Large-Margin Structured Prediction Extensions of Neural Networks for Automatic Speech Recognition

by

Suman V Ravuri

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Nelson Morgan, Chair
Professor Jerome Feldman
Professor Keith Johnson

Fall 2015

Large-Margin Structured Prediction Extensions of Neural Networks for Automatic Speech Recognition

Abstract

Large-Margin Structured Prediction Extensions of Neural Networks for Automatic Speech Recognition

by

Suman V Ravuri

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Nelson Morgan, Chair

Neural networks, especially those with more than one hidden layer, have re-emerged in Automatic Speech Recognition (ASR) systems as replacements to emission models based on Gaussian Mixture Models (GMMs). While the use of these so-called Deep Neural Networks (DNNs) has enjoyed widespread success due to improvements in recognition results, the exact source of better recognition accuracy is not entirely understood. Using a bootstrap resampling framework that generates synthetic test set data satisfying conditional independence assumptions of the model while still using real observations, I show that DNNs used for both feature generation and hybrid acoustic modeling help compensate for incorrect conditional independence assumptions and help fix poor phone duration estimates of the hidden Markov Model (HMM).

Despite these improvements, the large increase in word error rates for DNN-HMM systems on real data compared to synthetic data suggests that one can improve recognition performance by modifying the training criterion. Since neural networks are log-linear at the output layer, I propose using sequences of last hidden layers as input to a log-linear model, and training that model with large-margin criteria. These Structured Support Vector Machine (SVM) approaches allow us to more directly minimize errors relevant to automatic speech recognition, and provide some guarantees on test set error. First, I show how one can generate better features by combining a neural network with a hidden Markov Support Vector Machine (HMSVM). Then, I propose a hybrid DNN-Structured SVM acoustic model and an online training algorithm that iteratively updates alignments for faster convergence. Training of this model falls under a class of approaches known as sequence-discriminative training, which are used to train state-of-the-art systems. This DNN-latent Structured SVM model beats alternative methods to sequence-discriminative training by 1.0% absolute, while needing 33-66% fewer utterances to converge.

Finally, I analyze the Structured SVM approach to sequence-discriminative training and compare it to standard methods. I show how the loss function for boosted Maximum Mutual Information is an upper bound of the hinge loss for the Structured SVM, and how such a relaxation precludes the use of aggressive boosting parameters needed for better results. Finally, I analyze four of the most popular sequence-discriminative training criteria – Maximum Mutual Information, boosted Maximum Mutual Information, Minimum Phone Error, and state-level Minimum Bayes Risk – and the latent Structured SVM using the bootstrap resampling framework, and compare how different sequence-discriminative training criteria compensate for data/model mismatch. Structured SVM models perform better for real rather than synthetic data, likely because the model makes fewer distributional assumptions about the underlying data.

To Sarah Downs and the ’rents

Contents

Contents **ii**

List of Figures **v**

List of Tables **vii**

1 Introduction **1**

2 Background **3**

2.1 Automatic Speech Recognition 3

2.1.1 Lexicon 5

2.1.2 Features 6

2.1.3 Language Model 6

2.1.4 Acoustic Model 7

2.1.4.1 Frame Classification 9

2.1.4.2 Transition Model - Hidden Markov Model Phone Model . . 9

2.2 Neural Networks 10

2.2.1 Neural Networks in ASR 10

2.3 Mathematical Setup 12

2.4 Bootstrap Resampling 14

2.5 Sequence-Discriminative Training 15

2.6 Structured SVMs 16

2.6.1 Binary SVM 16

2.6.2 Model 17

2.6.2.1 Binary Classification 18

2.6.2.2 Multiway Classification 18

2.6.2.3 Hidden Markov Support Vector Machine 18

2.6.2.4 Log-linear Speech Recognition 19

2.6.3 Optimal Parameters 20

3 Analysis of Depth for Tandem Features and Hybrid Systems **22**

3.1 Introduction 22

3.2 Synthetic Data Generation 24
3.3 Experimental Setup 25
3.3.1 Data and Modeling 25
3.3.2 Tandem 26
3.3.3 DNN-HMM Hybrid Systems 26
3.3.4 Metrics and Alignments 27
3.4 Results 28
3.4.1 HMM Phone Loop 28
3.4.1.1 Tandem 28
3.4.1.2 Hybrid 28
3.4.2 Full Recognition System with Lexicon and Language Model 29
3.4.2.1 Tandem 29
3.4.2.2 Hybrid 30
3.5 Conclusion 30

4 Structured SVM Extensions of Neural Networks for Feature Extraction and Sequence-Discriminative Training 40
4.1 Introduction 40
4.2 Features 41
4.2.1 Introduction and Related Work 41
4.2.2 Structured SVM 42
4.2.2.1 Model 42
4.2.2.2 Training 44
4.2.2.3 Proposed Method 45
4.2.3 Experimental Setup 45
4.2.3.1 Aurora2 45
4.2.3.2 ICSI Meeting Corpus 46
4.2.4 Results 46
4.2.4.1 Frame Recognition 47
4.2.4.2 Speech Recognition 47
4.2.5 Conclusions and Future Work 48
4.3 Sequence-Discriminative Training 49
4.3.1 Introduction 49
4.3.2 Related Work 51
4.3.3 Latent Structured SVM Hybrid Acoustic Models 52
4.3.3.1 Experiments 55
4.3.3.2 Connection to boosted MMI 56
4.3.4 Experimental Setup 56
4.3.4.1 Data and Language Model 56
4.3.4.2 Recognition System 56
4.3.5 Results 57
4.3.6 Conclusion 60

5 Analysis of Sequence Discriminative Training Criteria for DNN-HMM ASR Systems 61
5.1 Introduction 61
5.2 Data and Base Experimental Setup 62
5.3 Batch Size 62
5.3.1 Results 63
5.4 Log-Sum-Exp Upper Bound to Hinge Loss 63
5.4.1 Experimental Setup and Results 65
5.5 Bootstrap Resampling Analysis of Sequence Discriminative Training Criteria 68
5.5.1 Experimental Setup 68
5.5.2 Results 69
5.6 Conclusion 69

6 Conclusion 72
6.1 Contributions and Future Work 72
6.2 Beyond 74

Bibliography 76

List of Figures

2.1 *Toy view of an Automatic Speech Recognition System.* 4

2.2 *Lexicon.* 5

2.3 *Illustration of phoneme /ah/ in different contexts to illustrate co-articulation effects.* 8

2.4 *Blue lines separate the beginning, middle, and end states for phone /aw/.* 9

2.5 *HMM phone model for triphone "s-t+er" ".* 10

2.6 *Illustration of neural networks in "Tandem"-based systems* 12

2.7 *Illustration of implied mathematical setup for speech recognition.* 13

2.8 *Illustration of the bootstrap resampling framework to generate synthetic test data at the phone level.* 14

2.9 *Illustration of Hidden Markov Support Vector Machine for three consecutive frames.* 18

3.1 *Reference vs. Model Phone Duration Histograms for MFCC and Tandem features and resampling units using an HMM Phone loop.* 36

3.2 *Reference vs. Model Phone Duration Histograms for for MFCC and Tandem features and resampling units using the full recognition system.* 37

3.3 *Reference vs. Model Phone Duration Histograms for GMM-HMM and DNN-HMM acoustic models and resampling units using the HMM phone loop.* 38

3.4 *Reference vs. Model Phone Duration Histograms for GMM-HMM and DNN-HMM acoustic models and resampling units using the full recognition system.* . . 39

4.1 *Diagram of the Hybrid MLP-Structured SVM Model for two consecutive frames. The parameters from the input features to the hidden units are those of a standard MLP, while the parameters from the hidden units to outputs, and time transitions, are trained using a Structured SVM.* 42

4.2 *Histogram for activation values for hidden layer nodes.* 43

4.3 *The figure represents the decode score for the word "cat" using monophone states.* 54

4.4 *Word Error Rate vs. Number of training utterances seen for different sequence-discriminative training criteria.* 59

5.1 *Mixture weight* $\frac{\exp(b\mathcal{L}(y^*,\hat{y}_{(n)})+\alpha^\intercal(\phi(h,\hat{W}_{(n)},\hat{S_{(n)}})-\phi(h,W^*,S^*)))}{Z}$ *of n-th best path* $\phi(h, \hat{W}_{(n)}, \hat{S}_{(n)})$ *used for gradient calculation* $\nabla_\alpha L(\alpha,\theta)$. *The x-axis represents the n-th best path on a* $\log_{10}$ *scale, while the y-axis represents the mixture weight on a* $\log_{10}$ *scale.* . . 66

5.2 *Effect on word error rate of using candidate alignments that violate the margin vs. using all alignments.* 67

List of Tables

3.1 *Phone Error Rate for HMM Phone Loop for different types of resampled data (top), and relative degradation among different types of features (bottom).* 29
3.2 *Phone Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom) for MFCC and Tandem features. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.* 31
3.3 *Word Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom) for MFCC and Tandem features. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.* 31
3.4 *Phone Error Rate using HMM Phone Loop for different types of resampled data (top), and relative degradation between different types of features (bottom), for GMM-HMM and DNN-HMM acoustic models. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.* 32
3.5 *Phone Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom), for GMM-HMM and DNN-HMM acoustic models. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.* 33
3.6 *Word Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom), for GMM-HMM and DNN-HMM acoustic models. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.* 34

4.1 *Frame Error Rate on Cross-Validation Set of Aurora2 for both the multi-layer perceptron (MLP) and MLP-Structured SVM (MLP-SSVM).* 47
4.2 *Frame Error Rate on Cross-Validation Set of the ICSI Meeting Corpus for both the multi-layer perceptron (MLP) and MLP-Structured SVM (MLP-SSVM).* . . . 47
4.3 *Average WER for several systems under different noise conditions on the Aurora2 corpus. Bold numbers indicate best performance. Note that, as mentioned before, MLP use the Krylov Subspace Descent optimization method.* 48

4.4 *WER for several systems on the large vocabulary section of the ICSI meeting corpus. Note that, as mentioned before, MLP use the Krylov Subspace Descent optimization method.* . . . 48

4.5 *Word Error Rates for baseline systems. CE refers to cross-entropy, MMI maximum mutual information, bMMI boosted MMI, MPE minimum phone error, and sMBR state-level minimum Bayes risk.* . . . 58

4.6 *Effect of loss unit and boosting parameter onthe performance of DNN-Latent Structured SVM systems.* $\lambda = 0.0001$*, size of the N-best list is 1,000.* . . . 58

4.7 *Effect of N-best list size on word error rate. For the frame model the boosting parameter is 5, while for phone it is 3.* . . . 58

4.8 *The effect of updating phone model temporal parameters on word error rate. The boosting parameter is 5 for the frame model and 3 for the phone model.* . . . 59

4.9 $\lambda = 0.0001$*, for frame, boosting parameter is 5, while for phone, it is 3.* . . . 60

5.1 *Effect of Batch Size on performance.* . . . 63

5.2 *Word error rates for log-sum-exp upper bound criteria by boosting parameters and number of sequences used to calculate the gradient. Numbers to the left of the slash are for the objective function summed over margin-violating alignments, while those to the right of the slash are for alignments summed over margin- and non-margin-violating alignments.* . . . 67

5.3 *Phone Error Rate at different resampling levels. GMM refers to Gaussian Mixture Model, CE to cross-entropy trained DNN, MMI and bMMI Maximum Mutual Information (MMI) and boosted MMI respectively, MPE Minimum Phone Error, sMBR state-level Minimum Bayes Risk, LSSVM-F Frame-level loss latent Structured SVM, and LSSVM-P phone-level loss latent Structured SVM.* . . . 70

5.4 *Word Error Rate at different resampling levels. GMM refers to Gaussian Mixture Model, CE to cross-entropy trained DNN, MMI and bMMI Maximum Mutual Information (MMI) and boosted MMI respectively, MPE Minimum Phone Error, sMBR state-level Minimum Bayes Risk, LSSVM-F Frame-level loss latent Structured SVM, and LSSVM-P phone-level loss latent Structured SVM.* . . . 71

Acknowledgments

When I was a first-year Ph.D. student, I used to read dissertations instead of papers because, at that time, I didn't have the requisite background knowledge to understand many papers. One section that always struck me about the dissertations were the acknowledgements: how did these dissertators work with so many people?

Now that I look back on my graduate career, I do have (and am happy) to thank a great many people who have helped me along the way. My advisor, Morgan, has literally changed the way I have looked at the world and how to understand it. Prior to graduate school, I had been enamored of the math that underpin many of the algorithms I learned in college, and the math seemed to be the silver bullet. Morgan re-oriented my focus toward empiricism, and how to make good scientific experiments that tell us something about the world. I've learned to love empiricism and the surprises it can shine on the world, and how this approach can be complementary (and sometimes more illuminating) than a theoretical one. I still have a great appreciation of theory, but certainly my world is much richer because I now have an intuitive grasp on how to view the world from an empiricist perspective.

While the forms at Berkeley state that I have one advisor, in reality I am lucky to have two more. Steven Wegmann, the head of the Speech Group at the International Computer Science Institute, has been a great mentor and has been so much fun to work with. He has been excellent at letting me know when my own mathematical thinking has been unclear, which has forced me to become a better researcher. I've also had incredible fun in espousing an apostate view and analyzing recognition systems at a time when such analysis has been sparse. I am in awe of my third advisor, Andreas Stolcke, who seems to know everything there is to know about speech. Although we did not directly worked on ASR problems, I was always impressed on how he had a grasp on how well certain algorithms would work, and when issues were bugs versus actual limitations of an algorithm.

My fellow graduate students and postdocs have been amazing, thoughtful, fun, and kind people, who have been quick to help me out when I've been stuck. I am particularly indebted to Oriol Vinyals, Bernd Meyer, and Lara Stoll. Oriol, because he showed me about deep learning at a time when it was out of favor. I've learned a lot about fighting for one's beliefs in the face of scientific skepticism, and has made an indelible mark on my psyche. Bernd helped me out at a time when I was stuck in an extremely thorny experimental issue, one which he had little putative gain for him, but was distressing to me. That he took that much time to help me get through this issue speaks to his kindness, which I try to emulate. Lara literally gave me a wonderful job when I was out of graduate school for a stint, and it was helpful in ways too numerous to enumerate here. That, and the cookies!

And oh, my, there are so many more. Arlo Faria taught me a lot about speech recognition when I was a first-year grad student and knew nothing about the subject. Mary Knox has been a wonderful friend and sounding board (and great office mate). Hang Su is the nicest and most competent graduate student, and seemingly has an answer to every question I have asked. My labmates Shuo-Yiin Chang and TJ Tsai are incredibly nice and always helpful. Adam Janin has been incredibly helpful at looking beyond the hype in research, and has forced me to think more clearly about my work. Erin and Justin Summers have always kept me sane, and Justin especially shares my love of fiction. Ma'ayan Bresler and Gireeja Ranade have been great friends, and I will miss the dinners. Rohit Prabhavalkar, with whom I spent a summer at Microsoft Research and whom I occasionally I crossed paths at conferences, is amazingly interesting and a great post-conference travel buddy.

I also want to thank the Speech Group at Microsoft Research for the one halcyon summer that I spent there. Mike Seltzer was a wonderful mentor and friend, and Li Deng, Dan Povey, Jasha Droppo, Geoff Zweig, and Dong Yu were wonderful people to bounce ideas off of, whether not not they were related to speech.

I would also like to thank my undergraduate advisor, Dan Ellis, for the interesting projects we worked on, and for continuing advice while I was a graduate student.

Special thanks goes to my dissertation committee, Profs. Jerome Feldman and Keith Johnson, for being so accommodating and responsive throughout the dissertation process. It's hard for me to imagine the task of a dissertation reader – to provide insightful comments on a long document about recondite ideas outside of one's expertise –, so to receive such helpful feedback is little short of amazing to me.

Lastly, and certainly not least, I want to thank my wife, Sarah Downs, for keeping my temperament even through the ups and downs of grad school, and my parents, Padmaja and Sreenivas Ravuri, for being unconditionally supportive (and not asking my when my dissertation will be complete).

And if I missed anybody, it's because I am scatterbrained and not because you aren't awesome. Please let me know, and I'll buy you a drink.

P.S. The material is based, in part, upon work supported by the National Science Foundation (NSF) under Grant No. IIS-1450916 and the NSF Graduate Research Fellowship Program.

Chapter 1

Introduction

The focus of this thesis is to make explicit and directly address a problem assumed by many researchers in the Automatic Speech Recognition (ASR) community: that the probabilistic models used for recognizing speech are not very good. Despite the revival of neural networks in ASR pipelines and significant advances since 2011, computer performance still lags human performance by at least a factor of two for conversational speech (and in practice, probably more, as ASR systems are often tuned for good performance on a particular test set). Somewhat more abstractly, if we think of speech as a stochastic process with some unknown distribution, our models, even with neural networks, do not well represent this underlying stochastic process. Directly modeling this unknown distribution has so far been extremely difficult, and it is hard to know what exactly is this true distribution, should it exist.

Although neural networks do not completely solve the modeling problem, they do indeed ameliorate some long-standing problems in speech recognition, as I will show in subsequent chapters. Moreover, the mathematic structure of neural networks, along with traditional rules of probability used to tie together various components of the recognizer, allow us to answer a different question: given that our models are poor, how can we train systems to minimize the word error rate? The mathematical structure exploited is linearity, and the tool I use to train systems is the Structured Support Vector Machine (SVM). By removing the probabilistic interpretation of these models, I show empirically that this approach can train systems better than standard methods used to train state-of-the-art speech recognition systems.

If one were to give an elevator pitch for this work, it would be this: given that our model family, from which $P_{model}(O, W)$ is selected through a training procedure, does not include the true distribution $P_{true}(O, W)$, how can we train our systems to improve performance? Unpacking this statement requires me to show three things: that indeed $P_{model}(O, W) \neq P_{true}(O, W)$, even with neural networks; that Structured SVM training of neural networks better maximizes performance than existing methods; and how the proposed method, and extant training methods, actually improve performance. These, broadly, are the main contributions outlined in the thesis, and the descriptions of each chapter are as follows:

- **Chapter 2** is a short introduction to Automatic Speech Recognition, neural networks in ASR, Structured SVMs, and the bootstrap resampling framework used to analyze recognition systems.

- **Chapter 3** analyzes the statistical properties of neural networks for Tandem neural network features and DNN-HMM hybrid systems. In particular, I use a bootstrap resampling framework to generate synthetic test data to study how neural network architecture helps compensate for data/model mismatch, and to examine what role depth plays.

- **Chapter 4** proposes hybrid neural network-Structured SVM models for Tandem feature extraction and sequence-discriminative training. Large-margin training of Tandem features allows for temporal structure to be introduced to the model and circumvents overtraining issues found in hybrid neural network-conditional random field approaches. Large-margin sequence-discriminative training outperforms the four most popular methods, which are typically used to train state-of-the-art ASR systems. Furthermore, this method requires far fewer utterances to converge compared to other methods.

- **Chapter 5** analyzes sequence-discriminative training methods. This analysis is split into two parts. The first compares the Structured SVM criterion to the more popular boosted MMI, and illustrates how different design choices allow the former criterion to outperform the latter. Then, I use the bootstrap resampling framework used in Chapter 3 to compare how different sequence-discriminative training criteria compensate for data/model mismatch.

- **Chapter 6** concludes the thesis, and gives some thoughts on possible future directions.

Chapter 2

Background

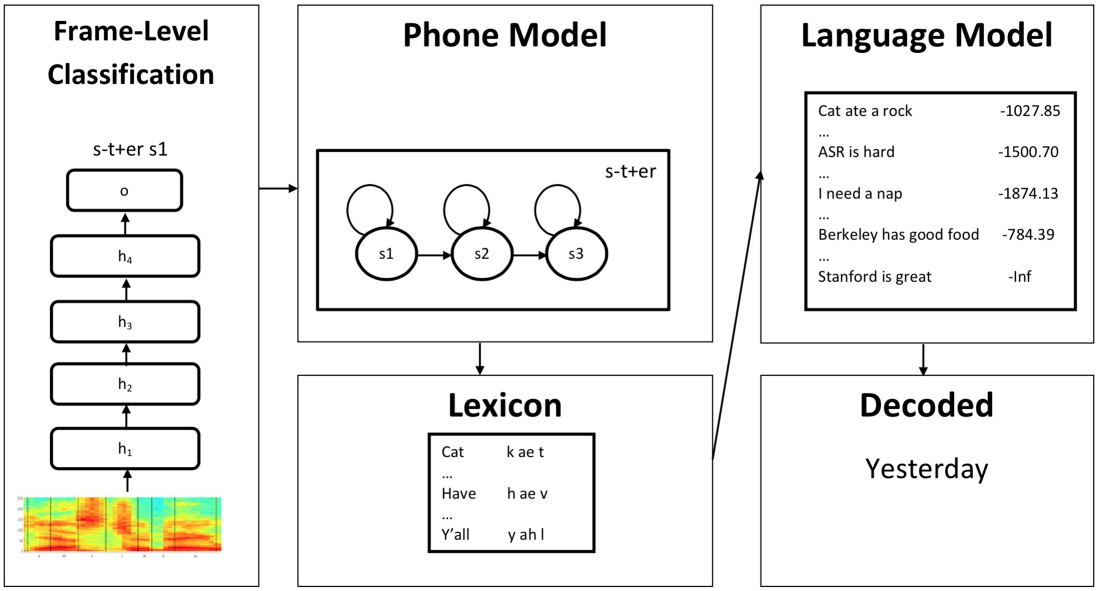
The focus of this thesis is the failure of probabilistic models used for automatic speech recognition (ASR) and an approach on how to compensate for poor modeling assumptions. At first glance, this topic seems a bit curious, since ASR has undergone a bit of a renaissance over the past four years. Word error rate (WER) (the standard performance metric) of a popular corpus, Switchboard, has decreased by nearly 50% relative, and with thousands of hours transcribed data, conversational speech recognition in English is within a factor of two of human-level of performance, while being nearly an order of magnitude worse only 15 years ago.

Much of the gain in the past four years stems from a renewed research interest in neural networks (currently rebranded as “deep learning”) for automatic speech recognition. Most of the improvement has occurred by replacing one part of the ASR system while leaving much of the rest of the system relatively untouched. I will show that, in a particular sense, using neural networks helps fix some statistical problems with extant ASR systems, but some rather egregious ones remain. Then I propose a method of training speech recognition systems to obtain better performance, given that we know that our current models are poor (and in may be poor for some time to come). Then, I will analyze how well this proposed method works.

Much of what I have written is at a high level, and this chapter will give requisite details of my approach for both the analysis and proposed improvements to the system.

2.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the process by which a computer, given a piece of audio containing speech, transcribes what was said in that piece of audio. Despite the ease with which humans are able to perform this task under a wide variety of conditions – room reverberation and noise seem to have little effect for a human while being extremely deleterious for an automatic system – this task is exceedingly difficult for computers, except in limited circumstances. If the recording is relatively free of noise, the microphones are



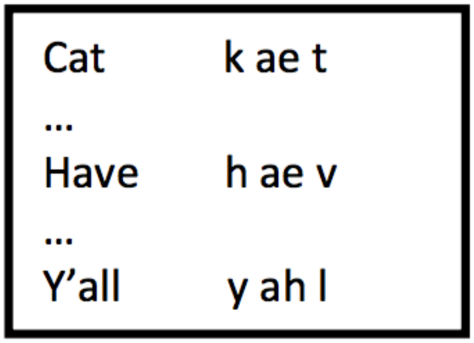
Cat ate a rock	-1027.85
…	
ASR is hard	-1500.70
…	
I need a nap	-1874.13
…	
Berkeley has good food	-784.39
…	
Stanford is great	-Inf

Cat	k ae t
…	
Have	h ae v
…	
Y'all	y ah l

Figure 2.1: *Toy view of an Automatic Speech Recognition System.*

placed near the speaker, and the vocabulary is extremely limited (such as a person speaking digits), recognition error rates are often below 1%, and for read speech, error rates are about 3-6%. For conversational speech, the best systems achieve about 8-15% word error rate [57], while human recognition on this task is roughly 4% [36]. Even the "best" system is likely not generalizable across corpora, as the system is highly complex: two types of neural networks perform frame-level audio to state classification, and three types of language models are combined to achieve this result, and the variance among test sets suggests that the actual performance may be closer to 15% than 8%. Despite this caveat, the decrease in word error rate is a tremendous achievement: in 1997, word error rates were 44% for this task [36].

Automatic Speech Recognition systems assume that the acoustics and associated words of an utterance, denoted O and W respectively, are a stochastic process, which we model as $P_{model}(O, W)$. While automatic speech recognition systems attempt to transduce speech into text, most modern systems do not perform direct word transcription (although systems such as [37] and [9] perform orthographic transcription). Instead, acoustic observations are transduced into phonemes, which are then transduced into words. Four major components – feature extraction, an acoustic model, a lexicon (alternatively called a pronunciation dictionary), and a language model – comprise a modern speech recognition system, illustrated in Figure 2.1. Probability theory and assumptions allow us to decompose $P_{model}(O, W)$ into



Cat	k ae t
…	
Have	h ae v
…	
Y’all	y ah l

Figure 2.2: *Lexicon.*

constituent components:

$$\begin{aligned} P_{model}(O,W) &\approx P_{model}(O|W)P_{model}(W) \\ &= \sum_{S} P_{model}(O|W,S)P_{model}(S|W)P_{model}(W) \\ &\approx \sum_{S} P_{model}(O|S)P_{model}(S|W)P_{model}(W) \\ &\approx \max_{S} P_{model}(O|S)P_{model}(S|W)P_{model}(W) \end{aligned}$$

where S is a sub-phoneme unit called a context-dependent state, $P_{model}(O|S)$ is the acoustic model, $P_{model}(S|W)$ the lexicon, and $P_{model}(W)$ the language model, which are further described below. The equation makes three approximations for computation efficiency purposes. The first is that the parameters of the language model are independent of the acoustic model, the second that the acoustics are conditionally independent of the word given the state, and the third that the best word sequence is well-approximated by the best state sequence associated with the word sequence. This last approximation is denoted as the Viterbi approximation.

2.1.1 Lexicon

Perhaps the least “automatic” part of ASR is the lexicon, or alternatively called the pronunciation dictionary, as shown in Figure 2.2. As the name suggests, for each word available to be decoded, a human gives one or more phonemic pronunciations. This allows the recognizer to train on phones, which are far fewer than the number of words and are shared among words. This setup, however, leads to rigid assumptions about word pronunciation, which can often fail if a foreign speaker uses the system. If multiple pronunciations are given per word, sometimes an estimate of the probability is also given. In addition to these procrustean pronunciation assumptions, I will show in the next chapter that the lexicon also modifies posterior phone length. That the lexicon itself does not explicitly model duration suggests that this constraint likely needs to be revisited.

2.1.2 Features

Although recent results suggest that one can obtain nearly equivalent performance using only raw waveforms as input to a DNN-HMM system ([56]), in general one must compute features in order to obtain reasonable recognition performance. Using features other than raw waveforms is particularly important in the presence of noise or reverberation, as waveforms can be particularly affected by these artifacts. Standard recognizers generally calculate features every 10 ms using a 25 ms window, and much of the processing for a conventional feature, such as mel-scaled cepstral coefficients (MFCCs) or perceptual linear prediction (PLP), broadly mirrors processing that occurs up to the basilar membrane. MFCCs and PLP features are typically 13-dimensional, and include first and second differences to account for temporal dynamics.

Since features are often calculated on time lengths much shorter than the average phone duration (usually 6-9 frames), features are sometimes concatenated with some number of frames of context – ±4 frames is typical – to incorporate longer duration. One issue with including longer term features, especially if they are modeled by Gaussian Mixtures, is that these features are generally are too high-dimensional for GMM classifiers to model. This issue is sidestepped by performing dimensionality reduction using linear discriminant analysis, and these features are denoted as LDA features in later chapters.

Features used in this dissertation include MFCCs, PLP, and LDA features.

2.1.3 Language Model

The language model (LM) attempts to model the prior probability of words $P(W)$ in a manner efficient for decoding. Typical LMs used for ASR make a trigram assumption; i.e., the current word is conditionally independent of all other previous words given the previous two. Mathematically, this is expressed as:

$$P(w_1, \ldots, w_n) = P(w_1, w_2) \prod_{i=3}^{n} P(w_i | w_{i-1}, w_{i-2})$$

The naive maximum likelihood estimate is:

$$P_{ML}(w_i | w_{i-1}, w_{i-2}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-1}, w_{i-2})}$$

where $c(w_{i-1}, w_{i-2}, w_i)$ are counts of particular word sequences on a large text corpus. This approach, however, suffers from the problem that there will likely be word sequences in test data that are unseen in training, and a naive approach would give 0 probability to those unseen word sequences. To deal with this issue, some probability is introduced to unseen word sequences through a "backoff" mechanism. Of the many such methods (see [32]), one that works particularly well for ASR is Kneser-Ney (KN) smoothing, which is calculated recursively as:

$$P_{kn}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - D, 0)}{c(w_{i-1})} + \gamma \frac{|w_{i-1} : c(w_{i-1}, w_i) > 0|}{\sum_{w_i} |w_{i-1} : c(w_{i-1}, w_i) > 0|}$$

$$\begin{aligned} P_{kn}(w_i|w_{i-1}, w_{i-2}) = & \frac{\max(c(w_{i-2}, w_{i-1}, w_i) - D, 0)}{c(w_{i-2}, w_{i-1})} \\ & + \frac{D|(w_{i-2}, w_{i-1}) : c(w_{i-2}, w_{i-1}, w_i) > 0|}{c(w_{i-2}, w_{i-1}, w_i)} P_{kn}(w_i|w_{i-1}) \end{aligned}$$

The intuition is that certain words, such as "Francisco", occur in fewer contexts than the word "the", and thus the probability of the former should be smaller than that of the latter. A more detailed explanation can be found in [11]. While KN discounting seems like a heuristic, a modified version that includes different discounts for n-gram size introduced in [11] can be interpreted as approximate inference for a n-gram language model based on a Hierarchical Pitman Yor process [66].

Recurrent neural networks (RNN) for language modeling have recently been introduced into the ASR pipeline, although at the time of writing, it is not included in the main decoder. Instead, RNN language models are used to re-score a N-Best list of decode hypotheses. Though not explored in this dissertation, please see [40] for more details.

2.1.4 Acoustic Model

If the language model estimates the probability of observing a particular word sequence, and the lexicon specifies one or more phoneme sequences for a particular word, then at first glance it seems that the acoustic model should try to classify phone sequences for a given observation. While this is true in a broad sense, two linguistic phenomena preclude us from naively using phoneme sequences as labels. The first is that the acoustics for a given phone are different at the beginning, middle, and end of phones, as shown in Figure 2.4. This leads to splitting of phones into a beginning, middle, and end state, and the HMM duration models account for this particular phenomena.

The second linguistic phenomenon is co-articulation, in which the previous and subsequent phonemes change the articulation of a particular phone. For instance, the acoustics of "/ae/" in cat are different than that in shaft. Figure 2.3 illustrates the different acoustics for the phoneme "/ah/" in different contexts. One way to account for this phenomenon is to predict "triphones", phones[1] with left and right context, but even for English, which has relative few phonemes – 39 –, the number of possible triphones are 39^3. Many of these

[1] Unlike in linguistics, the terms phone and phoneme are often used interchangeably in the ASR community.

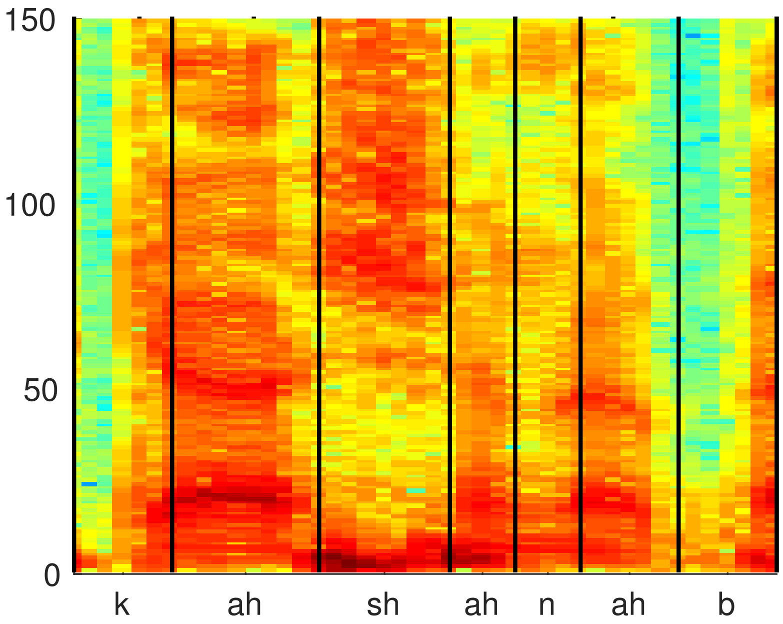


Figure 2.3: *Illustration of phoneme /ah/ in different contexts to illustrate co-articulation effects.*

triphones do not occur in a particular corpus, and moreover, many previous and subsequent phonemes – such as "/k/" and "/b/" in cat and bat – shape the middle phoneme in the same way. Thus, triphones states are clustered to reduce the number of possible labels. While many methods exist, by far the most popular one is [77], which clusters triphones based on a decision tree which asks questions about the articulators, such as if the left context is a nasal, etc.

The actual acoustic model comprises two components – frame-level classification and sequence modeling – that can be considered nearly independent components, and decomposes as follows:

$$P_{model}(O,S) = \prod_{i=1}^{N} P_{frame}(O_i|S_i) P_{trans}(S_1) \prod_{i=2}^{N} P_{trans}(S_i|S_{i-1})$$

Broadly speaking, the acoustic model of speech recognition systems is a hidden Markov Model (HMM), though the HMM used in automatic speech recognition has somewhat more specialized frame and sequence classification than HMMs used for other tasks.

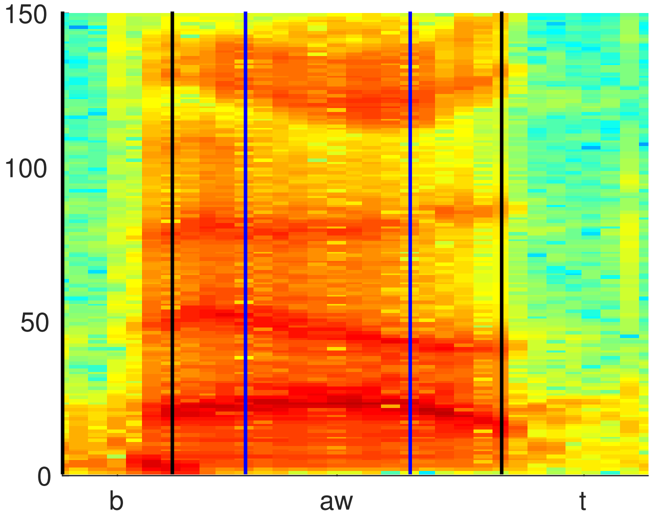


Figure 2.4: *Blue lines separate the beginning, middle, and end states for phone /aw/.*

2.1.4.1 Frame Classification

Until recently, frame classification was performed by a Gaussian Mixture Model (GMM), on a 25-ms window frame of features, every 10 ms.

$$P(O_i|S_i) = \sum_j c_j \mathcal{N}(O_i|\mu_{S_j}, \sigma_{S_j})$$

where:

$$\mathcal{N}(O|\mu, \sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp(-\frac{1}{2}(o-\mu)^\intercal \Sigma^{-1}(o-\mu))$$

$$c_j \geq 0, \sum_j c_j = 1$$

The use of GMM classifiers was, until recently, widespread, since they are able to model a wide variety of frame-level features, and yielded good recognition results.

2.1.4.2 Transition Model - Hidden Markov Model Phone Model

HMM phone models, shown in Figure 2.5, structurally encode the beginning, middle, and end states, and for an utterance, multiple “phone” models are concatenated. This forces the phone duration to be at least 3 frames long, and except for silence phones, skipping is not generally allowed. Moreover, having a linear model reduces the state space, as it reduces the number of possible alternative states. Although the phone model does enforce a minimum length duration, it is not a very good duration model. I will show in the next chapter that without neural networks, the phone model poorly estimates the duration of phones.

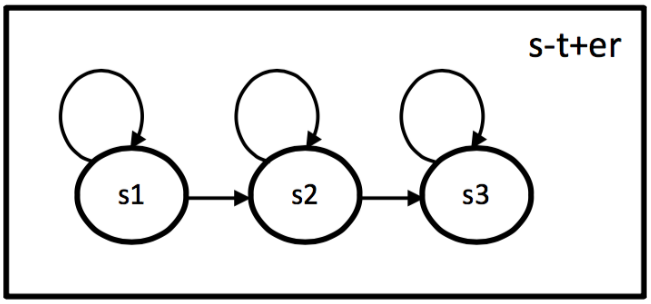


Figure 2.5: *HMM phone model for triphone "s-t+er" ".*

2.2 Neural Networks

Neural networks are classifiers that perform logistic regression after one more more non-linear transformations. Although other types of non-linear transformations exist – notably convolutional layers – the standard non-linear transformation is a matrix multiplication followed by an element-wise non-linearity:

$$h_i = \begin{cases} g(W_i o), & \text{if} \quad i = 1 \\ g(W_i h_{i-1}), & \text{otherwise} \end{cases}$$

where g is the element-wise non-linear function, W_i the parameters for layer i, and o the input. Standard non-linearities include the sigmoid, tanh, and rectified linear units. The output non-linearity is typically the softmax:

$$p_j = \frac{\exp(w_j^\intercal h_n)}{\sum_k \exp(w_n^\intercal h_n)}$$

The parameters w_j are the logistic layer parameters for category i. The hidden layer h_n can be considered a learned feature.

2.2.1 Neural Networks in ASR

There are two ways in which neural networks are generally used in Automatic Speech Recognition. The first is as a replacement to the Gaussian Mixture Model, which scores the observation given the state $P(O_i|S_i)$ as $\sum_i c_i \mathcal{N}(o|\mu_{S_i}, \sigma_{S_i})$. The neural network classifier estimates a probability of the state given the observation as $P(S_i|O_i) = \frac{\exp(\theta_{S_i}^\intercal h_i)}{\sum_{i=1}^N \sum_s \exp(\theta_s^\intercal h_i)}$ so the NN-HMM acoustic model is expressed as follows:

$$\begin{aligned} P(S|O) &= \prod_i \frac{\exp(\theta_{S_i}^\intercal h_i)}{\sum_{i=1}^N \sum_s \exp(\theta_s^\intercal h_i)} \prod_{i=2}^N P(S_i|S_{i-1}) \\ &= \prod_i \frac{\exp(\theta_{S_i}^\intercal h_i)}{Z} \prod_{i=2}^N P(S_i|S_{i-1}) \end{aligned}$$

Using a neural network estimator requires a modification of the overall probability model, since such "hybrid" acoustic models estimate $P(S|O)$ instead of $P(O|S)$. Another application of Bayes' rule modifies the speech recognition equation to a usable form:

$$\begin{aligned} \underset{W,S}{\operatorname{argmax}} P(W|O) &= \underset{W,S}{\operatorname{argmax}} \frac{P(W)P(O|S)P(S|W)}{P(O)} \\ &= \underset{W,S}{\operatorname{argmax}} \frac{P(W)\frac{P(S|O)P(O)}{P(S)}P(S|W)}{P(O)} \\ &= \underset{W,S}{\operatorname{argmax}} \frac{P(W)P(S|O)P(S|W)}{P(S)} \end{aligned}$$

$P(S)$, prior probability of states, is estimated assuming independence among states $P(S) = \prod_i P(S_i)$ from frame labels and typically, the term $\frac{p(S_i|O_i)}{p(S_i)}$ is called the "scaled likelihood." When first proposed in 1988 [4, 41, 5][2], the state sequences were phones rather than triphones, as training neural networks with triphones was at the time computationally intractable.

While the neural network estimator enjoyed a number of advantages – such as not needing to model the observations $P(O)$ directly and the ability to use input features with higher dimensionality and a longer temporal context – early promising results from GMM-HMM acoustic models, and its faster training and decoding speed compared to NN-HMMs decreased the amount of research spent on neural network acoustic models during the 1990s-2000s.

Research in NN-HMM acoustic modeling has experienced a revival since 2011 after a series of promising results, and seems to be the product of four trends: increased computing speed (especially GPUs) to allow for faster training and decoding with neural networks; a substantial increase in training data, allowing for more samples to be trained; the capacity to train on context-dependent triphones instead of monophones; and the ability to train multiple hidden layers. The ability to train multiple hidden layers, either through unsupervised pre-training, or through lots of training data and new hidden non-linearities, such as the rectified linear unit (ReLU), revived interest in what is now called the "deep" approach. Now DNN-HMM acoustic models (trained with 5-9 hidden layers) represent part of the state-of-the-art recognition system. On Switchboard and Call Home Corpora, DNN-HMM acoustic models outperform GMM-HMM systems by roughly 30% relative [70].

[2] Although the most cited reference,[6], was published 7 years after the initial proposal.

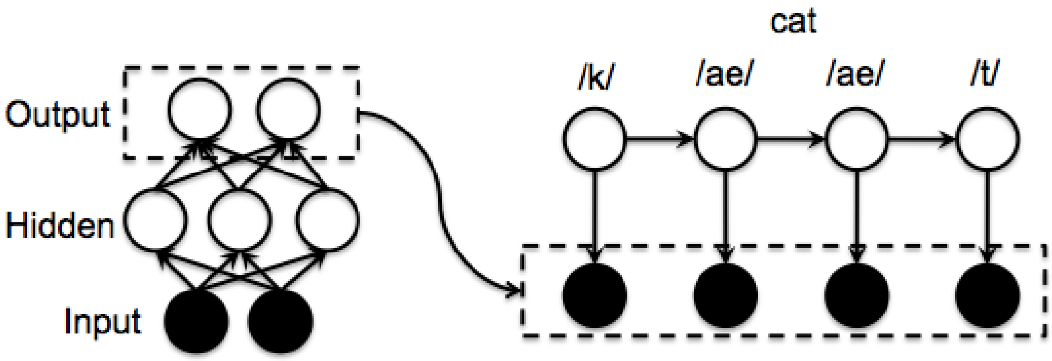


Figure 2.6: *Illustration of neural networks in "Tandem"-based systems*

The second way in which neural networks are used in automatic speech recognition is to augment features for a GMM-HMM or DNN-HMM acoustic model, as shown in Figure 2.6. The standard way to generate such features is to train neural network model that discriminates among phonemes, using a feature such as a MFCC or PLP. Then, dimensionality reduced (usually via principle component analysis) log posteriors are appended to a base-level feature – in "Tandem" – and are used as observations for a GMM/HMM-based system [27]. Such features seem to add extra useful phonemic information, and in general help speech recognition by at least 1% (and generally more) absolute on large-vocabulary tasks.

With the revival of the "deep" approach, new neural network models were explored for Tandem systems. I performed some of this research. In [75], Oriol Vinyals and I compared "deep" neural networks, trained with pre-training and stochastic gradient descent, to a single hidden layer network. In [74], we also compared Recurrent Neural Network (RNN) models, trained with a modified Hessian-Free methods, to Deep Neural Networks trained with Hessian-free and pretraining and stochastic gradient descent methods.

Using deep and recurrent neural networks improved recognition performance compared to a single layer neural network. The Recurrent Neural Network, in particular, which included temporal parameters, seemed naturally suited to the task as it learned sequences. Training such models, however, is prohibitively difficult, and led to alternate approaches for including temporal modeling. This approach is described in Chapter 4.

2.3 Mathematical Setup

In automatic speech recognition, we are given a domain-specified task that we need to solve; for instance, we may want a system that transcribes a conversation in a meeting room, where everyone has their own lapel microphone. We collect a corpus of speech in meetings, transcribe it by hand to obtain the ground truth word sequence, and try to build models

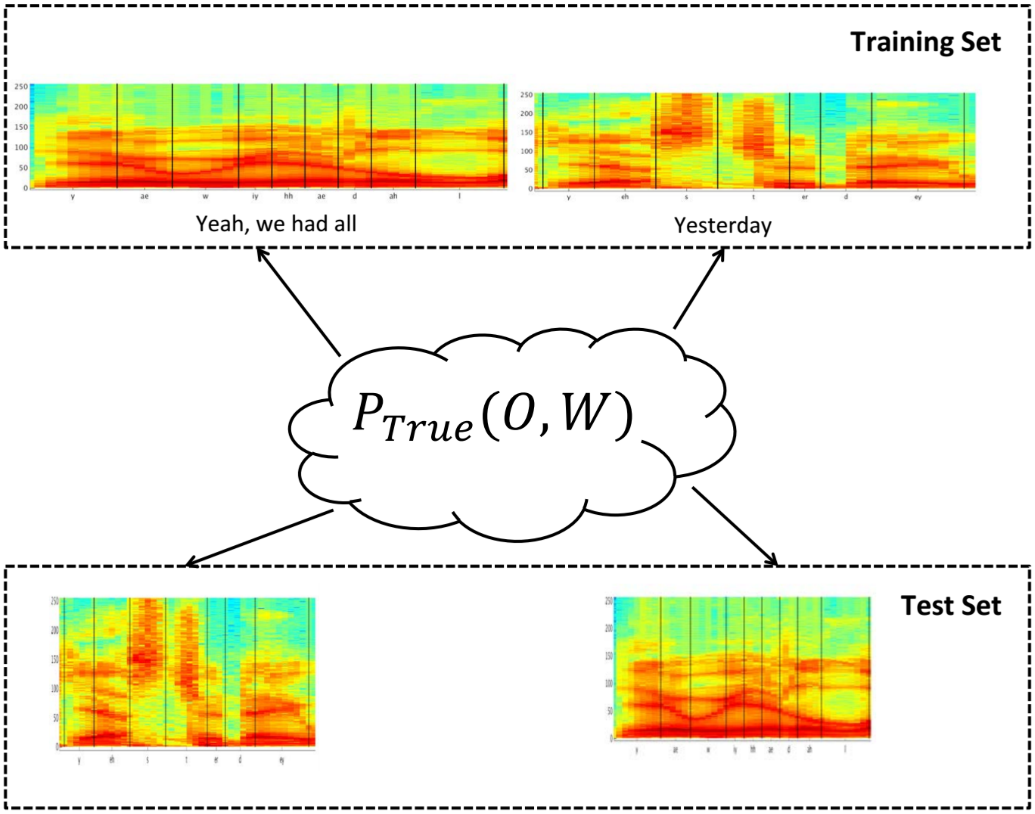


Figure 2.7: *Illustration of implied mathematical setup for speech recognition.*

that are able to perform this transcription automatically. The corpus we collect is used to train the parameters of the model, and assuming test data is similar to the training data, we expect the system to "perform well." Performing well depends on choice of metric, but in most situations, models are trained to (hopefully) minimize the word error rate:

$$WER(\hat{W}, W) = \frac{substitutions + deletions + insertions}{|W|}$$

Implicit in this setup is that speech, which consists of acoustic observations O and words W, is itself a stochastic process distributed $O, W \sim P_{true}(O, W)$. Our observations O correspond to features in the previous section, and we model this stochastic process with $P_{model}(O, W)$. We do not know what $P_{true}(O, W)$ is, but instances of the training set can be considered draws from the distribution. Concretely, when we create a training set, we consider each utterance a draw from $P_{true}(O, W)$. Then, we try to find a set of parameters α for the probability model $P_{model}(O, W)$ such that $P_{model}(O, W) \approx P_{true}(O, W)$. Then, when we encounter an unknown observation O we would like to output a word sequence $\hat{W}$ such that it minimizes the word error rate. Mathematically, we would like to select parameters α

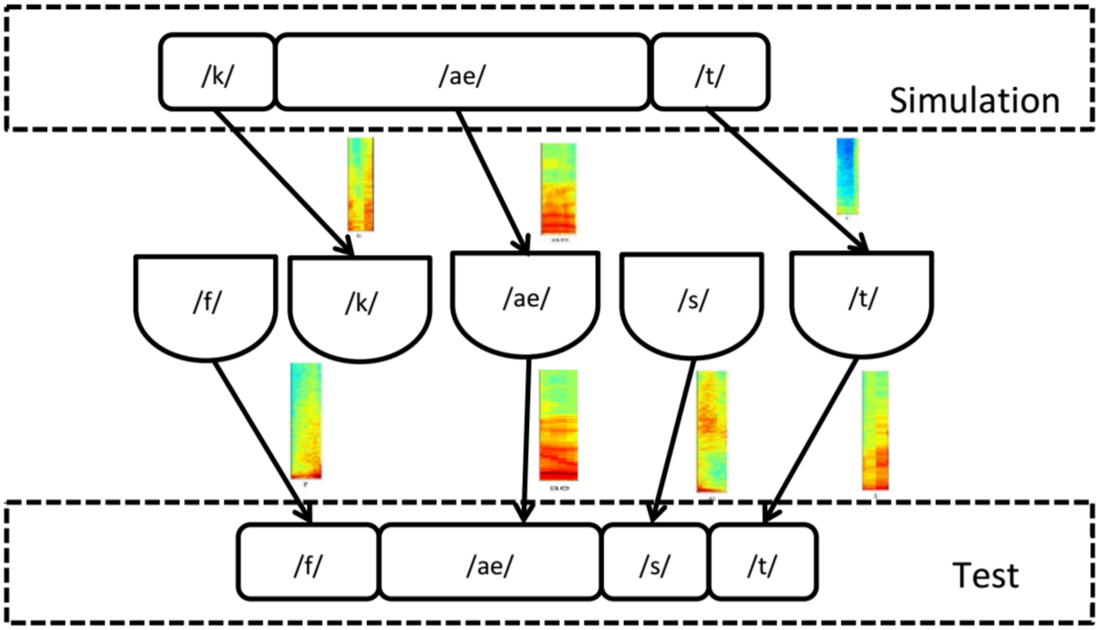


Figure 2.8: *Illustration of the bootstrap resampling framework to generate synthetic test data at the phone level.*

from a model family $\mathcal{A}$ to minimize risk:

$$\min_{\alpha \in \mathcal{A}} \mathbb{E}_{P_{true}(O,W)}[\mathcal{L}(\hat{W}, W)]$$

where $\mathcal{L}(\hat{W}, W)$ is the loss for predicting $\hat{W}$ when the correct word sequence is W. The standard loss is word error rate, but I will study alternative losses in subsequent chapters.

In typical speech recognition decodes, we use Maximum a Posteriori (MAP) decoding for the hypothesized word sequence $\hat{W} = \text{argmax}_W P_{true}(W|O)$. Supposing $P_{model}(O,W) \approx P_{true}(O,W)$, and we would like to minimize word error rate, MAP decoding is suboptimal, as it minimizes expected sentence instead of word error rate. Instead, we would like to find the hypothesized word sequence $\hat{W}$ which minimizes $\min_{\hat{W}} \mathbb{E}_{P_{true}(O,W)}[\mathcal{L}(\hat{W}, W)|O]$, but if our modeling assumptions were true, then $\min_{\hat{W}} \mathbb{E}_{P_{model}(O,W)}[\mathcal{L}(\hat{W}, W)|O]$ should yield a usable proxy, as mentioned in [21]. Thus, a question we would like to ask is if $P_{model}(O,W) \approx P_{true}(O,W)$, which is the focus of the next section.

2.4 Bootstrap Resampling

Access to $P_{true}(O,W)$ is rather difficult, but if our models do indeed match this distribution, we can generate synthetic data from the model, and see if recognition performance is the same for both real and simulated data. That is, we generate synthetic test data to match $P_{model}(O, W)$, and calculate:

$$\mathbb{E}_{P_{model}(O,W)}[\mathcal{L}(\hat{W}, W)]$$

The assumption is that if $P_{model}(O, W) \approx P_{true}(O, W)$, then error should be approximately equal. That $P_{model}(O, W) \neq P_{true}(O, W)$ is not a particularly surprising finding, but changing the simulation distribution from which we model the data and determining how robust models are to data/model mismatch could yield insights in how to fix our models. One particularly rigid assumption of the HMM model is that consecutive frames are independent given the state. We would like to determine how this conditional independence assumption affects our acoustic model, while still using real observations from the data. Through alignment, we can determine the state label for every frame. Since data can be considered a draw from $P_{true}(O, W)$, we can obtain a sample for $P(O|S = s)$, by accumulating features for which the frame label is $S = s$.

Using a bootstrap methodology [15] allows us to generate synthetic data that use real observation and latent alignment data, but have very specific conditional independence assumptions. The process is best explained by example. Suppose that we want to generate test data that is conditionally independent at the phone level. Using a simulation set, we accumulate all features associated with different phones and place them in separate urns. Then, using the test set alignment, we draw with replacement the phones from the different urns, and the resulting features are conditionally independent at the phone level, but the observations themselves are derived from real data. Figure 2.8 illustrates this process. This methodology was first proposed in [18] to analyze GMM-HMM systems. A more formal description of the data generation can be found in Section 3.2.

In this way, we can vary the level of conditional dependence from the frame level, which matches the conditional independence assumption of the model, to the more realistic sampling levels to determine the model robustness to data/model mismatch. In the next chapter, I study this problem for neural network features and DNN-HMM hybrid acoustic models to determine if depth helps robustness to poor conditional independence assumptions of the model. In the following chapter, I show that while depth does help robustness against conditional independence assumption problems, it is still true that $P_{model}(O, W) \neq P_{true}(O, W)$. This will lead me to propose alternative training criteria.

2.5 Sequence-Discriminative Training

The standard training pipeline of DNN-HMM acoustic models is currently rather complex. First, GMM-HMM models are trained using monophone alignments, and using those parameters as initialization, are re-trained using context-dependent triphones. These GMM-HMM models are used to generate labels for the neural networks, which are trained discriminatively assuming independence across frames. If the conditional independence assumptions were true, then this form of maximum likelihood (ML) training would likely yield optimal results. ML training would give us a model $P_{model}(O, W)$ that satisfies $P_{model}(O, W) = P_{true}(O, W)$, and during decode, the average word error rate can be minimized by the decision rule [3]:

$$\hat{W} = \operatorname*{argmin}_{W'} \sum_{W} WER(W, W') P_{model}(W|O)$$

[21] first implemented this "minimum Bayes Risk" decoding for automatic speech recognition. Despite the theoretical elegance, since $P_{model}(O, W) \neq P_{true}(O, W)$, further training the maximum likelihood trained models with alternative criteria improves recognition performance. These criteria use language model information and train on entire utterances, unlike ML-trained DNNs which are trained frame-wise, and thus are called sequence-discriminative training criteria.

Implicitly, the four most widely used training criteria – Maximum Mutual Information (MMI), boosted MMI, Minimum Phone Error (MPE), and state-level Minimum Bayes Risk (sMBR) – are trying to perform approximate learning on the true risk. There are two issues that make direct modeling difficult. The first is that we do not have access to the true probability distribution, and the second is optimizing directly for loss is a difficult learning problem. To circumvent these issues, extant sequence discriminative training criteria make different approximations to the true risk. In Minimum Phone Error and state-level Minimum Bayes Risk training criteria, the true risk is approximated as follows:

$$\operatorname*{argmin}_{\alpha \in \mathcal{A}} \mathcal{R} \approx \operatorname*{argmax}_{\alpha \in \mathcal{A}} \mathbb{E}_{P_{emp}(O)} \mathbb{E}_{P_{model}(W|O)} [P(\hat{S}, S)]$$

where S are phones for MPE and triphone states for sMBR, and the raw accuracy P is the number of correct units minus the number of insertions, calculated without substitutions or deletions for efficiency purposes. Maximum Mutual Information (MMI) [2] and boosted MMI [47] make somewhat different approximation:

$$\mathbb{E}_{P_{emp}(O,W)}[\log(1 + \sum_{\hat{W} \neq W} \exp(-(bP(\hat{S}, S) + \log \frac{P_{model}(W|O)}{P_{model}(\hat{W}|O)}))]$$

the boosted MMI model is explored more fully in Chapter 5.

Instead of these standard approximations, in Chapter 4, I will propose an alternative method based on large-margin training, more popularly known as Structured Support Vector Machines (SVMs).

2.6 Structured SVMs

2.6.1 Binary SVM

The support vector machine is a linear binary classification algorithm which learns a hyperplane θ that maximizes the distance between it and the positive and negative training examples. Mathematically, the algorithm attempts to maximize the difference $\max_\theta y^* \theta^\intercal \mathbf{h} - \hat{y} \theta^\intercal \mathbf{h}$

– where $\mathbf{h}$ is the input features, y^* the correct label, $\hat{y}$ the incorrect one, and $y^*, \hat{y} \in \{+1, -1\}$ – subject to the constraint that $\|\theta\| = 1$, so as not to create degenerate solutions. $y^*\theta^\intercal\mathbf{h}$ and $\hat{y}\theta^\intercal\mathbf{h}$ can be interpreted as the score for the correct and incorrect label, respectively. SVMs are generally recast into the standard constrained optimization problem:

$$\min_{\theta,\xi\geq 0} \frac{1}{2}||\theta||^2 + C\sum_i \xi_i$$

$$\forall i \text{ s.t. } y_i^*\theta^\intercal\mathbf{h} \geq 1 - \xi_i$$

where ξ_i are slack variables, and the subscript i indexes the training example.

Structured SVMs are extensions of this “large-margin” method for structured prediction.

2.6.2 Model

While there seems not to exist a single definition of “structured prediction” in the literature, various authors proposing models to perform structured prediction ([35, 65, 67]) agree that object being predicted, such as a sequence or tree, is more complex than a multiclass label. Speech recognition, which predicts sequences of words from acoustic observations, certainly satisfies this implied definition.

While structured prediction in general can be difficult, a subclass of problems for which inference and training are simpler are linear families. One such family is the log-linear one, for which the probability of a structure is scored as:

$$P(y|x) = \frac{\exp(\sum_j \theta_j^\intercal \phi_j(\mathbf{h}, \mathbf{y}))}{\sum_{\hat{\mathbf{y}}} \exp(\sum_j \theta_j^\intercal \phi_j(\mathbf{h}, \hat{\mathbf{y}}))} = \frac{\exp(\theta^\intercal \phi(\mathbf{h}, \mathbf{y}))}{\sum_{\hat{\mathbf{y}}} \exp(\theta^\intercal \phi(\mathbf{h}, \hat{\mathbf{y}}))}$$

where $\phi(\mathbf{h}, \mathbf{y}) \in \mathbb{R}^n$ is the feature function, which maps the prediction, which may be of variable size, into a fixed-length vector (and can be thought of as encoding features and structure in the model), and $\theta \in \mathbb{R}^n$ are the parameters of the model. Structured SVMs are the large-margin training alternative to log-linear models. At test time, inference is:

$$\operatorname*{argmax}_{\mathbf{y}} \theta^\intercal \phi(\mathbf{h}, \mathbf{y})$$

The training objective is:

$$\min_{\theta,\xi_i} \frac{1}{2}||\theta||^2 + C\sum_i \xi_i$$

$$\forall i \text{ s.t. } \hat{\mathbf{y}}_\mathbf{i} \neq \mathbf{y}_\mathbf{i}^*, \quad \theta^\intercal(\phi(\mathbf{h}, \mathbf{y}_\mathbf{i}^*) - \phi(\mathbf{h}, \hat{\mathbf{y}}_\mathbf{i})) \geq \mathcal{L}(\hat{y}_i, y_i^*) - \xi_i, \quad \xi_i \geq 0$$

where $\mathbf{y}_\mathbf{i}^*$ and $\hat{\mathbf{y}}_\mathbf{i}$ are the label and prediction for sample i, respectively, $\mathcal{L}(\hat{y}_i, y_i^*)$ is the loss function, and ξ_i is the slack variable. At a high level, the optimization problem tries to maximize the difference between $\theta^\intercal\phi(\mathbf{h}, \mathbf{y}_\mathbf{i}^*)$, the score for the correct output, and $\theta^\intercal\phi(\mathbf{h}, \hat{\mathbf{y}}_\mathbf{i})$, the score for the incorrect one. Although the equation looks slightly strange at first glance, hopefully the following examples will provide intuition for the model, which in reality is quite simple.

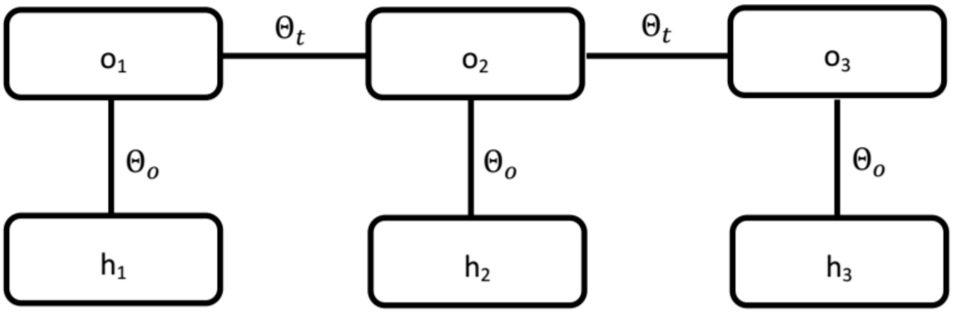


Figure 2.9: *Illustration of Hidden Markov Support Vector Machine for three consecutive frames.*

2.6.2.1 Binary Classification

As a warmup, let us cast the standard SVM as a Structured SVM. Letting θ_B be the parameters for classic binary SVM and h as the feature vector, the Structured SVM form of the binary SVM is:

$$\theta = \begin{bmatrix} \theta_{\mathbf{B}} \\ \theta_{\mathbf{B}} \end{bmatrix} \quad \text{and} \quad \phi(\mathbf{h}, \mathbf{y}) = \begin{bmatrix} \mathbf{h}1_{y_i=1} \\ \mathbf{h}1_{y_i=-1} \end{bmatrix}$$

Inference and training simplify to the binary SVM for $0-1$ loss.

2.6.2.2 Multiway Classification

Consider a standard multiway classification, where $\mathbf{h} \in R^d$ is a d-dimensional input feature, $\Theta \in R^{k\times d}$, where k is the number of classes. Inference is $\text{argmax}_i\, \Theta\mathbf{h}$. In a Structured SVM approach, let us define:

$$\theta = \begin{bmatrix} \theta_{\mathbf{1}} \\ \theta_{\mathbf{2}} \\ \cdots \\ \theta_{\mathbf{k}} \end{bmatrix} \quad \text{and} \quad \phi(\mathbf{h}, \mathbf{y}) = \begin{bmatrix} \mathbf{h}1_{y_i=1} \\ \mathbf{h}1_{y_i=2} \\ \cdots \\ \mathbf{h}1_{y_i=k} \end{bmatrix}$$

Then $\text{argmax}_{\mathbf{y}}\, \theta^\intercal \phi(\mathbf{h}, \mathbf{y}) = \text{argmax}_i\, \Theta\mathbf{h}$, and for $0-1$ loss, training simplifies to the standard one-vs-all SVM.

2.6.2.3 Hidden Markov Support Vector Machine

So far, it is not at all obvious what this type of notation buys us except for more complication. In fact, $\phi(\mathbf{h}, \mathbf{y})$ can encode more structured models through use of indicator functions. As an example, let us create a "Markovian"-like model, in which observations in time are linked. Formally, define N to be the length of an utterance, $\mathbf{h}$ the input features of the entire utterance

$$\mathbf{h} = \begin{bmatrix} \text{—}\mathbf{h_1}\text{—} \\ \text{—}\mathbf{h_2}\text{—} \\ \vdots \\ \text{—}\mathbf{h_N}\text{—} \end{bmatrix}$$

$P \in \{1, \dots, k\}$ the output phone set , $\mathbf{y} \in P^n$ the prediction, and θ the parameters of the Hidden Markov SVM. θ includes Θ_o and Θ_t (shown in Figure 2.9), but the weights are stacked as follows to create a single vector:

$$\theta = \begin{bmatrix} \theta_\mathbf{1} \\ \dots \\ \theta_\mathbf{k} \\ \theta_{11} \\ \theta_{12} \\ \dots \\ \theta_{kk} \end{bmatrix} \text{ where } \Theta_o = \begin{bmatrix} \text{—}\theta_\mathbf{1}\text{—} \\ \text{—}\theta_\mathbf{2}\text{—} \\ \dots \\ \text{—}\theta_\mathbf{k}\text{—} \end{bmatrix} \text{ and } \theta_{ij} = [\Theta_t]_{ij}$$

The feature function for the HMSVM is defined as:

$$\phi(\mathbf{h}, \mathbf{y}) = \begin{bmatrix} \sum_i^N \mathbf{h_i} 1_{y_i=1} \\ \sum_i^N \mathbf{h_i} 1_{y_i=2} \\ \dots \\ \sum_i^N \mathbf{h_i} 1_{y_i=k} \\ \sum_{i=2}^N 1_{y_{i-1}=1, y_i=1} \\ \sum_{i=2}^N 1_{y_{i-1}=1, y_i=2} \\ \dots \\ \sum_{i=2}^N 1_{y_{i-1}=k, y_i=k} \end{bmatrix}$$

Intuitively, $\phi(\mathbf{h}, \mathbf{y})$ can be thought of as a feature function associated with the prediction $\mathbf{y}$, and the feature function "turns on" correct features to interact with the right parts of the model θ, and the prediction $\mathbf{y}$ is obtained by finding the highest scoring prediction among all possibilities. In practice, the HMSVM inference problem is solved using a Viterbi algorithm. $\mathcal{L}(\mathbf{\hat{y}_i}, \mathbf{y_i^*})$ can be a $0-1$ loss on sentences or loss on frames.

2.6.2.4 Log-linear Speech Recognition

As a preview for later parts of the dissertation, one can show that standard ASR recognizers fit this model, provided that the output distribution is in a log-linear form. The decode equation can be be expressed as:

$$
\begin{aligned}
&\operatorname*{argmax}_{W,S} \log p(W) + \log p(S|O) + \log p(S|W) - \log p(S) \\
=&\operatorname*{argmax}_{W,S} \log p(W) + \sum_{i=1}^{N} \log P(S_i|O_i) + \sum_{i=2}^{N} \log p(S_i|S_{i-1}) + \log p(S|W) - \sum_{i=1}^{N} \log p(S_i) \\
=&\operatorname*{argmax}_{W,S} \log p(W) + \sum_{i=1}^{N} \log \frac{\exp(\theta_{S_i}^\intercal h_i)}{Z} + \sum_{i=2}^{N} \log p(S_i|S_{i-1}) - \sum_{i=1}^{N} \log p(S_i) \\
=&\operatorname*{argmax}_{W,S} \log p(W) + \sum_{i=1}^{N} \theta_{S_i}^\intercal h_i + \sum_{i=2}^{N} \theta_{S_i,S_{i-1}} - \sum_{i=1}^{N} \alpha_{S_i}
\end{aligned}
$$

where $P(S_i|O_i)$ is logistic regression on the last hidden layer.

Re-expressing the decode equation in Structured SVM notation:

$$
\theta = \begin{bmatrix} \theta_{\mathbf{1}} \\ \dots \\ \theta_{\mathbf{k}} \\ \theta_{11} \\ \theta_{12} \\ \dots \\ \theta_{kk} \\ \alpha_1 \\ \dots \\ \alpha_k \\ \alpha_{lmsf} \end{bmatrix} \quad \text{and} \quad \phi(\mathbf{h},\mathbf{S},\mathbf{W}) = \begin{bmatrix} \sum_i^N \mathbf{h_i} 1_{y_i=S_1} \\ \sum_i^N \mathbf{h_i} 1_{y_i=S_2} \\ \dots \\ \sum_i^N \mathbf{h_i} 1_{y_i=S_k} \\ \sum_{i=2}^N 1_{y_{i-1}=S_1,y_i=S_1} \\ \sum_{i=2}^N 1_{y_{i-1}=S_1,y_i=S_2} \\ \dots \\ \sum_{i=2}^N 1_{y_{i-1}=S_k,y_i=S_k} \\ -\sum_i^N 1_{y_i=S_1} \\ -\sum_i^N 1_{y_i=S_2} \\ \dots \\ -\sum_i^N 1_{y_i=S_k} \\ \log p(W) \end{bmatrix}
$$

Here, the α_{lmsf} is the language model scaling factor. Unlike the previous examples, exact inference is computationally intractable, as the number of possible candidate decodes is generally higher than available memory. In this case, decoding uses a beam search to limit the number of candidate word hypotheses. The loss here can be word error rate, but can be other types of losses such a frame error rate or phone error rate. The effect of various losses is explored more fully in Chapter 4.

2.6.3 Optimal Parameters

In a classification setting, ideally one would like to optimize θ to minimize test set risk:

$$
\min_{\theta \in \Theta} \mathbb{E}_{P_{true}(O,W)}[\mathcal{L}(W,\hat{W})]
$$

where the loss function $\mathcal{L}(W, \hat{W})$ can be a word-error-like loss. Unfortunately, we do not have access to the $P_{true}(O, W)$. Instead, the standard approximation is to minimize empirical risk:

$$\min_{\theta \in \Theta} \mathbb{E}_{P_{true}(O,W)}[\mathcal{L}(W, \hat{W})] \approx \min_{\theta \in \Theta} \frac{1}{N} \sum_{u} \mathcal{L}(y, \hat{y})$$

where $\mathcal{L}(W, \hat{W})$ is replaced with $\mathcal{L}(y, \hat{y})$, as calculating word error rate may not be ideal loss function for speech recognition (since minimizing word error rate does not allow sharing of error among words, unlike a phone error rate). Unfortunately, there exist two problems with directly minimizing empirical risk: we have no guarantees on test set performance; and 0-$\mathcal{L}(y, \hat{y})$ loss – a generalization of 0-1 loss for binary classification – is difficult to minimize directly. Structured SVMs make two modifications to direct loss minimization to address these two issues. For guarantees on test set performance, if we can keep $\|\theta\|$ small (which is equivalent to maximizing the margin), $\|\phi(\mathbf{h}, \mathbf{y})\|$ is bounded, the number of indicator functions in $\phi(\mathbf{h}, \mathbf{y})$ is finite, and the number of possible decodes is finite[3], there exists generalization result such as [39]. For loss minimization, we use a modified hinge loss, which is a convex upper bound to $0 - \mathcal{L}(y, \hat{y})$ loss. Two standard options for this upper bound are to keep the slope at -1 but move the x-intercept to $\mathcal{L}(y, \hat{y})$, or to keep the x-intercept at 1 and modify the slope to $-\mathcal{L}(y, \hat{y})$. The former is called margin rescaling while the latter slack rescaling. For this work, I focus on margin rescaling, as decoding is more efficient and learning more stable.

[3] This condition is true if utterances have a finite length.

Chapter 3

Analysis of Depth for Tandem Features and Hybrid Systems

3.1 Introduction

The resurgence of neural networks as a research focus for Automatic Speech Recognition (ASR) systems emerges from a growing body of empirical evidence showing that the use of such models improves word recognition performance. Since the work reported in [13], modifications to neural network models have led to a steady drop in recognition error rates, and perhaps unsurprisingly, more research has focused on exploring new models rather than trying to understand what exactly is causing improvements in existing ones.

Despite the rapid evolution in neural network models for ASR – proposed models such as CTC-trained RNNs [37] attempt to replace the now-standard DNN-HMM acoustic model–, one element has remained constant across a variety of systems: a frame-level classification with a neural network using multiple hidden layers. One subclass of models is the so-called "hybrid" system, in which the GMM frame classifier of the GMM-HMM system is replaced with a DNN. Within this subclass, there have notable attempts in understanding how these systems improve recognition. [29] compared DNN-HMM to GMM-HMM systems by comparing DNN models to GMMs on phone error rate, noise robustness, and speaking rate, and concluded that DNNs are likely better frame estimators than GMM. A separate attempt – [68] – measured the ASR performance after each step of MFCC processing. Recently, [43] showed that hidden units of deeper layers encoded more specific phonemic information, while also stripping away seemingly uninformative properties such as gender. For deep Tandem [27] features, there was an early attempt at comparing depth in [73], comparing frame error rates of a three-hidden-layer MLP to one with a single layer and its effect on word error rate in noise-added conditions.

My work adds to this body of literature by trying to understand if and how neural networks modify the statistical properties of the models we use for Automatic Speech Recognition. In particular, given that we assume speech to be a stochastic process with distribution $P_{true}(O,W)$, and we represent this random process with a model with distribution $P_{model}(O,W) \approx P_{model}(O|W)P_{model}(W)$, a natural question to ask is how well do our models match the true distribution. I would like to understand how the model mismatch – i.e., the difference between P_{model} and P_{true} – affects ASR performance. Unfortunately, direct access to $P_{true}(O,W)$ is difficult, so I instead construct synthetic data to match the conditional independence assumptions of our models, and measure performance as we relax those assumptions.

I use the resampling process described in Section 2.4 and [18, 44, 10], which uses simulation and novel sampling process to generate pseudo test data that deviate from the HMM in a controlled fashion. These processes allow one to generate pseudo data that, at one extreme, agree with all of the model's assumptions, and at the another extreme, match the original data. In between, one can precisely control the degree of data/model mismatch. By measuring recognition performance on this pseudo test data, one is able to quantify the effect of this controlled data/model residual on recognition accuracy. The novel sampling process, called resampling, was adapted from Bradley Efron's work on the bootstrap [15, 16]. Segment level resampling creates pseudo test data by randomly sampling (with replacement) labeled—using forced alignment—segments from real test data; the resulting pseudo test data is independent between the segments and inherits whatever dependence present in the segments. In this chapter, I use frame-, state-, and phone-level resampling.

In this work, I explore how Tandem features and depth of the neural networks used to generate those features affect the statistical properties of ASR models. While it may seem a bit dated to explore Tandem features given the widespread use of hybrid systems, the similarities between the two systems – both generate "derived" features based on supervised training on phone-like alignments – may provide insights into both Tandem and hybrid systems. Moreover, "deep" Tandem features have been known to improve recognition performance even in hybrid systems ([69], and such as in low resource settings [50]), so they are worth studying in their own right.

Moreover, I extend this analysis to DNN-HMM hybrid systems to explore what, if any, role depth plays for robustness to conditional independence assumptions in the model. If, as stated in [43], putatively ancillary information such as gender is stripped from deeper hidden layers, one should expect these deeper hidden layers to be less conditionally dependent as data is resampled at more realistic sampling levels. If, on the other hand, depth is merely a better frame classifier, then one should expect the effect of depth to be constant across resampling levels.

One additional issue is that since ASR systems are rather complex, it is not at all obvious how changes to one part of the system affects downstream processing. Even for a basic ASR system, a Deep Neural Network (DNN) or Gaussian Mixture Model (GMM) frame classifier estimates the (scaled-)likelihood of a context-dependent triphone for a particular feature, a temporal model such as a hidden Markov Model (HMM) generates phone sequence estimates from frame likelihoods, a lexicon restricts allowable phone sequences to those consistent with actual words, and a language model provides likelihood estimates for sequences of words. Problems fixed at a feature level may already be fixed later, or may break a hack used in another part of the system. As a result, I perform our analysis with just the HMM as a phone loop without a language model or lexicon constraints, and then for the entire recognition pipeline. For the HMM phone loop, I calculate phoneme error rates and marginal phone duration lengths of the predictions as the test set data moves from matching the conditional independence assumptions of the model to more realistic test data. Then I redo this analysis when I include the language model and lexicon, and also include word error rate results.

In this work, I would like to address five questions: 1) are neural network features more robust to data/model mismatch than more standard frame-level features, 2) do DNN-HMM models better handle data/model mismatch than GMM-HMM ones, 3) does depth provide more robustness to data/model mismatch, 4) how does the choice of feature or classifier affect expected phone duration of predictions using only the HMM phone loop, and 5) does including language model information change the predicted phone duration length? For features, the experiments suggest that neural network features are quite a bit more robust to data/model mismatch than MFCCs, and depth provides additional robustness to state- and phone-level statistical dependence. Moreover, using neural network features help fix poor duration modeling assumptions of the acoustic model. For DNN- and GMM-HMM systems, the results are somewhat more complicated, but perhaps are more interesting. For the HMM phone loop, neural network based systems are better phone recognizers than the GMM-HMM system for all synthetic test data, but it does not necessarily provide more robustness to data/model mismatch. Including a lexicon and language model model improves performance of all systems, but more for GMM-HMM models, such that for state-level resampled data, performance improvements for DNN-HMM systems are much smaller. For data that makes fewer independence assumptions, however, DNNs provide additional robustness to data/model mismatch. Moreover, the optimal depth of neural networks seems to be a function of amount of dependence in the data: for state- and phone-level data, shallower networks perform better, while deeper networks are better for more realistic data. Finally, the language model fixes also phone duration modeling, while using neural networks in both the feature and model setting provides this benefit without needing to include lexicon and language model constraints.

3.2 Synthetic Data Generation

An underlying assumption of generative models such as the GMM-HMM acoustic model or the "ersatz"-ly generative DMM-HMM[1] one is that the data are generated according to the model distribution. While very few researchers believe that these models accurately represent $P_{true}(O|W)$, the hope is that these models are not so misspecified that recognition performance greatly suffers. Ideally, one would like to check if $P_{model}(O|W)P_{model}(W) \approx P_{true}(O,W)$, but since direct measurement of the divergence between these distributions is difficult, I instead create synthetic data from $P_{synthetic}(O,W)$, which has specified assump-

[1] since the output distribution is a scaled rather than true likelihood

tions about how the data are generated, and check to what extent recognition performance improves with these assumptions.

The assumptions studied in this chapter are the conditional independence assumptions of the HMM phone model, which are given the current state, consecutive frames are independent. A generative model that matches these independence assumptions draws a sequence of frame labels F from a distribution $P(F)$ and observations O_t from $P(O_t|F_t)$, and the complete model is $P(F)\prod_{t=1}^{n} P(O_t|F_t)$. Then, one can relax these independence assumptions and track how performance degrades. Since the acoustic model predicts state sequences, one natural relaxation would be to generate synthetic data that is independent at the state level; i.e., the state sequence $S \sim P(S)$ and $O_t \sim P(O_t|S_t)$. O_t are now the n consecutive frames associated with state $S_t = s_t$. One can similarly generate data independent at the phone , and it is reasonable to study these latter two types of synthetic data as recognition systems transduce state sequences to phone, and then word sequences. The process which generates the four aforementioned types of data samples $U \sim P(U)$ and $P(O_t|U_t)$.

One question is how to specify the distributions $P(U)$ and $P(O_t|U_t)$. Ideally, one would like to draw from $P_{true}(U)$ and $P_{true}(O_t|U_t)$, but since these distributions are not readily available, I substitute the empirical distribution for the true distribution. For $P(U)$, one can consider U obtained from the alignments of the test data to be draws from $P_{true}(U)$. Similarly, one can interpret features associated with unit u to be draws from $P_{true}(O_t|U_t = u)$. The bootstrap resampling framework [15, 16] used in this work accumulates all features associated with unit u into urn B_u. One usually populates the urn using features from a simulation set independent of the training and test sets, though I follow [44] and use the test set as the simulation set as this substitution leads to little change in recognition performance as sampling from the test set uses the same speakers. Then, the synthetic data are generated by first drawing a unit sequence U from the test set alignment, and for each unit $U_t = u_t$, drawing an observation with replacement from bin B_{u_t}.

3.3 Experimental Setup

3.3.1 Data and Modeling

I use the spontaneous meeting portion of the ICSI meeting corpus [30], recorded with near-field microphones. The training set consists of 23,739 utterances – 20.4 hours – of speech across 26 speakers[2]. The training set is based on meeting data used for adaptation in the SRI-ICSI meeting recognizer [8]. I use a disjoint 20 hour set from the ICSI meeting corpus as a test set for phone loop recognition with neural network features, as phone loop recognition is quite fast. Since decoding using the full recognition system is much slower, I use a test set comprising 58 minutes of speech, taken from ICSI meetings portions of the NIST Rich Transcription Evaluation Sets 2002 [55], 2004 [53], and 2005 [54]. I also use this test set for phone loop recognition with DNN- and GMM-HMM acoustic models, as the high

[2] I follow the training and test set splits used in [44, 10].

dimensionality of resampled features precludes the use of the 20 hour test set. Resampling for this latter test set is performed 5 times to estimate the variance in sampling.

3.3.2 Tandem

I use HTK version 3.4 for MFCC calculation, acoustic modeling, and decoding. The mel-cepstra are standard 13-dimensional features, including energy, with first and second derivatives, and the MFCCs are mean-normalized at the utterance level. I use HDecode with a wide beam search (300) for decoding. To evaluate recognition accuracy, the reference and the decoded utterances are text normalized before the NIST tool sclite is used to obtain word error rate (WER).

The acoustic models use cross-word triphones and are estimated using maximum likelihood. Each triphone is a three-state linear HMM with no skipping, except for the silence phone. The output distribution is a single Gaussian, since I am not necessarily interested in the best results but merely those for analysis. Maximum likelihood training roughly follows the HTK tutorial: monophone models are estimated from a "flat start", duplicated to form triphone models, clustered to 2,500 states and re-estimated. Previous work [44, 52, 10] use this HTK setup.

I use a trigram language model (LM) [8] that was trained at SRI by interpolating a number of source LMs; these consist of webtext and the transcripts of the following corpora: Switchboard, meetings (CMU, ICSI, and NIST), Fisher, Hub4-LM96, and TDT4. The language model is renormalized after removing words not present in the training dictionary. The perplexity of this meeting room LM is around 70 on our test set. To be compatible with the SRI LM, I use the SRI pronunciation dictionary, which includes two extra phones compared to the CMU phone set – "puh" and "pum" – to model hesitations.

In this chapter, I compare MFCC to Tandem features. 9 Frames of MFCC features serve as input to the neural network, which is trained using TNet [34]. The number of hidden layers for this study range from 1-4, and I found severely degraded performance using more than 4 hidden layers. Each layer consists of 1,500 hidden units, as this produced the best results in initial experiments, and each hidden unit uses a sigmoid non-linearity. The networks are pretrained [28] before cross-entropy training. The labels are 42 phone targets, generated from alignments using a GMM-HMM baseline system with 2,500 states and 8 Gaussians per mixture. Training converged for all neural networks after 13-15 epochs.

3.3.3 DNN-HMM Hybrid Systems

The experimental setup for the DNN-HMM hybrid systems mirrors that for Tandem features in that we use resampled data to determine at which level hybrid systems compensate for poor conditional independence assumptions. Since classifiers, rather than features, are the object of study, one must take care that the baselines are sufficiently strong to make valid comparisons. There are two ways in which the GMM systems must be strengthened so that they reflect a good baseline system. The first is that since the neural network supplants the Gaussian Mixture Model classifier, using merely a single Gaussian and comparing those results to those of a neural network classifier would yield the incorrect conclusion that GMMs are strictly inferior. Thus, the GMM baseline systems used in this work are mixtures of 16 Gaussians, as those yield the best results on this corpus. The second is that better features may reduce the gap between GMM and DNN performance, and stronger baselines are a more accurate reflection on relative performance. For instance, raw time signals such as the ones used in [68], [56], and [22] may work well for certain types of neural network models, but will not work well for GMMs. For both DNNs and GMMs, however, LDA features are superior to MFCCs, in which 9 frames of concatenated MFCCs are projected down to a 39-dimensional space using linear discriminant analysis, and so I use these features for this analysis.

I created a new Kaldi [48] setup for this analysis, adapted from the Switchboard system. GMM-HMM systems are trained using 2,500 states and 40k Gaussians. Models are initially trained on MFCC features with first and second derivatives. Then the GMM-HMM system is retrained using LDA+MLLT features, and use 9 frames of context, akin to the Switchboard setup. Finally, speaker-adaptive training (SAT) is performed using per-conversation-side feature-space maximum likelihood linear regression (fMLLR) transforms, which is referred to as LDA+MLLT+SAT. Both MLLT and fMLLR are used on conversation sides with no access to ground truth speakers.

Alignments from the GMM-HMM systems and the LDA+MLLT+SAT system are used to train the DNN models, using a neural network with 2,048 hidden units per layer, with the number of hidden layers varying from 1 to 6. Using more than 6 hidden layers results in significantly worse results, and thus are not included in the experiment. Restricted Boltzmann Machine (RBM) pretraining [28] is performed until the final hidden layer, with each hidden layer using a sigmoid nonlinearity. Then the neural networks are cross-entropy trained using alignments from the GMM-HMM systems, which converged after 12-15 epochs depending on the layer.

3.3.4 Metrics and Alignments

Our study tracks three metrics: phone error rate, word error rate, and phone duration. The reference phone lengths depend on alignments, which are generated using the same alignments used for training of the neural networks. Since alignments using different features may differ by 10% [60], and using the same alignment for both training and test may bias results in favor of neural network-based systems, I also performed a preliminary study on alignment agreement and calculated frame error using alignments generated from a 8 Gaussian mixture GMM-HMM system using 3 hidden layer MLP features. While I found alignment disagreement to be around 5%, the relative ordering of performance of features did not change, so for this work I report on only reference phone lengths generated from MFCC alignments.

In addition to the above caveats, phone durations are generated from state-level alignments, which are subject to misalignment. In particular, alignments that are unable to locate particular phones will default to the minimum duration of three frames, due to structural constraints of the three-state Bakis phone hidden Markov Models. Figures 3.1 and 3.2 show a large percentage of phones with a duration of 3 frames in frame-level resampled and original test data, but this mode is more likely due to alignment error than another effect.

For resampling experiments, extra care must be taken as lengths of utterances change at the state- and phone-level. After test utterances are regenerated under the sampling framework, for neural network feature experiments, I realign the sampled data using an 8 Gaussian mixture GMM-HMM system with MFCC features and use those alignments as a reference. I also compare against alignments using MLP-based features, but found no significant difference in results. For hybrid systems, reference alignments are generated using a 16 Gaussian mixture GMM-HMM system, since those labels are less likely to bias results against GMM-HMM systems.

3.4 Results

3.4.1 HMM Phone Loop

3.4.1.1 Tandem

Table 3.1 shows the phone error rate for MFCC features and neural network features by depth. When the conditional independence assumptions are matched at the frame level, there is little benefit replacing MFCCs with Tandem features; in fact, MFCC features outperform MLP based features in all but the three-hidden-layer case. Starting with the state-level resampling, however, neural network features significantly outperform MFCC features. While the phone error rate degrades as the data becomes more realistic for all features, MLP-based features degrade less rapidly.

To understand why there is a significant difference between frame- and state-level results between the two types of features, it is instructive to look at Figure 3.1. At the frame-level, the expected duration of MFCC features match the durations from the alignment (including the spurious peak at the minimum phone duration length), but at more realistic sampling levels, the percentage of phones of duration 8 frames or longer is severely underestimated. In contrast, neural network features match the longer phone duration lengths more accurately. In some sense, MLP-based features, which allow for larger context to be used, are actually fixing the poor duration modeling assumptions of phone HMMs.

Among MLP-based classifiers, using two hidden layers instead of one seems to provide some modest robustness to data/model mismatch when moving from frame- to state-level resampling (shown in the bottom portion of Table 3.1). At more realistic sampling levels, however, relative degradation seems to be flat.

3.4.1.2 Hybrid

As shown in Table 3.4, similar to Tandem-based systems, replacing GMMs with neural networks improves recognition performance across all sampling levels. At the frame level, phone error rate results are likely close to measurement error for neural network systems, as tran-

	MFCC	MLP 1HL	MLP 2HL	MLP 3HL	MLP 4HL
frame	15.77	19.74	15.94	13.73	15.88
state	83.83	42.58	33.18	34.34	35.36
phone	86.74	51.60	43.47	42.42	42.11
original	93.10	61.64	59.56	58.39	58.95
frame/state	431%	116%	108%	150%	122%
state/phone	3.47%	21.2%	31.0%	23.5%	19.1%
phone/original	7.33%	19.5%	37.0%	37.6%	40.0%

Table 3.1: *Phone Error Rate for HMM Phone Loop for different types of resampled data (top), and relative degradation among different types of features (bottom).*

script word error rate is likely around 2-5%, and one would expect word errors to propagate to the phone level. Comparing neural networks to GMMs, it actually looks like that neural networks are as, or less, robust to poor conditional independence assumptions compared to GMMs. The relative degradation from frame- to state-resampled and state- to phone-resampled data is higher for neural networks than for GMMs, while the relative degradation from phone-resampled to original data is much lower for neural network classifiers than for GMMs.

Among the neural network systems, it is interesting to note that the "optimal depth" depends on the conditional independence of the test data. For frame-resampled data, a relatively shallow neural network – 3 hidden layers – performs as well as deeper systems. For state-resampled data, using more than 4 hidden layers does not yield better recognition results, while using more than 4 hidden layers for phone-resampled data actually degrades results. For the original test data, the best the best neural network architecture has 5 hidden layers. The results suggest that depth can help compensate for dependency within the data, despite not including more temporal information than shallower networks.

As shown in Figure 3.3, using better features seems to correct for the GMM-HMM phone loop underestimating longer phones; however, the GMM-HMM still estimates more shorter phones than expected. On the other hand, neural networks help fix both problems, and the depth of neural networks has negligible effect on estimated phone duration.

3.4.2 Full Recognition System with Lexicon and Language Model

3.4.2.1 Tandem

As expected, including language model information substantially improves phone error rate results for all features, as shown in Table 3.2, and better phone recognition correlates well – but not perfectly – with word recognition results, shown in Table 3.3. The result is not terribly surprising, as the lexicon restricts allowable phone sequences to correspond to actual words. Moreover, compared to results using only the HMM phone loop, the system is also more robust to conditional independence assumption mismatches across all features, and especially for MFCCs. Figure 3.2 shows one possible cause: the underestimates of phones 8 frames or higher has now vanished. Including a language model seems to fix the poor duration estimates of the HMM phone model.

Even though the LM does fix data/model mismatch problems, MLP features are still more robust than MFCCs, especially at the state level (shown in the bottom part of Table 3.2). Moreover, depth up to three hidden layers seems to improve this type of robustness to statistical dependence at the state level. At other sampling levels, relative degradation seems to be static.

3.4.2.2 Hybrid

Table 3.5 shows phone error rate results for GMM-HMM and DNN-HMM systems. Similar to the results using the HMM phone loop (Table 3.4), DNN-HMM systems outperform GMM-HMM ones across all sampling levels. Including the lexicon and language model information improves results for both types of systems, and for DNN-HMM systems, sometimes obviates the need for deeper networks to achieve better performance. In particular, the phone error rate for frame-resampled data does not decrease for increased depth, while for state-resampled data, the amount of improvement is marginal. For phone-resampled and original data, however, increasing depth still improves results more substantially, suggesting that depth compensates for data dependency. Despite this result, hybrid models do not necessarily provide additional robustness to data/model mismatch compared to GMM-HMM systems.

Word recognition results – Table 3.6 – broadly follow phone recognition results in that DNN-HMM systems outperform GMM-HMM systems across all sampling levels, increasing depth yields no improvement for frame-resampled data, and “optimal” neural networks is deeper as the data/model mismatch becomes more pronounced. Interestingly, unlike the aforementioned phone recognition results, which did not exhibit performance degradation as the models become deeper, for state-resampled, phone-resampled, and original data, the word error rate increases if the number of layers increases. For state-resampled data, the optimal depth is 3 hidden layers, while for phone-resampled and original data, the optimal depth is 4 or 5 hidden layers. Increasing the number of hidden layers beyond these depths results in performance degradation.

3.5 Conclusion

In this chapter, I study how neural network models improve performance of automatic speech recognition systems. I track how neural network features and classifiers improve ASR systems by testing performance on phone classification and phone duration modeling in two settings: using only a HMM phone loop, and in a full recognition system. Moreover, I compare how neural network models cope with data/model mismatch by comparing recog-

	MFCC	MLP 1HL	MLP 2HL	MLP 3HL	MLP 4HL
frame	4.52 (.03)	3.02 (.01)	2.57 (.08)	2.11 (.02)	2.28 (.02)
state	8.37 (.19)	5.02 (.10)	3.80 (.13)	2.47 (.24)	4.20 (.08)
phone	15.8 (.18)	10.1 (.28)	7.00 (.16)	5.03 (.11)	6.52 (.20)
original	32.1	27.6	22.6	21.0	21.6
frame/state	85.2%	66.2%	47.9%	17.1%	84.2%
state/phone	88.8%	101%	84.2%	103%	55.2%
phone/original	103%	173%	223%	317%	231%

Table 3.2: *Phone Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom) for MFCC and Tandem features. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.*

	MFCC	MLP 1HL	MLP 2HL	MLP 3HL	MLP 4HL
frame	1.02 (.11)	0.78 (.05)	0.86 (.05)	0.70 (0.0)	0.80 (0.0)
state	7.30 (.23)	4.46 (.21)	3.40 (.19)	4.48 (.29)	3.92 (.18)
phone	20.6 (.42)	13.6 (.34)	9.52 (.19)	9.48 (.36)	8.82 (.46)
original	44.6	40.0	32.6	31.6	31.8
frame/state	616%	472%	295%	540%	390%
state/phone	182%	205%	180%	111%	125%
phone/original	117%	194%	242%	233%	261%

Table 3.3: *Word Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom) for MFCC and Tandem features. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.*

nition performance on the original test data to that on data resampled to better match the model's conditional independence assumptions. Interestingly, increasing data dependence beyond the frame level requires increased depth in the neural networks for optimal performance. Moreover, the neural network features themselves fix poor phone duration modeling assumptions of the hidden Markov Model, although using features that includes longer term information can also modestly improve phone duration estimation. These poor duration modeling assumptions, however, are already fixed by including the dictionary and language model.

Prima facie, it seems as if duration modeling should be handled by the HMM phone model, or barring that, the acoustic model. That the lexicon and language model, which in and of itself do not explicitly model phone duration, also fix phone durations seems especially

	GMM	NN 1HL	NN 2HL	NN 3HL	NN 4HL	NN 5HL	NN 6HL
frame	3.4 (.01)	2.2 (4E-4)	2.1 (3E-4)	1.8 (8E-5)	1.8 (8E-5)	1.9 (3E-5)	1.8 (2E-4)
state	12.2 (.04)	8.5 (.17)	7.8 (.15)	7.5 (.14)	7.4 (.13)	7.4 (.15)	7.4 (.15)
phone	26.0 (.13)	20.4 (.12)	19.2 (.07)	18.6 (.07)	18.2 (.13)	18.4 (.11)	18.5 (.12)
original	47.1	31.3	29.9	29.1	28.9	28.8	29.1
frame/state	259%	286%	271%	317%	311%	311%	311%
state/phone	113%	140%	146%	148%	146%	149%	150%
phone/original	81.2%	53.4%	55.7%	56.5%	58.8%	56.5%	57.3%

Table 3.4: *Phone Error Rate using HMM Phone Loop for different types of resampled data (top), and relative degradation between different types of features (bottom), for GMM-HMM and DNN-HMM acoustic models. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.*

	GMM	NN	NN	NN	NN	NN	NN
		1HL	2HL	3HL	4HL	5HL	6HL
frame	1.5 (4E-3)	1.0 (7E-4)	1.0 (2E-3)	1.0 (2E-3)	1.0 (9E-4)	1.0 (1E-3)	0.9 (5E-4)
state	3.7 (8E-3)	2.8 (.01)	2.7 (.02)	2.7 (.02)	2.6 (.01)	2.6 (.02)	2.6 (.02)
phone	7.9 (7E-3)	6.5 (.02)	6.3 (.04)	6.1 (.05)	6.0 (.03)	6.0 (.04)	5.9 (.04)
original	18.2	16.7	15.7	15.3	15.1	15.1	14.8
frame/state	147%	180%	170%	170%	160%	160%	189%
state/phone	114%	132%	130%	126%	122%	122%	127%
phone/original	130%	156%	153%	151%	152%	152%	151%

Table 3.5: *Phone Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom), for GMM-HMM and DNN-HMM acoustic models. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.*

	GMM	NN 1HL	NN 2HL	NN 3HL	NN 4HL	NN 5HL	NN 6HL
frame	1.8 (.01)	1.5 (5E-3)	1.5 (0.0)	1.5 (2E-3)	1.5 (2E-3)	1.5 (2E-3)	1.4 (7E-3)
state	6.3 (.06)	6.0 (.05)	5.8 (.03)	5.6 (.04)	5.7 (.04)	5.7 (.06)	5.8 (.04)
phone	11.8 (.02)	10.7 (.07)	10.4 (.01)	10.3 (.02)	10.1 (.03)	10.1 (.01)	10.4 (.03)
original	26.2	24.9	23.4	22.9	22.4	22.4	22.5
frame/state	250%	300%	287%	250%	273%	280%	307%
state/phone	87.3%	78.3%	79.3%	83.9%	77.2%	77.2%	82.5%
phone/original	122%	133%	125%	122%	122%	122%	116%

Table 3.6: *Word Error Rate using full recognition system for different types of resampled data (top), and relative degradation between different types of features (bottom), for GMM-HMM and DNN-HMM acoustic models. Numbers in parentheses refer to standard deviation of error across 5 runs of resampled data.*

troubling. We encounter these problems when we tune recognizers: we scale language model scores to account for, among other things, score mismatch between the acoustic and language models, only to include a separate word insertion penalty, because increasing language model scaling factors now results in hypothesized word sequences with fewer longer words. That neural networks “fix” the phone duration model suggest that other improvements in the model remain.

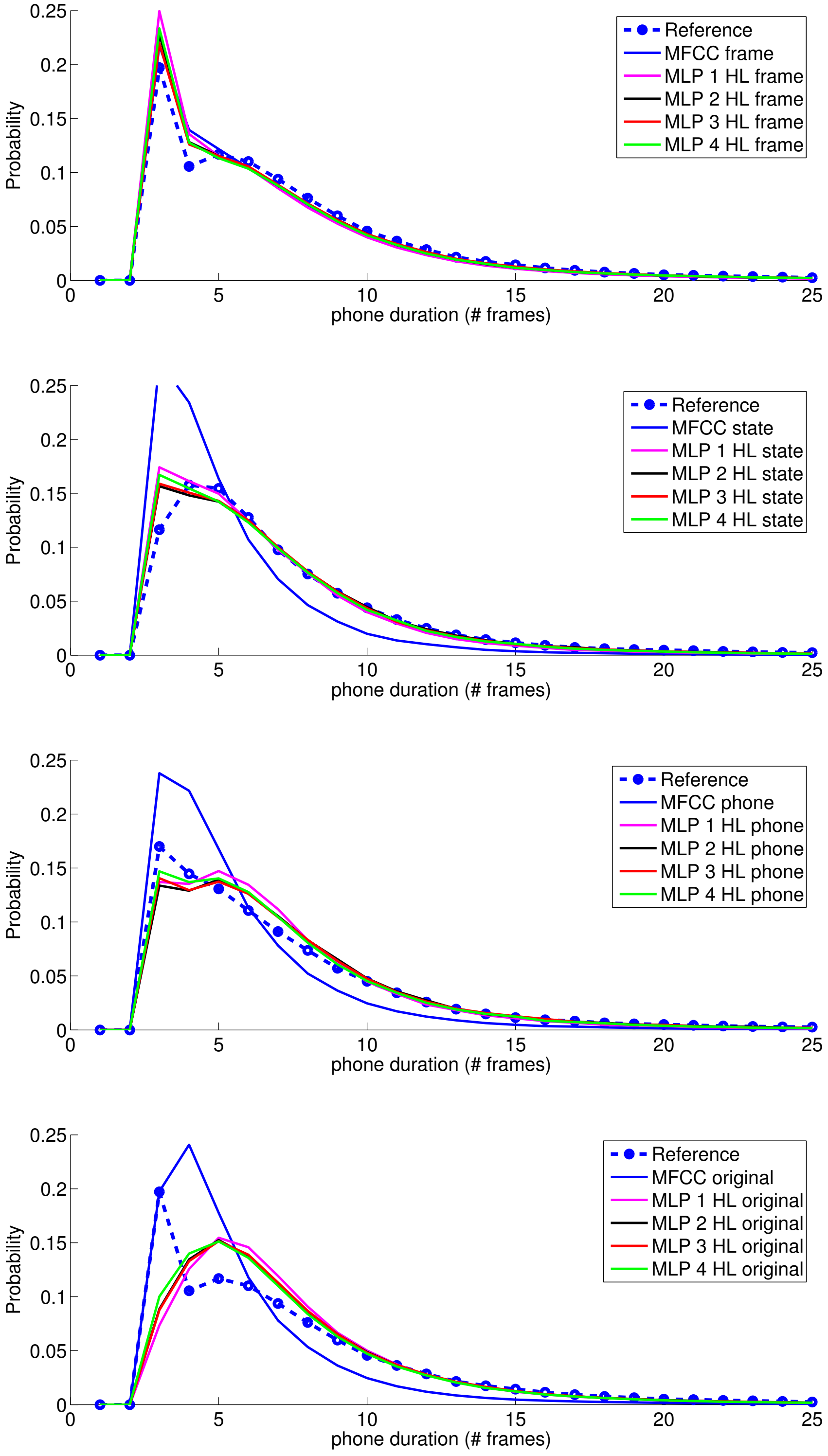


Figure 3.1: *Reference vs. Model Phone Duration Histograms for MFCC and Tandem features and resampling units using an HMM Phone loop.*

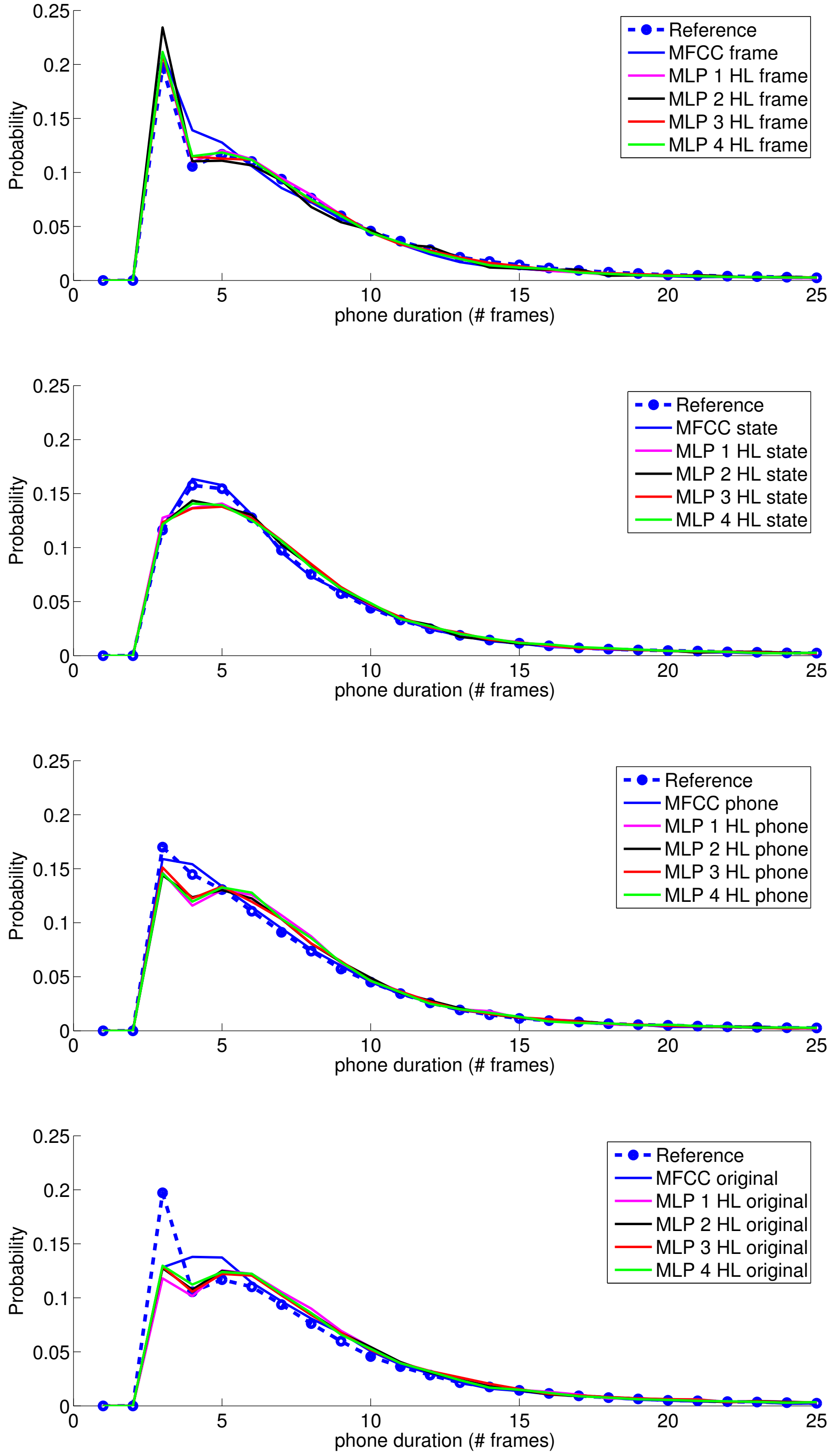


Figure 3.2: *Reference vs. Model Phone Duration Histograms for for MFCC and Tandem features and resampling units using the full recognition system.*

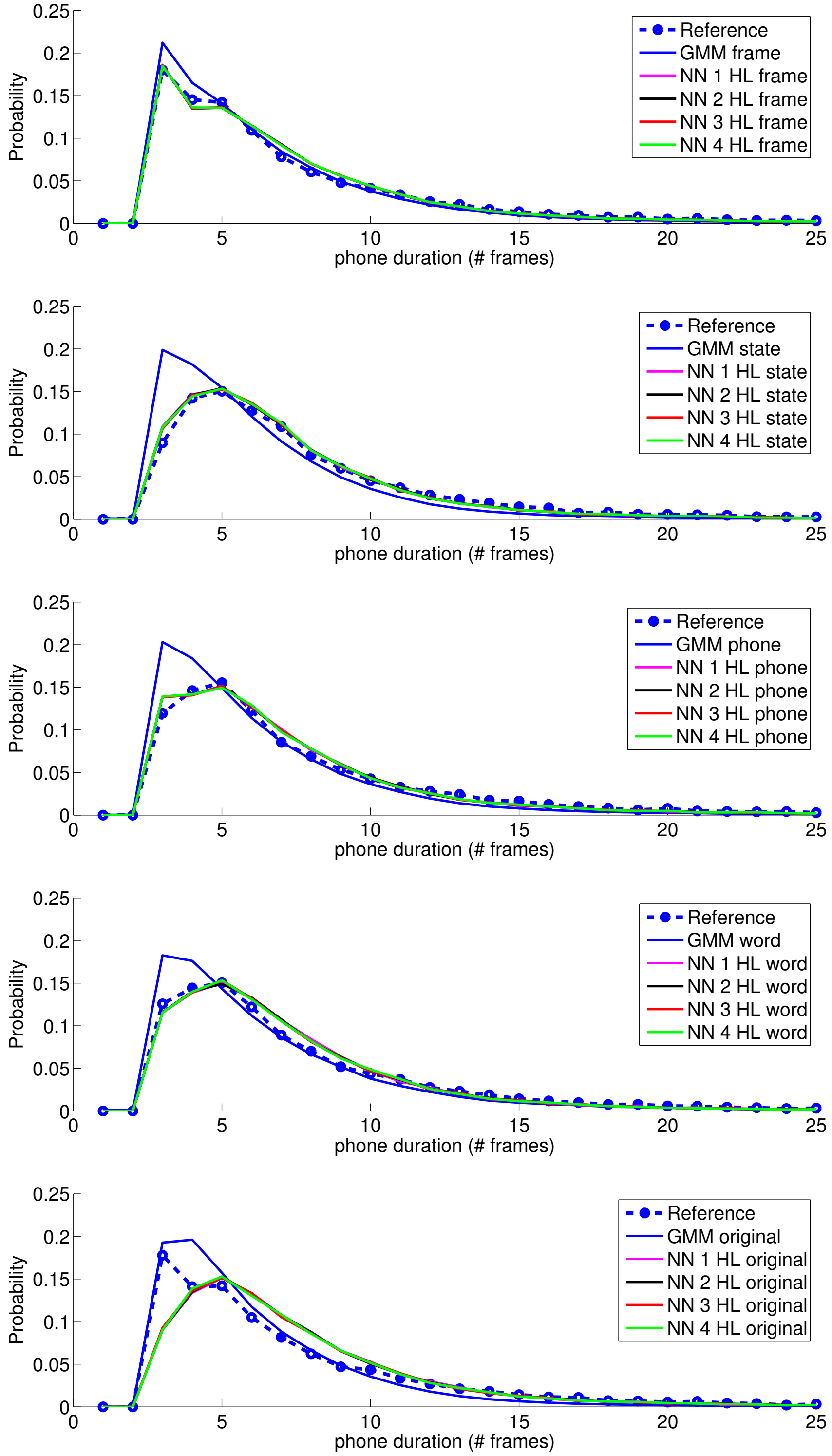


Figure 3.3: *Reference vs. Model Phone Duration Histograms for GMM-HMM and DNN-HMM acoustic models and resampling units using the HMM phone loop.*

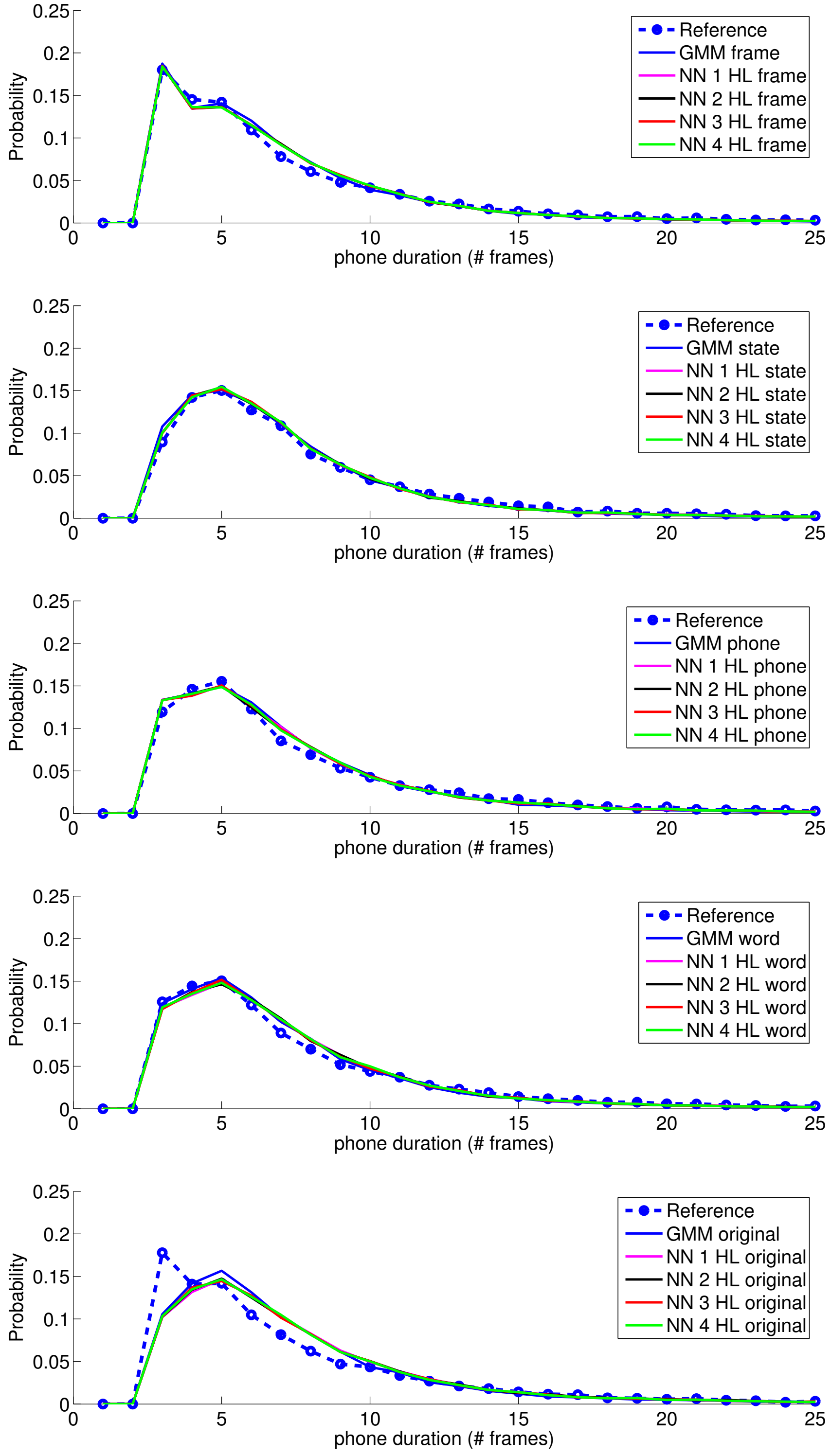


Figure 3.4: *Reference vs. Model Phone Duration Histograms for GMM-HMM and DNN-HMM acoustic models and resampling units using the full recognition system.*

Chapter 4

Structured SVM Extensions of Neural Networks for Feature Extraction and Sequence-Discriminative Training

Preliminary versions of this chapter appeared in [52] and [51].

4.1 Introduction

A typical problem for connectionist speech recognition is to classify the sequence of words uttered from a series of hidden layer vectors. The standard approach is to use a 3-state Bakis HMM with a log-linear output distribution to model the hidden layer vectors. As seen in the previous chapter, however, the conditional independence assumptions made by those models do not well represent an underlying stochastic process of speech. Exploiting the assumption that output distributions are log-linear, I propose extending neural network models to perform structured prediction using Structured SVMs. After introducing a proof-of-concept example – extending neural network features with a type of Structured SVMs to perform sequence prediction – I reinterpret the decoder as a log-linear model, and perform large-margin training. This approach falls under the class of techniques known as sequence-discriminative training, which is typically used to train state-of-the-art automatic speech recognition systems. The proposed method beats four of the most popular methods – Maximum Mutual Information (MMI), boosted MMI (bMMI), Minimum Phone Error (MPE), and state-level Minimum Bayes Risk (sMBR) – on a conversation speech recognition task, while needing far fewer utterances to train. The first half of this chapter comprises the feature extraction work, while the second half focuses on sequence-discriminative training.

4.2 Features

4.2.1 Introduction and Related Work

Using multi-layer perceptrons (MLPs) to model acoustics has had a long history in automatic speech recognition (ASR). One approach that has proven successful, especially for feature combination, is the Tandem method [27]. In the Tandem approach, 9-15 frames of a base feature, such as PLP, MFCC, or more exotic features, are trained with a MLP on a phone recognition task. Processed log posteriors from the MLP are appended to standard features such as MFCCs, which are then used as input to an acoustic model.

While the context frames allow the MLP to model longer temporal regions, a possible problem with this approach is that the MLP does not explicitly model any temporal dynamics. In more traditional acoustic modeling, a number of researchers over the years have tried to extend the MLP model to handle time transitions. The “hybrid” NN-HMM approach of [4, 41, 5] included HMM-style parameters between two consecutive output layers, and the model was trained using maximum likelihood (ML). With the renewed interest in neural networks using a “deep approach,” a number of new discriminative training criteria has been proposed or adapted from GMM-HMM systems: MMI/MPE [76], boosted MMI [70], and state-level minimum Bayes risk [17]. While comparatively less research has focused on Tandem systems, there have been some notable efforts. [74] explored the use of recurrent neural networks to replace the MLP, using second-order Hessian-free optimization for training. [42] and [49] proposed a hybrid system consisting of a linear-chain conditional random field and a multi-layer perceptron, and improve upon the MLP baseline on a phone recognition task.

Much of the difficulty of augmenting an MLP-based systems with time transitions and using a purely discriminative model is that, empirically, the system can easily become overtrained. In the original hybrid NN-HMM acoustic model and its successors trained on discriminative criteria, the probability of the phone given the input feature is divided by the probability of state to create an ersatz generative model. In the hybrid CRF-MLP approach of [49], very clever normalizations were used to combat what the authors call “a low entropy frame output.”

In this work, I propose introducing temporal structure using the framework of Structured SVMs, and in particular the Hidden Markov Support Vector Machine (HMSVM), first introduced in [1]. Figure 4.1 shows the proposed hybrid system. The architecture from the input layer to hidden layer is a multi-layer perceptron, while the parameters from the hidden layer to output layer, and those between output layers, are part of the Structured SVM.

There are two reasons, one theoretical and one empirical, to suggest that such a hybrid discriminatively-trained system may work. The theoretical reason is that generalization error of a support vector machine, with a couple modifications, also hold true for Structured SVMs (see [65] for a example of this bound) provided that the VC-dimension is finite. Bounded inputs provide a finite VC-dimension, and hidden layers based on tanh or sigmoid (as used in this work) non-linearities are bounded. Thus, at first glance, the generalization result may allow us to circumvent the overtraining problem. Empirically, researchers working on

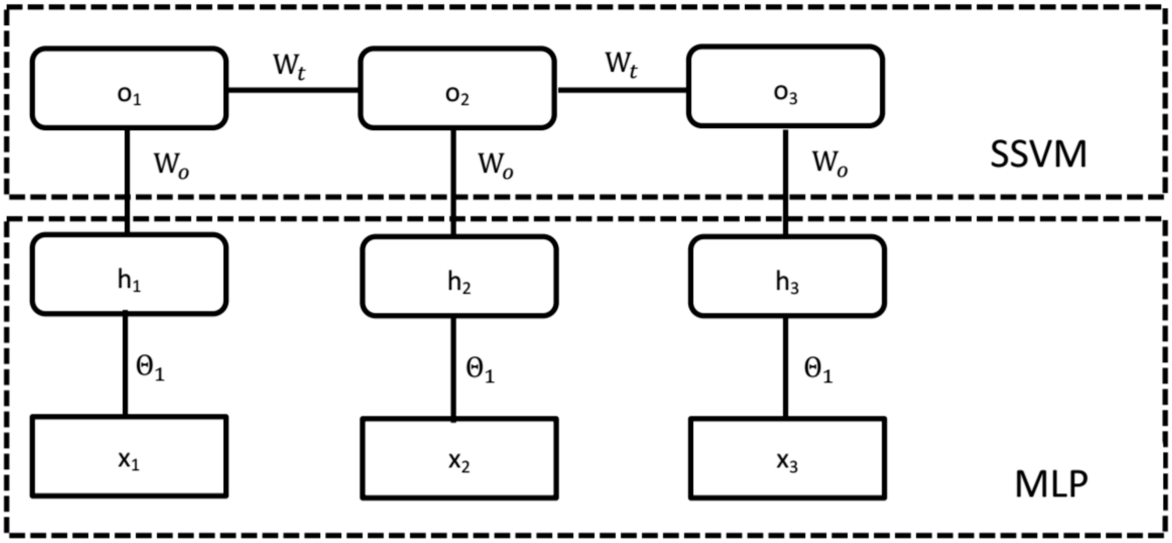


Figure 4.1: *Diagram of the Hybrid MLP-Structured SVM Model for two consecutive frames. The parameters from the input features to the hidden units are those of a standard MLP, while the parameters from the hidden units to outputs, and time transitions, are trained using a Structured SVM.*

other tasks such as part-of-speech tagging ([63]) have noted that Structured SVMs seem to work well when input features are binary. As shown in Figure 4.2, the hidden units of a multi-layer perceptron with sigmoid units serve as an approximation to binary features.

Structured SVMs have been successfully applied to other areas of automatic speech recognition. [81] used Structured SVMs as a type of meta-learning algorithm to improve ASR results; using log-probabilities from competing HMM word hypothesis and a language model as input features, the authors used a Structured SVM to improve inference and optimize segmentations for the Structured SVM. This work was later extended to large-vocabulary tasks in [80] by focusing on sub-word units and adding a parameter for a prior.

In the following sections, I provide an overview of the Structured SVM and the hybrid MLP-Structured SVM Tandem system, and show results on noisy and large-vocabulary corpora.

4.2.2 Structured SVM

4.2.2.1 Model

I use a Hidden Markov Support Vector Machine, first introduced in [1]. Informally, the HMSVM includes the same temporal parameters as a HMM, but no normalization across exiting states is needed. Moreover, the output distribution of the HMM is replaced by a multi-class SVM. More formally, define N to be the length of an utterance, $\mathbf{h}$ the input

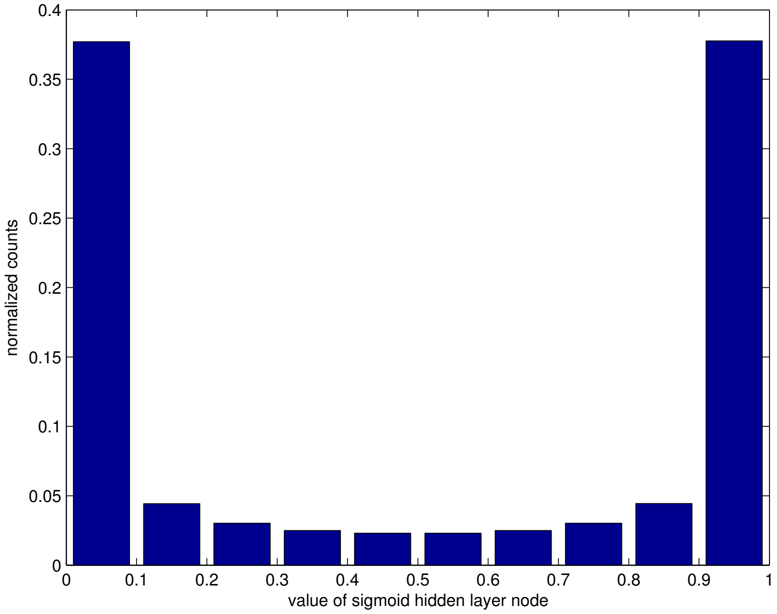


Figure 4.2: *Histogram for activation values for hidden layer nodes.*

features of the entire utterance,

$$\mathbf{h} = \begin{bmatrix} \text{—}\mathbf{h_1}\text{—} \\ \text{—}\mathbf{h_2}\text{—} \\ \vdots \\ \text{—}\mathbf{h_N}\text{—} \end{bmatrix}$$

$\mathcal{P} \in \{1, \ldots, k\}$ the output phone set , $\mathbf{y} \in \mathcal{P}^n$ the prediction, and $\mathbf{w}$ to be the model of the Hidden Markov SVM. $\mathbf{w}$ includes W_o and W_t, but the weights are stacked as follows to create a single vector $\mathbf{w}$:

$$\mathbf{w} = \begin{bmatrix} | \\ \mathbf{w_1} \\ | \\ \vdots \\ | \\ \mathbf{w_k} \\ | \\ w_{11} \\ w_{12} \\ \vdots \\ w_{kk} \end{bmatrix} \quad \text{where} \quad W_o = \begin{bmatrix} \text{—}\mathbf{w_1}\text{—} \\ \text{—}\mathbf{w_2}\text{—} \\ \vdots \\ \text{—}\mathbf{w_k}\text{—} \end{bmatrix} \quad \text{and} \quad [W_t]_{ij} = w_{ij}$$

At test, the best prediction $\mathbf{y}$ is solved as follows:

$$\operatorname*{argmax}_{\mathbf{y}} \mathbf{w}^{\intercal}\phi(\mathbf{h},\mathbf{y})$$

where

$$\phi(\mathbf{h},\mathbf{y}) = \begin{bmatrix} \sum_i^N \mathbf{h_i} 1_{y_i=1} \\ \sum_i^N \mathbf{h_i} 1_{y_i=2} \\ \vdots \\ \sum_i^N \mathbf{h_i} 1_{y_i=k} \\ \sum_{i=2}^N 1_{y_{i-1}=1, y_i=1} \\ \sum_{i=2}^N 1_{y_{i-1}=1, y_i=2} \\ \vdots \\ \sum_{i=2}^N 1_{y_{i-1}=k, y_i=k} \end{bmatrix}$$

Intuitively, $\phi(\mathbf{h},\mathbf{y})$ can be thought of as a feature function associated with the prediction $\mathbf{y}$, and the feature function "turns on" correct features to interact with the right parts of the model $\mathbf{w}$. In practice, the optimization problem is solved using a Viterbi algorithm.

At training time, I maximize the margin between the correct possible phone sequence and all incorrect ones. This leads to the constrained optimization problem:

$$\min_{\mathbf{w},\xi_i} \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i$$
$$\text{s.t. } \forall i \;\; \mathbf{\hat{y}_i} \neq \mathbf{y_i^*} \;\; \mathbf{w}^{\intercal}(\phi(\mathbf{h},\mathbf{y_i^*}) - \phi(\mathbf{h},\mathbf{\hat{y}_i})) \geq 1 - \xi_i, \; \xi_i \geq 0$$

The training objective is similar to the binary SVM, with one modification: there must be a margin between $\mathbf{y_i^*}$ and all incorrect $\mathbf{\hat{y}_i}$.

4.2.2.2 Training

The optimization problem in the previous subsection should give us pause, as the number of possible constraints is exponential in length of sequence. While cutting-plane training algorithms exist for reducing training complexity to linear in data size, optimization becomes increasingly more expensive as the size of dataset, and therefore the number of constraints, increases. Instead, I solve the problem in the unconstrained form. Rewriting the optimization problem and setting $\lambda = \frac{1}{C}$ yields:

$$\min_{\mathbf{w}} \frac{\lambda}{2}||\mathbf{w}||^2 + \sum_i \max(1 - \operatorname*{argmax}_{\mathbf{\hat{y}_i}\neq\mathbf{y_i^*}} \mathbf{w}^{\intercal}(\phi(\mathbf{h},\mathbf{y_i^*}) - \phi(\mathbf{h},\mathbf{\hat{y}_i})), 0)$$

We use an extension of the PEGASOS algorithm [59] for Structured SVMs. PEGASOS is a projected subgradient descent algorithm, and convergence is independent of training set size. While subgradients are extremely efficient to calculate (since for each sequence, the subgradient requires only a Viterbi output), I lose the ability to easily check for convergence. In practice, however, convergence can be monitored by checking performance on a held-out set.

4.2.2.3 Proposed Method

Parameter estimation for a hybrid MLP-Structured SVM System consists of two steps: training a multi-layer perceptron model with $W_t = 0$, and then updating parameters W_o and W_t using a Structured SVM approach. For the first stage, I train a standard multi-layer perceptron using a modified second-order method called Krylov Subspace Descent in [72], as I observed the best results (both for baseline Tandem systems and the hybrid model) using this method. In the second step, I propagate the input features to the hidden layer, and using the hidden layer as inputs, train the Structured SVM using the PEGASOS algorithm.

During inference, I follow the approach in [81], and consider the model to be a conditional random field trained with large margin optimization, to generate frame-level posteriors. I also tried to compare against a more standard MLP-CRF hybrid system, but could not obtain results that matched the baseline Tandem results, likely because of the overtraining issue mentioned in [49]. I perform Karhunen-Loève Transform on the log posteriors per frame, and append those features to standard MFCCs.

4.2.3 Experimental Setup

In this study, I compare 13-dimensional perceptual linear prediction (PLP) features with first and second derivatives discriminatively trained either by a MLP or MLP-Structured SVM, and processed using the Tandem approach. This feature is appended to a 13-dimensional MFCC with first and second derivatives in all the experiments.

I test this hybrid model on Aurora2 [45] and the large-vocabulary section of the ICSI Meeting Corpus [30], to check the model's performance in noisy conditions and for a large-vocabulary task, respectively. The two subsections below provide more detail on the experimental setups.

4.2.3.1 Aurora2

The Aurora2 data set is a connected digit corpus which contains 8,440 sentences of clean training data and 70,070 sentences of clean and noisy test data. The test set comprises 10 different noises (subway, babble, car, exhibition, restaurant, street, airport, train-station, MIRS-filtered subway, and MIRS-filtered street) at 7 different noise levels (clean, 20dB, 15dB, 10dB, 5dB, 0dB, -5dB), totaling 70 different test scenarios, each containing 1,001 sentences. All systems are trained only on the clean training set but tested on the entire test set.

The parameters for the HTK decoder used for this experiment are the same as that for the standard Aurora2 setup described in [45]. The setup uses whole word HMMs with 16 states with a 3-Gaussian mixture with diagonal covariances per state; skips over states are not permitted in this model. This is the setup used in the ETSI standards competition. More details on this setup are available in [45].

4.2.3.2 ICSI Meeting Corpus

For the large vocabulary task, I use the spontaneous meeting portion of the ICSI meeting corpus [30], recorded with near-field microphones. The training set consists of 23,739 utterances – 20.4 hours – of speech across 26 speakers. The training set is based on meeting data used for adaptation in the SRI-ICSI meeting recognizer [61]. The test set comprises 58 minutes of speech, taken from ICSI meetings from the NIST Rich Transcription Evaluation Sets 2002 [55], 2004 [53], and 2005 [54].

I use HTK version 3.4 for MFCC calculation, acoustic modeling, and decoding (ICSI's feacalc is used for PLP features used for Tandem systems). The mel-cepstra are standard 13-dimensional features, including energy, with first and second derivatives, and the MFCCs are mean-normalized at the utterance level. I use HDecode with a wide beam search (300) for decoding. Decoded utterances are text normalized before NIST's sclite tool is used to calculate word error rate (WER).

The acoustic models use cross-word triphones and are trained using maximum likelihood. Each triphone is modeled by a three-state HMM with a discrete linear transition to prevent skipping. The output distribution for each state is modeled by a GMM with 8 components per mixture with diagonal covariance. Training roughly follows the standard recipe, in which monophone models are estimated from a "flat start", duplicated to form triphone models, clustered to 2,500 states, and then re-estimated.

4.2.4 Results

The details of every system reported in this Section are as follows:

- MLP: The MLP is a single hidden-layer multi-layer perceptron with 2,000 hidden units. This number is chosen as it gives the best results on both the Aurora2 and ICSI meeting corpora. The inputs to the MLP are 13-dimensional PLP features with first and second derivatives, and 9 frames of context are used per frame. The multi-layer perceptron is trained with Krylov Subspace Descent, as performance was better than similar networks trained with Hessian-free [38] or stochastic gradient descent. The neural network was trained with 8 sweeps through the data on Aurora2, and 20 on the ICSI Meeting Corpus.

- MLP-Structured SVM: The hybrid structure also uses 2,000 hidden units. The Structured SVM is trained with PEGASOS, extended for use with Structured SVMs. Around 1,500 epochs were used Aurora2 data and about 3,000 epochs were used on the ICSI Meeting Corpus. A batch size of 128 was used (meaning that each epoch used 128 sequences, which constitutes 1.4% and 0.5% of Aurora2 and ICSI Meeting Corpus respectively). A λ of 0.25 was used for Aurora2, and 0.5 for the ICSI Meeting Corpus, although performance was similar for the two values.

4.2.4.1 Frame Recognition

When training both models, I held out around 10% of the training utterances – 800 for Aurora2 and 2,170 for the ICSI Meeting Corpus – to test for convergence of the MLP and MLP-Structured SVM. I get error rates for a frame classification task from the held out data by using our systems. Tables 4.1 and 4.2 show the frame error rate on Aurora2 and the ICSI Meeting Corpus, respectively. As expected, the MLP-Structured SVM model decreases phone error rate over the MLP baseline. For the smaller Aurora2 task, the MLP-Structured SVM has a 22.7% relative improvement over the standard MLP baseline, while for the larger vocabulary task, the relative improvement is a more modest 8.4%.

System	MLP	MLP-SSVM
FER	10.28%	7.94%

Table 4.1: *Frame Error Rate on Cross-Validation Set of Aurora2 for both the multi-layer perceptron (MLP) and MLP-Structured SVM (MLP-SSVM).*

System	MLP	MLP-SSVM
FER	39.91%	36.83%

Table 4.2: *Frame Error Rate on Cross-Validation Set of the ICSI Meeting Corpus for both the multi-layer perceptron (MLP) and MLP-Structured SVM (MLP-SSVM).*

4.2.4.2 Speech Recognition

Typical results on the Aurora2 test set using the ETSI setup report accuracies (or mean accuracy) across the 10 noises at 7 noise conditions. Instead, I report word error rate (WER), as this is the standard metric used for ASR performance, and a reduction in errors typically corresponds fairly well to common costs of using a system (for instance, how often a system must back off to a human operator). Moreover, I average across noises and report scores for each noisy condition, to see how the system degrades as SNR decreases. I also include a "10dB+ avg." that calculates WER across all noises and conditions at SNRs of 10dB and higher. With the exception of the two cleanest conditions, all results are significant with a p-value of 0.02 using the differences of proportions significance test.

Table 4.3 shows the results for the different systems on Aurora2. In almost every condition, except for the noisiest case (−5dB), the hybrid system improves upon an standard MLP

Tandem baseline, trained with second order methods, a difficult baseline.[1] In particular, the MLP-Structured SVM system improves upon the MLP baseline by 7.9% relative. The best relative improvements seem to occur in the cleaner cases, and taper off in more mismatched conditions.

Table 4.4 shows results for the large vocabulary section of the ICSI meeting corpus. Including Tandem features to the standard MFCCs improves performance by 1.3% absolute over the MFCC baseline. Swapping those features with the MLP-Structured SVM improves results by another 1.1%. All results are significant with a p-value of 0.05 using the differences of proportions significance test.

SNR	MFCC	MLP	MLP-SSVM
Clean	0.97%	0.54%	**0.50%**
20dB	5.99%	1.46%	**1.36%**
15dB	15.66%	3.85%	**3.48%**
10dB	36.62%	10.83%	**9.99%**
5dB	64.29%	28.75%	**27.44%**
0dB	84.66%	58.29%	**57.91%**
-5dB	92.21%	**84.20%**	85.18%
10dB+ avg.	14.08%	4.10%	**3.83%**

Table 4.3: *Average WER for several systems under different noise conditions on the Aurora2 corpus. Bold numbers indicate best performance. Note that, as mentioned before, MLP use the Krylov Subspace Descent optimization method.*

System	MFCC	MLP	MLP-SSVM
WER	33.2%	31.9%	**30.8%**

Table 4.4: *WER for several systems on the large vocabulary section of the ICSI meeting corpus. Note that, as mentioned before, MLP use the Krylov Subspace Descent optimization method.*

4.2.5 Conclusions and Future Work

In this work, I propose a hybrid MLP-Structured SVM model, and show how to use a system in a "Tandem" approach. In both noisy and large-vocabulary tasks, the MLP-Structured SVM improves upon a Tandem baseline trained with second-order methods.

[1] The result is among the best Tandem results in our lab, regardless if the MLP architecture is "shallow" or "deep". Please refer to [71] for comparison.

There are a few ways in which the model could be improved. One way could be improving optimization. Currently, the model is trained in two stages: first, as standard MLP; and then as a standard Structured SVM. The reasoning behind splitting optimization into two stages is that performing joint optimization would break the convexity of the Structured SVM and the nice theoretical convergence properties of the Structured SVM training algorithm. On the other hand, there is no reason to believe that the hidden units after MLP training are optimal for a Structured SVM, and perhaps alternating between the two types of training phases could yield better results.

For actual modeling, it is by no means obvious that a HMSVM is the optimal Structured SVM for the hybrid system. One simple extension would be to investigate second- or third-order Markov parameters for improving performance; or perhaps another structure, such as a tree, would improve both phone and word recognition. For the multilayer perceptron, with the current interest in deep architectures, it would be interesting to determine the optimal number of hidden layers for this hybrid approach.

4.3 Sequence-Discriminative Training

4.3.1 Introduction

Statistical speech recognition rests on the assumption that a word sequence W and its associated acoustics O is a stochastic process distributed according to $O, W \sim P_{true}(O, W)$. Modern Automatic Speech Recognition (ASR) systems attempt to model this process with $P_{model}(O, W)$, typically comprising four major components: neural networks for frame-level triphone classification, Hidden Markov Models (HMMs) for state-level sequence classification, a lexicon for phone-to-word transduction, and a language model that estimates the likelihood of word sequences. An application of elementary probability theory allows us to combine these separate models:

$$\begin{aligned}
\hat{W} &= \operatorname*{argmax}_{W} P_{model}(W|O) \\
&\approx \operatorname*{argmax}_{W} P_{model}(O|W) P_{model}(W) \\
&= \operatorname*{argmax}_{W} \sum_{S} P_{model}(O, S|W) P_{model}(W) \\
&= \operatorname*{argmax}_{W} \sum_{S} P_{model}(O|S) P_{model}(S|W) P_{model}(W) \\
&\approx \operatorname*{argmax}_{W,S} P_{model}(O|S) P_{model}(S|W) P_{model}(W) \\
&= \operatorname*{argmax}_{W,S \in \mathcal{S}_W} P_{model}(O|S) P_{model}(W)
\end{aligned}$$

where S denotes the state sequence and $P_{model}(S|W)$ the lexicon.[2] Since, in general, pronunciation dictionaries only define allowable phone sequences without more detailed relative probabilities, the decode equation in the final line involves a search over $\mathcal{S}_W$, the set of states consistent with word sequences.

If $P_{model}(O,W)$ could accurately model $P_{true}(O,W)$, then independent training of each of these separate components, and Minimum Bayes Risk (MBR) decoding [21] would likely yield an accurate recognition. HMM modeling assumptions, however, are rather poor: in addition to the results outlined in the previous chapter, [18] showed that if the data were actually distributed according to the HMM modeling assumptions, word error rates would drop from 30-60% to 1-5% even with weak frame-level classification and suboptimal MAP decoding in GMM-HMM systems (which minimizes expected sentence instead of expected word error rate). Moreover, since MBR decoding presupposes accurate estimates of $P_{true}(W|O)$, it is perhaps not surprising that implementing Minimum Bayes Risk decoding with model posteriors more modestly improves recognition performance than expected.

Many of the standard fixes used to improve word recognition performance, such as raising language model scores by a scaling factor in the exponent, violate the traditional rules of probability while partially fixing poor modeling assumptions. As mentioned by [25], the result is that the full decoding model more resembles a log-linear model. Denoting h_t to be the last hidden layer of a Deep Neural Network (DNN) system (augmented by 1 to accommodate a bias term), α_s as the logistic regression layer weights for state s (augmented by $b_s - \log P(s)$, the bias with the log prior of state s subtracted), the model transition log-probabilities α_{s_{i-1},s_i}, and the language model scaling factor α_{lmsf}, we can more accurately represent the decoding problem as:

$$\begin{aligned}
&\operatorname*{argmax}_{W,S\in\mathcal{S}_W} \log P_{model}(O|S) + \log P_{model}(W)\\
=&\operatorname*{argmax}_{W,S_i\in\mathcal{S}_W} \sum_i (\log P_{model}(O_i|S_i) + \log P_{model}(S_i|S_{i-1})) + \alpha_{lmsf} \log P_{model}(W)\\
=&\operatorname*{argmax}_{W,S\in\mathcal{S}_W} \sum_i (\alpha_{s_i}^{\intercal} h_{t_i} + \alpha_{s_{i-1},s_i}) + \alpha_{lmsf} \log P_{model}(W)
\end{aligned}$$

Even this log-linear model does not accurately model P_{true}, so let us discard the probabilistic interpretation – i.e., I keep the same model parameters as before but eschew the notion that decoding scores are representative of probabilities – and ask a slightly different question: for model parameters α in model family $\mathcal{A}$, what parameters will minimize our true risk

$$\min_{\alpha\in\mathcal{A}} \mathcal{R} \equiv \min_{\alpha\in\mathcal{A}} \mathbb{E}_{P_{true}(W,O)}[\mathcal{L}(\hat{W},W)]$$

where $\mathcal{L}$ is the word error rate. Here risk corresponds to the expected word error from a random test set.

[2] The lexicon defines phone, not state, sequences, but the two are trivially related.

While directly optimizing for risk is likely difficult for a finite training set (though [24] showed theoretically that one could use perceptron-like update to obtain an exact loss gradient on an "infinite" training set), there exist methods which minimize surrogate objectives that also give nice theoretical guarantees. Structured Support Vector Machines (SVMs) provide one such method for linear models, providing theoretical guarantees that true risk is not much higher than training set error (see [39] for a typical proof). Such a guarantee assumes that the input features to the Structured SVM are bounded, which is trivially true for hidden units with sigmoid non-linearities as used in this work.[3]

4.3.2 Related Work

Max-margin methods are not the only way to approach approximate optimization of Bayes risk, and sequence-discriminative training criteria have long attempted to minimize the risk equation through different approximations. The minimum phone error (MPE) [46] and state-level minimum Bayes risk (sMBR) [17, 31] criteria directly try to optimize for the risk through the approximation:

$$\operatorname*{argmin}_{\alpha \in \mathcal{A}} \mathcal{R} \approx \operatorname*{argmax}_{\alpha \in \mathcal{A}} \mathbb{E}_{P_{Emp}(O)} \mathbb{E}_{P_{Model}(W|O)}[\mathcal{P}(\hat{S}, S)]$$

where S are phones for MPE and triphone states for sMBR, and the raw accuracy $\mathcal{P}$ is the number of correct units minus the number of insertions, calculated without substitutions or deletions for efficiency purposes. Maximum Mutual Information (MMI) [2] and boosted MMI [47] make somewhat different approximation:

$$\mathbb{E}_{P_{emp}(O,W)}[\log(1 + \sum_{\hat{W} \neq W} \exp(-(bP(\hat{S}, S) + \log \frac{P_{model}(W|O)}{P_{model}(\hat{W}|O)}))]$$

which substitutes empirical risk for true risk, and a log-loss for true loss. Boosted MMI uses a soft margin, inspired by the work of [58], who applied large margin Gaussian Mixture Models (GMMs) to phoneme recognition. To the best of our knowledge, neither of these approximations have theoretical guarantees on test set error.

There have been some more recent attempts to include Structured SVM criteria – first introduced in [65] and later extended by [67] – into speech recognition: [79] augments the standard ASR model with per-phone acoustic model scaling factors learned through a cutting-plane algorithm, while more recent work on hybrid systems attempts to learn the output and transition model parameters using a frame-based loss [82], showing an improvement over cross-entropy trained neural networks on TIMIT phone recognition. There have also been attempts to incorporate Structured SVM criteria into segmental ASR models: see [64] for a comparison of different segmental models under different loss functions. Finally, [26] directly incorporated margin-terms into MMI and MPE criteria for a hidden CRF extension

[3] One can also make similar claims about rectified linear units, assuming that the norm of each row of the pre-nonlinearity weight matrix W in calculation Wx is bounded.

to GMMs, but were unable to significantly improve upon results of the MPE baseline on a large-vocabulary recognition task.

4.3.3 Latent Structured SVM Hybrid Acoustic Models

To connect speech recognition to Structured SVMs, note that the log-linear speech recognition model can be compactly expressed as:

$$\log p(W|O) = \alpha^\intercal \phi(h, W)$$

where α comprises the model parameters, $\mathbf{h} = (h_n^{(1)}, \dots, h_n^{(t)})$ constitutes the acoustic observations in the form of the sequence of final hidden layer activations, and feature function $\phi(h, W) \in \mathbb{R}^n$ encodes information about the features and the structure of the model.

In the concrete example of a hidden Markov Support Vector Machine (HMSVM) [1], α includes α_k, defining the hyperplane associated with class k, and $\alpha_{i,j}$, parameterizing the state transitions, while $\phi(\mathbf{x}, \mathbf{y})$ effectively defines a hidden Markov model via indicator functions that select the appropriate terms from α:

$$\alpha = \begin{bmatrix} \alpha_{\mathbf{1}} \\ \dots \\ \alpha_{\mathbf{k}} \\ \alpha_{11} \\ \alpha_{12} \\ \dots \\ \alpha_{kk} \end{bmatrix} \qquad \phi(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \sum_i^N x_i 1_{y_i=1} \\ \dots \\ \sum_i^N x_i 1_{y_i=k} \\ \sum_{i=2}^N 1_{y_{i-1}=1, y_i=1} \\ \sum_{i=2}^N 1_{y_{i-1}=1, y_i=2} \\ \dots \\ \sum_{i=2}^N 1_{y_{i-1}=k, y_i=k} \end{bmatrix}$$

Thus, to obtain the decoding score for labeling $\mathbf{y} = (4, 5, 6)$ of input features $\mathbf{x} = (x_1, x_2, x_3)$, one simply performs the dot product [4]:

$$\alpha^\intercal \phi(\mathbf{x}, (4, 5, 6)) = \alpha_4^\intercal x_1 + \alpha_5^\intercal x_2 + \alpha_6^\intercal x_3 + \alpha_{4,5} + \alpha_{5,6}$$

For more examples of this formalism, please see [67].

SVMs (and structured extensions) seek to minimize:

$$\begin{aligned} &\min_{\alpha \in \mathcal{A}} \mathbb{E}_{P_{true}(W,O)}[\mathcal{L}(\hat{W}, W)] \\ \text{where} \quad &\hat{W} = \operatorname*{argmax}_W \alpha^\intercal \phi(h, W) \end{aligned}$$

by making the following three approximations: replacing true with empirical risk $\mathcal{R} \approx \frac{1}{N} \sum_i \mathcal{L}(\hat{W}, W)$, upper-bounding the loss $\mathcal{L}(\hat{W}, W)$ which is not differentiable with respect

[4] Computing $\operatorname{argmax}_{\mathbf{y}} \alpha^\intercal \phi(\mathbf{x}, \mathbf{y})$ additionally requires efficient inference

to α with sub-differentiable hinge loss $[\mathcal{L}(\hat{W}, W) + \alpha^\intercal(\phi(h, \hat{W}_i) - \phi(h, W_i))]_+$[5], and keeping $\|\alpha\|$ small to limit generalization error. This leads to the (margin-rescaled) Structured SVM:

$$\min_{\alpha,\xi} \frac{\lambda}{2}||\alpha||^2 + 1^\intercal \xi$$
$$\text{s.t. } \forall i, \quad \hat{W}_i \neq W_i \quad \alpha^\intercal(\phi(h, W_i) - \phi(h, \hat{W}_i)) \geq \mathcal{L}(W_i, \hat{W}_i) - \xi_i$$

where $W_i, \hat{W}_i$ are the ground-truth and estimated word sequences, respectively. Unfortunately, the current form of the equation cannot be applied to acoustic model training, as updating parameters requires a state-level alignment; resorting to a frame-level loss based on fixed state-level alignment does not provide a good reflection of word error rate.

Instead, I propose an extension to the Structured SVM which includes latent variables to describe the alignment. Note that the optimal alignment l_i^* can be calculated as:

$$l_i^* = \operatorname*{argmax}_{l_i \in \mathcal{W}_i^*} \alpha^\intercal \phi(h, W_i, l_i)$$

where $l_i \in \mathcal{W}_i$ is the set of alignments associated with the reference word sequence W_i. If loss is based on words, then this form is equivalent to the Latent Structural SVM first proposed in [78]. I will also consider other units for loss (denoted below as y_i instead of W_i) which depart from the formalism in that work.

Incorporating latent variables and recasting the constrained optimization as unconstrained:

$$L(\alpha, \theta) = \frac{\lambda}{2}||\alpha||^2 + \sum_i [\max_{\hat{y}_i, \hat{l}_i, \hat{W}_i} (\mathcal{L}(y_i, \hat{y}_i) + \alpha^\intercal \phi(h, \hat{W}_i, \hat{l}_i)) - \max_{l_i \in \mathcal{W}_i} \alpha^\intercal \phi(h, W_i, l_i)]_+$$

Here α consists of parameters of the output layer of the DNN, the language model scaling factor, and transition model parameters. ϕ corresponds to the structure of the log-linear model. In addition, I would also like to update the other parameters of the deep neural network θ using gradient descent. Fig. 4.3 illustrates a simplified model using monophone states. Algorithm 1 shows the full training procedure, employing stochastic sub-gradient descent for optimization.

This hybrid DNN-LSSVM model exhibits two nice properties, one practical and the other more theoretical. The computational advantage is that the sub-gradient is sparse – unlike gradients for other discriminative training criteria. To see this, define the loss-augmented alignment as:

$$\phi(h, \bar{W}_i, \bar{l}_i) = \operatorname*{argmax}_{\hat{y}_i, \hat{l}_i, \hat{W}_i} (\mathcal{L}(y_i, \hat{y}_i) + \alpha^\intercal \phi(h, \hat{W}_i, \hat{l}_i))$$

and $\phi(h, W_i, l_i^*) = \operatorname{argmax}_{l_i \in \mathcal{W}_i} \alpha^\intercal \phi(h, W_i, l_i)$. Then if $\mathcal{L}(y_i^*, y_i) + \alpha^\intercal \phi(h, \bar{W}_i, \bar{l}_i) \geq \alpha^\intercal \phi(h, W_i, l_i^*)$, the subgradient (omitting, for clarity, the L2 penalty on α) is:

$$\nabla_\alpha L = \phi(h, \bar{W}_i, \bar{l}_i) - \phi(h, W_i, l_i^*)$$

[5] $[x]_+ = \max(x, 0)$

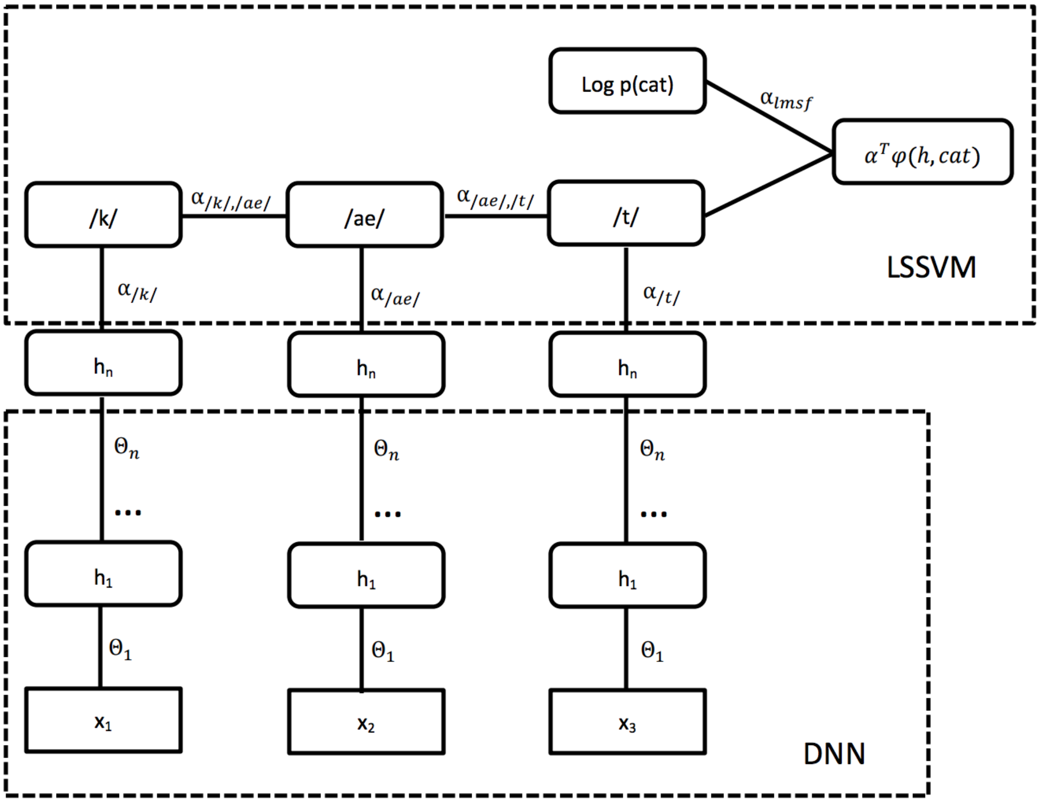


Figure 4.3: *The figure represents the decode score for the word “cat” using monophone states.*

otherwise it is 0.

For a particular frame, the error vector backpropagating through the output and earlier layers of the DNN contains only two non-zero values. Although not pursued in this work, for systems with a large number of context-dependent triphones (such as a recent state-of-the-art recognizer with 32k outputs [57]), exploiting the sparsity of this model could lead to large speed improvements: backpropagation of the error signal through the output layer requires subtraction of $2|H|$ vectors instead of a multiplication of $|O| \times |H|$ matrix with a $|H|$ vector (where $|O|$ and $|H|$ are the number of output and hidden units in the last hidden layer, respectively).

A more theoretical observation is that the boosting parameter b in boosted MMI is equivalent to L2 regularization parameter in the Latent SSVM framework. To see this, note that in the boosted optimization problem:

$$\min_{\alpha,\xi} \frac{\lambda}{2}||\alpha||^2 + 1^\intercal \xi$$

$$\text{s.t. } \forall i, \hat{W}_i \neq W_i \quad \alpha^\intercal(\phi(h, W_i, l_i^*) - \phi(h, \hat{W}_i, \hat{l}_i)) \geq b\mathcal{L}(y_i, \hat{y}_i) - \xi_i$$

dividing the constraints by b and making a transformation of variables $\alpha' = \frac{\alpha}{b}$ and $\xi' = \frac{\xi}{b}$

leads to the equivalent optimization problem:

$$\min_{\alpha',\xi'} \frac{b\lambda}{2}||\alpha'||^2 + 1^\intercal\xi'$$
$$\text{s.t. } \forall i, \hat{W}_i \neq W_i \quad \alpha^\intercal(\phi(h, W_i, l_i^*) - \phi(h, \hat{W}_i, \hat{l}_i)) \geq \mathcal{L}(y_i, \hat{y}_i) - \xi_i'$$

Since I use different regularization parameters for the Structured SVM parameters α and DNN parameters θ, I will use the boosting parameter b for the Latent Structured SVM parameters, and regularization constant λ for DNN parameters.

Algorithm 1 DNN-Latent SSVM training algorithm

```
Split training set $T$ into $K$ batches of size $N$, denoted $T_0 \dots T_k$
Set initial learning rate to $\beta_0$ and learning rate decrease to $\gamma$
for each $i$ in $0 \dots k-1$ do
    $\beta = \gamma^i \beta_0$
    Calculate $l^* = \text{argmax}_{l \in \mathcal{Y}}\, \alpha^\intercal \phi(h, y_i^*, l)$ for each utterance in $T_i$
    Generate N-Best List.
    Calculate $\hat{y}_i, \hat{l}_i = \text{argmax}_{y_i, l_i}\, \mathcal{L}(y_i^*, y_i) + \alpha^\intercal \phi(h, y_i, l_i)$ for each i in batch
    if $\mathcal{L}(y_i^*, y_i) + \alpha^\intercal \phi(h, \hat{y}_i, \hat{l}_i) \geq \alpha^\intercal \phi(h, y_i^*, l_i^*)$ then
        $\nabla_{\alpha^{(t)}} L = \lambda\alpha + \frac{1}{k}\sum_{i=1}^{k}(\phi(h, \hat{y}_i, \hat{l}_i) - \phi(h, y_i^*, l_i^*))$
        $\alpha^{(t+1)} \leftarrow \alpha^{(t)} - \beta \nabla_{\alpha^{(t)}} L$
        $\theta^{(t+1)} \leftarrow \theta^{(t)} - \beta \nabla_{\theta^{(t)}} L$, where $\nabla_{\theta^{(t)}} L$ is the gradient with respect to the neural network parameters
    end if
end for
```

4.3.3.1 Experiments

Given this framework, I would like to study five problems. The first is to determine which units of loss – frame-level, state-level, phone-level, or word-level – yield the best recognition. The latter three losses are measured as the number of substitutions plus deletions plus insertions, and do not need a raw accuracy approximation since I only need one loss-augmented alignment. Our second question is to understand how sensitive the models are to the boosting/regularization parameter. Third, since loss-augmented inference $\max_{\hat{y}, \hat{W}, \hat{l}} \mathcal{L}(y, \hat{y}) + \alpha^\intercal \phi(h, \hat{W}, \hat{l})$ currently uses an N-best list for search, I would like to understand how the size of the N-best list affects recognition performance. Fourth, in initial experiments, I discovered that convergence of this model requires fewer utterances than other sequence-discriminative training criteria, which I wish to quantify. Finally, I would like to evaluate performance on an independent test without extra parameter tuning. Further connections between latent Structured SVM and boosted MMI, and a comparison of how these methods compensate for data/model mismatch, are explored in the next chapter.

4.3.3.2 Connection to boosted MMI

Since the proposed method is not the first margin-inspired one, I would like to connect the SVM criterion to the more familiar boosted MMI. The analysis is similar to [26]. As a setup, define $G(\beta; \mathcal{B}) \equiv \log_\beta \sum_{b \in \mathcal{B}} \beta^b$ for $\beta > 1$. Note that $G(\beta; \mathcal{B}) \geq \max_{b \in \mathcal{B}} b$, is monotonically decreasing for increasing β, and $G(\beta; \mathcal{B}) \to \max_{b \in \mathcal{B}} b$ as $\beta \to \infty$. Also, note that raw phone accuracy is related to its loss by $\mathcal{L}(l^*, l) = |l| - \mathcal{P}(l^*, l)$, where $|l^*|$ is the number of frames. Defining $d\phi(h, l, l^*) \equiv \phi(h, l) - \phi(h, l^*)$, the boosted MMI criterion for one utterance in Structured SVM notation is:

$$\begin{aligned}
&\operatorname*{argmax}_{\alpha \in \mathcal{A}} \log \frac{\exp(\alpha^\intercal \phi(h, l^*))}{\sum_l \exp(\alpha^\intercal \phi(h, l) - b|l^*| + b\mathcal{L}(l^*, l))} \\
&= \operatorname*{argmin}_{\alpha \in \mathcal{A}} \log(1 + \sum_{l \neq l^*} \exp(\alpha^\intercal d\phi(h, l, l^*) + b\mathcal{L}(l^*, l))))
\end{aligned}$$

Changing the bases of the natural logarithm and e to $\log_\beta$ and β, respectively, adding L2-regularization, and taking the limit as $\beta \to \infty$ recovers the SVM criterion.

4.3.4 Experimental Setup

4.3.4.1 Data and Language Model

I use the spontaneous portion of the ICSI meeting corpus [30], recorded with near-field microphones. The training set consists of 23,739 utterances – 20.4 hours – across 26 speakers. The training set is based on meeting data used for adaptation in the SRI-ICSI meeting recognizer [8]. The test set comprises 58 minutes of speech, taken from ICSI meetings portions of the NIST Rich Transcription Evaluation Sets 2002 [55], 2004 [53], and 2005 [54]. Previous work [18, 44, 52, 10] use this setup with an HTK recognizer, as described in [44].

4.3.4.2 Recognition System

I use Kaldi [48] for recognition, using a setup adapted from the Switchboard recipe. GMM-HMM systems are trained using best-performing parameters of 2,500 states and 40k Gaussians. Models are initially trained on MFCC features with first and second derivatives. Then the GMM-HMM system is retrained using LDA+MLLT features, akin to the Switchboard setup. Finally, speaker-adaptive training (SAT) is performed using per-conversation-side feature-space maximum likelihood linear regression (fMLLR) transforms, which I refer to as LDA+MLLT+SAT.

Alignments from the GMM-HMM systems and the LDA+MLLT+SAT system are used to train the DNN models, using a 6-hidden-layer neural network with 2,048 hidden units per layer, as these parameters produced the best results in initial experiments. Restricted Boltzmann Machine (RBM) pretraining [28] is performed until the final hidden layer, with each hidden layer using a sigmoid nonlinearity. Then the DNN is cross-entropy trained using alignments from the GMM-HMM systems, which converged after 15 epochs.

I then update the cross-entropy-trained DNN using four sequence-discriminative training models: MMI, boosted MMI, MPE, and sMBR. Some effort is made to ensure that each baseline sequence-discriminative training system was tuned for optimal performance. Each system converged after 3 epochs, with lattices regenerated after the first epoch of training. Neither more epochs of training, nor more lattice regeneration, produces better results on this corpus. For MMI, and bMMI, frames are dropped according to the standard recipe, and a boosting value of 0.05 yields best results. I also perform some initial experiments with L2-regularization, but this gave no benefit on the sequence-discriminative training systems. I also tuned learning rate, but found the optimal parameter to be the standard 0.00001.

For most experiments, the DNN-Latent Structured SVM system is trained on one sweep through the data, except for convergence and testing on independent test sets which is trained on two sweeps. The L2-regularization parameter on the weights was set to 0.0001. Since alignments in this framework are regenerated after every batch, I find that a much higher learning rate of 0.0002 could be used. Due to the aggressive step size, some utterances with poor alignments cause a temporary high bias to the silence phone: removing alignments which contain 1.5 seconds more silence than the "loss-augmented alignment" fixes this problem. This occurs for fewer than 1% of the utterances. Batch size is set to 512 utterances (after which alignments and N-best lists are generated for the following batch), and learning rate decay is set to 0.98, so that the learning rate at the end of the epoch is roughly half that at the beginning. N-best lists are generated from lattices using a unigram language model, akin to other sequence-discriminative training criteria, and the "loss-augmented alignment" is searched via an N-best list of size 1,000 unless otherwise noted. DNN-Latent Structured SVM training uses initial parameters from the cross-entropy trained DNN-HMM system.

I use a trigram language model (LM) [8] that was trained at SRI by interpolating a number of source LMs; these consist of webtext and the transcripts of the following corpora: Switchboard, meetings (CMU, ICSI, and NIST), Fisher, Hub4-LM96, and TDT4. I renormalize the language model after removing words not present in the training dictionary. The perplexity of this meeting room LM is around 70 on the test set. To be compatible with the SRI LM, I use the SRI pronunciation dictionary, which includes two extra phones compared to the CMU phone set – "puh" and "pum" – to model hesitations.

4.3.5 Results

Table 4.6 shows the the effect of loss and boosting parameters on ASR performance. In nearly every case (except for word loss with boosting parameter 1) the proposed systems beat the other sequence-discriminative training approaches, shown in Table 4.5. In particular, the best frame-level loss based system reduces error by 2.6% absolute compared to a cross-entropy trained baseline system, compared to 1.7% absolute with a state-level MBR trained system. The relative improvements of the system are in line with a comparative study of sequence-discriminative trained systems in [70].

Somewhat surprisingly, frame-level loss seems to outperform other types of loss, albeit by a small margin. Of the remaining loss units, phone-level loss seems to perform the best, although the differences between phone, state, and word level loss are fairly small.

CE	MMI	bMMI	MPE	sMBR
22.7	21.3	21.2	21.1	21.0

Table 4.5: *Word Error Rates for baseline systems. CE refers to cross-entropy, MMI maximum mutual information, bMMI boosted MMI, MPE minimum phone error, and sMBR state-level minimum Bayes risk.*

Loss/Boost	1	3	5	7	9
frame	20.2	20.3	**20.1**	20.4	20.5
state	20.7	20.7	20.6	20.7	20.7
phone	20.7	**20.3**	20.5	20.5	20.5
word	21.2	20.6	20.6	20.6	20.6

Table 4.6: *Effect of loss unit and boosting parameter onthe performance of DNN-Latent Structured SVM systems.* $\lambda = 0.0001$*, size of the N-best list is 1,000.*

Table 4.7 shows the effect of the size of the N-best list for the best-performing of the frame-loss and phone-loss models. The optimal size seems to be about 1,000, although increasing or decreasing the size of the N-best list by a factor of two seems not to make much difference.

N-best size	100	500	1000	2000
frame	20.2	20.6	**20.1**	20.3
phone	20.7	20.5	**20.3**	20.5

Table 4.7: *Effect of N-best list size on word error rate. For the frame model the boosting parameter is 5, while for phone it is 3.*

Table 4.8 shows the effect of updating the transition model in the best model for each type of loss unit. In this case, I do not normalize the probabilities from the outgoing states to sum to 1. This necessitated a change in the weighted FST composition algorithm, as FSTs are composed under a log semiring in the standard recipe, under the assumption that the language and HMM models are roughly probabilities. For updating the time transitions, I instead use a tropical semiring, which generally produces ASR results that are the same or 0.1% worse than graphs produced with a log semiring. In any case, updating the time transitions seems not to have a material effect either way. For loss and phone-level units, the results are the same, while results were slightly better for word and slightly worse for state. It is likely that updating transition parameters does not improve recognition results

Update time transitions?	No	Yes
frame	**20.1**	**20.1**
state	**20.6**	20.9
phone	**20.3**	**20.3**
word	20.6	**20.4**

Table 4.8: *The effect of updating phone model temporal parameters on word error rate. The boosting parameter is 5 for the frame model and 3 for the phone model.*

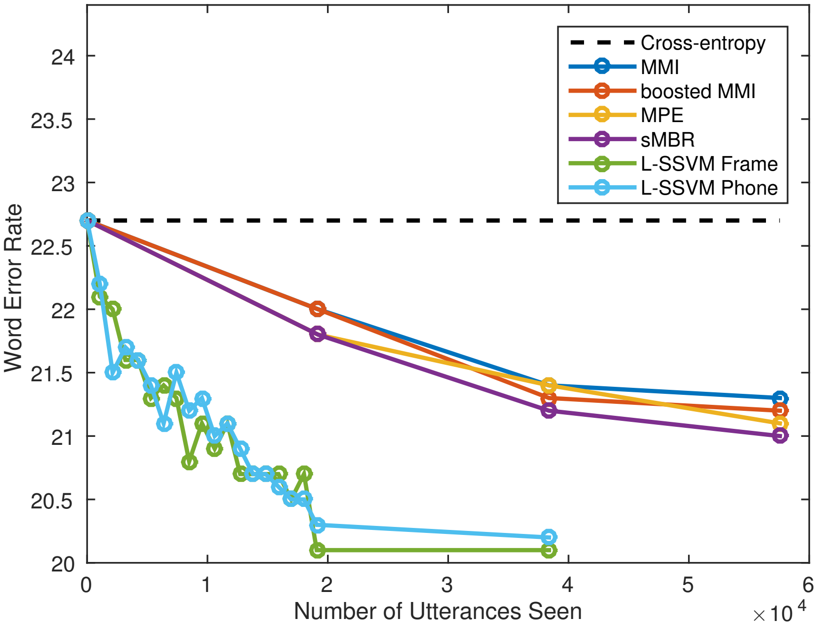


Figure 4.4: *Word Error Rate vs. Number of training utterances seen for different sequence-discriminative training criteria.*

In initial experiments comparing the latent SSVM, which update alignments after every batch, to a regular SSVM whose alignments were updated only after each epoch, I found that the latent SSVM converged to a better model after seeing fewer training utterances. Figure 4.4 compares two LSSVM systems to the standard sequence-discriminative training criteria. I note that the proposed model needs $33-66\%$ fewer utterances to converge, although with an N-best list of size 1,000, processing time per utterance seems to be roughly 50% longer than standard systems.

Finally, given that these models are implicitly tuned on the test set, I want to determine their performance on an independent test set. I compare the best frame-level and phone-level loss models to standard sequence-discriminative systems on the dev and eval portions of the AMI meeting corpus under the individual headset microphone (IHM) condition. Each set consists of roughly 8 hours of speech; more details can be found at [7]. As is shown in Table 4.9, the latent Structured SVM models outperform the sequence-discriminative training criteria, and the results are statistically significant with $p < 0.001$ using the matched pairs sentence segment word error test [20]. The boosted MMI system is not included here as results on the AMI dev and eval sets were not better than those from the cross-entropy model.

	AMI Dev	AMI Eval
CE	37.2	42.6
MMI	36.0	41.3
MPE	35.0	39.8
sMBR	35.0	39.9
LSSVM-frame	34.6	39.1
LSSVM-phone	**34.5**	**38.9**

Table 4.9: $\lambda = 0.0001$*, for frame, boosting parameter is 5, while for phone, it is 3.*

4.3.6 Conclusion

In this work, I propose hybrid DNN-Latent Structured SVM acoustic models. These systems outperform strong sequence-discriminative trained baselines, while often requiring fewer than half the utterances to converge.

Some directions for future research include comparing our method on a larger task to see if both the performance and sample complexity generalize. Initial results using Kaldi's AMI Setup seem to match those on the ICSI meeting corpus, but more work is needed. Second, currently, the "loss-augmented alignment" in the training algorithm requires both lattice generation and an N-best list, the latter of which seems to increase the processing time per utterance roughly 50% compared to that for extant sequence-discriminative training criteria. Future work will include methods for faster search.

Caveats aside, DNN-Latent Structured SVM acoustic models seem to offer a promising alternative to sequence-discriminative training criteria. Moreover, this framework is not specific to the DNN-HMM paradigm, and could be used with other acoustic models such as the LSTM, or another approximately log-linear model, such as [37] or [83].

Chapter 5

Analysis of Sequence Discriminative Training Criteria for DNN-HMM ASR Systems

5.1 Introduction

The previous chapter proposed a sequence-discriminative training criterion for hybrid systems based on large-margin techniques. These systems improve recognition performance compared to standard sequence-discriminative training criteria, and often need 33-66% fewer utterances to converge. Despite these promising results, it is not entirely obvious what differences account for the improved results and convergence speed. In this chapter, I analyze how the proposed LSSVM criteria are able to outperform other methods, and I perform this analysis in two ways. First, since both boosted MMI and latent Structured SVM both minimize upper bounds to empirical risk $\mathbb{E}_{P_{emp}(O,W)}[\mathcal{L}(\hat{y}, y^*)]$, I study how differences in objective function and other parameters, such as the boosting parameter and the frequency of alignment regeneration, affect recognition performance. Moreover, to better understand the connection between the two sequence-discriminative training criteria, I propose a log-sum-exp upper bound to the hinge loss in Structured SVMs and relate this to boosted MMI. This modification allows us to test the differences among the LSSVM, boosted MMI, and the "hybrid" approach which represent the log-sum-exp upper bound, to determine if better boosted MMI performance is possible. It also allows us to answer an open question mentioned in previous work ([26], [70]): why does including a margin term to Maximum Mutual Information criteria provide at most a marginal benefit? [26] noticed that adding a margin term to MMI yielded no positive results, while in [70], adding boosting terms had a 0.0-0.2% improvement on word error rate, depending on the choice of test set. Indeed, in the previous chapter, adding a margin only improved results by a statistically insignificant 0.1%.

Finally, I end the chapter with a broader comparison of sequence-discriminative training criteria for DNN-HMM systems. To perform this analysis, I use the bootstrap resampling framework of Chapter 3 to analyze the depth of neural networks. This analysis was performed once before for Minimum Phone Error of GMM-HMM systems [19], but robustness to conditional independence assumptions are not known for state-level minimum Bayes Risk, the latent SSVM training criteria, the MMI family of criteria, and for DNN-HMM hybrid systems.

5.2 Data and Base Experimental Setup

Since all experiments in this chapter share the experimental setup, I include the details here. As in the previous chapter, I use the spontaneous portion of the ICSI meeting corpus [30], recorded with near-field microphones. The training set consists of 23,739 utterances – 20.4 hours – across 26 speakers. The training set is based on meeting data used for adaptation in the SRI-ICSI meeting recognizer [8]. The test set comprises 58 minutes of speech, taken from ICSI meetings portions of the NIST Rich Transcription Evaluation Sets 2002 [55], 2004 [53], and 2005 [54].

I use the Kaldi setup described in the previous chapter. GMM-HMM systems are trained using best-performing parameters of 2,500 states and 40k Gaussians. Models are initially trained on MFCC features with first and second derivatives. Then the GMM-HMM system is retrained using LDA+MLLT features, akin to the Switchboard setup. Finally, speaker-adaptive training (SAT) was performed using per-speaker feature-space maximum likelihood linear regression (fMLLR) transforms, which I refer to as LDA+MLLT+SAT.

Alignments from the GMM-HMM systems and the LDA+MLLT+SAT system are used to train the DNN models, using a 6-hidden-layer neural network with 2,048 hidden units per layer. Restricted Boltzmann Machine (RBM) pretraining [28] is performed until the final hidden layer, with each hidden layer using a sigmoid nonlinearity. Then DNNs are trained using the cross-entropy objective using alignments from the GMM-HMM systems, which converged after 15 epochs.

5.3 Batch Size

Before studying the connection between boosted MMI and latent Structured SVMs more directly, let us investigate one ancillary property of the LSSVM training algorithm – how frequently labels and lattices[1] are re-estimated – to determine its effect on performance. The reason for studying this seemingly unimportant aspect is that, if smaller batch sizes yield better performance for LSSVM, then perhaps decreasing the batch size will also yield similar improvements for boosted MMI.

In this section, I use the same base experimental setup described in Section 5.2. Using the cross-entropy trained DNNs as initialization, the neural networks are trained with the LSSVM-frame criteria, the boosting parameter is 5, the l2 parameter 0.001, and the initial

[1] which are used to find the loss-augmented alignment

learning rate parameter is 0.0002 for batch sizes under 2,048, and 0.00005 otherwise, as these parameters produce the best results. Learning rate decay is set such that at the end of the epoch, the learning rate was half the initial learning rate. Cross-entropy trained DNNs achieve a word error rate of 22.5%.

5.3.1 Results

Batch Size	WER
256	19.9
512	19.8
1024	20.3
2048	20.7
4096	20.8
8192	20.8
Full	20.6

Table 5.1: *Effect of Batch Size on performance.*

The results, shown in Table 5.1, demonstrates the relatively straightforward result that increasing the number of utterances before regenerating lattices does indeed deteriorate recognition performance. This is interesting from the perspective of existing sequence-discriminative training criteria, since standard methods regenerate lattices after a full epoch. The results here suggest that better results could be obtained by reducing the batch size. I investigate this phenomenon in the next section, as I look at the connection between the latent Structured SVM and boosted MMI.

5.4 Log-Sum-Exp Upper Bound to Hinge Loss

The new proposed criteria, which allows us to connect boosted MMI with the Structured SVM criteria, replaces the hinge loss with a log-sum-exp upper bound. Instead of the standard LSSVM criterion:

$$\min_{\alpha,\xi} \frac{\lambda}{2}||\alpha||^2 + \sum_i \max_j \xi_{ij}$$
$$\text{s.t. } \forall i, j, \quad \hat{W}_{ij} \neq W_i^* \quad \alpha^\intercal(\phi(h, W_i^*, S_i^*) - \phi(h, \hat{W}_{ij}, \hat{S}_{ij})) \geq b\mathcal{L}(y_i, \hat{y}_{ij}) - \xi_{ij}, \ \xi_{ij} \geq 0$$

where ξ_{ij} is the slack variable associated with utterance i and alignment j. The $\max \xi_{ij}$ is upper-bounded by $\log(\sum_j \exp(\xi_{ij}))$, leading to the optimization problem:

$$\min_{\alpha,\xi} \frac{\lambda}{2}||\alpha||^2 + \sum_i \log(\sum_j \exp(\xi_{ij}))$$

$$\text{s.t. } \forall i,j, \quad \hat{W}_{ij} \neq W_i^* \quad \alpha^\intercal(\phi(h, W_i^*, S_i^*) - \phi(h, \hat{W}_{ij}, \hat{S}_{ij})) \geq b\mathcal{L}(y_i, \hat{y}_{ij}) - \xi_{ij}, \ \xi_{ij} \geq 0$$

The associated unconstrained objective is:

$$L(\alpha, \theta) = \frac{\lambda}{2}||\alpha||^2 + \sum_i \log \sum_{\mathcal{M} \cup \{(W_i^*, S_i^*)\}} \exp(b\mathcal{L}(y_i, \hat{y}_i) + \alpha^\intercal \phi(h, \hat{W}_i, \hat{S}_i) - \alpha^\intercal \phi(h, W_i^*, S_i^*))$$

where:

$$\mathcal{M} \equiv \{(\hat{S}, \hat{W}) | \ \ \alpha^\intercal \phi(h, \hat{S}, \hat{W}) + b\mathcal{L}(y^*, \hat{y}) - \alpha^\intercal \phi(h, S^*, W^*) > 0, \hat{W} \neq W^*\}$$

is the set of alignments which violate the margin. The reason for this extra bit of notation will become apparent shortly. The gradient for utterance i has a particular intuitive form (shown without the regularization parameter):

$$\nabla_\alpha L(\alpha, \theta) =$$

$$\sum_{\mathcal{M} \cup \{(W_i^*, S_i^*)\}} \frac{\exp(b\mathcal{L}(y_i, \hat{y}_i) + \alpha^\intercal(\phi(h, \hat{W}_i, \hat{S}_i) - \phi(h, W_i^*, S_i^*)))}{Z} \phi(h, \hat{W}_i, \hat{S}_i) - \phi(h, W_i^*, S_i^*)$$

where Z, the partition function, is the sum of the exponentiated loss-augmented decode scores over all possible state and word sequences. The first term is merely a mixture of margin-violating alignments, weighted proportional to the loss-augmented decode score.

The log-sum-exp upper bound allows us to connect this upper bound to boosted MMI criteria. Although the two procedures do not look particularly similar at first glance, note that:

$$P(O|S)P(W) \equiv \frac{\exp(\alpha^\intercal \phi(h, S, W))}{Z}$$

Then, the objective function associated with boosted MMI is:

$$\begin{aligned}
L_{bMMI} &= -\sum_i \log \frac{P(O_i|S_i)P(W_i)}{\sum_{S_i, W_i} P(O_i|S_i)P(W_i) \exp(b\mathcal{L}(y_i, \hat{y}_i))} \\
&= -\sum_i \log \frac{\exp(\alpha^\intercal \phi(h, S_i^*, W_i^*))}{\sum_{\hat{S}_i, \hat{W}_i} \exp(\alpha^\intercal \phi(h, \hat{S}_i, \hat{W}_i)) + b\mathcal{L}(y_i^*, \hat{y}_i))} \\
&= \sum_i \log(\sum_{\hat{S}_i, \hat{W}_i} \exp(\alpha^\intercal \phi(h, \hat{S}_i, \hat{W}_i) + b\mathcal{L}(y_i^*, \hat{y}_i) - \alpha^\intercal \phi(h, S_i^*, W_i^*)) \\
&= \sum_i \log \sum_{\mathcal{M} \cup \mathcal{N} \cup \{(W_i^*, S_i^*)\}} \exp(b\mathcal{L}(y_i^*, \hat{y}_i) + \alpha^\intercal \phi(h, \hat{W}_i, \hat{S}_i) - \alpha^\intercal \phi(h, W_i^*, S_i^*)) \\
&= \sum_i \log(1 + \sum_{\mathcal{M} \cup \mathcal{N}} \exp(b\mathcal{L}(y_i^*, \hat{y}_i) + \alpha^\intercal \phi(h, \hat{W}_i, \hat{S}_i) - \alpha^\intercal \phi(h, W_i^*, S_i^*)))
\end{aligned}$$

where:

$$\mathcal{N} \equiv \{(\hat{S}, \hat{W})| \;\; \alpha^\intercal \phi(h, \hat{S}, \hat{W}) + b\mathcal{L}(y^*, \hat{y}) - \alpha^\intercal \phi(h, S^*, W^*) < 0, \hat{W} \neq W^*\}$$

is the set of alignments which do not violate the margin.

Thus, one can see the connection between the Structured SVM criteria, the log-sum-exa upper bound, and boosted MMI. The Structured SVM only uses one alignment, the log-sum-exp upper bound all margin-violating alignments, and boosted MMI, all possible alignments. If one were to rank candidate hypotheses by their loss-augmented score, then we can create systems that represent the Structured SVM on one end, boosted MMI on the other, and hybrid systems in between. To make this precise, let $(S_a, W_a) \geq (S_b, W_b)$ if $\alpha^\intercal \phi(h, S_a, W_a) + L(S^*, S_a) - \alpha^\intercal \phi(h, S^*, W^*) \geq \alpha^\intercal \phi(h, S_b, W_b) + L(S^*, S_b) - \alpha^\intercal \phi(h, S^*, W^*)$ (i.e., the score for alignment a is higher than score for b), and $(S_{(n)}, W_{(n)})$ be the n-th best alignment in the candidate alignment set. Then a natural question to ask is how many alignments are needed for optimal performance.

$$\log(1 + \sum_{n=1}^{N} \exp(b\mathcal{L}(y^*, y_{(n)}) + \alpha^\intercal \phi(h, W_{(n)}, S_{(n)}) - \alpha^\intercal \phi(h, W^*, S^*)))$$

Furthermore, should the candidate alignments only include those that violate the margin, or also include those that do not? Finally, I would like to understand how choices in number of candidate alignments to use affects the choice of optimal boosting parameter.

5.4.1 Experimental Setup and Results

To study the above questions, I train a system using the log-sum-exp upper bound criterion using data and initialization of neural networks described in 5.2. Candidate alignments – i.e. summands of the critierion – are taken from an N-Best list of 1,000. Alignments are ranked according to their “loss-augmented“ decode score, and the number of candidate alignments ranges from 1 to 1,000. In addition, boosting parameters are varied from .01 to 10, to determine how performance changes with different parameters. Moreover, I compare the criterion which only use candidate alignments from $\mathcal{M} \cup \{(W_i^*, S_i^*)\}$ to those that use all alignments. The former corresponds to margin-based criteria, while the latter corresponds to boosted MMI. Gradient descent is performed for one iteration through the data. Lattices are regenerated every 512 utterances, since as shown in Section 5.3, this yields the best results for the LSSVM criterion. Losses are calculated on a frame level, as frame-level loss performed among the best of all latent Structured SVM criteria studied in the previous chapter.

It should be noted that for this experimental setup, the tools used are for exploratory purposes only. In practice, one would not naively sum over all possible sequences, but use a forward-backward method for computational efficiency. For this experiment, however, I want to precisely control the number of candidate alignments. Moreover, since I would like to understand the effect of using only margin-violating hypotheses compared to using all

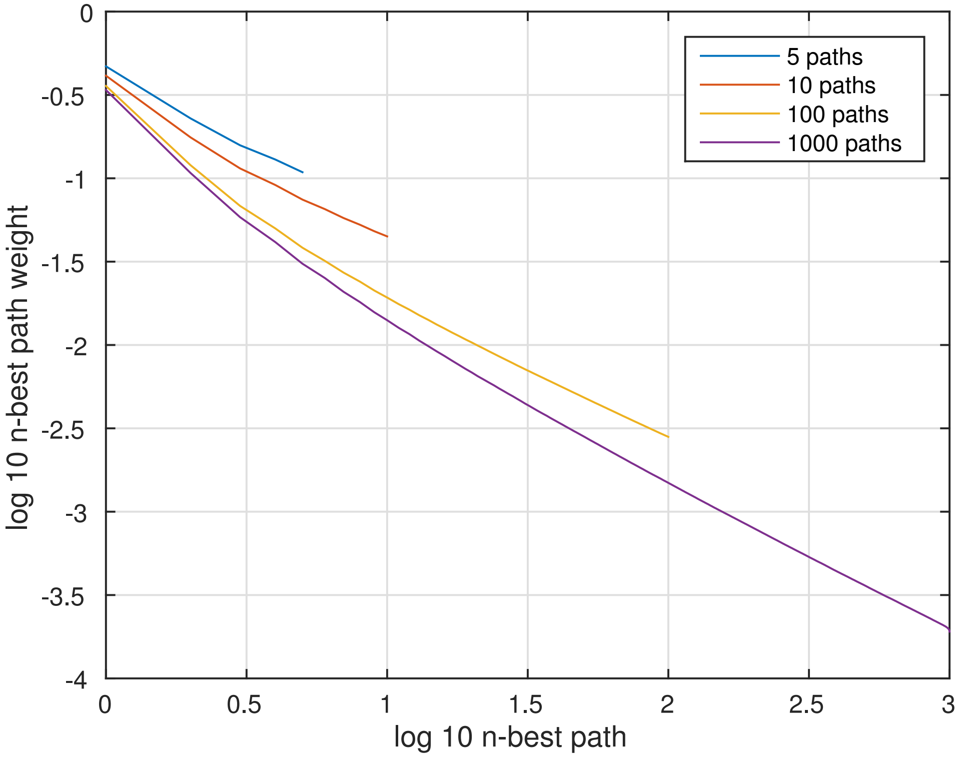


Figure 5.1: *Mixture weight* $\frac{\exp(b\mathcal{L}(y^*,\hat{y}_{(n)})+\alpha^\intercal(\phi(h,\hat{W}_{(n)},\hat{S_{(n)}})-\phi(h,W^*,S^*)))}{Z}$ *of n-th best path* $\phi(h,\hat{W}_{(n)},\hat{S}_{(n)})$ *used for gradient calculation* $\nabla_\alpha L(\alpha,\theta)$. *The x-axis represents the n-th best path on a* $\log_{10}$ *scale, while the y-axis represents the mixture weight on a* $\log_{10}$ *scale.*

alignments, and at present, a computationally efficient method for calculating the sum for only margin-violating alignments is not known, I choose the naive sum method. Performing true boosted MMI through one sweep of the data yielded a word error rate of 22.0%, which is in line with results presented in this section.

Table 5.2 shows the results for different choices of parameters. Using only the 1-best loss-augmented decode corresponds to the Structured SVM criteria, while using 1,000 corresponds to using boosted MMI with a soft margin. In the original works incorporating margin criteria to MMI, authors noted that adding a margin term had a minimal effect. In the current study, indeed adding a margin term does have a minimal effect, and the best boosting parameter is a rather low .01. On the other hand, reducing the number of loss-augmented decodes actually improves results (although the relative improvement is not quite as large as the table suggests – since each is trained for one epoch). Moreover, reducing the number of candidate alignments allows us to increase the boosting parameter: the boosting parameter that yields the best results for a single candidate alignment is 100 times higher than the one for 1,000. As shown in Figure 5.1, using more candidate decodes in the gradient computation decreases

boosting	Number of Sequences				
parameter	1	5	10	100	1000
.01	20.9/20.9	21.2/21.2	21.6/21.8	22.1/22.6	21.8/22.2
.1	20.7/20.7	21.1/21.2	21.6/21.8	22.1/22.6	22.2/22.5
1	20.1/20.1	21.1/21.1	21.6/21.7	22.5/22.6	22.3/22.6
10	20.4/20.4	22.1/21.9	22.5/22.5	22.6/22.5	22.6/22.5

Table 5.2: *Word error rates for log-sum-exp upper bound criteria by boosting parameters and number of sequences used to calculate the gradient. Numbers to the left of the slash are for the objective function summed over margin-violating alignments, while those to the right of the slash are for alignments summed over margin- and non-margin-violating alignments.*

the weight of incorrect word decodes that could be confused with the correct answer, while increasing the weight for those that are not confusable. Moreover, as shown in Figure 5.2, using only candidate alignments which violate the margin produces better recognition results, though the effect is only measureable for systems using at least 10 candidate sequences. It is perhaps these two reasons that suggest the counter-intuitive notion that more information is not better.

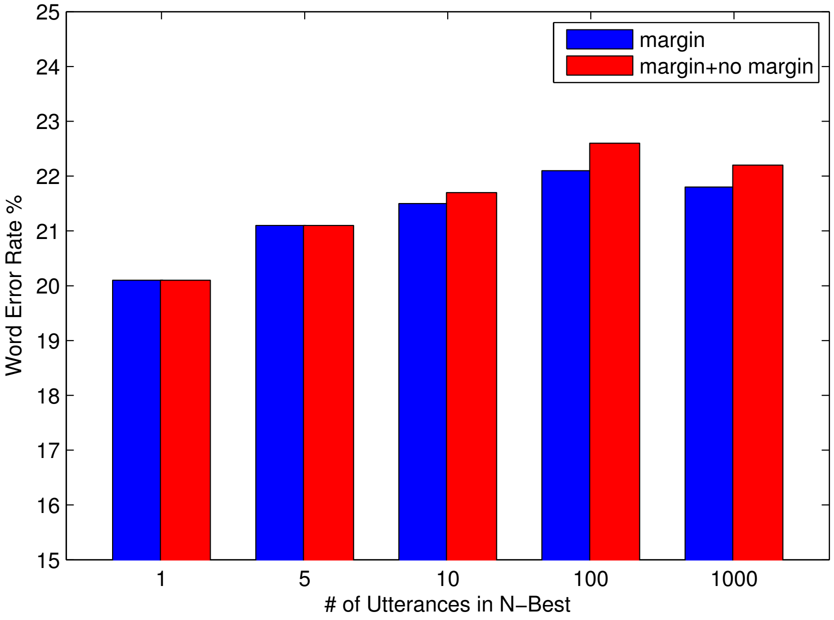


Figure 5.2: *Effect on word error rate of using candidate alignments that violate the margin vs. using all alignments.*

5.5 Bootstrap Resampling Analysis of Sequence Discriminative Training Criteria

The previous section empirically demonstrated that boosted MMI cannot be adapted through more frequent lattice re-generation or higher boosting parameters to match the SSVM training criteria. Also, the previous chapter showed that LSSVM criteria outperforms other methods in the word recognition task. What is not known, however, is exactly how these approaches improve word recognition, and what problems remain to be solved. Moreover, the beginning of this dissertation demonstrated that hybrid models did not match the underlying true distribution of speech, and it would be interesting to understand, for what, if any, poor modeling assumptions do the different families of sequence-discriminative training criteria – MMI, minimum risk, and Structured SVM – actually compensate. There is little previous work in this area, and perhaps a larger analysis can shed light on the different tradeoffs made by these different criteria. [19] performed an initial comparison, comparing maximum likelihood trained acoustic models with models trained with Minimum Phone Error (MPE) for GMM-HMM systems.

5.5.1 Experimental Setup

The setup prior to sequence-discriminative training is described in Section 5.2. In addition to the LSSVM criterion, I compare MMI, boosted MMI, MPE, and sMBR. Each system converged after 3 epochs, with lattices regenerated after the first epoch of training. Neither more epochs of training, nor more lattice regeneration, produces better results on this corpus. For MMI, and bMMI, frames are dropped according to the standard recipe, and a boosting value of 0.05 yields the best results. I also perform some initial experiments with L2-regularization, but this provides no benefit to the baseline sequence-discriminative systems. I also tuned learning rate, but find the optimal parameter to be the standard 0.00001.

The DNN-Latent Structured SVM system is trained on one sweep through the data. The L2-regularization parameter on the weights is set to 0.0001. Since alignments in this framework are regenerated after every batch, I find that a much higher learning rate of 0.0002 could be used. Due to the aggressive step size, some utterances with poor alignments cause a temporary high bias to the silence phone: removing alignments which contain 1.5 seconds more silence than the “loss-augmented alignment” fixes this problem. This occurs for fewer than 1% of the utterances. Batch size is set to 512 utterances (after which alignments and N-best lists were generated for the following batch), and learning rate decay is set to 0.98, so that the learning rate at the end of the epoch is roughly half that at the beginning. N-best lists are generated from lattices using a unigram language model, akin to other sequence-discriminative training criteria, and the “loss-augmented alignment” is searched from a N-best list of size 1,000.

5.5.2 Results

Tables 5.3 and 5.4 show the results for the various systems. Using sequence-discriminative training criteria yields performance equivalent to or worse than cross-entropy trained DNNs on a frame-resampled test set. At different resampling levels, optimal performance is achieved by different type of sequence-discriminative training criteria. Interestingly, at the frame level, the the word error rate of MPE and sMBR is worse than that for MMI-based criterion, while at state and phone levels, the MPE and sMBR criteria are better than competing methods for both phone and word recognition. For the original data, LSSVM criteria outperform other methods in both phone and word recognition.

The LSSVM criteria are interesting in their own right. For simulated data resampled at the frame and state levels, the LSSVM criteria are equivalent to or lag other approaches (including the cross-entropy trained DNNs) in both phone and word error rate, and are comparable at the phone level for word recognition. It is only for the original data that the LSSVM criteria are better. At first glance, this result seems a bit puzzling, but unlike other sequence-discriminative training criteria, LSSVM makes fewer distributional assumptions about the data.

5.6 Conclusion

I conclude the final technical chapter of this dissertation by trying to understand what type of design choices allow for better performance of the proposed Structured SVM training criteria. Much more aggressive alignment and lattice generation allow for better performance for LSSVM criteria, suggesting that existing sequence-discriminative training criteria could be helped by more frequent lattice regeneration. For boosted MMI, however, more frequent lattice generation yields neither better performance nor faster convergence. Moreover, by introducing the log-sum-exp upper bound to hinge loss, I study how performance changes as the criterion changes from one that behaves like the LSSVM to one that behaves more like boosted MMI. Counter-intuitively, using fewer candidate alignments actually improves performance. Finally, I use the bootstrap resampling framework to study how different sequence-discriminative training criteria cope with data/model mismatch. MMI and boosted MMI are best for state- and phone-resampled data, while MPE and sMBR are better for word-resampled data. By contrast, latent Structured SVM training criteria perform poorer when the test data matches the conditional independence assumptions of the model, but are better when the data makes fewer independence assumptions.

	GMM	CE	MMI	bMMI	MPE	sMBR	LSSVM-F	LSSVM-P
frame	1.5 (4E-3)	0.9 (5E-4)	0.9 (9E-4)	0.9 (1E-3)	0.9 (4E-4)	0.9 (9E-4)	1.0 (3E-3)	1.0 (3E-4)
state	3.7 (8E-3)	2.6 (.02)	2.4 (.02)	2.4 (.02)	2.3 (.01)	2.3 (.02)	2.7 (.02)	2.6 (.02)
phone	7.9 (7E-3)	5.9 (.04)	5.8 (.03)	5.8 (.03)	5.6 (.04)	5.6 (.02)	5.9 (.02)	5.9 (.01)
original	18.2	14.8	14.1	14.2	13.9	13.9	13.6	13.7
frame/state	147%	189%	167%	167%	156%	156%	170%	160%
state/phone	114%	127%	142%	142%	143%	143%	119%	127%
phone/original	130%	116%	151%	143%	145%	148%	131%	132%

Table 5.3: *Phone Error Rate at different resampling levels. GMM refers to Gaussian Mixture Model, CE to cross-entropy trained DNN, MMI and bMMI Maximum Mutual Information (MMI) and boosted MMI respectively, MPE Minimum Phone Error, sMBR state-level Minimum Bayes Risk, LSSVM-F Frame-level loss latent Structured SVM, and LSSVM-P phone-level loss latent Structured SVM.*

	GMM	CE	MMI	bMMI	MPE	sMBR	LSSVM-F	LSSVM-P
frame	1.8 (.01)	1.4 (0.007)	1.5 (.003)	1.5 (.003)	1.6 (.003)	1.7 (.003)	1.5 (.003)	1.5 (.003)
state	6.3 (.06)	5.8 (.04)	5.5 (.07)	5.5 (.05)	5.5 (.05)	5.5 (.18)	5.7 (.04)	5.6 (.05)
phone	11.8 (.02)	10.4 (.03)	9.8 (.02)	9.8 (.01)	9.5 (.91)	9.5 (.02)	9.7 (.03)	9.7 (.01)
original	26.2	22.5	21.0	21.0	20.8	20.7	19.8	19.9
frame/state	250%	307%	267%	267%	244%	224%	256%	273%
state/phone	87.3%	82.5%	78.2%	78.2%	72.7%	72.7%	70.2%	73.2%
phone/original	122%	116%	114%	114%	119%	118%	104%	105%

Table 5.4: *Word Error Rate at different resampling levels. GMM refers to Gaussian Mixture Model, CE to cross-entropy trained DNN, MMI and bMMI Maximum Mutual Information (MMI) and boosted MMI respectively, MPE Minimum Phone Error, sMBR state-level Minimum Bayes Risk, LSSVM-F Frame-level loss latent Structured SVM, and LSSVM-P phone-level loss latent Structured SVM.*

Chapter 6

Conclusion

6.1 Contributions and Future Work

As stated in the outset, the focus of this dissertation is two-fold: to better understand how exactly neural networks of varying depth are improving performance in automatic speech recognition systems, and to address those problems which neural networks did not help. The main tool for analyzing neural networks is the bootstrap resampling framework, which resamples test data to match the conditional independence assumptions in the model, and slowly relaxes those assumptions to determine how neural networks compensate for data/model mismatch. For Tandem features, neural networks offer better robustness to mismatch at the state level, and improve estimation of phone duration. For hybrid systems with base features that includes more temporal information, most of the gains occur moving from phone-resampled to original data, and modestly help the phone duration modeling. Despite these improvements, neural networks do not address the main problem: that the models we use to represent $P_{true}(O, W)$ are not very good.

Since these feed-forward neural networks are inherently frame-level classifiers, the output layer is a log-linear model, I investigate performing structured prediction with neural networks trained with large-margin criteria. The goal is to perform better structured prediction for automatic speech recognition while understanding that the structured prediction extensions do not correctly model speech. Thus, the idea is to use large-margin training in order to minimize error for these sub-optimal model families. In preliminary work on features, adding temporal transitions to the output layer of a feedforward MLP and training the model with large-margin criteria improves recognition performance, and circumvents overfitting issues found in purely a conditional maximum likelihood approach. Then, I extend this type of training to perform sequence-discriminative training of the acoustic model. This approach beats existing sequence-discriminative training approaches, which are used in state-of-the-art systems, while only needing 33-66% of the utterances to train.

In Chapter 5, I attempt to analyze the Structured SVM training criterion and compare it to more popular approaches. In particular, I compare the Structured SVM approach to boosted MMI, and show how the latter can be considered an upper bound on loss of the former. I show that, contrary to prior belief, training on less information via the Structured SVM approach can actually improve models. Then, I finally use the bootstrap resampling framework to compare sequence-discriminative training criteria. I show that MMI-based criteria seem to better compensate for incorrect conditional independence assumptions at the frame level (though not compared to cross-entropy-trained models), while MPE and sMBR help correct for poor conditional independence assumptions at the state and phone levels. The Structured SVM, on the other hand, yields the best results on the original data, since it makes fewer distributional assumptions compared to competing criteria.

While Structured SVM extensions of neural networks seem to improve performance of recognition systems, open questions still remain. One is that the current implementation of the training algorithm searches through N-best lists, which for experiments in this dissertation, are calculated from lattices. This process is more complex than it needs to be: it would be better to perform a loss-augmented decode directly than to generate a lattice, create an N-best list from the lattice, and search through that list. Recent work [14] shows how to annotate exact word or phone error with lattices for under 30 seconds of speech, which would obviate the need for N-best lists. Furthermore, it is an open question how large-margin sequence-discriminative training will scale as the number of states increases. The current maximum is 2,500, but for some systems trained on thousands of hours of data, tens of thousands of states are the norm. Moreover, how the Structured SVM approach performs as the amount of data scales is also an open question.

More theoretically, there are two questions that warrant further study. The first is related to the type of loss unit. It seems that a frame-, or depending of test set, phone-based loss yields the best word recognition results. Why this type of loss unit seems best when directly minimizing word error should provide superior results is currently unknown. The issue seems to be related to the multiple levels of hierarchy: words are composed of unknown phonemes, which themselves are composed of unknown alignments. While the latent structure attempts to infer the alignments, it assumes pronunciations from the lexicon. Future work will include multiple levels of hierarchy. Also, training these models from random initialization is an open question; future work will mirror what has been done previously in [33] for maximum a posteriori training of neural networks.

The second question is with the convex upper bound of the $0-n$ loss. For hinge loss, or its log-sum-exp upper bound, large differences between the alignment and the loss-augmented decode yield very large gradients. One potential problem is that poor alignments can lead to large but incorrect weight updates. Limiting these poor updates is a question worth studying. Moreover, perhaps it would be better to perform direct minimization of loss using structured perceptron [12] than a Structured SVM. There has some effort in [24] to perform this type of direct loss minimization, and it would be interesting to determine whether this direct update would yield better recognition performance.

Shifting our focus to analysis of ASR models, the bootstrap resampling framework presupposes hidden Markov Model acoustic models. With the re-introduction of recurrent neural network (RNN) acoustic models, which are not based on Markovian assumptions, an open question is how to generate data that matches the assumptions of the model, and understand how robust these models are to data/model mismatch. In some sense, recurrent models are Markovian at the feature level and may encode longer term information, though how useful this longer term information is is unknown. Neural-based translation systems, which map sentences to a fixed length vector, suggest encoding of longer-term information, and measuring what types of conditional independence assumptions these models fix is an open question [62].

6.2 Beyond

Improvements in recognition results by frame-level modeling with neural networks, and by language modeling with recurrent neural networks (though this component is not yet fully integrated into the decoder) suggest that the traditional components of automatic speech recognition systems may be supplanted in the near future. In fact, certain research directions reflect this belief. Of the many research directions that focus on neural networks for automatic speech recognition, one of the most ambitious attempts to replace the traditional acoustic model which predicts phone sequences with a system that performs orthographic transcription. [23] uses an orthographically trained recurrent neural network acoustic model and an n-gram language model to perform speech recognition. [9] proposes an attention model, which uses three components – a recurrent neural network that maps acoustic observations to a sequence of hidden layer vectors, a neural network "attention" component that learns which subset of hidden layer vectors are associated with which character, and another RNN that predicts from the subset and previous characters the current character – and a language model to perform word recognition. While the prenominate systems use a traditional language model, an even more radical approach uses a recurrent neural network character-based acoustic model and a character-based language model, discarding the n-gram language model entirely [37].

Although it is not entirely clear if acoustics, unlike language, exhibit long-term dependencies (which makes recurrent neural network modeling of acoustics less compelling than that for the language), one interesting question is whether character-based acoustic models can be combined with word-based RNN language models, since the latter has been shown to improve results for more traditional models. Unfortunately, since standard conditional independence assumptions are no longer valid for RNN language models, the search space can become exponential in the number of words. These models, however, are log-linear at the feature level, and it is entirely possible that the proper structured-prediction model and large-margin training could combine with these models. It would be interesting to see if such an idea would work well for a fully neural ASR system.

Even if progress is stymied pursuing a purely neural approach, the fact that these proposed systems are obviating the need for the pronunciation dictionary seems to me a worthy step. This "least automatic part of automatic speech recognition" has long been a source of poor recognition for speakers with foreign accents, who pronounce words differently than native speakers, and for unknown word modeling. Naively adding pronunciations to the lexicon at best seems inelegant, and at worst can substantially decrease recognition performance if too many pronunciations are included. Keeping the lexicon fixed is also a poor solution, as more data may introduce even more variation in phone models. Perhaps neural networks, or another machine learning model, can remove the lexicon, and the rigid constraints it enforces.

Bibliography

[1] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. "Hidden Markov Support Vector Machines". In: *Proceedings of the International Conference on Machine Learning*. 2003.

[2] Lalit Bahl, Peter Brown, Peter de Souza, and Robert Mercer. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP*. Vol. 11. IEEE, Apr. 1986, pp. 49–52. DOI: 10.1109/icassp.1986.1169179. URL: http://dx.doi.org/10.1109/icassp.1986.1169179.

[3] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Oakland, California: Holden-Day Company, 1977.

[4] Herve Bourlard and Nelson Morgan. *Merging Multilayer Perceptrons and Hidden Markov Models: Some Experiments in Continuous Speech Recognition*. Tech. rep. International Computer Science Institute - Technical Report TR-089-033, 1989.

[5] Herve Bourlard, Nelson Morgan, and Christopher J. Wellekens. "Statistical Inference in Multilayer Perceptrons and Hidden Markov Models with Applications in Continuous Speech Recognition". English. In: *Neurocomputing*. Ed. by FranoiseFogelman Souli and Jeanny Hrault. Vol. 68. NATO ASI Series. Springer Berlin Heidelberg, 1990, pp. 217–226. ISBN: 978-3-642-76155-3. DOI: 10.1007/978-3-642-76153-9_27. URL: http://dx.doi.org/10.1007/978-3-642-76153-9_27.

[6] Herve A. Bourlard and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993. ISBN: 0792393961.

[7] Jean Carletta. "Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus". In: *Language Resources and Evaluation* 41.2 (2007), pp. 181–190.

[8] Oliver Cetin and Andreas Stolcke. *Language modeling in the ICSI-SRI Spring 2005 meeting speech recognition evaluation system*. Tech. rep. International Computer Science Institute, 2005.

[9] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. "Listen, Attend and Spell". In: *CoRR* abs/1508.01211 (2015). URL: http://arxiv.org/abs/1508.01211.

[10] Shuo-Yiin Chang and Steven Wegmann. "On the Importance of Modeling and Robustness for Deep Neural Network Feature". In: *Proc. ICASSP*. 2015.

[11] Stanley F. Chen and Joshua Goodman. "An Empirical Study of Smoothing Techniques for Language Modeling". In: *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*. ACL '96. Santa Cruz, California: Association for Computational Linguistics, 1996, pp. 310–318. DOI: `10.3115/981863.981904`. URL: `http://dx.doi.org/10.3115/981863.981904`.

[12] Michael Collins. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 1–8. DOI: `10.3115/1118693.1118694`. URL: `http://dx.doi.org/10.3115/1118693.1118694`.

[13] George E. Dahl, Dong Yu, Li Deng, and Alex Acero. "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1 (2012), pp. 30 –42. ISSN: 1558-7916. DOI: `10.1109/TASL.2011.2134090`.

[14] Rogier van Dalen and Mark Gales. "Annotating Large Lattices with Exact Word Error". In: *Interspeech*. Dresden, Germany, Sept. 2015.

[15] Bradley Efron. "Bootstrap Methods: Another Look at the Jackknife". In: *Annals of Statistics* 7 (1979), pp. 1–26.

[16] Bradley Efron. *The Jackknife, the bootstrap and other resampling plans*. CBMS-NSF Reg. Conf. Ser. Appl. Math. Lectures given at Bowling Green State Univ., June 1980. Philadelphia, PA: SIAM, 1982. URL: `https://cds.cern.ch/record/98913`.

[17] Matthew Gibson and Thomas Hain. "Hypothesis Spaces for Minimum Bayes Risk Training in Large Vocabulary Speech Recognition". In: *In Proc. Interspeech*. 2006, pp. 2–4.

[18] Dan Gillick, Larry Gillick, and Steven Wegmann. "Don't multiply lightly: Quantifying problems with the acoustic model assumptions in speech recognition". In: *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011*. 2011, pp. 71–76. DOI: `10.1109/ASRU.2011.6163908`. URL: `http://dx.doi.org/10.1109/ASRU.2011.6163908`.

[19] Dan Gillick, Steven Wegmann, and Larry Gillick. "Discriminative training for speech recognition is compensating for statistical dependence in the HMM framework". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*. 2012, pp. 4745–4748. DOI: `10.1109/ICASSP.2012.6288979`. URL: `http://dx.doi.org/10.1109/ICASSP.2012.6288979`.

[20] Larry Gillick and Stephen Cox. "Some statistical issues in the comparison of speech recognition algorithms". In: *In Proc. of ICASSP*. 1989, pp. 532–535.

[21] Vaibhava Goel and William J. Byrne. "Minimum Bayes-risk automatic speech recognition." In: *Computer Speech and Language* 14.2 (2000), pp. 115–135. URL: `http://dblp.uni-trier.de/db/journals/csl/csl14.html#GoelB00`.

[22] Pavel Golik, Zoltán Tüske, Ralf Schlüter, and Hermann Ney. "Convolutional Neural Networks for Acoustic Modeling of Raw Time Signal in LVCSR". In: *Interspeech*. Dresden, Germany, Sept. 2015, pp. 26–30.

[23] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. "Deep Speech: Scaling up end-to-end speech recognition". In: *CoRR* abs/1412.5567 (2014). URL: `http://arxiv.org/abs/1412.5567`.

[24] Tamir Hazan, Joseph Keshet, and David A. McAllester. "Direct Loss Minimization for Structured Prediction". In: *Advances in Neural Information Processing Systems 23*. Ed. by J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 1594–1602. URL: `http://papers.nips.cc/paper/4069-direct-loss-minimization-for-structured-prediction.pdf`.

[25] Georg Heigold. "A log-linear discriminative modeling framework for speech recognition". Zsfassung in dt. und engl. Sprache; Aachen, Techn. Hochsch., Diss., 2010. PhD thesis. Aachen, 2010, XIV, 191 S. : graph. Darst. URL: `http://publications.rwth-aachen.de/record/51838`.

[26] Georg Heigold, Thomas Deselaers, Ralf Schlüter, and Hermann Ney. "Modified MMI/MPE: A Direct Evaluation of the Margin in Speech Recognition". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, 2008, pp. 384–391. ISBN: 978-1-60558-205-4. DOI: `10.1145/1390156.1390205`. URL: `http://doi.acm.org/10.1145/1390156.1390205`.

[27] Hynek Hermansky, Daniel P. W. Ellis, and Sangita Sharma. "Tandem connectionist feature extraction for conventional HMM systems". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on* 3 (2000), pp. 1635–1638. DOI: `10.1109/icassp.2000.862024`. URL: `http://dx.doi.org/10.1109/icassp.2000.862024`.

[28] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Comput.* 18.7 (July 2006), pp. 1527–1554. ISSN: 0899-7667. DOI: `10.1162/neco.2006.18.7.1527`. URL: `http://dx.doi.org/10.1162/neco.2006.18.7.1527`.

[29] Yan Huang, Dong Yu, Chaojun Liu, and Yifan Gong. "A Comparative Analytic Study on the Gaussian Mixture and Context Dependent Deep Neural Network Hidden Markov Models". In: *Interspeech 2014*. 2014. URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=230138`.

[30] Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. "The ICSI Meeting Corpus". In: *Proc. Interspeech*. 2003, pp. 364–367.

[31] Brian Kingsbury, Tara N. Sainath, and Hagen Soltau. "Scalable Minimum Bayes Risk Training of Deep Neural Network Acoustic Models Using Distributed Hessian-free Optimization." In: *Interspeech*. ISCA, 2012, pp. 10–13. URL: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2012.html#KingsburySS12.

[32] Reinhard Kneser and Hermann Ney. "Improved Backing-off for M-gram Language Modeling". In: Detroit, Michigan, USA, May 1995, pp. 181–184.

[33] Yochai Konig, Hervé Bourlard, and Nelson Morgan. "REMAP: Recursive Estimation and Maximization of A Posteriori Probabilities - Application to Transition-Based Connectionist Speech Recognition". In: *Advances in Neural Information Processing Systems 8*. Ed. by D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo. MIT Press, 1996, pp. 388–394. URL: http://papers.nips.cc/paper/1027-remap-recursive-estimation-and-maximization-of-a-posteriori-probabilities-application-to-transition-based-connectionist-speech-recognition.pdf.

[34] Stanislav Kontár. "Parallel training of neural networks for speech recognition". In: *Proc. 12th International Conference on Soft Computing MENDEL'06*. Brno, CZ: Brno University of Technology, 2006, p. 6. ISBN: 80-214-3195-4. URL: http://www.fit.vutbr.cz/research/view_pub.php?id=8180.

[35] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813.

[36] Richard P. Lippmann. "Speech Recognition by Machines and Humans". In: *Speech Communication*. 1997, pp. 1–15.

[37] Andrew L. Maas, Ziang Xie, Dan Jurafsky, and Andrew Y. Ng. "Lexicon-Free Conversational Speech Recognition with Neural Networks". In: *North American Chapter of the Association for Computational Linguistics*. Singapore, 2015.

[38] James Martens. "Deep learning via Hessian-free optimization". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. 2010, pp. 735–742. URL: http://www.icml2010.org/papers/458.pdf.

[39] David McAllester. "Generalization Bounds and Consistency for Structured Labeling in Predicting Structured Data". In: (2007). URL: http://nagoya.uchicago.edu/~dmcallester/colbounds.pdf.

[40] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. "Recurrent neural network based language model". In: *Interspeech 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. 2010, pp. 1045–1048. URL: http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html.

[41] Nelson Morgan and Herve A. Bourlard. “Continuous Speech Recognition Using Multilayer Perceptrons with Hidden Markov Models”. In: Albuquerque, New Mexico, USA, 1990, pp. 413–416.

[42] Jeremy Morris and Eric Fosler-Lussier. “CRANDEM: conditional random fields for word recognition.” In: *Interspeech*. ISCA, 2009, pp. 3063–3066. URL: `http://dblp.uni-trier.de/db/conf/interspeech/interspeech2009.html#MorrisF09`.

[43] Tasha Nagamine, Mike Seltzer, and Nima Mesgerani. “Exploring How Deep Neural Networks Form Phonemic Categories”. In: *Proc. Interspeech*. Dresden, Germany, 2015.

[44] Sree Hari Krishnan Parthasarathi, Shuo-Yiin Chang, Jordan Cohen, Nelson Morgan, and Steven Wegmann. “The blame game in meeting room ASR: An analysis of feature versus model errors in noisy and mismatched conditions”. In: *ICASSP'13*. 2013, pp. 6758–6762.

[45] David Pearce, Hans-Gunter Hirsch, and Ericsson Eurolab Deutschland Gmbh. “The Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions”. In: *in ISCA ITRW ASR2000*. 2000, pp. 29–32.

[46] Daniel Povey and Philip C. Woodland. “Minimum Phone Error and I-smoothing for improved discriminative training”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2002, May 13-17 2002, Orlando, Florida, USA*. 2002, pp. 105–108. DOI: `10.1109/ICASSP.2002.5743665`. URL: `http://dx.doi.org/10.1109/ICASSP.2002.5743665`.

[47] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah. “Boosted MMI for model and feature-space discriminative training”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2008, March 30 - April 4, 2008, Caesars Palace, Las Vegas, Nevada, USA*. 2008, pp. 4057–4060. DOI: `10.1109/ICASSP.2008.4518545`. URL: `http://dx.doi.org/10.1109/ICASSP.2008.4518545`.

[48] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. “The Kaldi Speech Recognition Toolkit”. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011.

[49] Rohit Prabhavalkar, Preethi Jyothi, William Hartmann, Jeremy Morris, and Eric Fosler-Lussier. “Investigations into the Crandem Approach to Word Recognition.” In: *HLT-NAACL*. The Association for Computational Linguistics, 2010, pp. 725–728. ISBN: 978-1-932432-65-7. URL: `http://dblp.uni-trier.de/db/conf/naacl/naacl2010.html#PrabhavalkarJHMF10`.

[50] Shakti P. Rath, Kate M. Knill, Anton Ragni, and Mark J. F. Gales. "Combining Tandem and Hybrid Systems for Improved Speech Recognition and Keyword Spotting on Low Resource Languages". In: *Proc. Interspeech.* Singapore, 2014.

[51] Suman V. Ravuri. "Hybrid DNN-Latent Structured SVM Acoustic Models for Continuous Speech Recognition". In: *ASRU 2015, IEEE Automatic Speech Recognition and Understanding Workshop.* 2015.

[52] Suman V. Ravuri. "Hybrid MLP/structured-SVM tandem systems for large vocabulary and robust ASR". In: *Interspeech 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014.* 2014, pp. 2729–2733. URL: `http://www.isca-speech.org/archive/interspeech_2014/i14_2729.html`.

[53] *Rt-04s evaluation data documentation,* ***http://www.itl.nist.gov/iad/mig/tests/rt/2004-spring/eval/docs.html*** *.*

[54] *Rt-05s evaluation data documentation,* ***http://www.itl.nist.gov/iad/mig/tests/rt/2005-spring/eval/docs.html*** *.*

[55] *Rt-2002 evaluation plan,* ***http://www.itl.nist.gov/iad/mig/tests/rt/2002/docs/rt02_eval_plan_v3.pdf.*** *.*

[56] Tara Sainath, Ron Weiss, Andrew Senior, Kevin Wilson, and Oriol Vinyals. "Learning the Speech Front-end With Raw Waveform CLDNNs". In: *Interspeech.* Dresden, Germany, Sept. 2015.

[57] George Saon, Hong-Kwang Jeff Kuo, Steven J. Rennie, and Michael Picheny. "The IBM 2015 English Conversational Telephone Speech Recognition System". In: *Interspeech.* Dresden, Germany, Sept. 2015.

[58] Fei Sha and Lawrence K. Saul. "Large Margin Gaussian Mixture Modeling for Phonetic Classification and Recognition". In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 14-19, 2006.* 2006, pp. 265–268. DOI: `10.1109/ICASSP.2006.1660008`. URL: `http://dx.doi.org/10.1109/ICASSP.2006.1660008`.

[59] Yoram Singer and Nathan Srebro. "Pegasos: Primal estimated sub-gradient solver for SVM". In: *In ICML.* 2007, pp. 807–814.

[60] Andreas Stolcke, Neville Ryant, Vikramjit Mitra, Wen Wang, and Mark Liberman. "Highly Accurate Phonetic Segmentation Using Boundary Correction Models and System Fusion". In: *Proc. IEEE ICASSP.* Florence: IEEE SPS, 2014, pp. 5589–5593. URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=209007`.

[61] Andreas Stolcke, Xavier Anguera, Kofi Boakye, zgr etin, Adam Janin, Mathew Magimai-doss, Chuck Wooters, and Jing Zheng. "The SRI-ICSI spring 2007 meeting and lecture recognition system". In: *Proc. NIST Rich Transcription Workshop, Springer Lecture Notes in Computer Science, 2007. 3.3 164.*

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27.* Ed. by Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger. Curran Associates, Inc., 2014, pp. 3104–3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[63] *SVM-HMM for Sequence Tagging, http://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html.*

[64] Hao Tang, Kevin Gimpel, and Karen Livescu. "A comparison of training approaches for discriminative segmental models". In: *Proc. Interspeech.* Singapore, 2014.

[65] Ben Taskar, Carlos Guestrin, and Daphne Koller. "Max-Margin Markov Networks". In: *Advances in Neural Information Processing Systems 16.* Ed. by S. Thrun, L.K. Saul, and B. Schölkopf. MIT Press, 2004, pp. 25–32. URL: http://papers.nips.cc/paper/2397-max-margin-markov-networks.pdf.

[66] Yee-Whye Teh. "A Hierarchical Bayesian Language Model Based on Pitman-Yor Processes". In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics.* ACL-44. Sydney, Australia: Association for Computational Linguistics, 2006, pp. 985–992. DOI: 10.3115/1220175.1220299. URL: http://dx.doi.org/10.3115/1220175.1220299.

[67] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. "Large margin methods for structured and interdependent output variables". In: *Journal of Machine Learning Research* 6 (2005), pp. 1453–1484.

[68] Zoltán Tüske, Pavel Golik, Ralf Schlüter, and Hermann Ney. "Acoustic Modeling with Deep Neural Networks Using Raw Time Signal for LVCSR". In: *Interspeech.* ISCA best student paper award Interspeech 2014. Singapore, Sept. 2014, pp. 890–894.

[69] Zoltán Tüske, Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "Context-Dependent MLPs for LVCSR: TANDEM, Hybrid or Both?" In: *Interspeech.* Portland, OR, USA, Sept. 2012, pp. 18–21.

[70] Karel Vesely, Arnab Ghoshal, Lukas Burget, and Daniel Povey. "Sequence-discriminative training of deep neural networks". In: *Interspeech 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013.* 2013, pp. 2345–2349. URL: http://www.isca-speech.org/archive/interspeech_2013/i13_2345.html.

[71] Oriol Vinyals and Nelson Morgan. "Deep vs. wide: depth on a budget for robust speech recognition". In: *Interspeech 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013.* 2013, pp. 114–118. URL: http://www.isca-speech.org/archive/interspeech_2013/i13_0114.html.

[72] Oriol Vinyals and Daniel Povey. "Krylov Subspace Descent for Deep Learning". In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012.* 2012, pp. 1261–1268. URL: `http://jmlr.csail.mit.edu/proceedings/papers/v22/vinyals12.html`.

[73] Oriol Vinyals and Suman Ravuri. "Comparing multilayer perceptron to Deep Belief Network Tandem features for robust ASR". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on.* IEEE. 2011, pp. 4596–4599.

[74] Oriol Vinyals, Suman Ravuri, and Daniel Povey. "Revisiting Recurrent Neural Networks for Robust ASR". In: IEEE International Confrence on Acoustics, Speech, and Signal Processing (ICASSP), 2012. URL: `http://research.microsoft.com/apps/pubs/default.aspx?id=164627`.

[75] Oriol Vinyals and Suman V. Ravuri. "Comparing multilayer perceptron to Deep Belief Network Tandem features for robust ASR". In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic.* 2011, pp. 4596–4599. DOI: `10.1109/ICASSP.2011.5947378`. URL: `http://dx.doi.org/10.1109/ICASSP.2011.5947378`.

[76] Guangsen Wang and Khe Chai Sim. "Sequential classification criteria for NNs in automatic speech recognition". In: *Proc. Interspeech.* Florence, Italy, 2011, pp. 441–444.

[77] Steve J. Young, Julian J. Odell, and Phil C. Woodland. "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proceedings of the Workshop on Human Language Technology.* HLT '94. Plainsboro, NJ: Association for Computational Linguistics, 1994, pp. 307–312. ISBN: 1-55860-357-3. DOI: `10.3115/1075812.1075885`. URL: `http://dx.doi.org/10.3115/1075812.1075885`.

[78] Chun-Nam John Yu and Thorsten Joachims. "Learning Structural SVMs with Latent Variables". In: *Proceedings of the 26th Annual International Conference on Machine Learning.* ICML '09. Montreal, Quebec, Canada: ACM, 2009, pp. 1169–1176. ISBN: 978-1-60558-516-1. DOI: `10.1145/1553374.1553523`. URL: `http://doi.acm.org/10.1145/1553374.1553523`.

[79] Shi-Xiong Zhang and M. J. F. Gales. "Structured Support Vector Machines for Noise Robust Continuous Speech Recognition". In: *Proc. Interspeech.* Florence, Italy, 2011, pp. 989–992.

[80] Shi-Xiong Zhang and Mark J. F. Gales. "Extending noise robust structured support vector machines to larger vocabulary tasks". In: *2011 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2011, Waikoloa, HI, USA, December 11-15, 2011.* 2011, pp. 18–23. DOI: `10.1109/ASRU.2011.6163898`. URL: `http://dx.doi.org/10.1109/ASRU.2011.6163898`.

[81] Shi-Xiong Zhang and Mark J. F. Gales. "Structured SVMs for Automatic Speech Recognition". In: *IEEE Transactions on Audio, Speech & Language Processing* 21.3 (2013), pp. 544–555. DOI: 10.1109/TASL.2012.2227734. URL: http://dx.doi.org/10.1109/TASL.2012.2227734.

[82] Shi-Xiong Zhang, Chaojun Liu, Kaisheng Yao, and Yifan Gong. "Deep Neural Support Vector Machines for Speech Recognition". In: *Proc. ICASSP*. 2015. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=244711.

[83] Geoffrey Zweig and Patrick Nguyen. "A segmental CRF approach to large vocabulary continuous speech recognition". In: *2009 IEEE Workshop on Automatic Speech Recognition & Understanding, ASRU 2009, Merano/Meran, Italy, December 13-17, 2009*. 2009, pp. 152–157. DOI: 10.1109/ASRU.2009.5372916. URL: http://dx.doi.org/10.1109/ASRU.2009.5372916.