i, \Lambda) \frac{\partial d}{\partial \Lambda}
\]

where

\[
W_{\text{MMI}}(O_i, \Lambda) = \frac{-\kappa}{e^{d\kappa} + 1}
\]

\[
\frac{\partial d}{\partial \Lambda} = \frac{\partial(\log a - \log b)}{\partial \Lambda} = \frac{1}{a} \frac{\partial a}{\partial \Lambda_a} - \frac{1}{b} \frac{\partial b}{\partial \Lambda_b}
\]

From the above equations, the gradient of the MMI loss function w.r.t. the model parameters can be seen as the product of the weight \( W_{\text{MMI}} \) and the differentiation of separation \( d \) w.r.t. model parameters. As the weight is a function of \( d \), we will expect utterances with different \( d \) to contribute to the gradient with different weights. From (5.31), there are two items in the gradient of separation. The item \( \frac{1}{a} \frac{\partial a}{\partial \Lambda_a} \) is similar to the gradient used in ML estimation and will make the likelihood of the training utterance \( O_i \) larger on the correct model \( A \). On the other hand, the item \( -\frac{1}{b} \frac{\partial b}{\partial \Lambda_b} \) will make the likelihood on wrong model \( B \) smaller. As we will show later, if LLR is used as the separation measure, the contribution of an utterance to the gradient of the loss functions of most DT methods can be represented as the product of weights and the differentiation of separation \( d \). We will focus on comparing the weights of different DT methods.

Similarly to MMI, the partial differentiation of MCE loss function w.r.t. the model parameters is derived as follows:

\[
\frac{\partial L_{\text{MCE}}(O_i, \Lambda)}{\partial \Lambda} = \frac{\partial}{\partial \Lambda} \left[ -\frac{e^{-\theta}}{e^{d\gamma} + e^{-\theta}} \right] = \frac{-e^{-\theta}}{(e^{d\gamma} + e^{-\theta})^2} \frac{\partial (e^{d\gamma} + e^{-\theta})}{\partial \Lambda} = \frac{-e^{-\theta + d\gamma}}{(e^{d\gamma} + e^{-\theta})^2} \frac{\partial d}{\partial \Lambda}
\]
and the partial differentiation of SME loss function w.r.t. the model parameters is:

\[
\frac{\partial L_{\text{SME}}(O_i, \Lambda)}{\partial \Lambda} = \frac{\partial}{\partial \Lambda} \left[ \frac{(\rho - d)}{1 + e^{-\gamma(\rho - d)}} \right] = \frac{-1}{1 + e^{-\gamma(\rho - d)}} - \frac{\rho - d}{(1 + e^{-\gamma(\rho - d)})^2} \frac{\partial (1 + e^{-\gamma(\rho - d)})}{\partial \Lambda}
\]

\[
= \frac{-1}{1 + e^{-\gamma(\rho - d)}} \left[ 1 + \frac{\gamma(\rho - d)e^{-\gamma(\rho - d)}}{1 + e^{-\gamma(\rho - d)}} \right] \frac{\partial d}{\partial \Lambda}
\]

(5.33)

We list the weights of DT methods together for better comparison:

\[
W_{\text{MMI}}(O_i, \Lambda) = \frac{-\kappa}{e^{d\kappa} + 1}
\]

(5.34)

\[
W_{\text{MCE}}(O_i, \Lambda) = \frac{-\gamma e^{-\theta + d\gamma}}{(e^{d\gamma} + e^{-\theta})^2}
\]

(5.35)

\[
W_{\text{SME}}(O_i, \Lambda) = \frac{-1}{1 + e^{-\gamma(\rho - d)}} \left[ 1 + \frac{\gamma(\rho - d)e^{-\gamma(\rho - d)}}{1 + e^{-\gamma(\rho - d)}} \right]
\]

(5.36)

A plot of the weight functions are shown in Fig. 5.10. Note that the maximum weights of MMI and MCE are normalized to one for better comparison. From the figure, we have the following observations:

- The weight function of SME is an approximation of a step function, with ripples at the transition area around \( \rho = d \). When an utterance is well separated, i.e. \( d > \rho \), there is no contribution to the gradient and hence no contribution to the update of the model parameters. This is similar to support vector machines (SVM), where the decision boundary is only determined by support vectors, i.e. the vectors falls within the margin. As a result of this, the decision boundary of SME is mainly determined by the utterances whose separation measure are below the margin.

- The weight function of MMI with \( \kappa = 0.5 \) (in subfigure (a)) looks similar to that of SME except that the transition is centered at \( d = 0 \). Using this weight function, correctly classified utterances contribute much less than wrongly classified utterances. The slope of the transition is controlled by \( \kappa \). If small \( \kappa \) is used (in subfigure (b)), correctly classified utterances will contribute more to the gradient and wrongly classified utterances will be smaller.
Figure 5.10: Contributions to the gradient of loss functions of DT methods.
• The weight function of MCE is a bell-shape symmetric function w.r.t. the line \( d = -\theta / \gamma \). When \( \theta = 0 \), weight is the highest for utterances near the decision boundary. The weight becomes smaller as the utterance get away from the decision boundary in both directions. A small weight for very wrongly classified utterances is helpful to reduce the effect of outliers on the parameters estimation. The center of MCE weight function can be shifted by adjusting \( \theta \) and the function can be fatter if smaller \( \gamma \) is used (see subfigure (b)).

From the analysis above, SME is designed based on the concept of margin maximization while MCE and MMI mainly focused on reducing empirical error. From the perspective statistical learning theory, SME is better than MCE and MMI in terms of increasing the margin (also separation measure) of training data given an acoustic model. However, MMI and MCE also have the potential of improving the margin of the model significantly. An important practical issue here is to decide the parameters of the techniques, i.e. \( \lambda \) for SME, \( \eta \) for MMI, and \( \gamma \) for MCE. The tuning of these parameters is very important for the final performance but this issue is not the focus of this study. Our focus is to show that by improving generalization capability of the model, no matter what technique is used, the robustness of speech recognition system can be improved.

5.6 Experiments

5.6.1 System Description

In this section, we study the effect of improving model generalization capability on speech recognition performance for both matched and mismatched testing cases. The performance of the DT methods is evaluated on Aurora-2 [69] and Aurora-3 [70] tasks. The acoustic models use standard “simple back-end” configurations, in which each digit is modeled by 16-state HMM with 3 Gaussian mixtures per state. Note that there is no short pause or ”sp” model in the acoustic modeling, and this does not have a significant effect on our experimental results. MFCC features are used for system training and testing and extracted using the WI007 feature extraction program provided by Aurora-2. There are 39 raw features, including 13 static features and their first and second order differential features. Cepstral energy \( \text{C0} \) is used instead of log energy. The order \( M \) in
Table 5.1: The settings for the Aurora-3 task.

<table>
<thead>
<tr>
<th>Nature of the task</th>
<th>Connected digital string in 5 European languages: Finnish, German, Spanish, Italian, and Danish.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling freq.</td>
<td>8000Hz</td>
</tr>
<tr>
<td>Sub-tasks</td>
<td>5 sub-tasks, 1 for each language</td>
</tr>
<tr>
<td>Test Scenarios</td>
<td>3 scenarios for each sub-task: well-match test, medium-mismatch test, and high-mismatch test. Refer to [70] for details.</td>
</tr>
<tr>
<td>Recognizer</td>
<td>HMM Toolkit</td>
</tr>
<tr>
<td>Acoustic model</td>
<td>16 states word model</td>
</tr>
<tr>
<td>Language Model</td>
<td>No language model</td>
</tr>
<tr>
<td>Test parameters</td>
<td>0 word insertion penalty</td>
</tr>
<tr>
<td></td>
<td>250 pruning threshold</td>
</tr>
</tbody>
</table>

(2.2) is set to 3 for generating first order differential features and 2 for generating second order differential features. This is different from the experiments in Chapter 4 where $M$ is set to 2 for both cases. This difference will cause minor differences in the baseline results but will not change the observations in this chapter.

In our experimental study, we will compare SME with MCE and MMI. As our purpose is to demonstrate how model generalization capability affects speech recognition robustness rather than to carry out a comprehensive comparison of the three model training criteria, we will only show the comparison on selected test scenarios on Aurora-2 task. Similar to SME, the implementation of MCE and MMI is also based on GPD. All the three DT methods uses N-best competing alignments (N=2) for confusion information. The parameters used in MCE are empirically determined as follows: the sigmoid parameters $\theta = 0$ and $\gamma = 0.1$ for multi-condition training of Aurora-2 task and $\gamma = 0.05$ for clean-condition training; the likelihood modification parameter $\eta = 0.066$ (in equation (13) of [48]). These settings are found to produce good results for MCE. For SME, the sigmoid parameter $\gamma = 2$ and the $\lambda$ in (5.13) is set to 5 if not explicitly specified. For MMI, the likelihood modification parameter $\eta = 0.03$ which produces good generalization capability of the model. Note that in the experimental studies of this chapter, the parameters are usually not fine-tuned for best performance.
Figure 5.11: Effect of SME, MCE, and MMI training methods on the separation measures of speech recognition. Four scenarios of Aurora-2 task are studied (one subfigure for each scenario). For each scenario, separation measures of utterances belonging to the scenario are computed using ML-, SME-, MCE-, and MMI-trained acoustic models and represented as histograms. The y-axis (i.e. x=0) is also plotted for easier analysis. The four scenarios include: (a) separation measures of clean training data; (b) separation measures of clean testing data; (c) separation measures of 10dB test data; (d) separation measures of -5dB test data.
5.6.2 Effect of SME on separation measures

Let’s first examine how well the DT methods improve the separation measure of the training and testing data. Note that a larger separation measure corresponds to a larger distance between a sample and the decision boundary and better separation. In Fig. 5.11, we compare the histograms of separation measures obtained using acoustic models trained by ML, MCE, MMI, and SME. The acoustic models are trained using clean training data. The features are processed by MVN [22] in an utterance-by-utterance fashion.

In Fig. 5.11(a), the histograms of separation measures of clean training data are shown. There are 8440 training utterances in the training set, hence there are 8440 separation measures also. From the figure, it is observed that the histogram of separation measures obtained with SME model is shifted right significantly compared to that obtained with ML model. Furthermore, there is a sharp slope around 9 in the histogram of SME. This is because when the training process stops, the final margin value is 9.14. These observations indicate that the SME-trained acoustic model has a larger margin and better generalization capability than the ML-trained acoustic model. After SME training, the area under the histogram on the left side of y-axis (x=0) is very small, which indicates a very small empirical risk, or string error. Besides SME, MMI and MCE also improves the margin significantly. In this specific study, SME produces best improvement of margin, followed by MMI and MCE.

In Fig. 5.11(b), the same study is carried out on the clean test data. There are totally 10,010 test utterances in the clean test set, the same as the following 10dB and -5dB test sets. Compared to ML, all the three DT methods significantly increases the separation measures of testing data as they have larger margins than ML as shown in Fig. 5.11(a). It is also observed that SME improves the separation measures of testing data the most and MCE the least, similar to the observations for training data in Fig. 5.11(a).

In Fig. 5.11(c), the separation measure histograms of 10dB test sets are shown. It is observed that the effect of all the DT methods becomes less significant in 10dB test set than in clean test set. One reason for this observation is that, as the mismatch becomes larger, the margin becomes less effective in covering the mismatch. Another reason is that the confusion pattern of noisy testing data may be different from that of clean training data. Hence, what DT learns from clean training data becomes less relevant on
Table 5.2: Comparison of SME effects on correctly and wrongly classified utterances. Correct refers to those utterances correctly classified by ML model, and Wrong refers to the rest utterances.

<table>
<thead>
<tr>
<th>Group</th>
<th>Clean</th>
<th>20dB</th>
<th>15dB</th>
<th>10dB</th>
<th>5dB</th>
<th>0dB</th>
<th>-5dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td>6.81</td>
<td>4.72</td>
<td>3.97</td>
<td>3.27</td>
<td>2.62</td>
<td>2.05</td>
<td>0.56</td>
</tr>
<tr>
<td>Wrong</td>
<td>3.81</td>
<td>2.23</td>
<td>1.68</td>
<td>1.29</td>
<td>0.68</td>
<td>-0.08</td>
<td>-0.71</td>
</tr>
</tbody>
</table>

noisy testing data. Despite this observation, the improvement of SME is still significantly better than that of MCE. The improvement of MMI is just slightly worse than that of SME. Note that the separation measures of most utterances evaluated with ML model are larger than -9. However, from the figure, only a small portion of them are covered by SME-trained acoustic model, in which there is a buffer zone with width=9.14. The majority of utterances that are wrongly classified by ML model are still wrongly classified by SME. This demonstrates the complexity of noise effect in speech recognition, which cannot be analyzed in a simple way.

In Fig. 5.11(d), the separation measure histograms of -5dB test sets are shown. As the SNR level is extremely low, the noise is more dominant than speech, and SME actually decreases the mean of the histogram. This observation is also true for MCE and MMI. However, the DT methods increase the area under the histogram on the right of y-axis, and this implies that the number of correctly recognized utterances is increased. The reason may be that the DT methods are able to improve separation measures for relatively good utterances, while they degrade separation measures for those bad utterances. We will show next that SME performs differently for good and bad utterances.

We also compare the SME effect on two groups of utterances, i.e. the group that is correctly classified by ML model and the group that is not. The comparison is shown in Table 5.2, where the average absolute increases of separation measures achieved by SME over ML are shown. From the table, it is obvious that SME performs better for those utterances already correctly classified utterances by ML model, i.e. those relatively good utterances. The reason for this is similar to the reason for the different effects of SME at different SNR levels. For utterances in relatively better conditions, the deviation in log likelihood domain is smaller, the SME training is more relevant, hence, the effect of SME is more obvious.
In summary, all the DT methods are able to improve the margin of the model significantly. Although the improvement varies from method to method, it is generally observed that improved margin on clean training data usually result in better separation of noisy test data if the SNR level is not too low. This suggests that big margin is beneficial to improve robustness. However, the conclusion may not generalize to cases where big mismatch exists.

5.6.3 Effect of Margin Size

An important question in SME training is the determination of the margin size. From our previous discussions, we expect that wide margin will make the acoustic model more general and robust. In this section, we will study the effect of margin size in SME model training and speech recognition. Similar study can also be performed for MCE and MMI by varying $\gamma$ and $\eta$. For brevity, we will just use SME as an example to show the effect of margin on robustness.

In SME, the margin is not fixed, but jointly estimated with the acoustic model parameters by using the GPD algorithm. The variable $\lambda$ is used to control the relative weights of the generalization term and the empirical risk term in the objective function of SME [see (5.13)]. Usually, larger $\lambda$ will produce larger margin and more general model. We now study four $\lambda$ values: 0.2, 1, 5, and 25. The acoustic model is trained from clean data and the features are processed by MVN.

The average recognition accuracies obtained by SME with different $\lambda$ values are shown in Table 5.3. From the table, it is observed that $\lambda=0.2$ produces poor performance, while the other three $\lambda$ values produce similar results. For $\lambda=0.2$, SME improves recognition performance significantly at high SNR levels (clean, 20dB), but decreases performance at low SNR levels (5dB, 0dB, -5dB). For the other three $\lambda$ values, SME improves performance at all SNR levels. The last row of the table shows the margins estimated by SME when the accuracies shown in the table are obtained. As we expected, larger $\lambda$ produces larger margin. Recognition results produced by MCE and MMI training are also shown for comparison. The performance improvement of MCE and MMI is less significant than the best performance improvement of SME at all SNR levels.
Table 5.3: Performance of SME with MVN processed MFCC features with different \( \lambda \) values on Aurora-2 task. The model is trained from clean data. ML represents the maximum likelihood baseline. The best MMI and MCE results are shown for comparison.

<table>
<thead>
<tr>
<th>SNR</th>
<th>ML</th>
<th>MCE</th>
<th>MMI</th>
<th>SME with different ( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>Clean</td>
<td>99.16</td>
<td>99.59</td>
<td>99.65</td>
<td>99.43</td>
</tr>
<tr>
<td>20dB</td>
<td>97.42</td>
<td>98.34</td>
<td>98.32</td>
<td>97.83</td>
</tr>
<tr>
<td>15dB</td>
<td>95.17</td>
<td>96.67</td>
<td>96.51</td>
<td>95.44</td>
</tr>
<tr>
<td>10dB</td>
<td>89.34</td>
<td>92.06</td>
<td>92.05</td>
<td>89.60</td>
</tr>
<tr>
<td>5dB</td>
<td>74.48</td>
<td>80.15</td>
<td>81.03</td>
<td>74.28</td>
</tr>
<tr>
<td>0dB</td>
<td>45.21</td>
<td>54.06</td>
<td>56.71</td>
<td>43.41</td>
</tr>
<tr>
<td>-5dB</td>
<td>17.81</td>
<td>22.76</td>
<td>25.68</td>
<td>17.12</td>
</tr>
<tr>
<td>0-20dB</td>
<td>80.33</td>
<td>84.25</td>
<td>84.92</td>
<td>80.11</td>
</tr>
<tr>
<td>Margin</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.00</td>
</tr>
</tbody>
</table>

To examine the reason of the different performance shown in Table 5.3, the separation measures of training data are now investigated. In Fig. 5.12, the histograms of separation measures of training data are shown with different \( \lambda \) values. These separation measures are obtained when the accuracies in Table 5.3 are obtained. Compared to ML, \( \lambda = 0.2 \) does not change the separation measures very much, except that the number of training utterances whose separation measures are less than the margin (1.00 in this case) is reduced. Hence, the generalization capability of acoustic model with \( \lambda = 0.2 \) is quite poor and this leads to poor recognition performance at mismatched testing scenarios when SNR level is low as shown in Table 5.3. When using \( \lambda \) values of 1, 5, and 25, SME improves the separation measures of training data significantly, and larger \( \lambda \) produces bigger improvement. However, the difference in recognition performance of the three cases are quite insignificant. It is also observed that using \( \lambda \) values of 1, 5, and 25 all produce better separation measure histogram than MCE and MMI. This explains the better performance of SME when \( \lambda \) is properly chosen in Table 5.3.

The observations in this section show that it is beneficial to improve the margin size during model training to improve the model’s generalization capability, while will translates into better robustness in noisy testing cases. It is also observed that it is not useful to use a very big margin. The reason for this may be complicated. One reason may be that SME and other DT methods are found to decrease the likelihood of the data
evaluated on the acoustic model. In other words, the fitness of the model to the training data is decreased after SME training, although the generalization capability is improved. Our study finds that the larger margin we pursue, the larger the likelihood degradation. As the current statistical speech recognition systems are based on the assumption that the acoustic model is able to represent the speech observations, a too much degradation in likelihood may violate this assumption. How to achieve good generalization capability while preserve likelihood is an open question and interesting research topic.

5.6.4 Speech Recognition Performance on Aurora-2

We first examine the performance of the DT methods with raw MFCC features. As the performance of SME is not very sensitive to the value of $\lambda$, we set $\lambda$ to be 5 for all following speech recognition experiments. The MFCC features used here are not processed by any feature compensation methods.

The performance of the DT methods on test data of different SNR levels for Aurora-2
task is shown in Table 5.4. Note that the R.R. columns represent the relative reduction of word error rate (WER) achieved by SME over ML baseline, i.e. $\text{R.R.} = \frac{\text{WER}_{ML} - \text{WER}_{SME}}{\text{WER}_{ML}} \times 100$, where WER=(1.0-Word Accuracy)*100%. From the table, all the three methods improve word accuracy significantly for both clean and multi-condition training schemes. In addition, SME usually produces the highest improvement and MCE the lowest. This is correlated with the capabilities of these methods in improving the margin as shown in Fig. 5.11. It is also observed that there are some differences between the two training schemes in terms of relative error rate reduction. In clean condition training, the DT methods perform better at high signal-to-noise ratio (SNR) levels (15dB and above) than at low SNR levels (5dB and below). This may be due to that the features at low SNR levels are too different from the clean training features, therefore, even more general model is not able to perform well. In multi-condition training, as the training data include noisy data down to 5dB, we see more even improvements at all SNR levels.

We also examine the performance of the methods for the 3 test sets in multi-condition training scheme in Table 5.5, as these three test sets represent different level of training-testing mismatches. In the multi-condition training scheme, the four kinds of noises in test set A are observed in the training data while the four kinds of noises in test set B are not. The test set C consists of two noises only, one is observed in the training data and the other is not. Note that there is also channel mismatch between test set C and the training data. Generally speaking, test set A represents the most matched scenario, and test set B represents the most mismatched scenario. From Table 5.7, it is observed that SME and MCE both produce the highest relative WER reduction for test set A and the least improvement for test set B. This is reasonable given their different levels of mismatches with the training data. It is also observed that MCE’s improvement on test set B and C is significantly less than SME’s improvement, while the two methods perform similar for matched test set A. This shows that for more mismatched scenarios, SME may have a larger advantage over MCE due to its better generalization property. MMI produces mixed results, with comparable performance as SME for test set B, but the least performance improvement for test set C.
Table 5.4: Performance of SME, MCE, and MMI with raw MFCC features on Aurora-2 task. Word accuracies of both clean and multi-condition training schemes are shown at different SNR levels. ML represents the maximum likelihood baseline. R.R. refers to the relative reduction of word error rate achieved by SME over the corresponding ML baseline results.

<table>
<thead>
<tr>
<th>SNR</th>
<th>Clean Condition</th>
<th>Multi-Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ML</td>
<td>MCE</td>
</tr>
<tr>
<td>Clean</td>
<td>99.04</td>
<td>99.61</td>
</tr>
<tr>
<td>20dB</td>
<td>94.36</td>
<td>97.40</td>
</tr>
<tr>
<td>15dB</td>
<td>85.58</td>
<td>92.27</td>
</tr>
<tr>
<td>10dB</td>
<td>66.82</td>
<td>75.23</td>
</tr>
<tr>
<td>5dB</td>
<td>39.20</td>
<td>45.86</td>
</tr>
<tr>
<td>0dB</td>
<td>17.14</td>
<td>21.56</td>
</tr>
<tr>
<td>0-20dB</td>
<td>60.62</td>
<td>66.47</td>
</tr>
</tbody>
</table>

Table 5.5: Performance of SME, MCE, and MMI with raw features on the three test sets of the Aurora-2 task using multi-condition training. Acc. refers to average word accuracy and R.R. is the relative WER reduction achieved over the corresponding ML baseline results.

<table>
<thead>
<tr>
<th>Test Set</th>
<th>ML Acc.</th>
<th>MCE Acc.</th>
<th>MMI Acc.</th>
<th>SME Acc.</th>
<th>ML R.R.</th>
<th>MCE R.R.</th>
<th>MMI R.R.</th>
<th>SME R.R.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>87.60</td>
<td>91.68</td>
<td>32.93</td>
<td>91.39</td>
<td>30.56</td>
<td>91.93</td>
<td>34.90</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>87.86</td>
<td>88.87</td>
<td>8.34</td>
<td>89.33</td>
<td>12.13</td>
<td>89.44</td>
<td>13.01</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>84.04</td>
<td>86.11</td>
<td>12.94</td>
<td>85.93</td>
<td>11.85</td>
<td>87.07</td>
<td>18.96</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>86.99</td>
<td>89.44</td>
<td>18.85</td>
<td>89.47</td>
<td>19.09</td>
<td>89.96</td>
<td>22.82</td>
<td></td>
</tr>
</tbody>
</table>

5.6.5 Interaction with MVN

As we have discussed in previous sections, one theoretical difficulty in applying SME, or more generally all DT methods, to noisy speech recognition is the mismatch between training and testing distributions. In this section, we will reduce the training-testing mismatch and examine its effect in SME training. A simple and effective feature normalization method, MVN [22], is used to process both the training and testing features before model training and testing. Each dimension of the 39 MFCC features are processed by utterance-based MVN individually. MCE and MMI are also evaluated on the processed features for comparison.
Table 5.6: Performance of SME, MCE, and MMI with MVN-processed features on Aurora-2 task.

<table>
<thead>
<tr>
<th>SNR</th>
<th>Clean Condition</th>
<th>Multi-Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ML</td>
<td>MCE</td>
</tr>
<tr>
<td>Clean</td>
<td>99.16</td>
<td>99.59</td>
</tr>
<tr>
<td>20dB</td>
<td>97.42</td>
<td>98.34</td>
</tr>
<tr>
<td>15dB</td>
<td>95.17</td>
<td>96.67</td>
</tr>
<tr>
<td>10dB</td>
<td>89.34</td>
<td>92.06</td>
</tr>
<tr>
<td>5dB</td>
<td>74.48</td>
<td>80.15</td>
</tr>
<tr>
<td>0dB</td>
<td>45.21</td>
<td>54.06</td>
</tr>
<tr>
<td>-5dB</td>
<td>17.81</td>
<td>22.76</td>
</tr>
<tr>
<td>0-20dB</td>
<td>80.33</td>
<td>84.25</td>
</tr>
</tbody>
</table>

Table 5.7: Performance of SME and MCE with MVN-processed features on the three test sets of the Aurora-2 task using multi-condition training.

<table>
<thead>
<tr>
<th>Test Set</th>
<th>ML</th>
<th>MCE</th>
<th>MMI</th>
<th>SME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>MCE</td>
<td>Acc.</td>
<td>R.R.</td>
</tr>
<tr>
<td>A</td>
<td>91.44</td>
<td>93.92</td>
<td>28.97</td>
<td>93.88</td>
</tr>
<tr>
<td>B</td>
<td>91.64</td>
<td>93.42</td>
<td>21.29</td>
<td>93.68</td>
</tr>
<tr>
<td>C</td>
<td>91.13</td>
<td>93.51</td>
<td>26.83</td>
<td>93.41</td>
</tr>
<tr>
<td>Average</td>
<td>91.46</td>
<td>93.64</td>
<td>25.53</td>
<td>93.71</td>
</tr>
</tbody>
</table>

The performance of SME, MCE, and MMI on the MVN-processed features is shown in Table 5.6. From the table, it is observed that all the methods significantly improves the performance for both clean and multi-condition training. SME outperforms MCE and MMI again and achieves about 28% relative error rate reduction for both clean and multi-condition training. If we compare the results in Table 5.6 and those in Table 5.4, we can see that SME gains further performance when working with MVN in terms of average relative error rate reduction. For example, for clean condition training, as MVN reduces the mismatch between noisy test features and clean training features, SME produces higher improvement in low SNR levels (5dB and below). However, the relative error rate reduction in 20dB and 15dB are decreased. For multi-condition training, we see better performance of SME in all SNR levels. Good interaction between MCE/MMI and MVN is also observed.

The experimental results show good interaction between DT methods and feature normalization method MVN. After MVN, the global mean and variance of both training
Figure 5.13: System with MVN and HLDA. Each feature dimension is processed by MVN individually. All the normalized features are then transformed by HLDA. The ML-trained model is refined by SME.

and testing data become zero and one, respectively. Hence, the assumption of same distribution for both training and testing data in the statistical learning theory is less violated. We expect SME, MCE, and MMI to work well with other feature domain methods as well.

Similar to Table 5.5, we also examine the performance of the DT methods on the 3 test sets for multi-condition training scheme in Table 5.7. Compare the results of these two tables, it is observed that the performance gap between test set A and test set B and C becomes smaller after MVN processing. This is because after the MVN processing, the feature distortion is reduced and training set can better represent test set B and C. Furthermore, it is also observed in Table 5.7 that the improvement of SME over MCE is larger for test set B than for test set A. This is similar to the observation in Table 5.5 and further shows the generalization capability of SME-trained models.

From the experimental results so far, it is observed that SME, MCE, and MMI generally improve the robustness of speech recognition significantly. SME usually performs better as it is the most aggressive in improving the margin of model. For the following experiments, we will only use SME to illustrate the usefulness of big margin for robustness.

5.6.6 Interaction with HLDA

Linear transformation is a popular technique in feature dimension reduction and extraction of useful features. In this section, we will investigate the interaction between SME and one of such transformation, the heteroscedastic linear discriminative analysis (HLDA) [155]. The system block diagram is shown in Fig. 5.13. We extend the 39 features to 52 features by appending the third-order differential features. The features are processed by MVN individually, and then used to build the ML model. After that, HLDA reduces
Table 5.8: Performance of SME with MVN-processed and HLDA-transformed features on Aurora-2 task.

<table>
<thead>
<tr>
<th>SNR</th>
<th>Clean Condition</th>
<th>Multi-Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ML</td>
<td>SME</td>
</tr>
<tr>
<td>Clean</td>
<td>99.34</td>
<td>99.73</td>
</tr>
<tr>
<td>20dB</td>
<td>97.57</td>
<td>98.53</td>
</tr>
<tr>
<td>15dB</td>
<td>95.00</td>
<td>96.75</td>
</tr>
<tr>
<td>10dB</td>
<td>88.54</td>
<td>92.27</td>
</tr>
<tr>
<td>5dB</td>
<td>71.70</td>
<td>80.70</td>
</tr>
<tr>
<td>0dB</td>
<td>38.85</td>
<td>55.89</td>
</tr>
<tr>
<td>-5dB</td>
<td>14.19</td>
<td>24.74</td>
</tr>
<tr>
<td>0-20dB</td>
<td>78.33</td>
<td>84.83</td>
</tr>
</tbody>
</table>

the feature dimension from 52 to 39. SME operates on HLDA-transformed features and model.

The performance of SME on the HLDA transformed features are shown in Table 5.8. By comparing Table 5.8 and Table 5.6, we have several observations. First, HLDA has different yet consistent effects on clean and multi-condition training schemes. For clean-condition training, HLDA improves recognition accuracy over ML baseline at high SNR levels (Clean and 20dB) but causes degradation at low SNR levels. This is reasonable as the training data here is clean, a higher joint probability of clean training data and their correct transcriptions may reduce the empirical risk but increase the generalization risk of testing on mismatched cases. For multi-condition training, HLDA improves performance in most SNR levels as noisy data are used for training. Only results in 0dB and -5dB are degraded as these two SNR levels are not used for training. Second, with HLDA, SME becomes slightly more effective in terms of average relative error rate reduction, especially for multi-condition training. The results here suggest that both HLDA and MVN are complementary to the SME training.

### 5.6.7 Interaction with TSN

In this section, we will combine the TSN in Chapter 4 and SME to see whether they are complementary to each other. Both training and testing samples are processed by MVN and TSN in sequence as described in Chapter 4. Note that there is no extra smoothing in
the TSN processing. The TSN-processed features are used to train both the ML model and SME model.

The results of TSN+SME are shown in Table 5.9. From the table, it is observed that while MVN+TSN improves performance significantly over MVN when ML training is used, MVN+TSN does not deliver significant improvement over MVN when SME is used. Similar observation is also observed when ARMA filter [28] is used with SME. This shows that the effects of SME and TSN (and possibly other temporal filters) are quite overlapped. The reason may be this. Temporal filters, such as TSN and ARMA filter, improve the robustness of speech recognition mainly by smoothing feature trajectories such that those high modulation frequency noisy spikes will be removed. However, temporal filters may only be effective when the distortion in the feature domain is relatively small. If the overall shape of the feature trajectory is distorted, temporal filters may fail to improve the performance. DT methods, such as SME, improve robustness by increasing the model’s margin to tolerate a certain degree of deviations of log likelihood scores, which is most likely due to small degree of distortion of features. Results on SME and MCE show that when the training-testing mismatch gets bigger, the improvement of SME and MCE gets smaller. Therefore, both SME (or more generally DT methods) and temporal filters are able to cope with small distortions in feature domain and less effective in dealing with large distortions. Their effects are similar and overlapped. Therefore, applying temporal filters such as TSN prior to SME training may not improve the performance significantly as compared to the case where TSN is not used.

<table>
<thead>
<tr>
<th>SNR</th>
<th>Scheme (Feature Compensation / Model Training)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MVN / ML</td>
<td>MVN+TSN / ML</td>
<td>MVN / SME</td>
<td>MVN+TSN / SME</td>
</tr>
<tr>
<td>Clean</td>
<td>99.16</td>
<td>99.07</td>
<td>99.68</td>
<td>99.58</td>
</tr>
<tr>
<td>20dB</td>
<td>97.42</td>
<td>97.63</td>
<td>98.51</td>
<td>98.45</td>
</tr>
<tr>
<td>15dB</td>
<td>95.17</td>
<td>95.78</td>
<td>96.85</td>
<td>96.73</td>
</tr>
<tr>
<td>10dB</td>
<td>89.34</td>
<td>91.26</td>
<td>93.09</td>
<td>92.96</td>
</tr>
<tr>
<td>5dB</td>
<td>74.48</td>
<td>81.19</td>
<td>82.93</td>
<td>83.35</td>
</tr>
<tr>
<td>0dB</td>
<td>45.21</td>
<td>58.93</td>
<td>58.67</td>
<td>60.66</td>
</tr>
<tr>
<td>-5dB</td>
<td>17.81</td>
<td>29.23</td>
<td>24.90</td>
<td>27.89</td>
</tr>
<tr>
<td>0-20dB</td>
<td>80.33</td>
<td>84.96</td>
<td>86.01</td>
<td>86.43</td>
</tr>
</tbody>
</table>
5.6.8 Speech Recognition Performance on Aurora-3

We also evaluate our approach on Aurora-3 task, in which the data were recorded in real noisy environments. Our evaluations are based on raw MFCC and MVN-processed MFCC features. HLDA is found to degrade the ML baseline performance on Aurora-3 task and hence ignored in our experiments.

The performance of SME with raw MFCC features on Aurora-3 is shown in Table 5.10. From the results, we have several observations. First, SME improves recognition accuracy for all cases except for the high-mismatch (HM) of German. This shows that by making the model more general, better performance can be obtained in realistic tasks. Second, SME usually produces higher relative error rate reduction in more matched cases. In most cases, the improvement for well-match (WM) is always the highest, followed by medium-mismatch (MM), and improvement is usually the lowest for HM. This is because in more mismatch training-testing cases, what SME learns from the training data is less relevant to the recognition of test data. The mismatch in HM may be beyond the generalization capability of the SME-trained acoustic model to tolerate. Similar results are observed in Aurora-2, where performance at very low SNR levels is usually less improved due to the high level of mismatch.

The performance of SME with MVN-processed MFCC features is shown in Table 5.11. Similar to results in Table 5.10, SME improves recognition accuracies significantly. This further shows the good interaction between MVN and SME.

5.7 Summary

In this chapter, we studied the approach of improving model generalization capability for better robustness of speech recognition. It is shown both analytically and empirically that, by improving acoustic model’s generalization capability, speech recognition performance can be improved significantly even when the testing data are mismatched from the training data. Furthermore, experimental results showed that SME, MCE, and MMI work well with feature domain method MVN. As MVN reduces the mismatch between the training and testing data, DT-trained acoustic model is able to better tolerate the
Table 5.10: Performance of SME with raw MFCC features on Aurora-3 task. The three training schemes are: well-matched (WM), medium-mismatch (MM) and high-mismatch (HM). In averaged results, the weights of WM, MM and HM are 40%, 35% and 25%, respectively.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Finnish</th>
<th></th>
<th>Spanish</th>
<th></th>
<th>German</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ML</td>
<td>SME</td>
<td>Imp.</td>
<td>ML</td>
<td>SME</td>
<td>Imp.</td>
</tr>
<tr>
<td>WM</td>
<td>92.00</td>
<td>96.97</td>
<td>62.13</td>
<td>86.08</td>
<td>94.69</td>
<td>61.85</td>
</tr>
<tr>
<td>MM</td>
<td>69.36</td>
<td>78.39</td>
<td>29.47</td>
<td>73.28</td>
<td>84.53</td>
<td>42.10</td>
</tr>
<tr>
<td>HM</td>
<td>42.61</td>
<td>56.47</td>
<td>24.15</td>
<td>41.29</td>
<td>54.05</td>
<td>21.73</td>
</tr>
<tr>
<td>Avg.</td>
<td>71.73</td>
<td>80.34</td>
<td>30.47</td>
<td>70.40</td>
<td>80.97</td>
<td>35.72</td>
</tr>
</tbody>
</table>

| Scheme | Danish | | Italian | | Average | |
|---|---|---|---|---|---|
| | ML | SME | Imp. | ML | SME | Imp. | ML | SME | Imp. |
| WM | 77.92 | 89.24 | 51.28 | 94.70 | 97.02 | 43.77 | 88.26 | 94.10 | 49.75 |
| MM | 53.11 | 64.41 | 23.09 | 85.30 | 86.38 | 7.35 | 72.07 | 78.94 | 24.59 |
| HM | 38.01 | 43.14 | 8.28 | 40.58 | 45.62 | 8.48 | 47.03 | 54.39 | 13.89 |
| Avg. | 59.26 | 69.02 | 23.97 | 77.88 | 80.45 | 11.60 | 72.29 | 78.87 | 23.74 |

mismatch. This suggest that feature domain methods are complementary with the proposed approach of improving model’s generalization capability. The study presented in this chapter is useful for researchers who would like to improving ASR robustness using margin-based model training methods.

Table 5.11: Performance of SME with MVN-processed MFCC features on Aurora-3 task.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Finnish</th>
<th></th>
<th>Spanish</th>
<th></th>
<th>German</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ML</td>
<td>SME</td>
<td>Imp.</td>
<td>ML</td>
<td>SME</td>
<td>Imp.</td>
</tr>
<tr>
<td>WM</td>
<td>89.24</td>
<td>97.82</td>
<td>79.74</td>
<td>93.16</td>
<td>96.35</td>
<td>46.64</td>
</tr>
<tr>
<td>MM</td>
<td>76.68</td>
<td>89.12</td>
<td>53.34</td>
<td>86.55</td>
<td>89.28</td>
<td>20.30</td>
</tr>
<tr>
<td>HM</td>
<td>79.65</td>
<td>82.90</td>
<td>15.97</td>
<td>81.65</td>
<td>83.31</td>
<td>9.05</td>
</tr>
<tr>
<td>Avg.</td>
<td>82.45</td>
<td>91.05</td>
<td>48.98</td>
<td>87.97</td>
<td>90.62</td>
<td>22.00</td>
</tr>
</tbody>
</table>

| Scheme | Danish | | Italian | | Average | |
|---|---|---|---|---|---|
| | ML | SME | Imp. | ML | SME | Imp. | ML | SME | Imp. |
| WM | 85.12 | 91.82 | 45.03 | 94.59 | 97.79 | 59.15 | 91.02 | 95.61 | 51.09 |
| MM | 62.71 | 71.47 | 23.49 | 82.26 | 90.81 | 48.20 | 78.57 | 85.18 | 30.85 |
| HM | 62.38 | 72.49 | 26.88 | 81.02 | 83.99 | 15.65 | 78.27 | 81.86 | 16.56 |
| Avg. | 71.59 | 79.87 | 29.12 | 86.88 | 91.90 | 38.23 | 83.47 | 88.52 | 30.55 |
Chapter 6

Conclusions

In this thesis, we studied two methods to improve the robustness of speech recognition against noise distortions, namely the temporal structure normalization (TSN) filter and the approach of improving model’s generalization capability. These two methods work effectively on three popular Aurora benchmarking tasks in the field of noise robust speech recognition. In this chapter, we will summarize the contributions of this thesis and discuss possible future research directions.

6.1 Contributions

6.1.1 Temporal structure normalization filter

We conducted a study on how additive noise affects the PSD functions of speech features, which can be seen as the modulation spectrum of speech. Our study shows that the examined noise and speech features generally have very different PSD functions. In addition, when speech is corrupted by noise, the PSD function of noisy speech features is approximately the sum of the PSD functions of the clean features and the corrupting noise. Therefore, the general shape (e.g. flatness) of the PSD function of noisy speech features is dependent on the SNR of the speech signal.

Based on our observations, we proposed to use the TSN filter to reduce the noise effect in the modulation spectrum domain. The TSN filters are designed to normalize the PSD function of noisy features to that of the unobserved clean features. As the PSD function of the unobserved clean features are not available during speech recognition, we used a group of reference PSD functions instead. These reference PSD functions
can be trained from a large amount of clean features offline, and captures the essential characteristics of clean speech. During speech recognition, the PSD function of incoming feature trajectories are normalized to these reference PSD functions.

Our experimental results have shown that TSN is effective in both small vocabulary Aurora-2 task and large vocabulary Aurora-4 task. In addition, the TSN filter is found to work complementarily with another temporal filter: the ARMA filter, in the Aurora-2 task. However, in Aurora-4 task, it is better for TSN filter to work alone as the strong low-pass filtering of the ARMA filter may remove too much information that is needed to discriminate a large number of speech units. In most cases, TSN delivers equal or better performance compared to other state-of-the-art temporal filters. The optimal filter length is also studied. It is found that when an adequate number of filter weights is used, e.g. above 25 taps, the performance of the TSN filter is not very sensitive to the filter length.

To address the problem of long processing delay, we also developed the segment-based implementation of the TSN filter. It is found that besides providing better adaptability than the utterance-based TSN filter, the segment-based implementation also produces better results than utterance-based implementation.

The major advantage of the TSN filter over other temporal filter is it’s ability to adapt to different environment conditions automatically. Previous temporal filters are fixed once designed and hence are not able to adapt to different environment conditions. The proposed TSN filter, on the other hand, is designed online for each incoming utterance by utilizing the temporal information of the utterance. Therefore, the TSN filter is able to track noise distortion and provide customized filtering of the feature trajectories.

The TSN filter can also be viewed as a normalization technique that normalizes the second-order statistics of speech features, i.e. the PSD functions. Hence, the TSN filter is an natural extension of other normalization methods, such as histogram equalization, which normalizes the first-order statistics of speech features, i.e. the probability distribution functions.

### 6.1.2 Study of model generalization capability

We began our study by investigating the effect of noise on the log likelihood scores of speech features in Chapter 5. The investigation showed that noise distortion will cause
random deviation of noisy speech features from clean speech features in the log likelihood domain. Empirical study on selected speech and noise data showed that such deviation generally move phonemes to the decision boundary and increases the confusion between competing classes. As a result, speech recognition performance will be significantly affected.

To reduce the sensitivity of speech recognition to the random variation of log likelihood scores, it is desirable to improve the acoustic model’s generalization capability. DT methods, such as SME, MMI, and MCE, were used to improve the margin of acoustic model, which is a measure of model’s generalization capability. While the original DT methods, focus on clean speech recognition (matched training-testing case), our study focuses on robust speech recognition (mismatched training-testing case). Our experimental results on both Aurora-2 and Aurora-3 tasks show that it is beneficial to apply SME and other DT methods during the acoustic model training stage. Significant improvement in recognition performance is observed for both matched and modestly mismatched testing cases, while small improvement is still observed for severely mismatched testing cases.

We also found that the DT methods worked even better when the speech features were pre-processed by the mean and variance normalization (MVN). The reason is that after the MVN processing, the distribution of the clean and noisy features become closer, hence the assumption of statistical learning theory is less violated. Besides MVN, HLDA was also found to work well with SME. However, the including of the TSN filter does not improve the performance further significantly. This shows the overlapped effects of TSN and SME. In addition, we found that when adequate separation measures are obtained, further increase of separation measures did not necessarily result in better performance in speech recognition.

6.2 Suggestion for Future Work

Currently, only one group of reference PSD functions is used in the TSN filter design, one PSD function for each feature trajectory. We found that the PSD function of feature trajectory is in fact quite sensitive to the content of the utterance. Therefore, current TSN filter that normalizes the PSD functions of a single utterance to a single group of
reference PSD functions may be improved. Similar to class-based HEQ [26], it is possible to apply class-based TSN, i.e. using multiple groups of reference PSD functions, for better matching between the PSD functions of the incoming utterance and the reference PSD functions used for filter design.

The FIR implementation of TSN requires quite a long filter to perform well, e.g. more than 20 taps. If an infinite impulse response (IIR) filter structure is used, it is possible to reduce the length of the filter significantly. With a shorter filter, the computation complexity of filtering can be reduced.

Another possible improvement of TSN is to reduce the complexity of filter design process. Currently, the desired filter response is obtained by dividing the reference PSD function by the test PSD function. Hence, the obtained desired filter response is highly variable due to the large variability of the test PSD function. It is possible to pre-train multiple sets of TSN filters from some noisy training data. Each set of filter is used to represent a specific noise/SNR combinations. During testing, the correct set of filters are first identified and then used for feature filtering. In this way, we may obtain further gains as the filter response can be trained from a large amount of data rather than just one sentence or segment. In addition, it is not necessary to design the filter on the fly, hence computation complexity can be reduced.

For the framework of improving the generalization capability for better robustness, it can be further extended by combining the discriminative model training with discriminative feature transformation as shown in [156, 157]. The margin concept can also be used to estimate discriminative feature transformation matrix or temporal filters that could be used to replace the DCT matrix and dynamic feature extraction.

Another possible improvement is to use better optimization techniques in DT methods. In the current formulation, the objective function is not a convex function and global optimal solution is not guaranteed. If better optimization methods can be used, it is possible to obtain better performance.

The concept of improving the generalization capability may be combined with model adaptation methods. The challenge is to robustly estimate the model parameters using very limited adaptation data.
Appendix - Mathematical Representation of Noisy Modulation Spectra

We now derive the modulation spectra of the noisy speech signal generated from the trajectories of filterbank coefficients. If we assume the noise is additive and statistically independent from the speech signal, the noisy speech signal can be represented as:

\[ y(i) = x(i) + n(i) \]  

(1)

where \( y(i) \), \( x(i) \) and \( n(i) \) are the noisy speech, the clean speech and the noise, respectively; \( i \) is the speech sample index. Apply the STFT on both sides of (1), we get:

\[ Y(t, f) = X(t, f) + N(t, f) \]  

(2)

where \( Y(t, f) \), \( X(t, f) \) and \( N(t, f) \) are the 2-dimensional short-time Fourier coefficients of the \( y(i) \), \( x(i) \) and \( n(i) \), respectively; \( t \) and \( f \) are the frame index and acoustic frequency index, respectively. The spectrogram of the noisy speech is

\[ |Y(t, f)|^2 = |X(t, f)|^2 + |N(t, f)|^2 \]

\[ + 2|X(t, f)||N(t, f)| \cos(\theta(t, f)) \]  

(3)

where \( \theta(t, f) \) is the angle between \( X(t, f) \) and \( N(t, f) \) and represents the phase difference between the two. The phase item \( 2|X(t, f)||N(t, f)| \cos(\theta(t, f)) \) in (3) has zero expected value when noise and speech are independent and are less significant than the other two items. Hence it is often omitted in many speech enhancement techniques such as spectral subtraction \([11]\). For simplicity, we ignore the phase item and (3) becomes

\[ |Y(t, f)|^2 = |X(t, f)|^2 + |N(t, f)|^2 \]  

(4)
The spectrogram is further processed by filterbank analysis, which integrates the frequency bins into filterbanks. Let \( L \) and \( F \) be the number of filterbanks and frequency bins respectively, and \( g^l_f, f = 1, ..., F \) be the weights of the integration window for the \( l^{th} \) filterbank, with constraint \( \sum_{f=1}^{F} g^l_f = 1, l = 1, ..., L \). The calculation of the filterbank coefficients is as follows

\[
\begin{align*}
|Y^M(t, l)|^2 & = \sum_{f=1}^{F} g^l_f |Y(t, f)|^2 \\
& = \sum_{f=1}^{F} g^l_f (|X(t, f)|^2 + |N(t, f)|^2) \\
& = |X^M(t, l)|^2 + |N^M(t, l)|^2
\end{align*}
\]

where

\[
|X^M(t, l)|^2 = \sum_{f=1}^{F} g^l_f |X(t, f)|^2
\]

\[
|N^M(t, l)|^2 = \sum_{f=1}^{F} g^l_f |N(t, f)|^2
\]

and the superscript \( M \) denotes the filterbank domain.

From the filterbank trajectories, the modulation spectrum of the noisy speech is calculated by

\[
|Y_m(k, l)|^2 = \text{DFT}(|Y^M(t, l)|^2)^2
\]

\[
= \text{DFT}(|X^M(t, l)|^2 + |N^M(t, l)|^2)^2
\]

\[
= |X_m(k, l) + N_m(k, l)|^2
\]

\[
= |X_m(k, l)|^2 + |N_m(k, l)|^2
\]

\[
+ 2|X_m(k, l)||N_m(k, l)| \cos(\beta(k, l))
\]

where

\[
Y_m(k, l) = \text{DFT}(|Y^M(t, l)|^2)
\]

\[
X_m(k, l) = \text{DFT}(|X^M(t, l)|^2)
\]

\[
N_m(k, l) = \text{DFT}(|N^M(t, l)|^2)
\]
Chapter 6. Conclusions

$k$ is the modulation frequency index and $|Y_m(k, l)|^2$, $|X_m(k, l)|^2$ and $|N_m(k, l)|^2$ are the modulation spectra of the noisy speech, the clean speech and the noise generated from the $l^{th}$ filterbank, respectively. DFT represents the discrete Fourier transform, and $\beta(k, l)$ is the angle between $X_m(k, l)$ and $N_m(k, l)$. To make (8) easier to understand, we use a simpler representation:

$$P_y = P_x + P_n + Q$$  \hspace{1cm} (12)

where $P_y$, $P_x$ and $P_n$ are the modulation spectra of the noisy speech, clean speech and noise respectively; $Q$ represents the phase item $2|X_m(k, l)||N_m(k, l)|\cos(\beta(k, l))$ in (8).
References


REFERENCES


REFERENCES


REFERENCES

information estimation of hidden markov model parameters for speech recognition,”

of large vocabulary recognition systems,” Speech Communication, vol. 22, no. 4,

[51] D. Povey, Discriminative Training for Large Vocabulary Speech Recognition. PhD

[52] P. C. Woodland and D. Povey, “Large scale discriminative training of hidden
markov models for speech recognition,” Computer Speech and Language, vol. 16,

1993.

[54] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum
269, Apr 1967.


2000.

[57] N. Morgan, Q. Zhu, A. Stolcke, K. Sonmez, S. Sivadas, T. Shinozaki, M. Ostendorf,
P. Jain, H. Hermansky, D. Ellis, G. Doddington, B. Chen, O. Cetin, H. Bourlard,
and M. Athineos, “Pushing the envelop–aside,” IEEE Signal Processing magazine,

language processing for next-millennium communications services,” Proc. IEEE,
REFERENCES

[59] M. Benzeghiba, R. D. Mori, O. Deroo, S. Dupont, T. Erbes, D. Jouvet, L. Fis-
sore, P. Laface, A. Mertins, C. Ris, R. Rose, V. Tyagi, and C. Wellekens, “Auto-
matic speech recognition and speech variability: A review,” *Speech Communication*,

[60] H. Hermansky, “Perceptual linear predictive (PLP) analysis of speech,” *Journal of


[63] X. D. Huang, A. Acero, and H. W. Hon, *Spoken language processing: A guide to


[67] S. Young *et al.*, *The HTK book*. Cambridge university engineering department,
3.2.1 ed., 2006.


formance evaluation of speech recognition systems under noisy conditions,” in *Proc.

[70] Aurora document no. AU/255/00, *Baseline results for subsect of SpeechDat-Car
Finnish database for ETSI STQ WI008 advance front end evaluation*, Nokia, Jan
2000.
REFERENCES


182
REFERENCES


REFERENCES


184
REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


