NONPARAMETRIC BAYESIAN APPROACHES FOR ACOUSTIC MODELING

A Dissertation
Submitted to
the Temple University Graduate Board

In Partial Fulfillment
of the Requirements for the Degree
DOCTOR OF PHILOSOPHY

---

by
Amir Hossein Harati Nejad Torbati
July, 2015

Examining Committee Members:

Dr. Joseph Picone, Advisory Committee Chair, Department of ECE
Dr. Iyad Obeid, Department of ECE
Dr. Marc Sobel, Department of Statistics
Dr. Chang-Hee Won, Department of ECE
Dr. Slobodan Vucetic, Department of Computer Science
Dr. Kevin Buckley, External Reader, Department of ECE, Villanova University

ABSTRACT

NON-PARAMETRIC BAYESIAN APPROACHES FOR ACOUSTIC MODELING

Amir Hossein Harati Nejad Torbati

Doctor of Philosophy

Temple University, July 2015

Advisor: Dr. Joseph Picone

The goal of Bayesian analysis is to reduce the uncertainty about unobserved variables by combining prior knowledge with observations. A fundamental limitation of a parametric statistical model, including a Bayesian approach, is the inability of the model to learn new structures. The goal of the learning process is to estimate the correct values for the parameters. The accuracy of these parameters improves with more data but the model's structure remains fixed. Therefore new observations will not affect the overall complexity (e.g. number of parameters in the model). Recently, nonparametric Bayesian methods have become a popular alternative to Bayesian approaches because the model structure is learned simultaneously with the parameter distributions in a data-driven manner.

The goal of this dissertation is to apply nonparametric Bayesian approaches to the acoustic modeling problem in continuous speech recognition. Three important problems are addressed: (1) statistical modeling of sub-word acoustic units; (2) semi-supervised training algorithms for nonparametric acoustic models; and (3) automatic discovery of sub-word acoustic units.

We have developed a Doubly Hierarchical Dirichlet Process Hidden Markov Model (DHDPHMM) with a non-ergodic structure that can be applied to problems involving sequential modeling. DHDPHMM shares mixture components between states using two Hierarchical Dirichlet Processes (HDP). An inference algorithm for this model has been developed that enables DHDPHMM to outperform both its hidden Markov model (HMM) and HDP HMM

(HDPHMM) counterparts. This inference algorithm is shown to also be computationally less expensive than a comparable algorithm for HDPHMM.

In addition to sharing data, the proposed model can learn non-ergodic structures and non-emitting states, something that HDPHMM does not support. This extension to the model is used to model finite length sequences. We have also developed a generative model for semi-supervised training of DHDPHMMs. Semi-supervised learning is an important practical requirement for many machine learning applications including acoustic modeling in speech recognition. The relative improvement in error rates on classification and recognition tasks is shown to be *22%* and *7%* respectively. Semi-supervised training results are slightly better than supervised training (*29.02%* vs. *29.71%*). Context modeling was also investigated and results show a modest improvement of *1.5%* relative over the baseline system.

We also introduce a nonparametric Bayesian transducer based on an ergodic HDPHMM/DHDPHMM that automatically segments and clusters the speech signal using an unsupervised approach. This transducer was used in several applications including speech segmentation, acoustic unit discovery, spoken term detection and automatic generation of a pronunciation lexicon. For the segmentation problem, an F-score of *76.62%* was achieved which represents a *9%* relative improvement over the baseline system. On the spoken term detection tasks, an average precision of *64.91%* was achieved, which represents a *20%* improvement over the baseline system. Lexicon generation experiments also show automatically discovered units (ADU) generalize to new datasets.

In this dissertation, we have established the foundation for applications of non-parametric Bayesian modeling to problems such as speech recognition that involve sequential modeling. These models allow a new generation of machine learning systems that adapt their overall complexity in a data-driven manner and yet preserve meaningful modalities in the data. As a result, these models improve generalization and offer higher performance at lower complexity.

DEDICATION

To my parents, Fatemeh and Alireza,

for having supported me throughout my education and life.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Dr. Joseph Picone. His commitment to his students and the quality of their research (and writing) is exemplary. I have learned a great deal during the past five years that I have been working in his lab. Most importantly I learned how to conduct research with a positive attitude and how to overcome obstacles. His insistence on systematic approaches helped me to learn how to isolate problems and solve them one at a time. I am also thankful that he trusted my abilities and gave me space to conduct my research freely.

I am also thankful to all of my committee members: Dr. Obied, Dr. Sobel, Dr. Vucetic, Dr. Won and my external reader Dr. Buckley. I am especially thankful to Dr. Sobel because the talk he gave in our department's seminar series helped me find this dissertation topic. He was also kind enough to spend time discussing some of the concepts presented in this dissertation.

I am thankful to all the previous and current students at ISIP who provided a friendly, supportive and collaborative work environment. I am thankful to the College of Engineering and the Department of Electrical and Computer Engineering, both of which provided me with the financial support that helped me complete my PhD studies. I am thankful to the Graduate School at Temple University, which provided me with a PhD continuation grant that helped me to focus on finishing my dissertation. I am thankful to the University City Science Center and several external government funding agencies for their financial support of this dissertation.

The experiments conducted in this dissertation used Temple University's High Performance Computing (HPC) facilities that were provided by the National Science Foundation through a Major Research Instrumentation Grant (Grant Nos. CNS-09-58854 and CNS-1305190). I am also thankful to HPC technical staff that provided excellent support during my graduate studies.

Finally, I am thankful to my parents and family for their support throughout my life and for providing me with a good education and unconditional love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Over the past few decades, speech recognition research, much like other pattern recognition applications, has been focused on developing new statistical models and better machine learning algorithms to estimate the parameters of these models (Rabiner, 1989; Jelinek, 1997; Huang et al., 2001; Hinton, et al., 2012). Some of the most successful statistical modeling approaches from this era include hidden Markov models (HMMs), neural networks and more recently deep learning based neural networks. Parameter estimation algorithms for HMMs are often based on the Expectation Maximization (EM) Theorem (Dempster et al., 1977) using the computationally efficient Baum-Welch (BW) algorithm (Baum & Petrie, 1966), or the extended Baum-Welch (EBW) algorithm (McLachlan & Thriyambakam, 2008). However, during the past few years, despite of the availability of vast computational power and large amounts of data, the improvement of the performance of the state of the art systems has been at best marginal, and the ability of these systems to process previously unseen data, such as data recorded from new acoustic channels, is limited. This latter problem is related to the generalization problem (Bishop, 2007) and is a major long-term goal of this research. In this dissertation, we will explore nonparametric Bayesian solutions to address some of these limitations.

Generally, determining model complexity, which is an important part of the generalization problem, is difficult. An oversimplified model cannot describe the data and a very complex model is prone to overfitting. Model selection techniques usually need a huge amount of data and are computationally expensive (Bishop, 2007). Any selection methodology needs a criterion for

selecting a preferred model. There is not a widely accepted universal selection criterion (Ghahramani, 2010). Hence, this process is application specific and involves searching through a discrete space (e.g., a combinational search over models). The final result is also sensitive to the criterion used to guide the search.

Nonparametric Bayesian methods provide a mathematically elegant framework that allows inference of model structure and complexity without diluting the purity of modes or clusters (Sudderth, 2006). Figure 1-1 shows an example application of nonparametric Bayesian modeling to the problem of clustering. The complexity of the model in this case is proportional to the number of clusters. In a fully Bayesian framework, hyperparameters (i.e. parameters that control the complexity of the model) along with the model parameters can be learned automatically from the data. In other words, the data speaks for itself. Unlike in a model selection problem, the optimization of the model parameters is a continuous optimization problem and hence is more tractable. The number of clusters in this example varies with the amount of data and is optimized as part of the training process. A data-driven approach to infer the complexity of the model is extremely valuable in many pattern recognition applications.

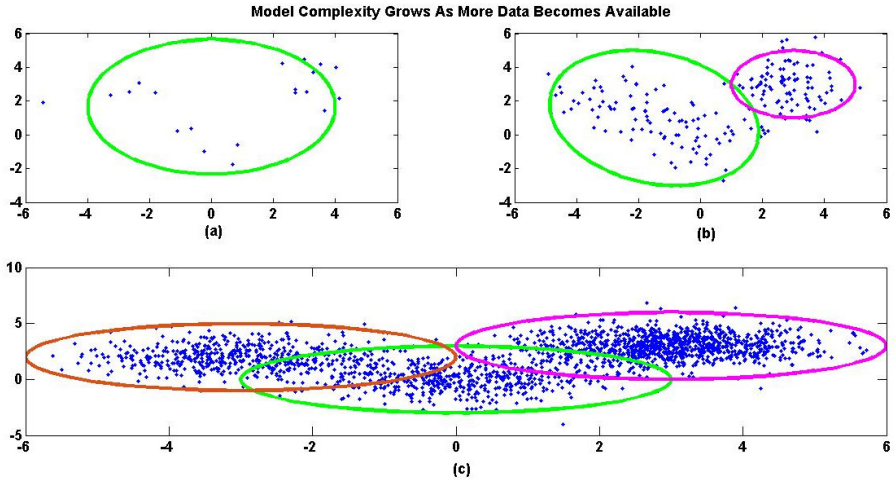


Figure 1-1. An example that shows the growth in model complexity as more data becomes available.

In speech recognition, selection of an appropriate model complexity and the optimal hyperparameters are among the most difficult and time-consuming parts of the training process. The selection process also has a direct impact on the performance of the system. A typical state of the art speech recognition system, which is composed of a collection of many simple HMMs, has a large number of degrees of freedom. Model complexity in this case is proportional to the complexity of the individual HMMs that compose the system. These HMMs consist of a topology, transition probability matrix, and an emission distribution model that typically uses a large number of Gaussian mixture models. Such systems have tens of thousands of these HMMs and often tens of millions of parameters must be estimated using a complicated bootstrapping process. It is difficult to guarantee good generalization with such complex systems. Therefore, a technology that optimizes model complexity as part of the training process is important.

In this chapter, we first review the problem of acoustic modeling in speech recognition and then review some known applications of nonparametric Bayesian modeling in speech recognition. Finally the outline of the dissertation will be presented and our contributions will be summarized.

1.1 Acoustic Modeling

The ultimate goal of speech recognition is to map the acoustic observations into word sequences. This problem can be formulated as (Bahl et al., 1983):

$$P(W \mid A) = \frac{P(A \mid W)P(W)}{P(A)}. \tag{1.1}$$

In this formulation, $P(W|A)$ is the probability of a particular word sequence given the acoustic observations. The goal is to find a sequence of words, W, that maximizes this probability. $P(W)$ is referred to as the language model since it provides the prior probability of words. $P(A)$ is the probability of the observed acoustic data and usually can be ignored. $P(A|W)$ is referred to as the acoustic model. Therefore, we can divide the speech recognition problem into two separate sub-

problems, namely language modeling and acoustic modeling, and solve each one independently. Our focus in this research is on the acoustic modeling problem.

Figure 1-2 shows a modular architecture for a typical speech recognizer based on (1.1). Input speech, which is a sampled data signal, is first converted into a representation that consists of D-dimensional vectors computed typically every *10* ms. This sequence of vectors is referred to as acoustic features. One of the most popular algorithms for performing this conversion uses Mel Frequency Cepstral Coefficients (MFCC) (Davis and Mermelstein, 1980). All experiments in this dissertation use the MFCC representation. The next step is to evaluate the likelihood that these feature vectors were generated by each acoustic model available in the system. This is a process we refer to as acoustic decoding. These acoustic models must be defined and trained before they can be used. The development of these acoustic models is the main topic of this dissertation. Finally we combine these likelihoods, referred to as the acoustic evidence, with an independently

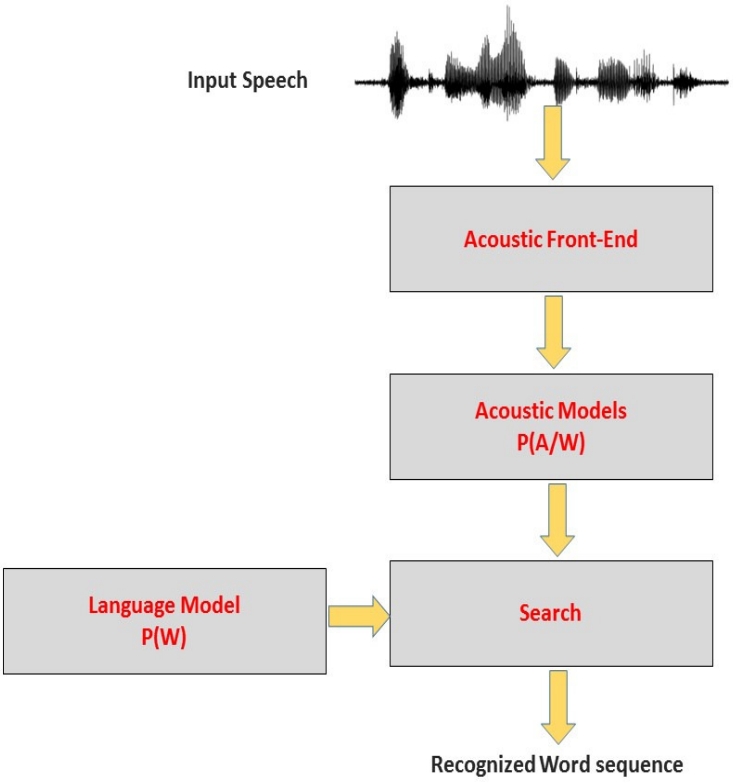


Figure 1-2. A modular architecture for a speech recognition system is shown.

trained language model to produce the most likely word sequence. This latter problem is referred to as the language modeling problem and is not addressed in this dissertation.

In early speech recognition systems (Furui, 1986), each word was modeled separately using a template matching approach based on dynamic time warping (DTW). This approach worked well for small vocabulary and isolated word speech recognition tasks. However, it is not scalable to large vocabulary continuous speech recognition problems. Since the number of words in a typical language is very large (and increases over time), modeling all words independently is not feasible. An alternative approach is to decompose words into a finite set of units common to all possible words, referred to as acoustic units. Different types of acoustic units such as phonemes (Lee, 1989), syllables (Ganapathiraju et al., 2001) and acoustically inspired units (Paliwal, 1990) have been explored over the years. Phonemes are the most popular units since they achieve good performance and have a direct connection to the linguistic properties of a language. Most successful commercial systems are based on phonetic units.

After selecting the type of the unit, a lexicon is needed that maps words into these units. The lexicon is a critical module of speech recognition system and is often one of the most costly resources to construct for a new language or application. Joint optimization of the lexicon and the acoustic units is one of several challenging problems in achieving high performance speech recognition. For example, if we want to introduce a new set of acoustic units, we must define or modify the lexicon to incorporate these units. In Chapter 5 of this dissertation we discuss the problem of automatically learning the acoustic units and lexicon from acoustic observations.

We also need to select a statistical model to be used as a model for each acoustic unit. Given a set of trained models and some new observations, we test all the models against the observations and select the model with the highest score (e.g. likelihood). The most popular models used in state of the art systems are left-to-right HMMs (Levinson et al., 1983; Picone, 1990) with mixtures of Gaussians emission distributions (Rabiner, 1989). An HMM is a generalization of a mixture model where latent variables are not independent of each other and

are related by a Markov chain. This makes them particularly attractive for modeling sequential observations. Most systems use a simple HMM with some predetermined number of states (e.g., *3*) for all units (Huang et al., 2001). Each state will use a predetermined number of mixture components, often ranging from *16* to *128* depending on the application.

State of the art speech recognizers usually use some form of context-dependent units instead of simple context-independent units (Schwartz et al., 1985). For example, phoneme-based systems usually have between *39-60* context independent (CI) phonemes (Huang et al., 2001). In order to improve the quality of models, we can incorporate the left and right context and define context dependent (CD) units (e.g. triphones). However, the number of units grows exponentially with an increase in the depth of the context. For example, number of possible triphones for a system with *42* phonemes are *42x42x42 = 74,088*. In any practical situation, many models will never have any observations associated with them and many will have just a few examples. This means the process of training CD models poses a serious data sparsity problem. The resulting system will perform worse than a CI system for a moderate or even a relatively large amount of training data.

The estimated parameters will have large variances, and sophisticated parameter sharing techniques must be employed (Young & Woodland, 1994). The most successful approach to tie states is based on a phonetic decision tree that is implemented as a binary tree with phonetic questions attached to its nodes. The tying is performed between corresponding states of all triphones with the same central phoneme.

Therefore a general algorithm to train acoustic models in a contemporary automatic speech recognition (ASR) system is as follow:

1. **Data Preparation:** We begin with speech data that consists of the sampled data and the corresponding word transcriptions. We do not require time alignments of the words to the sampled data (e.g., the start and stop time of each word). We will refer to these as word-level transcriptions. We convert the sampled data into an appropriate feature

representation (e.g., MFCC). We also need a lexicon that contains all possible words and their corresponding sub-word decompositions (e.g. phonemes).

2. **CI Training:** We train CI phonetic models, often referred to as monophone models, using the transcribed data and the EM algorithm. This step is usually performed using the self-organizing property of an HMM. We let the HMM segment data into different models and states as part of the training process.
3. **CD Training:** After training good monophone models, the next step is to clone monophones into CD models, referred to as triphones, by simply copying the emission distributions and transition matrix for all triphones with the same central phoneme. These cloned models are then trained using EM.
4. **State Tying:** The next step is to tie states to account for models that are underutilized and train the resulting tied models using several more iterations of EM.
5. **Mixture Splitting:** After training a model with one mixture component per state we increase the number of components per state by using an iterative splitting algorithm (Young et al., 2006). We retrain the models using several iterations of EM until we reach the desired trade-off between complexity and recognition accuracy.

This training procedure can be viewed as a semi-supervised approach to training acoustic models. The term semi-supervised refers to the fact that we use word-level transcriptions of the data but no time alignments between the words and corresponding feature vectors. Note also that transcription of the data in terms of phonemes, which here represent the acoustic units, are NOT required. This is an important practical consideration when training on large amounts of data. It greatly reduces the cost of developing training data for these systems.

In the following chapters, we will address three specific problems related to the processes above:

- Modeling sub-word units;
- Semi-supervised training algorithms for sub-word units;
- Data-driven discovery of the sub-word units and the lexicon.

Our goals are to reduce complexity and ultimately the time required to create a high performance system for a new application or language through the use of data-driven techniques based on nonparametric Bayesian models.

1.2 Nonparametric Bayesian Approaches in Speech Recognition

Nonparametric Bayesian models have been previously used in language modeling (Teh, 2006; Wood & Teh, 2009). Language modeling involves estimation of discrete probability distributions that model conditional probabilities of word sequences. Teh (2006) showed that incorporating a hierarchical Pitman-Yor process as a prior performs better than the interpolated Kneser-Ney smoothing algorithm and can approach the performance of the state of the art modified Kneser-Ney algorithm (Kneser & Ney, 1995). More importantly, it has been shown that the Kneser-Ney smoothing algorithm can be derived from an approximation of the inference algorithm for this model (Teh, 2006). A Pitman-Yor process is a generalization of a Dirichlet process and can model power-law distributions. This is relevant to language modeling because it has been shown experimentally that word frequencies in human language follow a power-law distribution (Pitman & Yor, 1997).

Nonparametric Bayesian approaches have been used previously in some acoustic modeling related applications as well. For example, Fox et al. (2011) have used a nonparametric Bayesian HMM in a speaker diarization problem where each meeting was modeled as a nonparametric Bayesian HMM. Each meeting was segmented into speaker homogenous regions. Results were competitive with the state of the art (Wooters & Huijbregts, 2007). This system was known as a hierarchical Dirichlet process hidden Markov model (HDPHMM).

Lee (2014) has proposed a nonparametric Bayesian model for acoustic unit discovery. This model was applied to a variety of problems in cognitive science (e.g., one-shot learning of spoken words and hierarchical linguistic structure discovery) and speech recognition (e.g., automatic learning of acoustic units and lexicon). We have compared some of our results in Chapter 5 to Lee's work on speech recognition.

In our preliminary research, we have studied the application of a Dirichlet Process Mixture (DPM) model to the speaker adaption problem (Harati et al., 2012). In that study, we showed that DPM can successfully replace the regression tree in a Maximum Likelihood Linear Regression (MLLR) model. Figure 1-3 compares the word error rate (WER) obtained for a speech recognizer with speaker adaption using monophone models for both DPM and a regression tree based MLLR algorithm. DPM improves performance over MLLR by *10%*.

More interestingly, clusters generated by DPM had acoustically and phonetically meaningful interpretations – the resulting clusters resemble broad phonetic classes. For example, distributions related to phonemes "*w*" and "*r*", which both belong to a broad phonetic class known as liquids

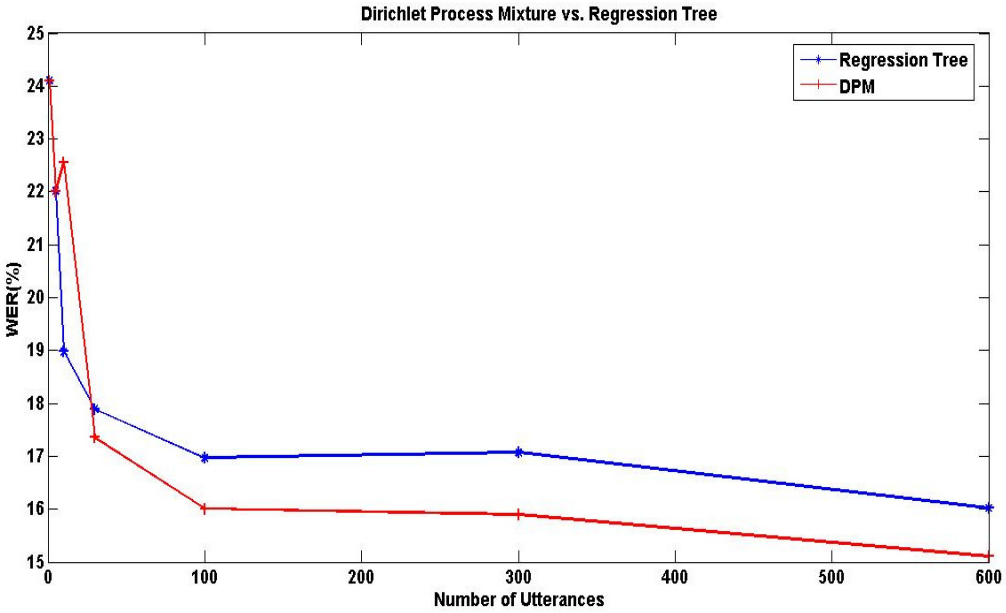


Figure 1-3. DPM-based clustering produced a *10%* improvement over a standard regression tree approach for an MLLR speaker adaptation task.

(Ladefoged & Johnson, 1993), were assigned to the same cluster. Also, the DPM-based model finds seven clusters, which is on the order of the number of broad phonetic classes used to describe English (Ladefoged & Johnson, 1993). The MLLR approach based on regression trees found *20* clusters. The nonparametric Bayesian algorithm found a solution with lower complexity and better performance.

This study was one of the motivations for this dissertation because it demonstrated that the nonparametric Bayesian framework could reduce complexity and yet preserve meaningful modalities in the data, making it a promising approach for acoustic modeling in speech recognition.

1.3 Dissertation Organization

Chapter 2: Nonparametric Bayesian Basics

In Chapter 2, a brief introduction to the nonparametric Bayesian framework will be presented. Nonparametric Bayesian modeling consists of a fairly vast family of models and distributions that parallels most of the well-known parametric models. However, in this chapter we limit our introduction to DPM, HDP and HDPHMM.

Chapter 3: Nonparametric Bayesian Approaches For Acoustic Modeling of Sub-Word Units

In this chapter, we discuss the limitations of HDPHMM for the problem of acoustic modeling of sub-word units. We introduce a new model that is the central contribution to this dissertation. This model, named a Doubly Hierarchical Dirichlet Process Hidden Markov Model (DHDPHMM), shares mixture components across all states. We will show this model works better than HDPHMM for problems similar to sub-word modeling and has the benefits of better accuracy and less computational complexity.

In this chapter, we also introduce necessary extensions for learning non-ergodic structures (e.g. left to right) and learning non-emitting states. These structures are required for acoustic

modeling in speech recognition. HDPHMM is limited to only learning ergodic structure. Non-emitting states are also usually needed when modeling finite sequences. We introduce two different generative DHDPHMMs that incorporate these additional properties. Finally, an inference algorithm based on a block-sampler algorithm introduced by Fox et al. (2011) is presented.

Chapter 4: Semi-Supervised Training of DHDPHMM

In Chapter 3, we introduced a DHDPHMM and its corresponding inference algorithm. However, DHDPHMM does not address the problem of semi-supervised learning which is critical to the way we train speech recognizers. In Chapter 4, we introduce another generative model and its approximation to solve this problem. Moreover, the context modeling problem is discussed and two approaches to address this problem are presented.

Chapter5: Acoustic Unit And Lexicon Discovery

Acoustic unit discovery is a critical issue in many speech recognition applications where there are limited linguistic resources or where limited training data is available for the target language. Further, one can argue that phonemes are artificial units and are not often ideal to model the complexity of what we observe in the acoustic data. In Chapter 5, we study the possibility of using HDPHMM/DHDPHMM based transducers for problems of speaker independent segmentation and acoustic unit learning.

The segmentation problem is studied for a speaker diarization task (Fox et al., 2011) and for acoustic unit segmentation (Harati et al., 2013) previously. We explore the segmentation properties of HDPHMM and then investigate the validity of the discovered units through extensive experimentation. We show that the discovered units are related to English phonemes.

We also investigate the validity of the discovered units through a completely unsupervised spoken term detection (STD) task. We provide an algorithm to automatically discover a lexicon given only acoustic data and word-level transcriptions. We compare a speech recognizer trained

using the discovered lexicon and acoustic units to a speech recognizer trained using traditional linguistic resources.

Chapter 6: Conclusion

In this final chapter we summarize the dissertation results and our contributions. We will also provide some comments on possible directions for future research on these topics.

1.4 Dissertation Contributions

The contributions of this dissertation are:

- Defined a new nonparametric HMM, DHDPHMM, that enables sharing of mixture components across states;
- Derived an inference algorithm for DHDPHMM;
- Developed a model to learn non-ergodic DHDPHMM/HDPHMM structures;
- Extended DHDPHMM to include non-emitting states;
- Defined a generative model for semi-supervised training;
- Derived an approximate algorithm based on this model to train DHDPHMM models in semi-supervised settings;
- Applied HDPHMM/DHDPHMM to automatic discovery of speaker independent acoustic units which we call automatically discovered units (ADU);
- Developed an unsupervised approach for a spoken term detection (STD) task based on the ADU algorithm;
- Developed a semi-supervised automatic algorithm to learn a lexicon based on an ADU transducer.

The broader impact of these contributions is to introduce the nonparametric Bayesian methods to the acoustic modeling problem. We have shown that our proposed model, DHDPHMM, and its semi-supervised counterpart can compete with the state of the art

algorithms. For example, DHDPHMM works better than discriminatively trained models on certain tasks while trained only using maximum likelihood (ML). We have also shown some of the most difficult problems in speech recognition, such as discovering acoustic units and lexicons, can be approached based on models introduced in this dissertation.

It should be noted that the improvements we demonstrate in this initial research exceeded what was demonstrated initially for the applications of neural networks (Bourlard and Morgan, 1993) in acoustic modeling and is comparable to the initial performance of deep learning-based neural networks (Mohamed et al., 2009; Yu & Deng, 2010). Yet, this approach is much simpler than deep learning and results in less complex models. Therefore, we hope this research motivates others to continue investigating applications of nonparametric Bayesian methods in acoustic modeling and other pattern recognition applications.

CHAPTER 2

NONPARAMETRIC BAYESIAN BASICS

Parametric approaches have been used in machine learning and pattern recognition applications since the mid-1900's. The phrase "parametric" was coined by statistician Jacob Wolfowitz (1942):

> Most of these developments have this feature in common, that the distribution functions of the various stochastic variables which enter into their problems are assumed to be of known functional form, and the theories of estimation and of testing hypotheses are theories of estimation of and of testing hypotheses about, one or more parameters ..., the knowledge of which would completely determine the various distribution functions involved. We shall refer to this situation ... as the parametric case, and denote the opposite case, where the functional forms of the distributions are unknown, as the non-parametric case (Wolfowitz, 1942).

These approaches provide reasonable performance with a fixed amount of complexity (Gelman et al., 2004). For some time, it was generally believed that such models could be arbitrarily improved through the use of larger datasets (Bacchiani et al., 2008; Ma & Schwartz, 2008). However, performance gains have leveled off for a variety of reasons, including the complex recording conditions embodied in these massive datasets. Using more data to train models improves the estimation of individual parameters but it usually does not translate to overall better performance since the model itself is fixed. Nonparametric non-Bayesian approaches have been also used (e.g. decision trees) but it has been shown (Breiman et al., 1984; Bramer, 2007) that they are prone to overfitting of the training data. It is difficult to control the complexity of these models in a rigorous manner. A number of ad hoc algorithms (e.g. pruning in decision trees) have been used instead (Bramer, 2007).

Nonparametric Bayesian approaches make it possible to learn the model structure (and the degree of the complexity) from the data without the risk of overfitting the model to the observations by biasing the model toward simpler structures. Like all Bayesian approaches, nonparametric Bayesian approaches use Bayes rule to combine the prior distributions with the observations (e.g. likelihoods) to estimate the posterior distribution for the models. This posterior implicitly contains the structure we have learned from the data. Depending on how we define the prior distribution we can define an unlimited number of nonparametric Bayesian models. In this dissertation we are interested in a very specific type of prior based on the Dirichlet process, and therefore we restrict our discussion to this form of prior.

Mixture models are a very popular basic building block in many machine learning applications and provide a framework for generating arbitrarily complex models with good convergence properties. For example, mixture models are used extensively in HMMs. A Dirichlet distribution is a parametric prior used frequently in Bayesian approaches involving mixture models. In this chapter we will review the Dirichlet distribution and its application in Bayesian modeling, including the use of mixture distributions. We then will introduce a nonparametric counterpart in which we replace the Dirichlet distribution with a Dirichlet Process (DP). Dirichlet processes were among the first priors used in nonparametric Bayesian modeling (Teh, 2010). Beside their applications in mixture modeling problems they also have been used as a building block for many other nonparametric models including the Hierarchical Dirichlet Process (HDP) (Teh & Jordan, 2006) and the infinite HMM (iHMM) (Beal, 2002). The latter is also known as an HDPHMM (Teh et al., 2006; Fox et al., 2011). These DP-based models form the basis for the work presented in this dissertation.

2.1 The Dirichlet Distribution

Consider a random variable x over a finite K-dimensional space $X=\{1,2,\ldots,K\}$. The probability mass function in this space can be represented by a K-dimensional vector $\pi=(\pi_1,\pi_2,\ldots,\pi_K)$ where $0\le\pi_k\le 1$ and $\sum_{k=1}^{K}\pi_k=1$. This vector can characterize a multinomial distribution that is defined as:

$$p(x_1,x_2,\ldots,x_N\mid\pi)=\frac{N!}{\prod_k m_k!}\prod_k \pi_k^{m_k}, \text{ where } m_k \triangleq \sum_n \delta(x_n,k). \tag{2.1}$$

Equation (2.1) can be used to calculate the probability of selecting a category or class among K possible classes. In this definition m_k is the number of observations of category k. Given N observations, π can be estimated using a maximum likelihood (ML) approach (Sudderth, 2006). ML is a point estimate which means it does not estimate the posterior distribution, but rather it estimates an important point (e.g. mean) of this distribution.

In the case of a multinomial distribution, the results are empirical frequencies of discrete categories. For example, for a specific observation, the probability of each category can be calculated by dividing the number of samples in that category by the total number of samples:

$$\hat{\pi}=\arg\max_{\pi}\sum_{n=1}^{N}\log p(x_n\mid\pi)=\left(\frac{m_1}{N},\frac{m_2}{N},\ldots,\frac{m_K}{N}\right). \tag{2.2}$$

However, if the number of data points is not large enough, ML estimation of π will have a high variance (e.g. the estimated value varies around the real value by a large amount) and some categories may even have a zero probability. Estimating zero probability for an event means we believe that event will never happen. In practice many events of interests are rare but have a nonzero probability of occurrence. Estimating a zero probability for the event is generally a bad idea. Over the years, particularly with language modeling, smoothing of these distributions has proven to be a better alternative (Zhai & Lafferty, 2004; Jelinek & Mercer, 1980).

An example of this problem is *N*-gram modeling of phonemes. For instance, consider the problem of finding the probability of *3*-grams of phonemes occurring in English. Given a finite amount of text, many *3*-grams will never be observed. If we model the problem using a multinomial distribution and use an ML approach to estimate the occurrence probabilities, the result will contain many zeroes or unrealistically small numbers. The estimated value for the probability of each *3*-gram will be a point estimate, in this case the mean, of the underlying distribution for these parameters.

An alternative approach is to infer π using a Bayesian approach (Gelman et al., 2004). We should define a prior on π in such a way that a posterior inferred by multiplying the prior and likelihoods remain in the same family of distributions. In Bayesian statistics, this particular property is named conjugacy (Gelman et al., 2004) and the prior is called a conjugate prior for the likelihood.

For example, the conjugate prior for the Gaussian distribution with known covariance is itself a Gaussian distribution. Consider *N* Gaussian observations $x_1,x_2,\ldots,x_N$. Suppose the covariance matrix Σ is known. We can place a normal prior over the mean with mean μ_0 and covariance Σ_0. This prior is indicated with $Norm(\mu_0,\Sigma_0)$. After observing *N* data points the posterior over the mean is found using Bayes Rule and given by:

$$p(\mu \mid x_1,\ldots,x_N,\Sigma,\mu_0,\Sigma_0) = Norm\left(\left(\Sigma_0^{-1}+\Sigma^{-1}\right)^{-1}+\left(\Sigma_0^{-1}\mu_0+\Sigma^{-1}\sum_{i=1}^{N}x_i\right),\left(\Sigma_0^{-1}+\Sigma^{-1}\right)^{-1}\right). \qquad (2.3)$$

In the case of a multinomial distribution, the conjugate distribution is a Dirichlet distribution (Teh, 2010):

$$P(\pi \mid \alpha) = Dir(\alpha_1,\alpha_2,\ldots,\alpha_K) \triangleq \frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma(\alpha_k)} \prod_k \pi_k^{\alpha_k - 1}, \text{ for } \alpha_k > 0. \qquad (2.4)$$

In this definition Γ is the gamma function and defined by:

$$\Gamma(z) = \int_0^\infty t^z e^{-t} \frac{dt}{t} . \tag{2.5}$$

A Gamma function is an extension of a factorial function to real and complex numbers (Milton et al., 1974). The concentration parameter, α, in (2.4) is proportional to the inverse of the variance (Teh, 2010).

Figure 2-1 shows a Dirichlet distribution for several values of α. The distribution depicted in this figure has three dimensions but since the sum of three probabilities is constrained to be one, we can visualize the distribution using only two dimensions. As Figure 2-1 shows, a larger value for α means the distribution has a higher concentration around the mean. In this particular example, α = (*1*,*1*,*1*) is equivalent to a uniform distribution. We can also see (2.4) places a probability distribution over π which itself is a probability distribution.

A Dirichlet distribution, like all other discrete distributions, can be represented by two sets of parameters: locations of the impulse functions and their corresponding weights. The impulse functions are often referred to as "atoms". For example, in a binomial distribution, there are exactly *2* atoms, *x = 0* and *x = 1*, and two corresponding weights, *P(x=0)* and *P(x=1)*. We can have an infinite number of atoms for a Dirichlet distribution.

The mean of Dirichlet distribution is given by:

$$E_\alpha[\pi_k] = \frac{\alpha_k}{\sum_j \alpha_j} . \tag{2.6}$$

If the parameter α is set symmetrically (e.g. set to equal values for all *K* dimensions):

$$\alpha_k = \frac{\sum_j \alpha_j}{K}, \tag{2.7}$$

and the variance of the distribution is given by (Gelman et al., 2004):

$$Var_\alpha[\pi_k] = \frac{K-1}{K^2\left(\sum_j \alpha_j + 1\right)} . \tag{2.8}$$

Equation (2.8) clearly shows that the variance of the Dirichlet distribution is inversely proportional to the concentration parameter α. This is demonstrated in Figure 2-1. Therefore, a Dirichlet distribution with a small concentration parameter (but greater than one) means a weak prior belief while large values of α means high confidence. Values less than one means we believe in extremes of the distribution (e.g. the distribution peaks around the corners).

Given some data we can obtain a posterior distribution for π using Bayes rule (by multiplying the prior and likelihood):

$$p(\pi \mid x_1, x_2, ..., x_N, \alpha) \propto p(\pi \mid \alpha)\, p(x_1, x_2, ..., x_N \mid \pi) \ . \tag{2.9}$$

By substituting from (2.1) and (2.4) we can write:

$$p(\pi \mid x_1, x_2, ..., x_N, \alpha) \propto \prod_{k=1}^{K} \pi_k^{\alpha_k + m_k - 1} \propto Dir(\alpha_1 + m_1, ..., \alpha_K + m_K) \ . \tag{2.10}$$

Equation (2.10) unlike (2.2) gives a distribution over π. The parameters of this distribution are learned from the observed data. However the learning process is influenced by the prior

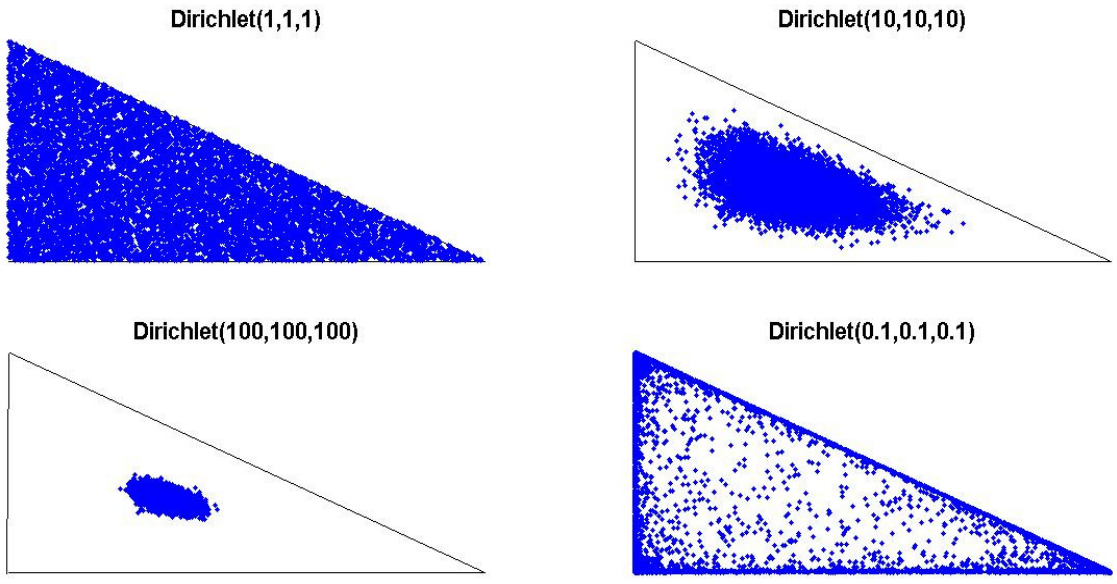


Figure 2-1. Dirichlet distributions are shown with different concentration parameters. A higher value of concentration parameter means the distribution has a higher concentration around the mean.

assumptions and the amount of data. When the amount of training data is small, we rely on the priors. As the amount of training data increases, the influence of the prior diminishes.

From (2.10) we can see α_k acts as a pseudo observation. A pseudo observation is a term used to weight our belief in the prior knowledge. It should be noted that when placing a symmetric concentration parameter on the prior we are in fact expressing belief in the equal probability of all outcomes. Mathematically pseudo observations act as actual observations though they are not really observed. Hence, we refer to them as a pseudo observations. The total number of pseudo observations, α_0, is equal to the sum of α_k:

$$\alpha_0 = \sum_k \alpha_k \ . \tag{2.11}$$

By considering this fact and (2.8) we can see the variance of the estimate decreases by increasing the number of pseudo observations. This reduction in variance comes with the cost of increasing the bias. In statistical modeling, we usually must trade off variance and bias. Obviously, we prefer an estimator with zero bias and variance (Bishop, 2007), but it is often impossible to achieve this in practice. In the case of the Bayesian estimator in (2.10) the bias of the estimator asymptotically decreases as more data becomes available, thereby making the estimator asymptotically unbiased, or as it is often described, consistent.

The predictive distribution for a new observation, which is the distribution of unseen data given observed data and priors, can be written using (2.1) and (2.10):

$$p\left(x_{new} \mid x_1,...,x_N,\alpha\right) = \frac{m_k + \alpha_k}{N + \sum_j \alpha_j} \ . \tag{2.12}$$

An example of the above discussion can be seen in language modeling for document retrieval. A language model assigns a probability to a document. One simple unigram language model is a multinomial language model (e.g. bag of words). If we define the language model for a document (D) as π_D, then for a sequence of independent terms we can write:

$$p\left(T_1,...,T_N \mid \pi_D\right) = \prod_{i=1}^{N} p\left(T_i \mid \pi_D\right) \ . \tag{2.13}$$

In this equation each $p(T_i \mid \pi_D)$ is a multinomial distribution.

Now consider a search engine application where we have some number of documents and a goal of finding the most relevant documents given a query. For each document *D*, we have to compute (2.13). To compute this probability, we have to compute π_D for all terms in the query. If we use the maximum likelihood solution in (2.2), we might get a zero probability for a document if one of the terms does not exist in the document. Obviously, it is not an acceptable solution for a search engine application. On the other hand, estimating π_D using a Dirichlet distribution as shown in (2.10) will solve this problem since it always gives a nonzero probability even if some of the terms are not presented in a document.

One of the main applications of Dirichlet distribution is in mixture modeling. Any convex combination of distribution functions that follow

$$\sum_{i=1}^{K} \pi_i f_i(x), \text{ where } \sum_{i=1}^{K} \pi_i = 1, \tag{2.14}$$

can be considered a mixture distribution. In a Bayesian setting, π is a draw from a Dirichlet distribution:

$$\pi \mid K \sim Dir(\alpha_1, ..., \alpha_k). \tag{2.15}$$

Then a generative mixture model can be defined as:

$$\begin{aligned} &\pi \mid \alpha \sim Dir(\alpha_1, ..., \alpha_K) \\ &z_i \mid \pi \sim Mult(\pi) \\ &\theta_k \mid G_0 \sim G_0 \\ &x_i \mid z_i, \{\theta_k\} \sim F(\theta_{z_i}). \end{aligned} \tag{2.16}$$

In this definition z_i is an indicator variable that shows the mixture component. G_0 is a continuous function from which the parameters of density function are sampled. For example, if density functions are assumed to be Gaussian and we restrict ourselves to conjugate priors, G_0 would be a Normal-Inverse-Wishart (NIW) distribution.

2.2 Dirichlet Process

A Dirichlet process (DP) is the generalization of Dirichlet distribution to an infinite number of atoms. DP is a distribution over distributions, or more precisely over discrete distributions. Formally, a Dirichlet process, $DP(\alpha,G_0)$, is "defined to be the distribution of a random probability measure G over Θ such that for any finite measurable partition $(A_1,A_2,\ldots,A_r)$ of Θ the random distribution $[G(A_1),\ldots,G(A_r)]$ is distributed as finite dimensional Dirichlet distribution" (Teh et al., 2006):

$$\left[G(A_1),...,G(A_r)\right] \sim Dir\left(\alpha G_0(A_1),...,\alpha G_0(A_r)\right) . \tag{2.17}$$

In this definition α is the concentration parameter and is proportional to the inverse of the variance. G_0 is the base distribution and is equal to the mean of the DP (e.g. $E\left(G(A)\right) = G_0(A)$).

A constructive definition for a Dirichlet process, given by Sethuraman (1994), is known as the Griffiths, Engen and McCloskey (GEM) construction, or the stick-breaking construction. This construction explicitly shows that samples from a DP are discrete with probability one:

$$\begin{aligned} &v_k \mid \alpha, G_0 \sim Beta(1,\alpha), \quad \theta_k \mid \alpha, G_0 \sim G_0, \\ &\beta_k = v_k \prod_{l=1}^{k-1}(1-v_l), \qquad G = \sum_{k=1}^{\infty} \beta_k \delta_{\theta_k} . \end{aligned} \tag{2.18}$$

Starting with a stick of length one, we break it at v_1 and assign the length to β_1. Then we recursively break the remaining part of the stick and assign the corresponding lengths to β_k. In this representation β can be interpreted as a random probability measure over positive integers and is denoted by $\beta \sim GEM(\alpha)$.

Another representation of a DP is the Polya urn process (Teh et al., 2006). In this approach, we consider i.i.d. samples from a DP and consider the predictive distribution over these draws:

$$\theta_i \mid \theta_1,...,\theta_{i-1},\alpha,G_0 \sim \sum_{k=1}^{i-1} \frac{1}{N-1+\alpha}\delta_{\theta_k} + \frac{\alpha}{N-1+\alpha} G_0 . \tag{2.19}$$

In the urn interpretation of (2.19), we have an urn containing several balls of different colors. We draw a ball and put it back in the urn and add another ball of the same color to the urn. With probability proportional to α we draw a ball with a new color.

To make the clustering property more clear, we should introduce a new set of variables that represent distinct values of the atoms (e.g. observed balls). Let $\theta_1^*,...,\theta_K^*$ be the distinct values and m_k be the number of θ_l associated with θ_k^*. We now have:

$$\theta_i \mid \theta_1,...,\theta_{i-1},\alpha,G_0 \sim \sum_{k=1}^{K} \frac{m_k}{i-1+\alpha}\delta_{\theta_k^*} + \frac{\alpha}{i-1+\alpha}G_0 \ . \tag{2.20}$$

Another useful interpretation of (2.20) is the Chinese restaurant process (CRP). In a CRP we have a Chinese restaurant with infinite number of tables. A new customer comes into the restaurant and can either sit at one of the occupied tables with probability proportional to the number of people already sitting there (m_k) or initiate a new table with probability proportional to α. In this analogy, each customer is a data point and each table is a cluster. Let z_i indicate the cluster associated with i^{th} observation. A CRP is the interpretation of the predictive distribution:

$$p\left(z_{N+1}=z \mid z_1,....,z_N,\alpha\right)=\frac{1}{\alpha+N}\left(\sum_{k=1}^{K} m_k\delta\left(z,k\right)+\alpha\delta\left(z,\bar{k}\right)\right) . \tag{2.21}$$

New customers, represented by z_{N+1}, tend to sit around crowded tables and eat the food served on that table with probabilities proportional to the number of people at that table, represented by m_k. The model tends to discover large clusters. However, sometimes with probability proportional to α, a customer sits at a new table (shown with $\bar{k}$). The model allows this new modality, or cluster, to be formed without readjusting the other existing clusters that presumably reflect valid modalities in the data. The CRP analogy demonstrates how data can be generated in a Dirichlet Process Mixture (DPM) model.

As an illustrative example, consider the problem of automatic acoustic unit discovery. Given a set of segments, the goal is to cluster the segments into a set of units that are collections of these

segments. However, the number of units is not known a priori. If we think of each "segment" as a customer then we see CRP acts as a prior distribution over the units.

A Dirichlet Process Mixture is defined as:

$$\begin{aligned} &\pi \mid \alpha \sim GEM(\alpha) \\ &z_i \mid \pi \sim Mult(\pi) \\ &\theta_k \mid G_0 \sim G_0 \\ &x_i \mid z_i, \{\theta_k\} \sim F(\theta_{z_i}) \ . \end{aligned} \tag{2.22}$$

In this model, observations, x_i, are sampled from an indexed family of distributions denoted by F. If F is assumed to be Gaussian then the result is an infinite Gaussian mixture model.

Figure 2-2 show a graphical model for DPM. It should be noted that a CRP induces priors that prefer simpler models (e.g. a small number of tables with a large number of customers per table) which means number of discovered units would be much smaller than the number of observed segments.

2.3 Hierarchical Dirichlet Process

A Hierarchical Dirichlet Process (HDP) is the natural extension of a Dirichlet process to problems with multiple groups of data. Usually, data is split into J groups a priori. For example, consider a collection of documents. If words are considered as data points, each document would be a group. We want to model data inside a group using a mixture model. However, we are also

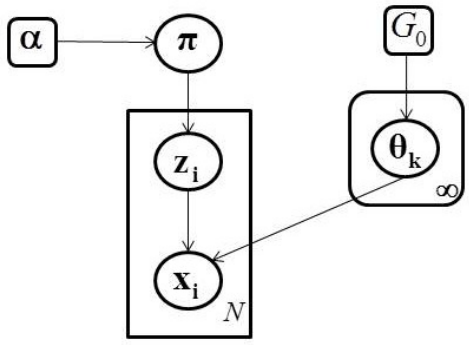


Figure 2-2. A graphical model for Dirichlet mixture model is shown.

interested in tying these mixtures together, i.e. to share clusters across all groups. Let's assume that we have an indexed collection of DPs with a common base distribution $\{G_j\} \sim DP(\alpha, G_0)$. Unfortunately this simple model cannot solve the problem since for continuous G_0 different G_j have no atoms in common. Samples from a continuous function are distinct with probability one.

The solution is to use a discrete G_0 with infinite (broad) support. One such choice is to draw G_0 from a Dirichlet process. An HDP is defined by (Teh & Jordan, 2006):

$$\begin{aligned} &G_0 \mid \gamma, H \sim DP(\gamma, H) \\ &G_j \mid \alpha, G_0 \sim DP(\alpha, G_0) \\ &\theta_{ji} \mid G_j \sim G_j \\ &x_{ji} \mid \theta_{ji} \sim F\left(\theta_{ji}\right) \quad \text{for } j \in J. \end{aligned} \tag{2.23}$$

In this definition H represents a prior distribution for the factor θ_{ji}. The parameter γ governs the variability of G_0 around H and α controls the variability of G_j around G_0. H, γ and α are hyperparameters of the HDP. Equation (2.23) is just one of several possible representations of an HDP (Teh & Jordan, 2006).

Another representation can be obtained by introducing an indicator variable, β:

$$\begin{aligned} &\beta \mid \gamma \sim GEM(\gamma) \\ &\pi_j \mid \alpha, \beta \sim DP(\alpha, \beta) \\ &\theta_k \mid H, \lambda \sim H(\lambda) \\ &z_{ji} \mid \pi_j \sim \pi_j \\ &x_{ji} \mid \{\theta_k\}_{k=1}^{\infty}, z_{ji} \sim F\left(\theta_{ji}\right). \end{aligned} \tag{2.24}$$

Both of these representations are depicted in Figure 2-3. Similar to DP, HDP also has a stick-breaking construction that can be used to obtain an inference algorithm. This is discussed next.

2.3.1 Stick-Breaking Construction

Because G_0 is a Dirichlet distribution, it has a stick-breaking representation:

$$G_0 = \sum_{k=1}^{\infty} \beta_k \delta_{\theta_k^{**}}, \tag{2.25}$$

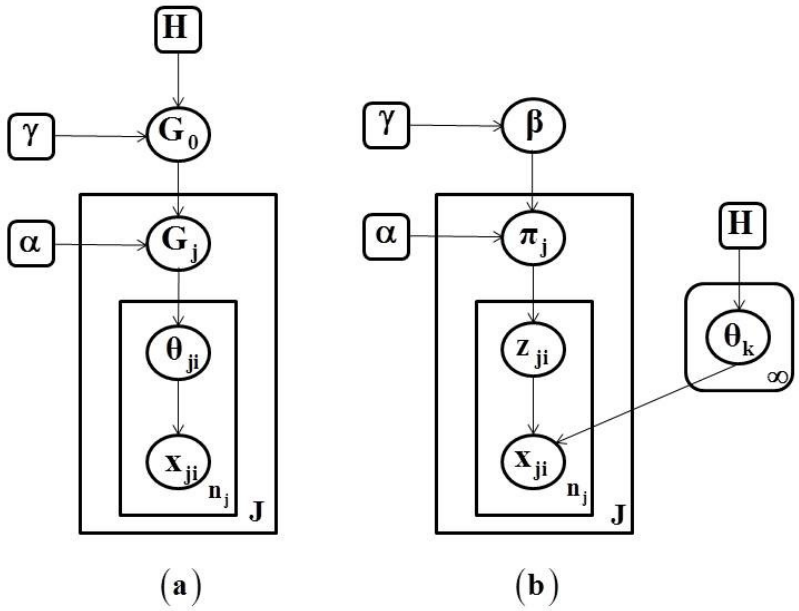


Figure 2-3. An HDP representation of (2.23) is shown in (a). An alternative indicator variable representation from (2.24) is shown in (b) (adapted from Teh et al., 2004).

where $\theta_k^{**} \sim H$ and $\beta = (\beta_k)_{k=1}^{\infty} \sim GEM(\gamma)$. Since support of G_j is contained within the support of G_0, we can write a similar equation to (2.25) for G_j:

$$G_j = \sum_{k=1}^{\infty} \pi_{jk} \delta_{\theta_k^{**}}. \tag{2.26}$$

Then we have:

$$\pi_j \sim DP(\alpha, \beta) \tag{2.27}$$

$$\begin{aligned} v_{jk} &\sim Beta\left(\alpha\beta_k, \alpha\left(1 - \sum_{l=1}^{k} \beta_l\right)\right) \\ \pi_{jk} &= v_{jk} \prod_{l=1}^{k-1} (1 - v_{jl}), \text{ for } k = 1, ..., \infty. \end{aligned} \tag{2.28}$$

The Chinese restaurant franchise (CRF) is the natural extension of Chinese restaurant process for HDPs. In a CRF, we have a franchise with several restaurants and a franchise-wide menu. The first customer in restaurant j sits at one of the tables and orders an item from the menu. Other customers either sit at one of the occupied tables and eat the food served at that table or sit at a

new table and order their own food from the menu. Moreover, the probability of sitting at a table is proportional to the number of customers already seated at that table. In this analogy, restaurants correspond to groups. Customer i in restaurant j corresponds to θ_{jt} (customers are distributed according to G_j). Tables are i.i.d. variables θ_{jt}^{*} distributed according to G_0. Finally, foods are i.i.d. variables represented by θ_{k}^{**}, and distributed according to H. If customer i at restaurant j sits at table t_{ji}, and that table serves dish k_{ji}, we have $\theta_{ji} = \theta_{jt_{ji}}^{*} = \theta_{k_{t_{ji}}}^{**}$. Each restaurant represents a simple DP and therefore a cluster over data points. At the franchise level we have another DP, but this time clustering is over tables.

Next, we can introduce several variables that will be used throughout this thesis: n_{jkt} is the number of customers in restaurant j, seated around table t, and who eat dish k; m_{jk} is the number of tables in restaurant j serving dish k and K is the number of unique dishes served in the entire franchise. Marginal counts are denoted with dots.

A CRF can be characterized by its state, which consists of dish labels $\boldsymbol{\theta}^{**} = \left\{\theta_{k}^{**}\right\}_{k=1,\ldots,K}$, tables $\left\{t_{ji}\right\}_{\substack{j=1,\ldots,J \\ i=1,\ldots,n_{\cdot\cdot}}}$ and dishes $\left\{k_{jt_{ji}}\right\}_{\substack{j=1,\ldots,J \\ i=1,\ldots,n_{\cdot\cdot}}}$. As a function of the state of the CRF, we also have the number of customers, $\mathbf{n} = \left\{n_{jtk}\right\}$, the number of tables, $\mathbf{m} = \left\{m_{jk}\right\}$, customer labels $\boldsymbol{\theta} = \left\{\theta_{ji}\right\}$ and table labels $\boldsymbol{\theta}^{*} = \left\{\theta_{jt}^{*}\right\}$ (Teh & Jordan, 2010). The posterior distribution of G_0 is given by:

$$G_0 \mid \gamma, H, \boldsymbol{\theta}^{*} \sim DP\left(\gamma + m_{\cdot\cdot}, \frac{\gamma H + \sum_{k=1}^{K} m_{\cdot k}\delta_{\theta_k^{**}}}{\gamma + m_{\cdot\cdot}}\right), \tag{2.29}$$

where $m_{\cdot\cdot}$ is the total number of tables in the franchise and $m_{\cdot k}$ is the total number of tables serving dish k. We can define the posterior for G_j:

$$G_j \mid \alpha, G_0, \boldsymbol{\theta}_{\mathbf{j}} \sim DP\left(\alpha + n_{j\cdot\cdot}, \frac{\alpha G_0 + \sum_{k=1}^{K} n_{j\cdot k}\delta_{\theta_k^{**}}}{\alpha + n_{j\cdot\cdot}}\right), \tag{2.30}$$

where $n_{j\bullet\bullet}$ is the total number of customers in restaurant j and $n_{j\bullet k}$ is the total number of customers in restaurant j eating dish k.

Conditional distributions can be obtained by integrating out G_j and G_0 respectively. By integrating out G_j from (2.30) we obtain:

$$\theta_{ji} \mid \theta_{j1},...,\theta_{j,i-1},\alpha,G_0 \sim \sum_{t=1}^{m_{j\bullet}} \frac{n_{jt\bullet}}{\alpha+n_{j\bullet\bullet}} \delta_{\theta^*_{jt}} + \frac{\alpha}{\alpha+n_{j\bullet\bullet}} G_0 , \tag{2.31}$$

and by integrating out G_0 from (2.29) we obtain:

$$\theta^*_{jt} \mid \theta^*_{j1},...,\theta^*_{j,t-1},\ \gamma,H \sim \sum_{k=1}^{K} \frac{m_{\bullet k}}{\gamma+m_{\bullet k}} \delta_{\theta^{**}_k} + \frac{\gamma}{\gamma+m_{\bullet\bullet}} H \ . \tag{2.32}$$

A draw from (2.29) can be obtained using:

$$\begin{aligned} &\beta_0,\beta_1,...,\beta_K \mid \gamma,G_0,\boldsymbol{\theta}^* \sim Dir\left(\gamma,m_{\bullet 1},...,m_{\bullet K}\right) \\ &G_0' \mid \gamma,H \sim DP\left(\gamma,H\right) \\ &G_0 = \beta_0 G_0' + \sum_{k=1}^{K} \beta_k \delta_{\theta^{**}_k}, \end{aligned} \tag{2.33}$$

and a draw from (2.30) can be obtained using:

$$\begin{aligned} &\pi_{j0},\pi_{j1},...,\pi_{jK} \mid \alpha,\boldsymbol{\theta}_{\mathbf{j}} \sim Dir(\alpha\beta_0,\alpha\beta_1+n_{j\bullet 1},...,\alpha\beta_K+n_{j\bullet K}), \\ &G_j' \mid \alpha,G_0 \sim DP(\alpha\beta_0,G_0') , \\ &G_j = \pi_{j0} G_j' + \sum_{k=1}^{K} \pi_{jk} \delta_{\theta^{**}_k} . \end{aligned} \tag{2.34}$$

From (2.33) and (2.34) we see that the posterior of G_0 is a mixture of atoms corresponding to dishes, and is an independent draw from $DP(\gamma,H)$. Similarly, G_j is a mixture of atoms at θ^{**}_k and an independent draw from $DP(\alpha\beta_0,G_0')$ (Teh & Jordan, 2010).

2.4 HDPHMM

Hidden Markov models (HMMs) are a class of doubly stochastic processes in which discrete state sequences are modeled as a Markov chain (Rabiner, 1989). In the following discussion we

will denote the state of the Markov chain at time t with z_t and the state-specific transition distribution for state j by π_j. The Markovian structure is represented by $z_t \mid z_{t-1} \sim \pi_{z_{t-1}}$. Observations are conditionally independent given the state of an HMM and are denoted by $x_t \mid z_t \sim F\left(\theta_{z_t}\right)$. In a typical fully ergodic HMM, the number of states is fixed and a matrix of dimension N states by N transitions per state is used to represent the transition probabilities.

An HDPHMM is an extension of an HMM in which the number of states can be infinite. At each state z_t we should be able to transition to an infinite number of states so the transition distribution should be a draw from a DP. On the other hand, we want reachable states from one state to be shared among all states so these DPs should be linked together. The result is an HDP. In an HDPHMM each state corresponds to a group (restaurant) and therefore, unlike HDP in which an association of data to groups is assumed to be known a priori, we are interested in inferring this association.

A major problem with original formulation of an HDPHMM is state persistence. HDPHMM has a tendency to make many redundant states and switch rapidly among them (Teh et al., 2006). This problem has been solved by introducing a sticky parameter, κ, to the definition of an HDPHMM (Fox et al., 2011):

$$
\begin{aligned}
&\beta \mid \gamma \sim GEM(\gamma) \\
&\pi_j \mid \alpha, \beta \sim DP(\alpha + \kappa, \frac{\alpha\beta + \kappa\delta_j}{\alpha + \kappa}) \\
&\psi_j \mid \sigma \sim GEM(\sigma) \\
&\theta_{kj}^{**} \mid H, \lambda \sim H(\lambda) \\
&z_t \mid z_{t-1}, \left\{\pi_j\right\}_{j=1}^{\infty} \sim \pi_{z_{t-1}} \\
&s_t \mid \left\{\psi_j\right\}_{j=1}^{\infty}, z_t \sim \psi_{z_t} \\
&x_t \mid \left\{\theta_{kj}^{**}\right\}_{k,j=1}^{\infty}, z_t \sim F\left(\theta_{z_t s_t}\right) .
\end{aligned}
\tag{2.35}
$$

The state, mixture component and observations are represented by z_t, s_t and x_t respectively. The indices j and k are indices of the state and mixture components respectively. The base distribution

that links all DPs together is represented by β and can be interpreted as the expected value of state transition distributions. The transition distribution for state j is a DP denoted by π_j with a concentration parameter α. Another DP, ψ_j, with a concentration parameter σ, is used to model an infinite mixture model for each state z_j.

The distribution H is the prior for the parameters θ_{kj}. If we want the posterior distribution over the parameters to remain in the same family as the prior, then H should be chosen to be a conjugate prior to the observation likelihood. Since the likelihood has a multivariate normal distribution, H should have normal inverse Wishart (NIW) distribution. Figure 2-4 shows a graphical representation of model in (2.35).

The generative definition in (2.35) does not clarify how to estimate the actual model given a finite amount of data. Several algorithms have been proposed (Beal et al., 2002; Teh et al., 2006). We will use a block sampler proposed by Fox et al. (2011) as the basis for our inference algorithm for models presented in subsequent chapters.

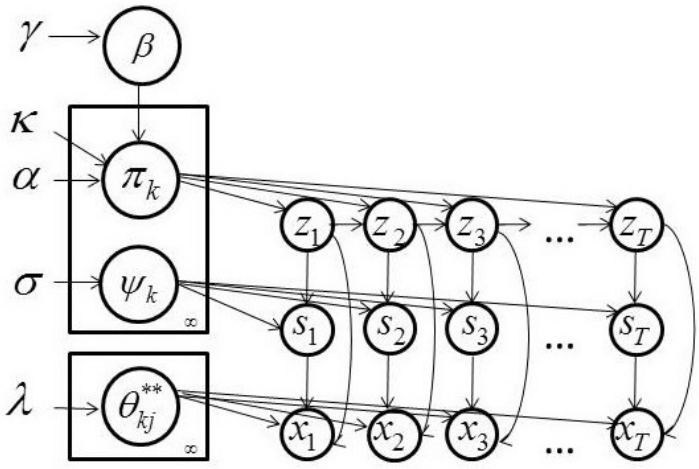


Figure 2-4. A graphical model of HDPHMM is shown (adapted from Fox, et al., 2011).

2.4.1 Block Sampler

A block sampler is based on using the Markovian structure of the model to improve the performance of the inference algorithm. A variant of Forward-Backward (FB) procedure (Ferguson, 1980) is incorporated in the sampling algorithm that enables us to sample the state sequence $z_{1:T}$ at once. This is an important computational issue since sampling each data point separately using Gibbs sampling significantly increases the computation time required by the inference algorithm. To achieve this goal, a fixed truncation level L should be used which in a sense reduces the model to a parametric model (Fox et al., 2011). However, it should be noted that the result is different from a classical parametric Bayesian HMM since the truncated HDP priors induce a shared sparse subset of the L possible states. In short, we obtain an approximation to the nonparametric Bayesian HDPHMM with the maximum number of possible states set to L. For most applications this does not cause a problem as long as we set L reasonably high. It is easy to determine experimentally what is a sufficiently large value of L.

The approximation used in this algorithm is the degree L weak limit approximation to the DP (Ishwaran & Zarepour, 2002), which is defined as:

$$GEM_L(\alpha) \triangleq Dir(\alpha / L,...,\alpha / L) \ . \tag{2.36}$$

Using (2.36) β is approximated as (Fox et al., 2010):

$$\beta \mid \gamma \sim Dir(\gamma / L,...,\gamma / L) \ . \tag{2.37}$$

We can write:

$$\pi_j \mid \alpha, \kappa, \beta \sim Dir(\alpha\beta_1,...,\alpha\beta_j + \kappa,...\alpha\beta_L) \ . \tag{2.38}$$

The posteriors are given by:

$$\begin{aligned} &\beta \mid \bar{\mathbf{m}}, \gamma \sim Dir(\gamma / L + \bar{m}_{\cdot 1},...,\gamma / L + \bar{m}_{\cdot L}) \\ &\pi_j \mid z_{1:T}, \alpha, \beta \sim Dir(\alpha\beta_1 + n_{j1},...,\alpha\beta_j + \kappa + n_{jj},...,\alpha\beta_L + n_{jL}) \ . \end{aligned} \tag{2.39}$$

The number of data that transitions from state j to state k is represented by n_{jk} where j and k both range from *1* to *L*. The data that is associated with a transition from state j to state k are clustered into m_{jk} groups. The back-off variable, $\bar{m}_{jk}$, which is a modified estimate of the number of clusters in the j^{th} state that transition to the k^{th} state, is defined as:

$$\bar{m}_{jk} = \begin{cases} m_{jk} & j \neq k, \\ m_{jj} - \omega_{j\cdot} & j = k. \end{cases} \tag{2.40}$$

This back-off variable is necessary because the addition of κ, the sticky parameter, introduces a bias.

We can explain this via the CRF metaphor. Without the sticky parameter, κ, the number of tables in restaurant j that order dish k are m_{jk}. However, κ introduces a preference for the restaurant specialty dish. It should be noted that each restaurant has one special dish that is also indexed with the same index as the restaurant (e.g. for restaurant j the special dish is indexed with j). A table, which is constrained to have the same dish served to all customers, considers a particular dish. Its initial decision might be overridden by the restaurant specialty dish, such as the feature of the day. This override procedure is simulated by a Bernoulli variable that decides whether to override the decision by tossing a coin with probability $\rho = \frac{\kappa}{\kappa + \alpha}$, where κ is the sticky parameter and α is the concentration parameter. The total number of overrides in restaurant j is represented by $\omega_{j\cdot}$. Therefore to obtain the total number of tables in restaurant j that considers dish k, before overriding happens, we need to reduce m_{jj} by $\omega_{j\cdot}$.

Finally an order L' weak limit approximation is used for the DP prior on the emission parameters:

$$\psi_k \mid z_{1:T}, s_{1:T}, \sigma \sim Dir\left(\sigma / L' + n'_{k1}, ..., \sigma / L' + n'_{kL'}\right). \tag{2.41}$$

The FB algorithm for the joint sample $z_{1:T}$ and $s_{1:T}$ given $x_{1:T}$ can be obtained by:

$$p(z_t, s_t \mid x_{1:T}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\psi}, \boldsymbol{\theta}) \propto p(z_t \mid z_{t-1}, x_{1:T}, \boldsymbol{\pi}, \boldsymbol{\theta})\, p(s_t \mid \psi_{z_t})\, p(x_t \mid \theta_{z_t, s_t}) \times p(x_{t+1:T} \mid z_t, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\psi}) \quad (2.42)$$

The right side of (2.42) has two parts: forward and backward probabilities (Rabiner, 1989). The forward probability includes $p(z_t \mid z_{t-1}, x_{1:T}, \boldsymbol{\pi}, \boldsymbol{\theta})\, p(s_t \mid \psi_{z_t})\, f(x_t \mid \theta_{z_t, s_t})$ and the backward probability includes $p(x_{t+1:T} \mid z_t, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\psi})$. Therefore, for the backward probabilities we have:

$$p(x_{t+1:T} \mid z_t, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\psi}) \propto A_{t,t-1}(z_{t-1}) \propto \begin{cases} \sum_{z_t} \sum_{s_t} p(z_t \mid \pi_{z_t - 1})\, p(s_t \mid \psi_{z_t}) f(x_t \mid \theta_{z_t, s_t})\, A_{t+1,t}(z_t) & t \le T \\ 1 & t = T+1 \end{cases} \quad (2.43)$$

$$A_{t,t-1}(z_{t-1}) \propto \begin{cases} \sum_{i=1}^{L} \sum_{l=1}^{L'} \pi_{ki} \psi_{il} f(x_t \mid \theta_{z_t, s_t})\, A_{t+1,t}(z_t) & t \le T, k = 1, \ldots L \\ 1 & t = T+1. \end{cases} \quad (2.44)$$

The variable $A_{t,t-1}$ is referred to as the backward message (Bishop, 2007). As a result we have (Fox et al., 2011):

$$p(z_t = k, s_t = j \mid x_{1:T}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\psi}, \boldsymbol{\theta}) \propto \pi_{z_{t-1} k} \psi_{kj} f(x_t \mid \theta_{z_t, s_t})\, A_{t+1,t}(z_t). \quad (2.45)$$

For Gaussian emission distributions, the components are given by $f(x_t \mid \theta_{z_t, s_t}) = \mathrm{N}(x_t; \mu_{kj}, \Sigma_{kj})$.

The complete block-sampler algorithm for estimation of HDPHMM is as follow (Fox et al., 2010):

1. Initialize with the previous of $\boldsymbol{\pi}^{(n-1)}, \boldsymbol{\psi}^{(n-1)}, \boldsymbol{\beta}^{(n-1)}$ and $\boldsymbol{\theta}^{(n-1)}$.

2. Initialize $A_{T+1,T}(k)$:

$$A_{T+1,T}(k) = 1, \text{ for } k = 1, 2, \ldots, L. \quad (2.46)$$

3. For $t \in \{T-1, \ldots, 1\}$ and $k \in \{1, \ldots, L\}$ compute:

$$A_{t,t-1}(k) = \sum_{i=1}^{L} \sum_{l=1}^{L} \pi_{ki} \psi_{il} N(x_{t+1}; \mu_{il}, \Sigma_{il}) A_{t+1,t}(i). \quad (2.47)$$

4. Sample the augmented state (z_t, s_t) sequentially and start from *t=1:*

a) For $(i,k) \in \{1,...,L\} \times \{1,...,L\}$ and $(k,j) \in \{1,...,L\} \times \{1,...,L'\}$ set:

$$n_{ik} = 0, n'_{kj} = 0 \text{ and } \Upsilon_{kj} = \varnothing. \tag{2.48}$$

b) For all $(k,j) \in \{1,...,L\} \times \{1,...,L'\}$ compute:

$$f_{k,j}(x_t) = \pi_{z_{t-1},k} \psi_{k,j} N(x_t; \mu_{k,j}, \Sigma_{k,j}) A_{t+1,t}(k) \ . \tag{2.49}$$

c) Sample the augmented state (z_t, s_t):

$$(z_t, s_t) \sim \sum_{k=1}^{L} \sum_{j=1}^{L'} f_{k,j}(x_t) \delta(z_t, k) \delta(s_t, j) \ . \tag{2.50}$$

d) Increase $n_{z_{t-1} z_t}$ and $n'_{z_t s_t}$; add x_t to the cached statistics:

$$\Upsilon_{k,j} \leftarrow \Upsilon_{k,j} \oplus x_t \ . \tag{2.51}$$

5. Sample auxiliary variables by simulating a CRF:

a) For each $(j,k) \in \{1,...,K\}^2$ set m_{jk}=0 and n=0:

1. For each customer in restaurant j eating dish k ($i = 1,...,n_{jk}$), sample:

$$x \sim Ber\left(\frac{\alpha \beta_k + \kappa \delta(j,k)}{n + \alpha \beta_k + \kappa \delta(j,k)} \right) . \tag{2.52}$$

2. Increase n and if $x=1$ increase m_{jk}.

b) For each $j \in \{1,...,K\}$, sample the override variables in restaurant j:

$$\omega_{j\bullet} \sim Binomial\left(m_{jj}, \frac{\rho}{\rho + \beta_j (1-\rho)} \right), \rho = \frac{\kappa}{\alpha + \kappa} \ . \tag{2.53}$$

6. Update β using:

$$\beta \mid \bar{\mathbf{m}}, \gamma \sim Dir(\gamma / L + \bar{m}_{\bullet 1}, ..., \gamma / L + \bar{m}_{\bullet L}) . \tag{2.54}$$

7. For $k \in \{1,...,L\}$:

a) Sample π_k and ψ_k:

$$\pi_k \sim Dir\left(\alpha\beta_1 + n_{k1},...,\alpha\beta_k + \kappa + n_{kk},...,\alpha\beta_L + n_{kL}\right),$$
$$\psi_k \sim Dir\left(\sigma / L' + n'_{k1},...,\sigma / L' + n'_{kL'}\right). \quad (2.55)$$

b) For $j \in \{1,...,L'\}$ sample:

$$\theta_{k,j} \sim p\left(\theta \mid \lambda, \Upsilon_{k,j}\right). \quad (2.56)$$

8. Set $\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}, \boldsymbol{\psi}^{(n)} = \boldsymbol{\psi}, \beta^{(n)} = \beta$ and $\boldsymbol{\theta}^{(n)} = \boldsymbol{\theta}$.

9. Optionally sample hyperparameters σ, γ, α and κ. Note that hyperparameters influence the complexity but are not part of the model themselves so sampling them is optional.

The above algorithm represents a single iteration of the inference algorithm. In order to obtain, samples from the posterior distributions of the parameters we have to iterate for several times and discard the values from the first few iterations.

Fox et al. (2011) have shown that for the block sampler the number of iterations is significantly lower than a direct sampler. Also if we set the initial values for hyperparameters carefully (e.g. tune the values by running some pilot experiments) we can reduce this number even further because the algorithm will find the stable values more quickly. In our experiments with modified block samplers, described in Chapters 3-5, we have discarded the first *200* samples and used the next *200* samples to estimate the value of the parameters.

2.4.2 Learning Hyperparameters

The hyperparameters σ, γ, α and κ can also be estimated like other parameters of the model (Fox et al., 2010):

Posterior for ($\alpha + \kappa$):

Consider the predictive distribution for a CRF:

$$p\left(t_{ji} = t \mid \mathbf{t}^{-ji}, n_{jt\cdot}^{-ji}, \alpha, \kappa\right) \propto \begin{cases} n_{jt\cdot}^{-ji} & t \in \{t_1,...,m_{j\cdot}\}, \\ \alpha + \kappa & t = t_{new}. \end{cases} \quad (2.57)$$

This equation can be written using (2.31) and (2.35). A customer table assignment follows a DP with concentration parameter $\alpha + \kappa$. Antoniak (1974) has shown that if $\beta \sim GEM(\gamma)$ and $z_i \sim \beta$, then the distribution of the number of unique values of z_i resulting from N draws from β has the following form:

$$p(K \mid N, \gamma) = \frac{\Gamma(\gamma)}{\Gamma(\gamma + N)} s(N, K) \gamma^K, \tag{2.58}$$

where *s(N,K)* is the Stirling number of the first kind. Using these two equations the distribution of the number of tables in the restaurant *j* is as follows:

$$p(m_{j\cdot} \mid \alpha + \kappa, n_{j\cdot}) = s(n_{j\cdot\cdot}, m_{j\cdot})(\alpha + \kappa)^{m_{j\cdot}} \frac{\Gamma(\alpha + \kappa)}{\Gamma(\alpha + \kappa + n_{j\cdot})}. \tag{2.59}$$

The posterior over $\alpha + \kappa$ is given by:

$$\begin{aligned} p(\alpha + \kappa \mid m_{1\cdot}, \ldots, m_{J\cdot}, n_{1\cdot}, \ldots, n_{J\cdot}) &\propto p(\alpha + \kappa)\, p(m_{1\cdot}, \ldots, m_{J\cdot} \mid \alpha + \kappa, n_{1\cdot}, \ldots, n_{J\cdot}) \\ &\propto p(\alpha + \kappa) \prod_{j=1}^{J} p(m_{j\cdot} \mid \alpha + \kappa, n_{j\cdot}) \\ &\propto p(\alpha + \kappa) \prod_{j=1}^{J} s(n_{j\cdot}, m_{j\cdot})(\alpha + \kappa)^{m_{j\cdot}} \frac{\Gamma(\alpha + \kappa)}{\Gamma(\alpha + \kappa + n_{j\cdot})} \\ &\propto p(\alpha + \kappa)(\alpha + \kappa)^{m_{\cdot\cdot}} \prod_{j=1}^{J} \frac{\Gamma(\alpha + \kappa)}{\Gamma(\alpha + \kappa + n_{j\cdot})}. \end{aligned} \tag{2.60}$$

The reason for the last line is that $\prod_{j=1}^{J} s(n_{j\cdot}, m_{j\cdot})$ is not a function of $\alpha + \kappa$ and therefore can be ignored. By substituting $\beta(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x + y)} = \int_0^1 t^{x-1}(1-t)^{y-1} dt$ and by considering that $\Gamma(x+1) = x\Gamma(x)$, we obtain:

$$p(\alpha + \kappa \mid m_{1\cdot}, \ldots, m_{J\cdot}, n_{1\cdot}, \ldots, n_{J\cdot}) \propto p(\alpha + \kappa)(\alpha + \kappa)^{m_{\cdot\cdot}} \prod_{j=1}^{J} \left(1 + \frac{n_{j\cdot}}{\alpha + \kappa}\right) \int_0^1 r_j^{\alpha + \kappa} (1 - r_j)^{n_{j\cdot} - 1} dr_j. \tag{2.61}$$

Noting that we have placed a *Gamma(a,b)* prior on $\alpha + \kappa$ we can write:

$$p\left(\alpha+\kappa, r, s \mid m_{1\cdot},...,m_{J\cdot}, n_{1\cdot},...,n_{J\cdot}\right) \propto \left(\alpha+\kappa\right)^{a+m_{\cdot\cdot}-1} e^{-(\alpha+\kappa)b} \prod_{j=1}^{J}\left(\frac{n_{j\cdot}}{\alpha+\kappa}\right)^{s_j} r_j^{\alpha+\kappa}\left(1-r_j\right)^{n_{j\cdot}-1}, \quad (2.62)$$

where s_j can be either one or zero. For marginal probabilities we obtain:

$$\begin{aligned} p\left(\alpha+\kappa \mid r, s, m_{1\cdot},...,m_{J\cdot}, n_{1\cdot},...,n_{J\cdot}\right) &\propto \left(\alpha+\kappa\right)^{a+m_{\cdot\cdot}-1-\sum_{j=1}^{J} s_j} e^{-(\alpha+\kappa)\left(b-\sum_{j=1}^{J}\log r_j\right)} \\ &= Gamma\left(\alpha+m_{\cdot\cdot}-\sum_{j=1}^{J} s_j, b-\sum_{j=1}^{J}\log r_j\right), \end{aligned} \quad (2.63)$$

$$p\left(r_j \mid \alpha+\kappa, r_{\setminus j}, s, m_{1\cdot},...,m_{J\cdot}, n_{1\cdot},...,n_{J\cdot}\right) \propto r_j^{\alpha+\kappa}\left(1-r_j\right)^{n_{j\cdot}-1} = Beta\left(\alpha+\kappa+1, n_{j\cdot}\right), \quad (2.64)$$

$$p\left(s_j \mid \alpha+\kappa, r, s_{\setminus j} m_{1\cdot},...,m_{J\cdot}, n_{1\cdot},...,n_{J\cdot}\right)\alpha\left(\frac{n_{j\cdot}}{\alpha+\kappa}\right)^{s_j} = Ber\left(\frac{n_{j\cdot}}{n_{j\cdot}+\alpha+\kappa}\right), \quad (2.65)$$

where in the above equations, the notation "\" in the subscript means the entire sequence with the exception of the j^{th} point. For example, $s_{\setminus j}$ means the sequence $\{s_1, s_2,..., s_{j-1}, s_{j+1},..., s_T\}$.

Posterior of γ:

Similar to (2.58), the distribution of the unique number of dishes served in the whole franchise is given by $p\left(K \mid \gamma, \bar{m}_{\cdot\cdot}\right) = s\left(\bar{m}_{\cdot\cdot}, K\right)\frac{\Gamma(\gamma)}{\Gamma\left(\gamma+\bar{m}_{\cdot\cdot}\right)}$. Therefore for the posterior distribution of γ we can write:

$$\begin{aligned} p\left(\gamma \mid K, \bar{m}_{\cdot\cdot}\right) &\propto p(\gamma)\, p\left(K \mid \gamma, \bar{m}_{\cdot\cdot}\right) \\ &\propto p(\gamma)\gamma^K \frac{\beta\left(\gamma+1, \bar{m}_{\cdot\cdot}\right)}{\gamma\Gamma\left(\bar{m}_{\cdot\cdot}\right)} \\ &\propto p(\gamma)\gamma^K \left(\gamma+\bar{m}_{\cdot\cdot}\right)\int_0^1 \eta^{\gamma}\left(1-\eta\right)^{\bar{m}_{\cdot\cdot}-1} d\eta\,. \end{aligned} \quad (2.66)$$

By considering the fact that prior over γ is *Gamma(a,b)*, we can write:

$$p\left(\gamma, \eta, \varsigma \mid K, \bar{m}_{\cdot\cdot}\right) \propto \gamma^{\alpha+K-1}\left(\frac{\bar{m}_{\cdot\cdot}}{\gamma}\right)^{\varsigma} e^{-\gamma(b-\log\eta)}\left(1-\eta\right)^{\bar{m}_{\cdot\cdot}-1}. \quad (2.67)$$

For the marginal distributions we have:

$$p\left(\gamma \mid \eta, \varsigma, K, \bar{m}_{\cdot\cdot}\right) \propto \gamma^{\alpha+K-1-\varsigma} e^{-\gamma(b-\log\eta)} = Gamma\left(\alpha+K-\varsigma, b-\log\eta\right), \quad (2.68)$$

$$p(\eta \mid \gamma, \varsigma, K, \bar{m}_{\cdot\cdot}) \propto \eta^{\gamma} (1-\eta)^{\bar{m}_{\cdot\cdot}-1} = Beta(\gamma+1, \bar{m}_{\cdot\cdot}), \tag{2.69}$$

$$p(\varsigma \mid \gamma, \eta, K, \bar{m}_{\cdot\cdot}) \propto \left(\frac{\bar{m}_{\cdot\cdot}}{\gamma} \right)^{\varsigma} = Ber\left(\frac{\bar{m}_{\cdot\cdot}}{\bar{m}_{\cdot\cdot} + \gamma} \right). \tag{2.70}$$

Posterior of σ:

The posterior for σ is obtained in a similar way to the derivation of the posterior for $\alpha+\kappa$. We use two auxiliary variables r' and s'. The final marginalized distributions are:

$$p(\sigma \mid r', s', K'_{1\cdot}, ..., K'_{J\cdot}, n_{1\cdot}, ..., n_{J\cdot}) \propto (\sigma)^{\alpha + K'_{\cdot\cdot} - 1 - \sum_{j=1}^{J} s'_j} e^{-\sigma\left(b - \sum_{j=1}^{J} \log r'_j\right)}, \tag{2.71}$$

$$p(r'_j \mid \sigma, r'_{\setminus j}, s', K'_{1\cdot}, ..., K'_{J\cdot}, n_{1\cdot}, ..., n_{J\cdot}) \propto r_j'^{\sigma} \left(1 - r_j'^{\sigma}\right)^{n_{j\cdot} - 1}, \tag{2.72}$$

$$p(s'_j \mid \sigma, r', s'_{\setminus j}, K'_{1\cdot}, ..., K'_{J\cdot}, n_{1\cdot}, ..., n_{J\cdot}) \propto \left(\frac{n_{j\cdot}}{\sigma} \right)^{s'_j}. \tag{2.73}$$

It should be noted that in cases where we use auxiliary variables we prefer to iterate several times before moving to the next iteration of the main algorithm.

Posterior of ρ:

Let us define without loss of generality: $\rho = \frac{\kappa}{\alpha + \kappa}$. By considering the fact that the prior on ρ is *Beta(c,d)* and $\omega_{jt} \sim Ber(\rho)$ we can write:

$$\begin{aligned} p(\rho \mid \omega) &\propto p(\omega \mid \rho)\, p(\rho) \\ &\propto Binomial\left(\sum_j \omega_{j\cdot}; m_{\cdot\cdot}, \rho \right) Beta(c, d) \\ &\propto Beta\left(\sum_j \omega_{j\cdot} + c, m_{\cdot\cdot} - \sum_j \omega_{j\cdot} + d \right). \end{aligned} \tag{2.74}$$

The hyperparameters control the complexity of the model. As these equations show, we can sample their posterior distributions like any other parameters of the model. If we iterate a few thousand times it will converge to the true distribution even when starting from non-informative priors. However, for speech applications we often deal with large amounts of data. Even the

relatively small datasets used in this dissertation are computationally challenging with each iteration requiring a few minutes of computation time when running on medium-sized cluster computer. Therefore in order to reduce the required number of iterations we have to carefully initialize these variables to values that are in the neighborhood of the desired values. These values can be found by running several tuning experiments with smaller subsets of data using different initial values, and then evaluating the final trained distributions for accuracy.

2.5 Conclusion

In this chapter, we have introduced the background necessary to understand the following chapters of this dissertation. We have covered introductory materials related to a Dirichlet distribution, a Dirichlet Process, a Hierarchical Dirichlet Process and HDPHMMs. Most of the material in this chapter was presented from a speech recognition/machine learning perspective. Therefore they lack the formal treatment usually preferred by researchers in mathematics or statistics. We encourage, interested reader to review some of the excellent references introduced in this chapter for a deeper understating of these concepts.

CHAPTER 3

NONPARAMETRIC BAYESIAN APPROACHES FOR ACOUSTIC MODELING OF SUB-WORD UNITS

The most important element of acoustic modeling is the statistical approach used to model the sub-word units. Most state of the art systems use left-to-right Hidden Markov Models (HMMs) with Gaussian mixtures to model phonetic units (Rabiner, 1989). HMMs, often referred to as doubly stochastic models, are parameterized both in their structure (e.g. number of states) and emission distributions (e.g. Gaussian mixtures).

Model selection methods such as the Bayesian Information Criterion (BIC) (Kadane & Lazar, 2004) are traditionally used to optimize the number of states and mixture components. However, these methods are computationally expensive and there is no consensus on an optimum criterion for selection (Kadane & Lazar, 2004).

In speech recognition applications, usually the number of states is assumed to be fixed for all models (e.g., *3*). Mixtures are trained progressively by starting from one mixture component per state and increasing the number of mixtures until a further increment does not improve the likelihood of the training data (or sometimes error rate on a development set). The number of mixtures per state is also a fixed parameter for all states and models (with exception of a silence model that usually has a different structure and complexity).

Beal et al. (2002) proposed a nonparametric Bayesian HMM with a countably infinite number of states. This model is known as an infinite HMM (iHMM) because it has an infinite number of hidden states. Teh et al. (2006) and Fox et al. (2011) proposed a different formulation,

HDPHMM, based on a hierarchical Dirichlet process (HDP) prior. HDPHMM is an ergodic model – a transition from an emitting state to all other states is allowed. However, in many pattern recognition applications involving temporal structure, such as speech processing, a left-to-right topology is required (Fink, 2008).

For example, in continuous speech recognition applications we model speech units (e.g. phonemes), which evolve in a sequential manner, using HMMs. Since we are dealing with an ordered sequence (e.g. a word is an ordered sequence of phonemes), a left-to-right model is preferred (Juang & Rabiner, 1991). The segmentation of speech data into these units is not known in advance and therefore the training process must be able to connect these smaller models together into a larger HMM that models the entire utterance. This task can easily be achieved using left-to-right HMMs (LR-HMM). If the data has finite length, the beginning and end of a sequence is typically modeled as two additional discrete events – non-emitting initial and final states (Fink, 2008). In the HDPHMM formulation, these problems are not addressed.

An HDPHMM, as well as a parametric HMM, models each emission distribution by data points mapped to that state. For example, it is common to use a Gaussian mixture model (GMM) to model the emission distributions. However, in an HDPHMM, the mixture components of these GMMs are not shared or reused. Sharing of such parameters is a critical part of most state of the art pattern recognition systems.

In this chapter, we introduce a model with two parallel hierarchies that enable sharing of data among different states. We refer to this model as a Doubly Hierarchical Dirichlet Process Hidden Markov Model (DHDPHMM) (Harati et al., 2014). We also introduce a general method to add non-emitting states to both HDPHMMs and DHDPHMMs. We develop a framework to learn non-ergodic structures from the data and present comprehensive experimental results for a standard phoneme classification task in speech processing.

3.1 Related Work

HMMs are parameterized both in their topology (e.g. number of states) and emission distributions. Most attempts to relax these parameterizations were focused on the second aspect. Bourlard (1993) and others proposed to replace Gaussian mixture models (GMMs) with a neural network based on a multilayer perceptron (MLP). It was shown that MLPs generate reasonable estimates of a posterior distribution of an output class conditioned on the input patterns (Bourlard & Morgan, 1993). This hybrid HMM-MLP system works slightly better than traditional HMM-GMMs, but the gain was not significant enough to justify adopting this new technology. Most of the gain can be recovered by using more sophisticated processing steps such as speaker adaptation.

More recently, renewed interest in using neural networks (NN) has emerged due to the introduction of deep learning approaches (Bengio, 2009; Bengio et al., 2013). Deep learning models solve some of the major problems associated with neural networks by allowing training of deep neural networks (DNN) through a procedure called pre-training and fine tuning. These networks are constructed using simpler building blocks like restricted Boltzmann machine (RBM) and utilize properties of these simple structures (e.g. lack of connections between hidden units in RBMs) to make inference algorithm practical. The result is a hybrid HMM-DNN that currently delivers state of the art performance on some speech recognitions tasks (Hinton et al., 2012; Rath et al., 2013; Sainath et al., 2012; Seide et al., 2011).

Besides these HMM-NN hybrid systems, several other approaches based on combinations of Markov based models and deep learning have been proposed recently. Conditional Random Fields (CRF), which are discriminative in nature, have been used (Morris & Fosler-Lussier, 2008; Gunawardana et al., 2005). Yu and Deng (2010) have proposed a deep structured CRF (DHCRF) for a phoneme recognition task that has been shown to produce better results compared discriminatively-trained HMMs. However, it should be noted some of the results reported for

models based on deep learning are not among the state of the art results. For example, Palaz et al. (2013) reported results of a hybrid CRF convolutional network that is close to an ML-trained HMM baseline in performance. Nonparametric non-Bayesian modeling of emission distributions (Lefèvre, 2003; Shang, 2009) have been used to replace GMMs, but improvements were marginal at best. These nonparametric non-Bayesian methods are especially prone to overfitting or over-smoothing (Wang et al., 2010).

Henter et al. (2012) introduced a new model named Gaussian process dynamical model (GPDM) to completely replace HMMs in acoustic modeling. This model is a nonparametric Bayesian model based on a Gaussian process (comparing to HDPHMM which is based on Dirichlet process) and supposedly solves some of the problems traditionally associated with HMMs such as duration modeling and stepwise constant evolution (Henter et al., 2012). However, this model is used only in speech synthesis and no results have been reported for speech recognition tasks. Another related (but independently developed) model also named GPDM has been reported by (Park & Yoo, 2011) for a phoneme classification task, but again the results for this model were actually worse than the baseline HMM system.

Petrov et al. (2007) introduced a data driven HMM that learns the structure of HMMs by utilizing a split-merge EM procedure. The model starts from a single state with only one Gaussian and is iteratively refined into more complicated structures. It has been shown that this model can compete with state of the art systems. The sprit behind this model is similar to HDPHMM but the approach is different. Fox et al. (2011) have applied an ergodic HDPHMM model to the problem of speaker diarization. Speaker diarization is the process of segmenting an audio stream into speaker-based segments. Classic speaker diarization algorithms usually consist of two stages: segmentation and clustering. Clustering segments into speaker labels is usually implemented with some form of hierarchical agglomerative clustering and is very sensitive to the specific threshold for cluster merging. It has been shown that HDPHMM can compete with state of the art systems for speaker diarization (Wooters & Huijbregts, 2007). HDPHMM has not been used for acoustic

modeling problem since it only provides an ergodic structure and does not address some of the issues (e.g. left to right structure, non-emitting states and computational cost) that are important in problems like sub-word modeling.

Steinberg et al. (2012, 2013) have proposed to use DPM to model sub-word units for two different English and Chinese phoneme classification tasks. They have shown that models discovered by the DPM approach have much fewer parameters compared to the baseline parametric GMM models while delivering comparable performance. The model used in their work is much simpler than models we have proposed in this chapter but some of the conclusions (such as learning optimal complexity) are similar.

Harati et al. (2012) has used a DPM-based algorithm in speaker adaption application to grow a tree using a bottom-up Euclidean distance based approach. Several inference algorithms were evaluated and it was shown that the proposed system can produce better results (e.g. *10%* improvement) comparing to a regression tree based approach as discussed in Chapter 1. Nonparametric Bayesian approaches were also used in speech segmentation and acoustic unit discovery problems (Harati el al., 2013; Lee & Glass, 2012), and this is discussed in Chapter 5.

Another trend is the application of parametric Bayesian methods in modeling of acoustic units. The most famous of these are variational HMMs where HMMs are treated in Bayesian framework using a Dirichlet distribution as a prior for transitions. Inference is done using variational method instead of Gibbs sampling. The results are promising especially for smaller sized datasets (Watanabe et al., 2003).

3.2 A Doubly Hierarchical Dirichlet Process Mixture Model

HDPHMM defined in (2.35) introduces a model with unbounded number of states that learns the model complexity (e.g. number of states, transition distribution and number of mixture components) from data. However, one problem with having an unbounded number of states is

that there are fewer data points to estimate the parameters (e.g. mean and covariance) for each state. Fox et al. (2011) introduced the sticky parameter κ that to some extent biases the model toward models with fewer states since consecutive data points would tend to stay within one state. However, each state's parameters are estimated independently of the other states. Sharing data, if performed carefully, can potentially improve the accuracy of the estimated parameters (Young et al., 1994). In this section we introduce a new model to address this problem. Instead of sharing data points directly we share mixture components between different states.

We can extend the model in (2.35) to address the problem of sharable mixture components. Equation (2.35) defines a model with a multimodal distribution at each state. In an HDPHMM formulation these distributions are modeled using a DPM model:

$$\begin{aligned}
&\psi_j \mid \sigma \sim GEM(\sigma) \\
&s_t \mid \{\psi_j\}_{j=1}^{\infty},\ z_t \sim \psi_{z_t} \\
&\theta_{kj}^{**} \mid H, \lambda \sim H(\lambda) \\
&x_t \mid \{\theta_{kj}^{**}\}_{k,j=1}^{\infty},\ z_t \sim F\left(\theta_{z_t s_t}\right),
\end{aligned} \tag{3.1}$$

where x_t and s_t are data and mixture component indicators respectively. The variable z_t is the state indicator and is assumed to be known. ψ_j is a DP distribution with a concentration parameter σ that models the infinite mixture model for state j. θ_{kj} is an emission distribution parameter that is sampled from H.

Equation (3.1) demonstrates that when the state assignment, z_t, for data point x_t is known, the mixture components can be sampled from a multinomial distribution with DP priors. Equation (2.35) also shows that each emission distribution is modeled independently of the other distributions. We first assign the state indicator variable z_t for each data point and then model all data points with the same z_t using a DPM.

As we have discussed in Section 2.3, HDP is an extension of a DPM to mixture modeling of grouped data. If the state assignment, z_t, is assumed to be known (or estimated) then an

HDPHMM divides the data points into multiple groups (e.g. each state forms one group). Therefore we should be able to model the emission distributions with another HDP.

In other words, we want all emission distributions to be linked together and to share mixture components. The resulting model will have two parallel hierarchies, one to model an infinite number of states and another one to allow sharing mixture components across states. Hence, it is referred to as a Doubly Hierarchical Dirichlet Process Hidden Markov Model (DHDPHMM).

Applying (2.23) we can write:

$$
\begin{aligned}
&\xi \mid \tau \sim GEM(\tau) \\
&\psi_j \mid \sigma, \xi \sim DP(\sigma, \xi) \\
&\theta_{kj}^{**} \mid H, \lambda \sim H(\lambda) \\
&s_t \mid \left\{\psi_j\right\}_{j=1}^{\infty}, z_t \sim \psi_{z_t} \\
&x_t \mid \left\{\theta_{kj}^{**}\right\}_{k,j=1}^{\infty}, z_t \sim F\left(\theta_{z_t s_t}\right),
\end{aligned}
\tag{3.2}
$$

where ζ is the DP used as the base distribution for HDP and τ and σ are hyperparameters. It should be noted that ψ_j is a DP with a base distribution ζ. By substituting (3.2) in (2.35) we can obtain a generative model for DHDPHMM:

$$
\begin{aligned}
&\beta \mid \gamma \sim GEM(\gamma) \\
&\pi_j \mid \alpha, \beta \sim DP(\alpha + \kappa, \frac{\alpha\beta + \kappa\delta_j}{\alpha + \kappa}) \\
&\xi \mid \tau \sim GEM(\tau) \\
&\psi_j \mid \sigma, \xi \sim DP(\sigma, \xi) \\
&\theta_{kj}^{**} \mid H, \lambda \sim H(\lambda) \\
&z_t \mid z_{t-1}, \left\{\pi_j\right\}_{j=1}^{\infty} \sim \pi_{z_{t-1}} \\
&s_t \mid \left\{\psi_j\right\}_{j=1}^{\infty}, z_t \sim \psi_{z_t} \\
&x_t \mid \left\{\theta_{kj}^{**}\right\}_{k,j=1}^{\infty}, z_t \sim F\left(\theta_{z_t s_t}\right).
\end{aligned}
\tag{3.3}
$$

The state, mixture components and observations are represented by z_t, s_t and x_t respectively. Similar to HDPHMM, j and k are indices of the state and mixture components respectively. The base distribution, β, can be interpreted as the expected value of state transition distributions. The

transition distribution for state j is a DP denoted by π_j with a concentration parameter α and sticky parameter κ.

Another DP distribution, ζ with concentration parameter τ, is used as the base distribution for the second HDP that models emission distributions. Each mixture is modeled with a DP, ψ_j, with a concentration parameter σ and base distribution ζ (forming the second HDP). Each DP models an infinite mixture model for a corresponding state z_j. However, as discussed above, the components of these mixture models are shared across different states because we link all mixture distributions using a second HDP.

If emission distributions are mixtures of Gaussians then θ_{kj} includes mean and covariance parameters. The distribution H is the prior for the parameters θ_{kj}. Similarly, the conjugate prior, H, is a Normal-Inverse-Wishart (NIW) distribution (Suderth, 2006). If we relax this conjugacy requirement we can have Gaussian priors on the mean and an independent inverse-Wishart distribution on the covariance for each Gaussian component.

Intuitively, DHDPHMM pools the data points while HDPHMM divides data points between different states. If we don't have enough data points in a particular state or a mixture component then the distribution parameters will be estimated poorly (e.g., mean and covariance). For example, in speech recognition systems we usually use features with a dimensionality of *39* which translates to $39+(39x40)/2+1=820$ free parameters per Gaussian mixture component (assuming a full covariance). In an HDPHMM, with no sharing of parameters, we can easily end up with an intractable number of parameters.

As an illustrative example, consider Figure 3-1 that depicts the sharing process using a simple scenario. Let's assume data point x_1 is drawn from an underlying Gaussian distribution, while x_4 and x_5 are drawn from a second Gaussian with similar parameters. Since we are dealing with sequential data points, an HMM would segment the data based on the order of occurrence. In this case x_1 is assigned to state *1* while x_4 and x_5 are assigned to state *2*. If the HMM is modeled using an HDPHMM, the Gaussian component associated with x_1 would be different from Gaussian

component associated with x_4 and x_5 (shown on the left side of Figure 3-1). However, DHDPHMM will automatically assign all three to only one Gaussian component (based on similarity of x_1, x_4 and x_5) and then reuse that component in both state *1* and state *2* (shown on the right side of Figure 3-1).

Note that this does not mean that DHDPHMM always assigns data from different states to a single component and then shares that component across states. Sharing of data points only happens when there is an underlying statistical similarity. For example, data points x_2 and x_3 form a component for state *1* while x_6 forms another component for state *2*, but these components are not shared across states. Each component is modeled using its own data points.

3.3 Inference Algorithm for DHDPHMM

An inference algorithm is required to learn the model parameters (e.g. number of observed states and emission parameters) from the data. One solution to this problem is the block sampler (Fox et al., 2011) discussed in the previous section. Here we present modifications of this block sampler for the inference of our DHDPHMM.

Using the "degree L weak limit" (Ishwaran and Zarepour, 2002) approximation to Dirichlet

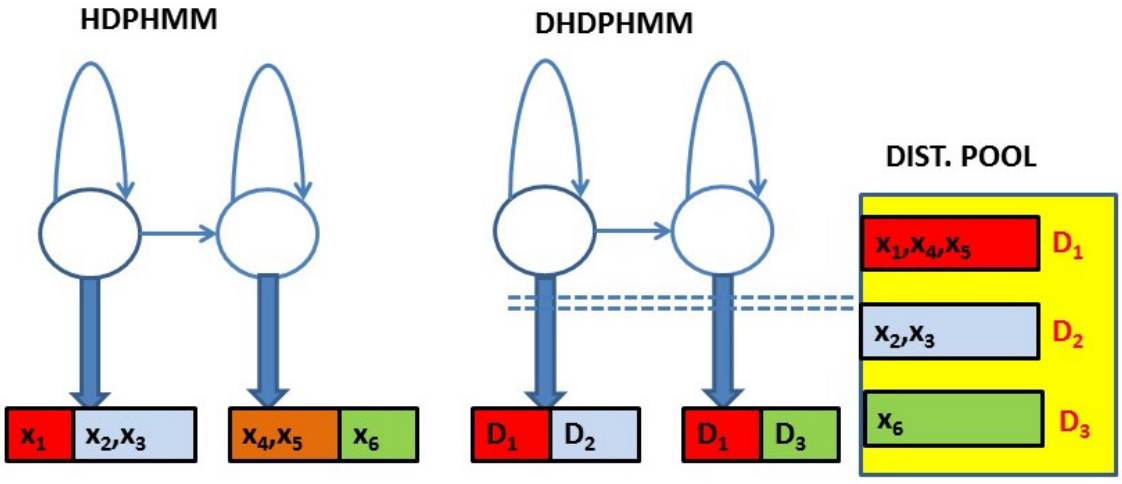


Figure 3-1. A comparison of HDPHMM and DHDPHMM is shown.

process in (2.36) for HDP emissions of (3.3), we can write the following equations (replacing L' with L):

$$\xi \mid \sigma \sim Dir\left(\frac{\sigma}{L'},...,\frac{\sigma}{L'}\right), \tag{3.4}$$

$$\psi_j \mid \xi,\tau \sim Dir\left(\tau\xi_1,...,\tau\xi_{L'}\right). \tag{3.5}$$

In writing (3.4) and (3.5) we have assumed that the base distribution, ζ, and the emission distributions, ψ_j, are approximated with same upper bound, L'. We can alternately use a different upper bound, $L'' \le L'$, to approximate ψ_j.

Following a similar approach to that in Fox et al. (2011), we write the posterior distributions for these equations as:

$$\xi \mid M',\tau \sim Dir\left(\frac{\tau}{L'}+M'_{\cdot 1},...,\frac{\tau}{L'}+M'_{\cdot L'}\right), \tag{3.6}$$

$$\psi_j \mid \sigma,\xi,z_{1:T},s_{1:T} \sim Dir\left(\sigma\xi_1+n'_{j1},...,\sigma\xi_{L'}+n'_{jL'}\right), \tag{3.7}$$

where M'_{jk} is the number of clusters in state j with mixture component k, and $M'_{\cdot k}$ is the total number of clusters that contain mixture component k. The number of observations in state j that are assigned to component k is denoted by n'_{jk}. The posterior distribution for τ, the hyperparameter in (3.6), can be written as:

$$P\left(\tau \mid n_{1\cdot},...,n_{J\cdot},M'_{1\cdot},...M'_{J\cdot}\right) \propto Gamma\left(a+M'_{\cdot\cdot}-\sum_{j=1}^{L'}s_j, b-\sum_{j=1}^{L'}\log r_j\right), \tag{3.8}$$

$$P\left(r_j \mid \tau, r_{\setminus j}, s, n_{1\cdot},...,n_{J\cdot},M'_{1\cdot},...M'_{J\cdot}\right) \propto Beta\left(\tau+1, n_{j\cdot}\right), \tag{3.9}$$

$$P\left(s_j \mid \tau, s_{\setminus j}, r, n_{1\cdot},...,n_{J\cdot},M'_{1\cdot},...M'_{J\cdot}\right) \propto Ber\left(\frac{n_{j\cdot}}{n_{j\cdot}+\tau}\right), \tag{3.10}$$

where r and s are auxiliary variables used to facilitate the inference for τ (Fox et al., 2011), and a and b are hyperparameters over a Gamma distribution.

We can summarize the modifications to the block sampler as follows:

1. Given the previous $\boldsymbol{\pi}^{(n-1)}, \boldsymbol{\psi}^{(n-1)}, \boldsymbol{\beta}^{(n-1)}$ and $\boldsymbol{\theta}^{(n-1)}$.

2. Compute backward messages $A_{T+1,T}(k)$ using (2.46) and (2.47).

3. Sample the augmented state (z_t,s_t) sequentially and start from *t=0* using (2.48), (2.49) and (2.50).

4. Accumulate the sufficient statistics using (2.51).

5. Sample auxiliary variables (**m**, **w**, $\bar{\boldsymbol{m}}$) similar to the block sampler algorithm using (2.52) and (2.53).

6. Sample a new auxiliary variable $\boldsymbol{M'}$ (to model the second HDP) by simulating a CRF:

   For each $j \in \{1,2,...,L\}, k \in \{1,2,...,L'\}$, set $\boldsymbol{M'}_{\text{jk}} = 0$ and n = 0.

   a) For each customer in restaurant *j* eating dish *k*, for $i = 1,...,n_{jk}$, sample:

$$x \sim Ber\left(\frac{\tau\xi_k}{n+\tau\xi_k}\right). \tag{3.11}$$

   b) Increase *n* and if *x = 1* increase $\boldsymbol{M'}_{jk}$.

7. Update β using (2.54).

8. Update the base distribution for the second HDP, ζ, using (3.6).

9. Sample π_k from (2.55) and ψ_k from (3.7).

10. Sample θ_{kj} from (2.56).

11. Set $\boldsymbol{\pi}^{(n)} = \boldsymbol{\pi}, \boldsymbol{\psi}^{(n)} = \boldsymbol{\psi}, \beta^{(n)} = \beta$ and $\boldsymbol{\theta}^{(n)} = \boldsymbol{\theta}$.

12. Optionally sample hyperparameters σ, γ, α and κ similar to the block-sampler algorithm and τ from (3.8).

By comparing the block sampler and the modified block sampler, we see the basic procedure is similar. However, the modified block sampler also samples an extra HDP (and related variables) that models the emission distributions.

One of the major computational parts of both algorithms is to calculate the likelihood of each data point for every Gaussian component. For example, computing backward messages, $A_{t,t-1}$, which are required in the FB algorithm, involves computing:

$$A_{t,t-1}(k) = \sum_{i=1}^{L}\sum_{l=1}^{L} \pi_{ki}\psi_{il} N(x_{t+1};\mu_{il},\Sigma_{il}) A_{t+1,t}(i), \tag{3.12}$$

where in this equation $N(x_i;\mu_k,\Sigma_k)$ is the likelihood of data point x_i for component k. We have LxL' Gaussians for HDPHMM while for DHDPHMM we have only L' Gaussians since we reuse Gaussians across states. The number of likelihood computations for DHDPHMM is less than HDPHMM by a factor proportional to L. In Section 3.7.2 we will further investigate this important practical issue and demonstrate its effect on inference time.

Furthermore, as stated above, for DHDPHMM we approximate both ζ and ψ_j with L'. However, we can approximate ψ_j with L'' ($L'' \le L'$). This allows us to constrain the states to only learn the specific number of mixture components during the training procedure and therefore gives us greater control during training. For example, in traditional acoustic model training, we use an iterative mixture splitting process described in Section 1.1. In DHDPHMM, we might first require that each model only has one Gaussian mixture model initially and after several iterations relax this restriction, allowing states to have mixtures with up to L' Gaussian components. For the remainder of this chapter we will assume $L' = L''$.

3.4 DHDPHMM with a Non-Ergodic Structure

A non-ergodic structure for the DHDPHMM can be achieved by modifying the transition distributions in (3.3). These modifications can also be applied to HDPHMM using a similar

approach (Harati et al., 2014).

3.4.1 Left-to-Right DHDPHMM with Loop Transitions

The transition probability from state j is modeled using a DP that has infinite support. It can be written as:

$$\pi_j \mid \alpha, \beta \sim DP(\alpha + \kappa, \frac{\alpha\beta + \kappa\delta_j}{\alpha + \kappa}) \,, \tag{3.13}$$

where α and κ are the concentration and sticky parameters respectively, and β is the global transition distribution that acts as the base distribution for this DP.

From (3.13) we can see the transition distribution has no topological restrictions and therefore (2.35) and (3.3) define ergodic HMMs. In order to obtain a left-to-right (LR) topology we need to force the base distribution of the Dirichlet distribution in (3.13) to only contain atoms to the right of the current state. This means β should be modified so that the probability of transiting to states left of the current state (i.e. states previously visited) becomes zero. For state j we define $V_j=\{V_{ji}\}$:

$$V_{ji} = \begin{cases} 0, & i < j, \\ 1, & i \geq j, \end{cases} \tag{3.14}$$

where i is the index for all following states. We can then modify β by multiplying it with V_j:

$$\beta' = \frac{\beta \bullet V_j}{\sum_i \beta_i V_{ji}}. \tag{3.15}$$

In the block sampler algorithm, we have:

$$\pi_j \sim Dir\left(\alpha\beta'_1 + n_{j1}, ..., \alpha\beta'_j + \kappa + n_{jj}, ..., \alpha\beta'_L + n_{jL}\right), j = 1, ..., L, \tag{3.16}$$

where n_{jk} are the number of transitions from state j to k. From (3.16) we can see that multiplying β with V_j biases π_j toward a left-to-right structure but there is still a positive probability to transit to the states left of j. If we leave π_j as in (3.16) the resulting model would be an LR model with possible loops. The model would be biased toward an LR structure but with the possibility of

forming loops. Models with an LR structure and possible loops will be denoted as LR-L. An example of this structure is shown in Figure 3-2(a).

3.4.2 Left-To-Right DHDPHMM

In order to obtain an LR model with no loops, we have to make sure all terms in (3.16) which are left of state *j* have a value of zero. This can be done by multiplying n_{jk} with V_j :

$$\pi_j \sim Dir\left(\alpha\beta'_1 + V_{j1}n_{j1},...,\alpha\beta'_j + \kappa + V_{jj}n_{jj},...,\alpha\beta'_L + V_{jL}n_{jL}\right), \quad j = 1,...,L. \tag{3.17}$$

V_j and β' are calculated from (3.14) and (3.15) respectively. This model always finds transitions to the right of state *j* and is referred to as an LR model.

Sometimes it is useful to have LR models that allow restricted loops to the first state. For

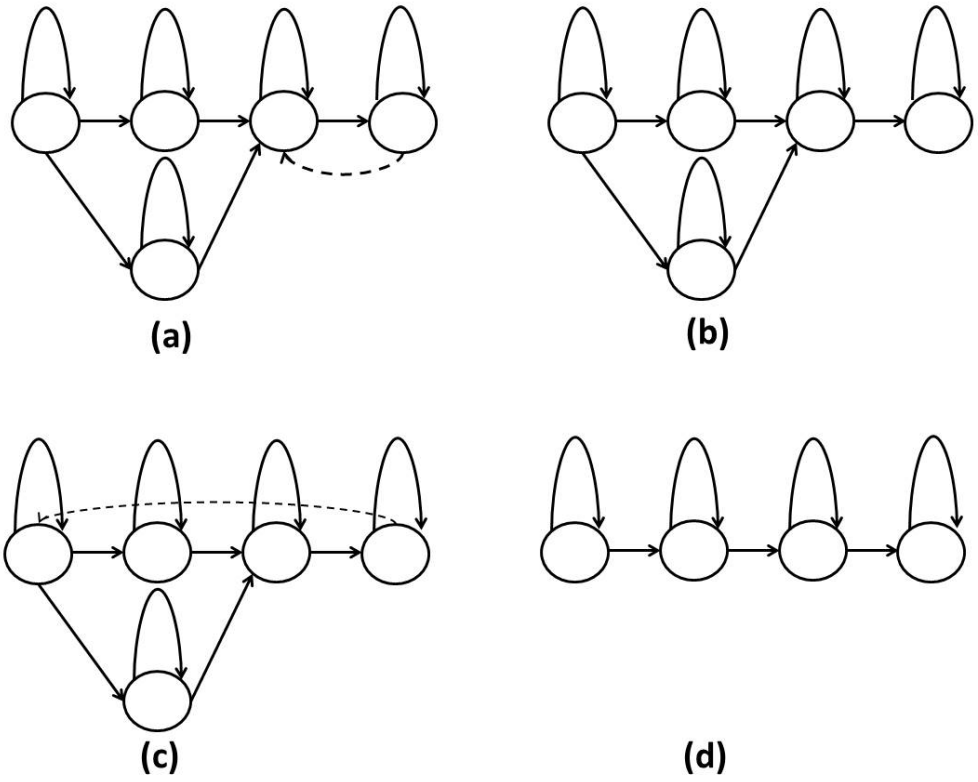


Figure 3-2. Different HMM structures for DHDPHMM are shown: (a) left to right with loops, (b) left to right with only self loops, (c) left to right with a loop to the first state, and (d) strictly left to right.

example, when dealing with long sequences, a sequence might have a local left to right structure but needs a reset at some point in time. To modify β to obtain an LR model with a loop to the first state (LR-LF) we can write:

$$. V_{ji} = \begin{cases} 0, & 0 < i < j, \\ 1, & i \geq j, i = 0. \end{cases} \tag{3.18}$$

β' can be calculated from (3.15) and π_j should be sampled from (3.17). Figure 3-2(b) and Figure 3-2(c) demonstrate the LR and LR-LF models respectively.

3.4.3 Strictly Left-to-Right DHDPHMM

The LR models described above allow for skip transitions that allow the model to learn parallel paths corresponding to different modalities. Sometimes more restrictions on the structure might be required. One such example is a strictly left to right structure (LR-S):

$$Vji = \begin{cases} 0, & i \neq j+1, \\ 1, & i = j+1. \end{cases} \tag{3.19}$$

An example of these models is shown in Figure 3-2(d). By comparing LR-S to other structures we can see that LR-S is restricted to only one path while other structures have multiple paths. These extra paths allow modeling of sequences that have varying lengths in addition to supporting modeling of multiple modalities in the data.

3.5 Initial and Final Non-Emitting States

In many applications, such as speech recognition, an LR-HMM begins from and ends with non-emitting states. These states are required to model the beginning and end of finite duration sequences. In practical systems, we use these non-emitting states to connect different HMMs. For example, the final emitting state of each HMM has a self-loop and is associated with observations. Similarly, a non-emitting final state has no such loops and is not associated with

observations. Entering the non-emitting state of model *j* therefore means we enter model *j+1*, without consuming an additional data point. Adding a non-emitting initial state is straightforward: the probability of transition into the initial state is *1*. The probability of a transition from this state is equal to π_{init} which is the initial probability distribution for an HMM without non-emitting states. However, adding a final non-emitting state is more complicated. In the following sections we will discuss two approaches that solve this problem.

3.5.1 Maximum Likelihood Estimation

Consider state z_i depicted in Figure 3-3. The outgoing probabilities for any state can be classified into three categories: (1) a self-transition (P_1), (2) a transition to all other states (P_2), and (3) a transition to a final non-emitting state (P_3). These probabilities must sum to 1: $P_1 + P_2 + P_3 = 1$. Suppose that we obtain P_2 from the inference algorithm. We will need to reestimate P_1 and P_3 from the data. This problem is, in fact, equivalent to the problem of tossing a coin until we obtain the first tails. Each head is equal to a self-transition and the first tails triggers a transition to the final state. This can be modeled using a geometric distribution (Pitman, 1993):

$$P(x=k) = (1-\rho)^{k-1}\,\rho\,. \tag{3.20}$$

Equation (3.20) shows the probability of $K - 1$ heads before the first tail. In this equation $1 - \rho$ is

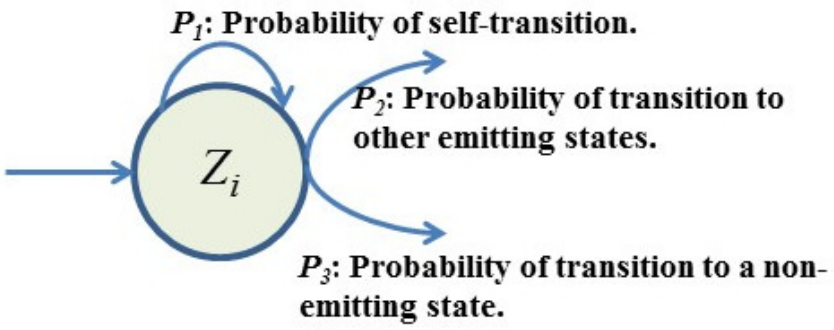


Figure 3-3. Outgoing probabilities for state z_i are shown.

the probability of heads (success).

We also have:

$$\frac{P_1}{1-P_2} = 1-\rho, \qquad \frac{P_3}{1-P_2} = \rho. \tag{3.21}$$

Suppose we have a total of N examples but for a subset of these, M_i, the state z_i is the last state of the model (S_M). It can be shown (Pitman, 1993) that the maximum likelihood estimation is obtained by:

$$\hat{\rho}_i = \frac{M_i}{\sum_{j \in S_M} k_j}, \tag{3.22}$$

where k_i are the number of self-transitions for state i. Notice that if z_i is never the last state, then $M_i = 0$, $P_3 = 0$ and $\rho = 0$. In other words, we only need to reestimate the transition probabilities if and only if we have some examples in the training data for which the state i is the last observed state.

3.5.2 Bayesian Estimation

Another approach to estimate transitions to a final non-emitting state, ρ_i, is to use a Bayesian framework. Since a Beta distribution is the conjugate distribution for a Geometric distribution, we can use a Beta distribution with hyperparameters (a,b) as the prior and obtain a posterior as (Gelman et al., 2004; Diaconis et al., 2010):

$$\rho_i \sim Beta\left(a + M_i, b + \sum_{j \in S_M} \left(k_j - 1\right)\right), \tag{3.23}$$

where M_i and S_M are the number of times which state z_i was the last state and set of all examples where state i is the last state respectively. Hyperparameters (a,b) can also be estimated using a Gibbs sampler if required (Quintana & Tam, 1996).

If we use (3.23) to estimate ρ_i we need to modify (3.16) to impose the constraint that the sum of the transition probabilities adds to one. This is a relatively simple modification based on the

stick-breaking interpretation of a Dirichlet process in (2.18). This modification is equal to assigning ρ_i to the first break of the stick and treating the remaining $1 - \rho_i$ portion as having a unit length. We can then use the standard stick-breaking algorithm iteratively.

3.6 An Integrated Model

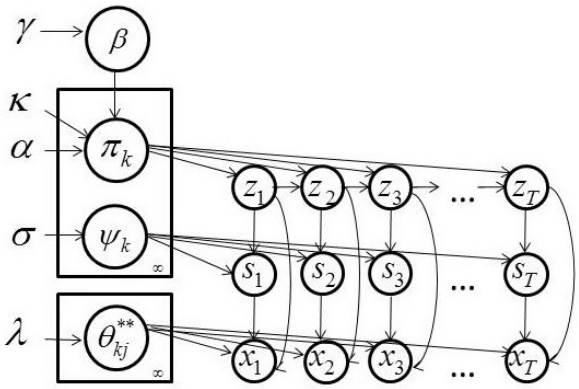
By incorporating the framework for learning non-ergodic structures into (3.3) we can write the definition for DHDPHMM for non-ergodic structures as:

$$
\begin{aligned}
&\beta \mid \gamma \sim GEM(\gamma) \\
&\beta'_j \mid V_j = \frac{V_j \cdot \beta}{\sum_i V_{ji}\beta_i} \\
&\pi_j \mid \alpha, \beta'_j \sim DP(\alpha+\kappa, \frac{\alpha\beta'_j + \kappa\delta_j}{\alpha+\kappa}) \\
&\xi \mid \tau \sim GEM(\tau) \\
&\psi_j \mid \sigma, \xi \sim DP(\sigma, \xi) \\
&\theta^{**}_{kj} \mid H, \lambda \sim H(\lambda) \\
&z_t \mid z_{t-1}, \{\pi_j\}_{j=1}^{\infty} \sim \pi_{z_{t-1}} \\
&s_t \mid \{\psi_j\}_{j=1}^{\infty}, z_t \sim \psi_{z_t} \\
&x_t \mid \{\theta^{**}_{kj}\}_{k,j=1}^{\infty}, z_t \sim F\left(\theta_{z_t s_t}\right),
\end{aligned}
\tag{3.24}
$$

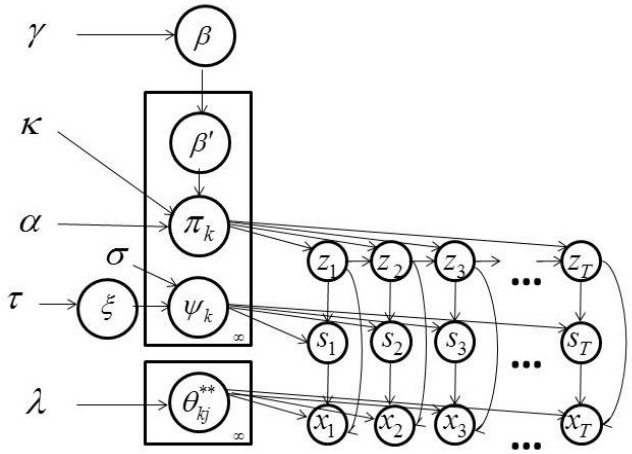
where in this definition x_t, s_t and z_t have similar interpretations as in (3.3), β is the global transition distribution and β' is the modified β (possibly the same as β if V_j is one). π_j, ψ_j, θ_{kj} and ξ also have similar interpretations to that in (3.3).

In this definition, V_i should be replaced with the appropriate definition from Section 3.4 based on the desired type of structure. For example if we want an LR model then V_i should be sampled from (3.14). Also note that by setting V_i to *1* we obtain the ergodic DHDPHMM in (3.3). A graphical representation of DHDPHMM is shown in Figure 3-4(b). An HDPHMM (Fox et al., 2011) is also displayed in Figure 3-4(a) for comparison.

We have not incorporated modeling of non-emitting states discussed above in (3.24). If we



(a)



(b)

Figure 3-4. Graphical representations are shown for: (a) an ergodic HDPHMM and (b) a DHDPHMM.

choose to use a maximum likelihood approach for estimating the non-emitting states then no change to this model is required (e.g. we can estimate these non-emitting states after estimating other parameters). However, if we choose to use the Bayesian approach then we have to replace the sampling of π_j in (3.24) with:

$$\begin{aligned} &\overline{w}, \overline{\chi} \sim MSB(\alpha, \beta, \kappa, j) \\ &\pi_j \mid \overline{w}, \overline{\chi} \sim \sum_k w_k \delta_{\chi_k}, \end{aligned} \tag{3.25}$$

where *MSB()* is a modified stick-breaking process. Equation (2.18) shows the basic stick-breaking algorithm – start with a stick of length one and then break the stick consecutively to obtain the

weights in (2.18). The locations of atoms, represented by δ in (2.18), are sampled independently from another distribution – G_0 in (2.18). In *MSB()*, we start from a stick of length (1-ρ_j) and sample the atoms from a discrete distribution that represents the transition probabilities:

$$MSB(\alpha,\beta',\kappa,j)=\begin{cases} for\, i=\{1,2,....\}: \\ v_i \mid \alpha,\kappa \sim Beta(1,\alpha+\kappa) \\ w_i \mid v_i,\rho_j = v_i(1-\rho_j)\prod_{l=1}^{k-1}(1-v_l) \\ \chi_i \mid \alpha,\beta',\kappa \sim \sum_k \frac{\alpha\beta'_k+\kappa\delta_{kj}}{\alpha+\kappa}\delta_k, \end{cases} \tag{3.26}$$

where v_i are sequences of independent variables drawn from a Beta distribution and w_i are stick weights. χ_i is the location of the atom that represents a transition to another state. χ_i determines which state we will transit to while w_i determines what is the probability to transit to this state.

By replacing (3.25) in (3.24) we can write:

$$\begin{aligned} &\beta \mid \gamma \sim GEM(\gamma) \\ &\beta'_j \mid V_j = \frac{V_j \cdot \beta}{\sum_i V_{ji}\beta_i} \\ &\rho_j \mid a,b,M_j,S_M \sim Beta\left(a+M_i, b+\sum_{j\in S_M}\left(k_j-1\right)\right) \\ &\overline{w},\overline{\chi} \sim MSB\left(\alpha,\beta'_j,\kappa,j\right) \\ &\pi_j \mid \overline{w},\overline{\chi} \sim \sum_k w_k \delta_{\chi_k} \\ &\xi \mid \tau \sim GEM(\tau) \\ &\psi_j \mid \sigma,\xi \sim DP(\sigma,\xi) \\ &\theta^{**}_{kj} \mid H,\lambda \sim H(\lambda) \\ &z_t \mid z_{t-1},\left\{\pi_j\right\}_{j=1}^{\infty} \sim \pi_{z_{t-1}} \\ &s_t \mid \left\{\psi_j\right\}_{j=1}^{\infty}, z_t \sim \psi_{z_t} \\ &x_t \mid \left\{\theta^{**}_{kj}\right\}_{k,j=1}^{\infty}, z_t \sim F\left(\theta_{z_t s_t}\right), \end{aligned} \tag{3.27}$$

where we have replaced DP with the modified stick-breaking process described above. Most of the results discussed above, including the inference algorithm, hold for this model as well.

3.7 Experiments

In this section we provide some experimental results which compare DHDPHMM with HDPHMM, HMM and several other models. First we investigate the scalability of DHDPHMM and compare it with HDPHMM. The experiments continue with artificial data and then proceed to a standard phoneme classification task.

3.7.1 Evaluation Methods

For the artificial data we compare the log-likelihood for a held-out subset of the data as defined by:

$$LL \mid m = \sum \log\left(P\left(x_i \mid m\right)\right), \tag{3.28}$$

where *LL* is the log-likelihood and *m* is the model under consideration. This equation determines the probability of generating data under a particular model. We also compare the resulting learned structure (topology and parameters) to the reference generative model.

For the phoneme classification task we compare the classification error rate on the evaluation and development subsets of the TIMIT (Garofolo et al., 1993) dataset. The classification error is defined as:

$$\%Err = 100 \times \left(1 - \frac{C}{C+I}\right), \tag{3.29}$$

where *C* is the number of correct and *I* is the number of incorrect decisions. We also compare complexity for HMM and DHDPHMM, and study how they behave as a function of the amount of training data.

3.7.2 A Computational Analysis of DHDPHMM

The main motivation behind DHDPHMM is the ability to share mixture components and therefore data points between different states. As discussed earlier when using the modified block

sampler algorithm we only deal with L' Gaussian distributions. The HDPHMM model has LxL' Gaussians to estimate.

Figure 3-5 shows a simplified analysis of the computational time spent for each of the major modules of the C++ inference algorithm implementation. Though the total time is somewhat data (or sound) dependent, the distribution of that time across tasks is relatively stable. This diagram was generated for */sh/* for $L = L' = 10$. We can see the likelihood calculations of Gaussian components are the most computationally expensive portion of the inference algorithm and typically consume between *50%* and *95%* of the total inference time. Therefore a reduction from LxL' to L' reduces the computation time considerably. Fortunately the computationally expensive portions of the code are easily parallelized using openMP (OpenMP Architecture Review Board, 2008), making the algorithms feasible for small and moderate size data sets.

Figure 3-6 provides a comparison of both algorithms for different values of L and L'. DHDPHMM's computational complexity is flat as the maximum bound on the number of states

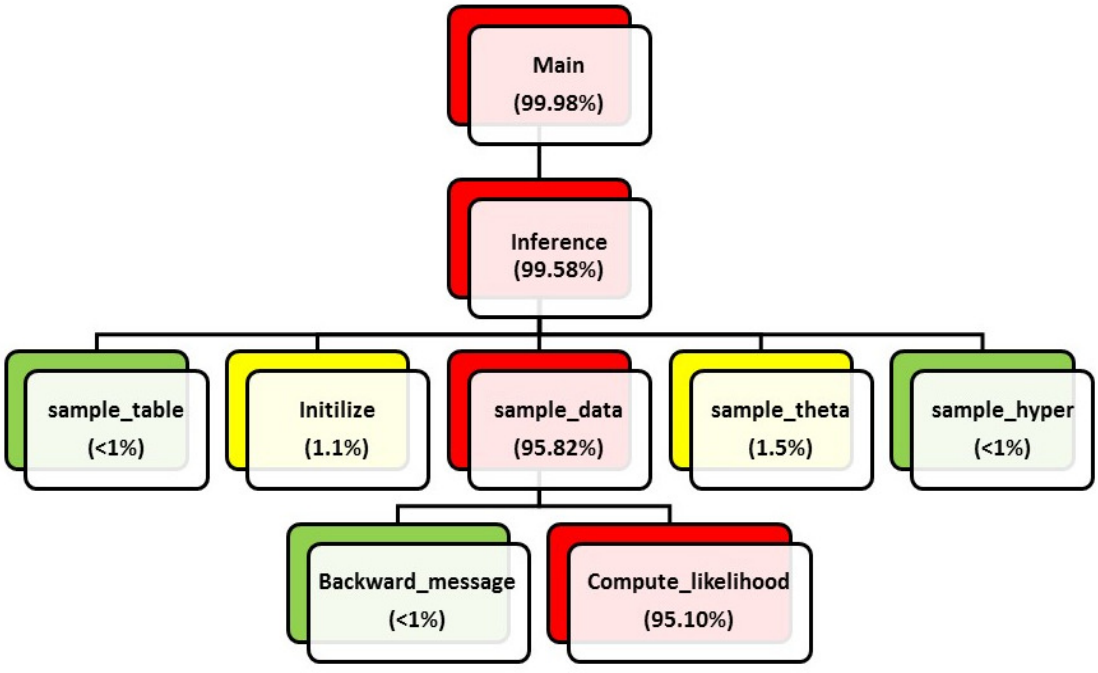


Figure 3-5. A decomposition of the computational time required by the block sampler algorithm is shown. The likelihood computation is the major computational bottleneck and consumes up to *95%* of the CPU time.

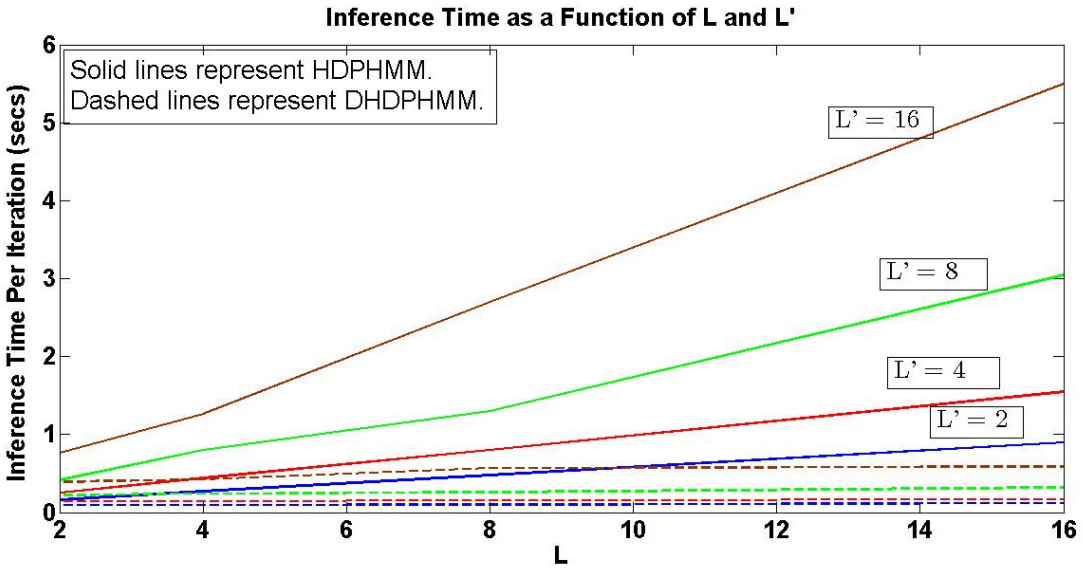


Figure 3-6. DHDPHMM improves the scalability relative to HDPHMM.

increases while the inference cost for HDPHMM grows linearly.

3.7.3 HMM-Generated Data

To demonstrate the basic efficacy of the model, we generated data from a *4*-state left to right HMM. The emission distribution for each state is a GMM with a maximum of three components, each consisting of a two-dimensional normal distribution. Three synthetic data sequences totaling *1900* observations were generated for training.

Three configurations have been studied: (1) an ergodic HDPHMM, (2) an LR HDPHMM and (3) an LR DHDPHMM. A Normal-inverse-Wishart distribution (NIW) prior is used for the mean and covariance. The truncation levels are set to *10* for both the number of states and the number of mixture components.

Figure 3-7 compares the average likelihood for different models for held-out data by averaging five independent chains. In Figure 3-7(a), the log-likelihoods are shown as a function of the number of iterations. In Figure 3-7(b) the topologies for the trained model are compared to the reference structure. The LR DHDPHMM discovers the correct structure while the ergodic HDPHMM finds a simpler HMM. LR DHDPHMM constrains the search space to left to right

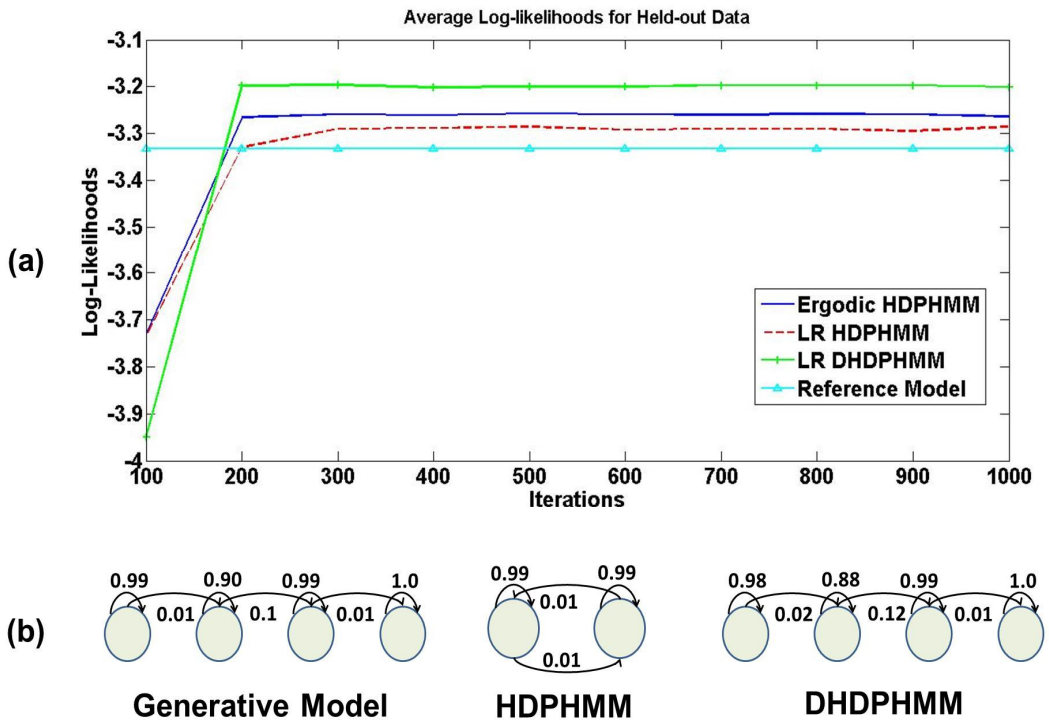


Figure 3-7. A comparison of the log-likelihoods of the proposed models to an ergodic model is shown in (a), while the corresponding model structures are shown in (b).

topologies while HDPHMM has a less constrained search space. Further, we can see that DHDPHMM has a higher overall likelihood.

While LR HDPHMM can find the structure that closely resembles the reference (this structure is not shown in this figure), its likelihood is slightly lower than the ergodic HDPHMM due to these constraints. However, LR DHDPHMM produces a *15%* (relative) improvement in likelihoods compared to the ergodic model. It is also interesting to note that the likelihoods of models discovered by all the nonparametric Bayesian algorithms are superior to the likelihood of the reference model itself.

This experiment proves that our model can discover the underlying structure of data. Moreover, we can see that DHDPHMM produces better likelihoods relative to HDPHMM. This is attributed to DHDPHMM's ability to share components. Each component is estimated using more data points because data is not forced to be associated with only one state. This feature,

tying sub-state parameters, has been used successfully in HMMs (Gu & Rose,2000). Here we are introducing this capability into nonparametric HMMs.

3.7.4 Phoneme Classification on the TIMIT Corpus

The TIMIT Corpus (Garofolo et al., 1993) is one of the most cited evaluation datasets used to compare new speech recognition algorithms. The data was phonetically transcribed by expert linguists and therefore is a natural choice to evaluate phoneme classification tasks. TIMIT contains *630* speakers (*438* male and *192* female) from eight main dialects of American English. Each speaker read *10* sentences that can be classified into three categories: (1) a set of sentences read by all speakers (referred to as the SA sentences), (2) a set of phonetically compact sentences that were designed to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts (referred to as the SX sentences), and (3) a set of phonetically diverse sentences that were selected to add diversity in sentence types and phonetic contexts (referred to as the SI sentences).

There are a total of *6,300* utterances where *3,990* are used in the training set and *192* utterances are used for the "core" evaluation subset (another *400* are used as a development set). We have also removed the SA sentences from both the training and evaluation sets, which leaves us with *3,637* training utterances. We followed the standard practice of building models for *48* phonemes and then map them into *39* phonemes (Gunawardana et al., 2005). Table 3-1 shows the *39* phoneme set and how we mapped *48* phonemes to *39* phonemes.

A standard Mel-frequency Cepstral Coefficients (MFCC) front-end has been used for feature extraction (Young et al., 2006). We have used a *25* msec hamming window with frame shift of *10* msec. Spectral analysis has been performed using a *40*-channel filter bank (filters are spaced based on Mel scale) with cut-off frequencies at *64* Hz and *8,000* Hz. A pre-emphasis coefficient of *0.97* has been used. We have used the first twelve Cepstral coefficients plus energy and their first and second derivatives to obtain a *39*-dimensional feature vector.

Table 3-1. A mapping from *48* phonemes to *39* classes is shown. This is a standard approach used for the TIMIT Corpus.

1	**iy**	2	**ih ix**	3	**eh**	4	**ae**
5	**ah ax ax-h**	6	**uw ux**	7	**uh**	8	**aa ao**
9	**ey**	10	**ay**	11	**oy**	12	**aw**
13	**ow**	14	**er axr**	15	**l el**	16	**r**
17	**w**	18	**y**	19	**m em**	20	**n en nx**
21	**ng eng**	22	**dx**	23	**jh**	24	**ch**
25	**z**	26	**s**	27	**sh zh**	28	**hh hv**
29	**v**	30	**f**	31	**dh**	32	**th**
33	**b**	34	**p**	35	**d**	36	**t**
37	**g**	38	**k**				
39	**bcl pcl dcl tcl gcl kcl epi pau h# cl**						

To minimize the effect of acoustic channel (e.g. microphones, environment) Cepstral Mean Subtraction (CMS) (Young et al., 2006) was applied to the MFCC features. The mean of the MFCCs was calculated for each utterance and subtracted from each MFCC vector for every frame. We have used an HTK front-end to compute MFCC features, apply CMS and computer the derivatives of the features (Young et al., 2006).

A Comparison to HDPHMM

In Table 3-2 we compare the performance of DHDPHMM to HDPHMM. We provide error rates for both the development and core subsets. In this table we have compared an LR model with two other models: a strictly LR topology and an ergodic model. DHDPHMM is consistently

Table 3-2 A comparison of error rates on TIMIT for LR DHDPHMM and HDPHMM is shown. LR DHDPHMM produces a *10%* reduction in error rate and a *15%* reduction in complexity.

Model	Dev Set (% Error)	Core Set (% Error)	No. Gauss.
LR HDPHMM 1	23.52%	24.40%	4628
LR HDPHMM 2	23.86%	25.14%	7281
Ergodic DHDPHMM	24.01%	25.42%	2704
Strictly LR DHDPHMM	39.03%	38.43%	2550
LR DHDPHMM	**20.51%**	**21.42%**	**3888**

better than its HDPHMM counterparts. Further, it can be seen that LR models perform better than ergodic models (as expected) while the LR-S models perform more poorly. This is due to the fact that the LR-S model constrains the best path to one path while the other LR models learn many parallel paths.

From the last column of this table we can see LR DHDPHMM finds *3,888* Gaussians for all *48* phonemes while two different LR HDPHMM models find *4,628* and *7,281* Gaussians for all phonemes respectively. These numbers show DHDPHMM can learn a less complex model that can explain the data better than a more complex model learned by HDPHMM. This is an important result that validates the basic philosophy of the nonparametric Bayesian approaches and also follows Occam's Razor (Rasmussen & Ghahramani, 2001).

It should also be noted after learning the structure and parameters using the block sampler, we refined all the models (reestimating only the means and transition probabilities) using a few more EM iterations. After this EM step, the *3,888* Gaussians are distinct distributions. If we do not apply the EM reestimation step the actual number of distinct Gaussians for DHDPHMM is less than *3,888* Gaussians (for this case, it would be about *1,050* Gaussians). For HDPHMM, the EM reestimation step does not change the number of Gaussians.

Figure 3-8 shows the structures for phonemes */aa/* and */sh/* discovered by DHDPHMM. It is clear that the model structure evolves with amount of data points, validating another characteristic of this type of nonparametric model. It is also important to note that the structure learned for each phoneme is unique and reflects the underlying differences between phonemes. Finally, note that the proposed model learns multiple parallel left-to-right paths. This is shown in Figure 3-8(b) where *S1-S2*, *S1-S3* and *S1-S4* depict three parallel models. These parallel paths model different clusters of speakers and to some extent the context. This is another example that shows sharing parameters enables the model to learn more complicated underlying structures while still estimating model parameters with sufficient accuracy.

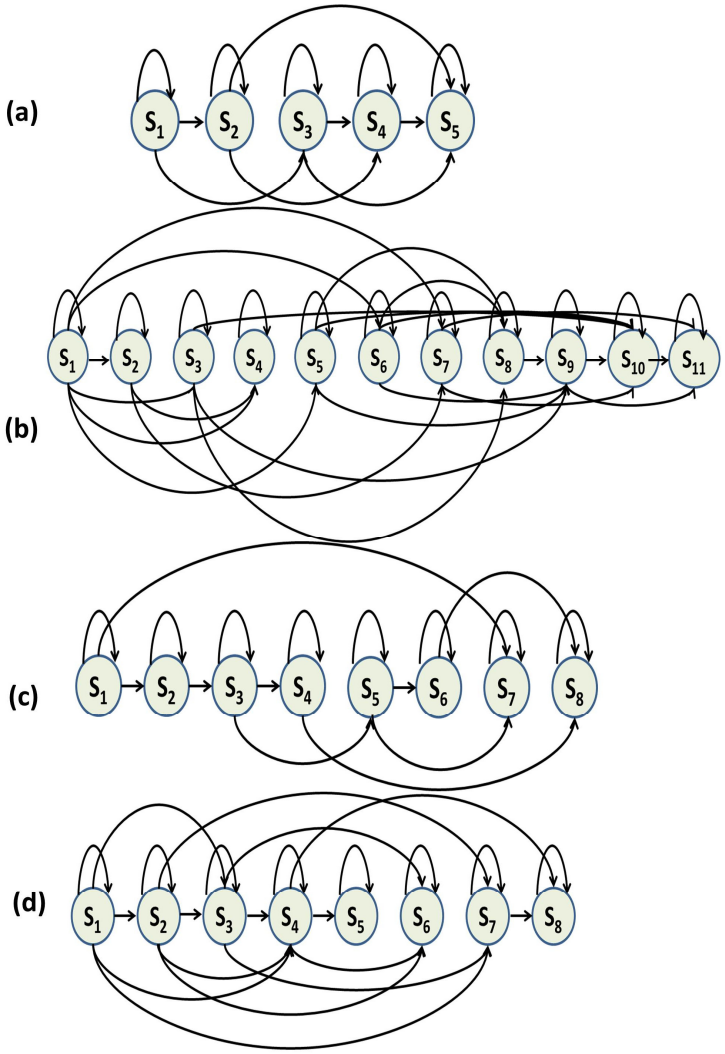


Figure 3-8. An automatically derived model structure is shown for a LR DHDPHMM model (without the first and last non-emitting states) for (a) *//aa//* with *175* examples (b) *//aa//* with *2,256* examples (c) *//sh//* with *100* examples and (d) *//sh//* with *1,317* examples.

Figure 3-9 shows the confusion matrix among different phonemes. From this confusion matrix we can see that most errors occur, as expected, between acoustically similar phonemes. In fact, if we use *5* broad phonetic classes, as displayed in Table 3-3, instead of using *39* phoneme classes, the classification error rate drops to *4.8%*.

A Comparison to Other Representative Systems

Table 3-4 shows a full comparison between DHDPHMM and both baseline and state of the art systems. The first three rows of this table show three-state LR HMMs trained using maximum likelihood (ML) estimation. HMM with *40* Gaussians per state performs better than other two and

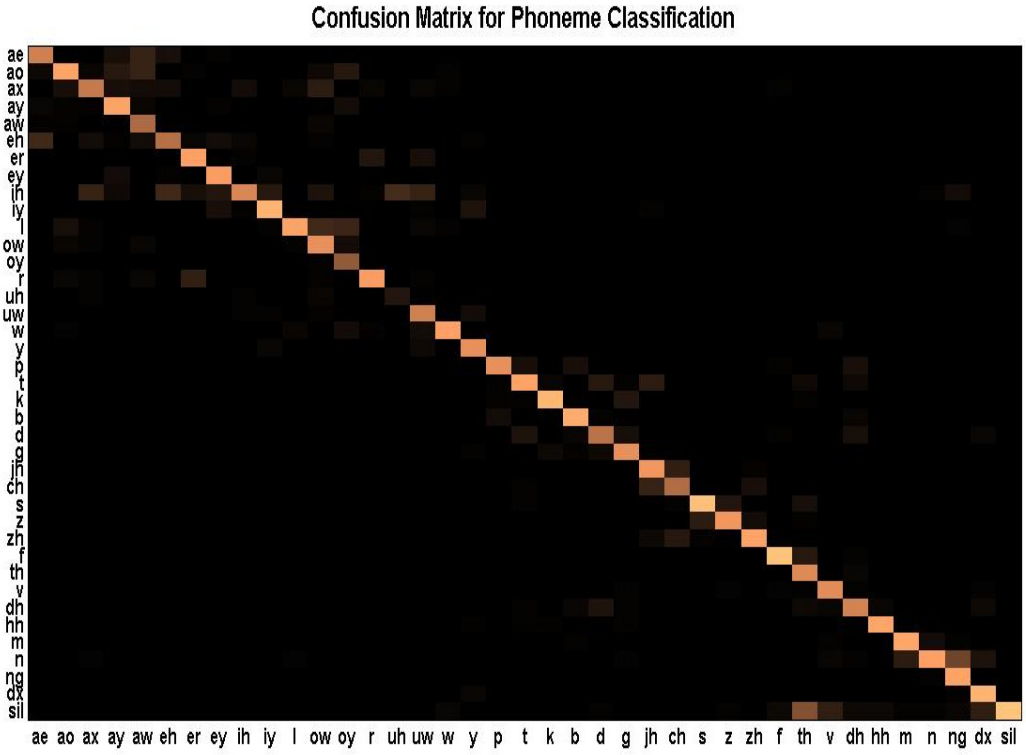


Figure 3-9. The confusion matrix for a phoneme classification experiment is shown.

has an error rate of *26.17%* on the core subset. Our LR DHDPHMM model has error rate of *21.42%* on the same subset of data (a *20%* relative improvement). It should be noted that the number of Gaussians used by this HMM system is *5,760* (set a priori) while our LR DHDPHMM uses only *3,888* Gaussians.

It should be noted that most of the differences between state of the art results in Table 3-4 can be attributed to various data preparation steps. For example, we have trained another model (not in this table) with same parameters as the one reported in this table with slightly different

Table 3-3. A mapping of phonemes to broad phonetic classes is shown.

Class	Phonemes
Vowels	aa, ae, ah, ao, ax, ay, aw, eh, el, er, ey, ih, ix, iy, l, ow, oy, r, uh, uw, w, y
Stops	p, t, k, b, d, g, jh, ch
Fricatives	s, sh, z, zh, f, th, v, dh, hh
Nasals	m, n, ng, en
Silence	sil, vcl, cl, epi, dx

Table 3-4. A comparison of phoneme classification algorithms is shown.

Model	Discriminative Training	Dev Set (% Error)	Core Set (% Error)
HMM (10 Gauss.)	No	28.44%	28.71%
HMM (20 Gauss.)	No	26.16%	27.33%
HMM (40 Gauss.)	No	25.01%	26.17%
HMM/MMI (20 Gauss.) (Gunawardana et al., 2005)	Yes	23.20%	24.60%
HCRF/SGD (Gunawardana et al., 2005)	Yes	20.30%	21.70%
Large Margin GMMs (Sha and Saul, 2006)	Yes	–	21.10%
GMMs/Full Cov. (Sha and Saul, 2006)	No	–	26.00%
SVM (Clarkson and Moreno, 1999)	Yes	–	22.40%
Data-driven HMM (Petrov et al., 2007)	No	–	21.40%
Direct Segment Model (Zweig, 2012)	Yes	–	21.70%
LR DHDPHMM	**No**	**20.51%**	**21.42%**

alignment between transcription and frames and the error changed to *21.81%* (from *21.42%*). Therefore, we can conclude all state of the art results in this table show similar underlying modeling capabilities. A simple statistical significance analysis (Gillick & Cox, 1989) also shows these differences are not statistically significant.

The fourth row of Table 3-4 shows the error rate for an HMM trained using a discriminative objective function (e.g. MMI). We can see discriminative training reduces the error rate. However, the model still produces a larger error rate relative to our ML trained DHDPHMM. This suggests that we can further improve DHDPHMM if we use discriminative training techniques. Several other state of the art systems are shown that have error rates comparable to our model. Data-driven HMMs (Petrov et al., 2007) unlike DHDPHMM, models the context, which seems to be one of the main reasons that it performs so well. We expect to obtain better results if we also use CD models instead of CI models.

Figure 3-10 shows the error rate vs. the amount of training data for both HMM and DHDPHMM systems. As we can see DHDPHMM is always better than the HMM model. For example, when trained only using *40%* of data, DHDPHMM performs better than an HMM using the entire data set. Also it is evident that HMM performance does not improve significantly when we train it with more than *60%* of the data (error rates for *60%* and *100%* are very close) while DHDPHMM improves with more data.

Figure 3-11 shows the number of Gaussians discovered by DHDPHMM versus the amount of data. The model evolves into a more complex model as it is exposed to more data. This growth in complexity is not linear – the number of Gaussians grows *33%* when the amount of data increases *5* times. This is expected due to the use of the DP prior constraints. If we want to change this behavior we would have to use other type of priors.

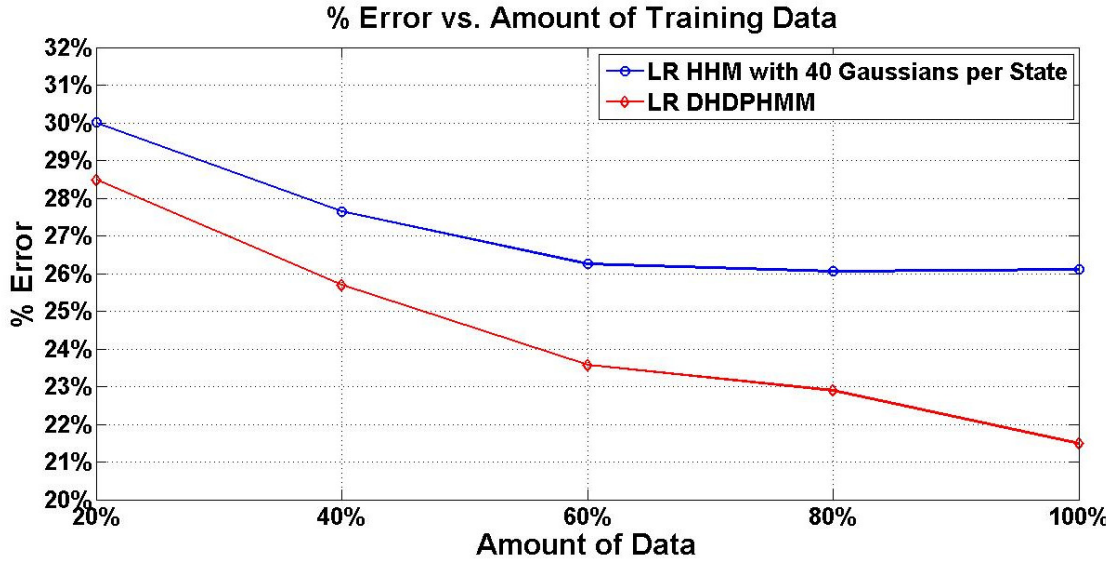


Figure 3-10. The error rate vs. the amount of training data is shown for LR DHDPHMM and LR HMM.

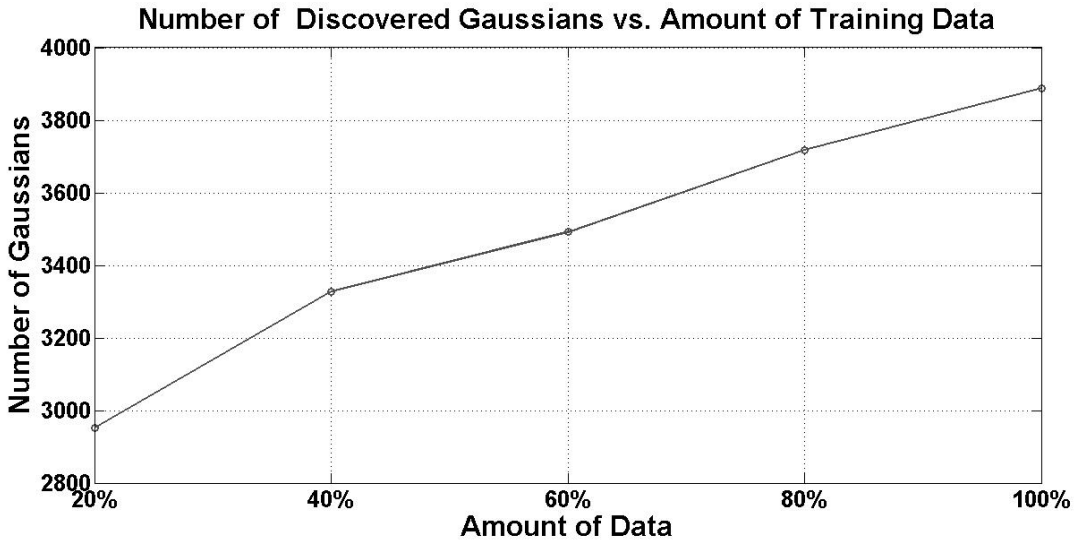


Figure 3-11. The number of discovered Gaussians is shown as a function of the amount of training data.

3.8 Conclusions

In this chapter we introduced an extension of HDPHMM, referred to as DHDPHMM that incorporates a parallel hierarchy to share data between states. We have also introduced methods to model non-ergodic structures. We demonstrated through experimentation that LR DHDPHMM outperforms both HDPHMM and its parametric HMM counterparts. We have also shown that despite the fact that we have only used ML training for DHDPHMM, its performance is better than a discriminatively trained HMM and is comparable to other state of the art discriminatively trained models.

This chapter describes the core contribution of this dissertation and establishes the underlying theory. Chapter 4 will focus on incorporating semi-supervised training and context modeling. In Chapter 5 we will investigate use of nonparametric Bayesian models for automatic discovery of acoustic units.

Extensions to this work not addressed in this dissertation are the effect of using priors other than a Dirichlet process. We have shown that complexity grows very slowly with the data size

because of the DP properties (only *33%* more Gaussians were used after increasing the size of the data five times). Therefore it makes sense to explore other types of prior distributions to investigate how it can affect the estimated complexity and overall performance.

Another possible direction is to replace HDP emissions with more general hierarchical structures such as a Dependent Dirichlet Process (MacEachern , 1999) or an Analysis of Density (AnDe) model (Tomlinson and Escobar, 1999). It has been shown that the AnDe model is the appropriate model for problems involving sharing among multiple sets of density estimators (Tomlinson & Escobar, 1999; Teh et al., 2006).

CHAPTER 4

SEMI-SUPERVISED TRAINING OF DHDPHMM

We have introduced a DHDPHMM in Chapter 3. We have shown these models can outperform HMMs (including discriminatively trained HMMs) and some other state of the art models in a completely supervised task such as classification. In a supervised task, labels and observations are given and the goal is to learn a mapping between them.

In an unsupervised task, on the other hand, we only have the observations and the goal is to infer certain structures or patterns in the data. An example of this type of task is clustering. Well-known solutions to this problem include the popular K-MEANS (Anderberg, 1973) and GMM (Bishop, 2007) algorithms. In statistical modeling this goal is usually stated as learning the distribution of the observations and their properties (Hastie et al., 2009). A middle ground between the supervised and unsupervised tasks is semi-supervised training. In semi-supervised problems we have partial information about the observations.

In most practical hierarchical pattern recognition applications, there are several hidden levels between the observations and the labels. For example, in speech recognition, the labels are words, the observations are feature vectors, and we hypothesize a hidden layer consisting of phonemes. Labeled training data is always labeled at the word level without time alignment, and we must somehow automatically infer the phoneme labels. The alternative, to label phonemes, is time-consuming, costly and prone to error. Lee et al (1989) demonstrated that statistical systems could actually achieve higher levels of performance by using word labels instead of phoneme labels, and treating the phoneme labels as a hidden unit. The training process in such situations is

referred to as semi-supervised training since only word labels are specified and the system must determine the time alignments of these labels as part of the training process.

The challenge in the training task then becomes to segment observations into consecutive blocks each aligned with one of the labels in the label sequence. Once this alignment is performed, we can infer a mapping between labels and observations. This procedure is usually an iterative algorithm; starting from an initial segmentation (e.g. uniform), we learn the mapping and then re-segment the data until convergence is achieved. The great success of HMM-based speech recognizers can be attributed to the relative ease of implementing such a semi-supervised training procedure by exploiting three things: (1) an HMM's sequential properties, (2) the EM algorithm's convergence properties and (3) the Viterbi algorithm's ability to time-align data. As we will show in the following section, this is not the case for DHDPHMM (or HDPHMM) and we need a new semi-supervised algorithm in order to be able to use these models for semi-supervised training.

A related problem is modeling of context. Modeling context is one of the major advances made in speech recognition in the 1990's (Lee, 1990). In CI systems, we use only one acoustic model for each phoneme. However, as explained in Section 1.1, it is well known that due to the physics and cognitive aspects of speech production, adjacent sounds influence one another. This is known as coarticulation (Ladefoged, 1993). Sounds can assimilate to the following sound (referred to as anticipatory assimilation) or a proceeding sound (progressive assimilation) (Jun, 1995). This suggests modeling context can improve the speech recognition task significantly (Schwartz et al., 1985). We will investigate effect of context modeling for DHDPHMM models in Section 4.2.

This chapter is divided into four sections. In Section 4.1 we introduce a model for semi-supervised training of DHDPHMM and derive its approximations. In Section 4.2 the context modeling problem will be investigated. In Section 4.3 experimental results are presented on the

TIMIT Corpus. In Section 4.4 we summarize the results of this chapter and propose some additional approaches to the context modeling problem.

4.1 Semi-Supervised Training of DHDPHMM Models

Let us begin by reviewing semi-supervised training of HMMs and then discuss why this procedure can't be used directly for DHDPHMM and HDPHMM. Next, we will introduce a generative model for semi-supervised training of DHDPHMM and a corresponding approximation algorithm that allows us to use DHDPHMMs in semi-supervised task similar to acoustic modeling.

HMM acoustic models are usually trained using a special extension of the EM algorithm also known as embedded training (Young et al., 2006). The procedure takes advantage of non-emitting states of individual HMMs to generate a composite HMM for each utterance. This is one reason why the treatment of non-emitting states presented in Section 3.5 was important. Typically we have several utterances and their corresponding transcriptions and a lexicon. A lexicon is used to map transcripts into a sequence of phonemes. However, as discussed before, the time alignment between the acoustic units (e.g., phonemes) and acoustic observations (e.g., feature vectors) is not known.

One solution to this problem is to build a composite HMM for each utterance. A composite HMM is nothing more than a concatenation of the individual HMMs into one large HMM. Since one of the most useful properties of any HMM is to segment the observations into states, the composite HMM will implicitly segment the utterance into individual phoneme HMMs and update the corresponding parameters for each HMM. This process is reminiscent of dynamic time warping (Furui, 1986) and is an integral part of any speech recognition training process. This procedure is executed for all utterances to complete a single iteration of the algorithm. Model

parameters are then updated once after all training utterances have been aligned and processed (Young et al., 2006). This is often referred to as batch mode training.

The above simple procedure is all that is required to train HMMs without having time alignments for each phoneme in the utterance. We must pay special attention to how silence (or non-speech) is inserted into this process so that we can account for arbitrary amounts of silence between words (Alphonso, 2003). However, the details of this process are beyond the scope of this dissertation.

4.1.1 Composite DHDPHMM Model

We cannot directly adapt the composite HMM approach to DHDPHMM because a DHDPHMM learns its structure (e.g. number of states and how they are connected) from the corresponding observations. Therefore, there are no initial models available to generate a composite model. However, we can introduce a generative model that enables us to build a composite DHDPHMM for semi-supervised training.

Let's assume we have a list of all models denoted by $\boldsymbol{m}$:

$$\boldsymbol{m} = [m_a, m_b, ..., m_{hh}, ..., m_{ie}, ..., m_s, ..., m_z], \tag{4.1}$$

where in this definition m_a is the model corresponding to the phoneme */a/*. Let us also define an array of utterances, U. Each utterance is indexed by an integer number u. For example, consider the 27^{th} utterance in our list, and let's assume for $u = 27$, we have: *U[27] = /hh ie s hh/*. Therefore it is clear that each utterance is a sequence of models defined in (4.1), and the model assignments for entire corpus are enumerated in U.

For each utterance we have an array, or list, of models. We can define a data structure, Q, that contains all models for all utterances, indexed by the position of the model in the list. For example, $Q[27] = \{1{:}m_{hh}, 2{:}m_{ie}, 3{:}m_s, 4{:}m_{hh}\}$. Let L_u denote the total number of models used for utterance u. We have:

$$Q[u] = \{1 : m_{n1}, 2 : m_{n2}, ..., L_u : m_{nL_u}\}. \tag{4.2}$$

We also need a function to return the model given the index of the model in the utterance, which we will define as $\varphi(i, u, Q)$:

$$\varphi(i,u,Q) = \{\text{Return the model indexed by i from } Q(u)\}. \tag{4.3}$$

In order to obtain a generative model for a composite DHDPHMM we need to develop a generative model for an utterance given the labels. Continuing our example of the 27^{th} utterance that consists of the models */hh ie s hh/*, we can represent these models with a Markov chain as shown in Figure 4-1. Consider an observation sequence O_j and utterance membership function, Φ, with length T:

$$O_j = o_1 o_2 o_3 ... o_T, \tag{4.4}$$

$$\phi(O_i) = I_1 I_2 ... I_T, \tag{4.5}$$

where I_k is the model index for the k^{th} observation o_k.

In order to generate O_j we first have to sample the Markov chain shown in Figure 4-1 to determine which model produced the observation o_k. We can write the boundary conditions as:

$$\begin{aligned} I_1 &= 1, \\ I_i \mid (I_{i-1} = L_u) &= L_u. \end{aligned} \tag{4.6}$$

The Markov relationship can be written as:

$$I_i \mid I_{i-1} \sim \varepsilon(I_{i-1}, u, Q) = \begin{cases} I_{i-1} & \text{with probability} \quad \rho_{\varphi(I_{i-1},u,Q)} \\ I_{i-1}+1 & \text{with probability} \quad 1-\rho_{\varphi(I_{i-1},u,Q)} \end{cases}, \tag{4.7}$$

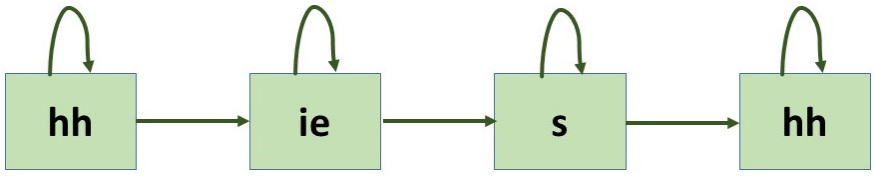


Figure 4-1. A Markov chain that represents a composite HMM is shown.

where $1 < I_i < L_u$ and $\rho_{\varphi(I_i,u,Q)}$ is the self-transition probability for the model indexed by I_i in utterance u. Starting from $i = 1$ we select the first model in the list and generate an observation using this model, and then we use (4.6) and (4.7) to select the next model. This is either the same model (a self-transition) or the next model in the Markov chain.

A formal definition based on (3.24) can be written as:

$$
\begin{aligned}
&\beta_{\varphi(I_t,u,Q)} \mid \gamma \sim GEM(\gamma) \\
&\beta'_j \mid V_j = \frac{V_j \cdot \beta_{\varphi(I_t,u,Q)}}{\sum_i V_{ji} \beta_{\varphi(I_t,u,Q)i}} \\
&\pi_{\varphi(I_t,u,Q)j} \mid \alpha, \beta'_j \sim DP(\alpha + \kappa, \frac{\alpha\beta'_j + \kappa\delta_j}{\alpha + \kappa}) \\
&\xi_{\varphi(I_t,u,Q)j} \mid \tau \sim GEM(\tau) \\
&\psi_{\varphi(I_t,u,Q)j} \mid \sigma, \xi \sim DP(\sigma, \xi) \\
&\theta^{**}_{\varphi(I_t,u,Q)kj} \mid H, \lambda \sim H(\lambda) \\
&z_{\varphi(I_t,u,Q)t} \mid z_{\varphi(I_t,u,Q)t-1}, \left\{\pi_{\varphi(I_t,u,Q)j}\right\}_{j=1}^{\infty} \sim \pi_{\varphi(I_t,u,Q)z_{t-1}} \\
&s_{\varphi(I_t,u,Q)t} \mid \left\{\psi_{\varphi(I_t,u,Q)j}\right\}_{j=1}^{\infty}, z_{\varphi(I_t,u,Q)t} \sim \psi_{\varphi(I_t,u,Q)z_t} \\
&x_t \mid \left\{\theta^{**}_{\varphi(I_t,u,Q)kj}\right\}_{k,j=1}^{\infty}, z_t \sim F\left(\theta_{\varphi(I_t,u,Q)z_t s_t}\right) \\
&I_t \mid \mathrm{I}_{t-1} \sim \varepsilon(I_{t-1}, u, Q),
\end{aligned}
\tag{4.8}
$$

where in this equation each DHDPHMM is indexed by $\psi(I_t,u,Q)$. Equation (4.8) defines a composite DHDPHMM for the utterance indexed by u. A composite DHDPHMM is defined for every utterance in the dataset. Therefore, to generate the entire dataset we have to first select an utterance index and then use this generative model to generate observations for that utterance.

The definition in (4.8) makes it clear how to generate observations for an utterance given a sequence of labels and a set of DHDPHMMs. We can also define an inference algorithm similar to a block sampler for HDPHMM (or modified block sampler for DHDPHMM) but the result would be extremely inefficient and from a computation point of view impractical because:

1. Convergence of the algorithm would be too slow because we have to sample I_t.

2. For sampling I_t we also need to sample the self-transition probability, ρ in (4.7), for every model. This sampling of ρ involves computing the likelihood that x_t belongs to the current or the next model. This is a time-consuming operation because we have to compute this for all observations for each and every utterance.
3. All models are updated sequentially and hence we cannot use coarse parallelization. This is a serious problem because we have at least 48 models and running them in parallel makes using DHDPHMM feasible but running them sequentially makes the algorithm impractical.

As a result we need to use some sort of approximation that preserves the basic properties of (4.8) while maintaining efficiency. In the next section we see how this can be done by leveraging the Viterbi algorithm.

4.1.2 Approximation of the Generative Model for Semi-Supervised Training

Exact inference of a composite DHDPHMM in (4.8) is computationally difficult because all models are linked together through I_t and x_t. However, if I_t was known, the composite DHDPHMM reduces to a collection of independent DHDPHMMs (x_t can be segmented based on I_t). We can divide this problem into two sub-problems: (1) segmentation of the observations into aligned blocks with labels, and (2) DHDPHMM training. The first problem is a well-known problem in speech recognition and can be solved using a forced alignment process that is based on the Viterbi algorithm (Alphonso, 2003). The second problem has already been addressed in Chapter 3.

Assuming that we have a set of phoneme DHDPHMM models, we can generate a regular composite HMM by connecting these HMMs together based on a given sequence of labels and utilizing non-emitting states. We can use the Viterbi algorithm to find a time alignment between observations and states for the composite HMM. However, since these states can also be assigned to phoneme DHDPHMMs we can also find the alignment between the labels and observations.

Suppose the initial probabilities, transition probabilities and output probabilities are given by π_{init}, π, and b, respectively. The Viterbi algorithm can be described as follows (Viterbi, 1967; Alphonso, 2003; Huang et al., 2001):

1. Initialization:

$$\begin{aligned} V_1(i) &= \pi_{init}(i) b_i(o_i) \\ B_1(i) &= 0 \end{aligned} \qquad 1 \le i \le N. \tag{4.9}$$

2. Induction :

$$V_t(j) = \max_{1 \le i \le N} \left[V_t(i) \pi_{ij} \right] b_j(o_i), \qquad 2 \le t \le T, 1 \le j \le N, \tag{4.10}$$

$$B_t(j) = \arg\max_{1 \le i \le N} \left[V_t(i) \pi_{ij} \right], \qquad 2 \le t \le T, 1 \le j \le N. \tag{4.11}$$

3. Termination and back-tracking:

$$s_T^* = \arg\max_{1 \le i \le N} [B_T(i)], \tag{4.12}$$

$$s_t^* = B_{t+1}(s_{t+1}^*), t = T-1, T-2, ..., 1, \tag{4.13}$$

$$S^* = \left(s_1^*, s_2^*, ..., s_T^* \right). \tag{4.14}$$

where S^* is the optimum path through the composite HMM and as stated above can be used directly to find the time alignment between the observations and the sequence of labels.

The Viterbi algorithm can find a time alignment if we already have the HMM models. But, we don't have the HMM models yet. Therefore we need to use a two-step iterative approach in which we estimate the model structure and parameters in the first step and align the observations in second step. The resulting algorithm is as follow:

1. Initialize the alignment using a heuristic method.
2. Use the alignment to generate a list of examples for each DHDPHMM.
3. Use examples generated in Step 2 to train each DHDPHMM using the modified block sampler of Chapter 3.

4. Use models obtained in Step 3 to time align the labels to observations using the Viterbi algorithm (by first generating a composite HMM for each utterance).
5. If the maximum number of iterations is reached or a convergence criterion is met, exit. Otherwise, return to Step 2.

In this algorithm convergence can be checked by calculating the average log-likelihood of the models for training or a separate development data set can be used. Initialization can be done using one of several heuristic methods that are discussed in Section 4.3.

4.2 Context Modeling

Context modeling has been previously discussed in Section 1.1. One of most popular methods for context modeling takes into account the left and right context of a sound (Lee, 1990). This type of acoustic model is referred to as a triphone. For example, "*k-ae+t*" represents */ae/* proceeded by */k/* and followed by /t/. The center context, /ae/, defines the monophone class with which this triphone is associated for clustering or tying. The number of potential triphones in a typical speech recognizer is very large (e.g. *42x42x42= 74088*). Many triphones occur infrequently and have little or no data associated with them in a typical training corpus. Therefore, we have to share parameters among triphones to achieve good performance.

Sharing parameters can be achieved in many different ways (e.g. tie the models or tie the states) but the most common approach is to use a phonetic decision tree to cluster triphone states (Young & Woodland, 1994). A decision tree is a binary tree that classifies data by asking binary (e.g. yes/no) questions. The questions are based on the linguistic properties of the sound, such as "is the left phoneme a stop?" The tree is built by successively splitting the data by selecting the question that causes more entropy reduction. We usually set a threshold on the minimum number of data points in a single node and stop growing the tree once the number of data points reaches

this threshold. We typically assume each state is modeled by a single Gaussian and use the parameters of this model for all likelihood calculations required by the tying algorithm.

Though it is possible to model context in a nonparametric Bayesian framework by using an additional level in our hierarchy that models relationships between models and states, due to time and resource constraints, we did not develop this approach in this dissertation. We have instead used two approaches to explore context modeling in a DHDPHMM: (1) tie triphones using a decision tree framework, or (2) direct context modeling based on broad phonetic classes.

Decision Tree Based Tying

In this approach we first train CI models using the semi-supervised training algorithm discussed in Section 4.1. After training CI models we follow the standard procedure to obtain a decision tree and further train the resulting CD models using the EM algorithm (Young & Woodland, 1994). However, since the algorithms for generating the decision tree only use a single Gaussian distribution to represent each state, we have to perform some model preparation before we can apply this technique. The algorithm is as follows:

1. Train CI models using a semi-supervised algorithm.
2. Clone CI models into the appropriate CD models.
3. Generate a copy of the CI models where all the mixtures are collapsed into a single Gaussian. This can be done by using a weighted average approach: the single mixture approximation is generated by computing a weighted average of all the mixture components.
4. Use the models generated in Step 3 to generate sufficient statistics (mean, covariance and counts for each Gaussian) by training the data using several iterations of the EM algorithm.
5. Use the generated sufficient statistics in Step 4 to grow a phonetic decision tree.

6. Discard the models generated in Step 3 and instead use the original CI models trained in step 1 along with the tree generated in Step 5 to train the final triphone models (using EM).

In this approach it should be noted that we first fix the structure for all triphones that share the same phoneme as the center context, then use a phonetic decision tree to cluster states for all triphones with the same center context, and finally reestimate the tied triphone models using EM.

Using Broad Phonetic Class Context

An alternative approach is to use context based on the broad phonetic class (BPCC). Since we have five different broad phonetic classes (see Table 3-3), the number of models would be *5x5x48 = 1,200* models. Even for this modest number of models we have models that are not observed in the training data or are represented with only few examples. For these models we can back off to the corresponding CI model. The procedure for broad phonetic context modeling is as follows:

1. Train CI models using a semi-supervised algorithm.
2. Clone CI models into BPCC models. Each BPCC is initialized based on its central phoneme (e.g. all BPCC with central phoneme */hh/* will be initialized with CI */hh/* model).
3. Scan the training transcription and identify BPCCs with a small number of examples.
4. For models with a sufficient number of examples (e.g. *100*) train the BPCC models using EM. Other models are automatically backed off to their CI counterparts.

Again we fix the structure for all BPCC models with the same central phoneme and only use EM to further adjust CI parameters to the specific context. Note that state tying is not used.

4.3 Experiments

In this section we provide some experimental results that compare supervised training versus semi-supervised training algorithms for DHDPHMM. We use two different methods to initialize the semi-supervised training algorithm. Both semi-supervised and supervised results will be compared with the baseline and other state of the art systems. Furthermore, we also compare CD trained DHDPHMMs with a comparable HMM system.

4.3.1 Evaluation Methods

In the phoneme recognition problem, unlike phoneme classification, the boundaries between subsequent phonemes are not known (during the recognition phase) and should be estimated along with phoneme labels. During the recognition process we have to decide if a given frame belongs to the current group of phonemes under consideration or we have to initiate a new phoneme hypothesis. This decision is made by considering both the likelihood measurements and the language model probabilities. All systems compared in this section use a bigram language model. However, the training procedure and optimization of each language model is different and has some effect on the reported error rates.

In the following we define *% Correct* and *% Error* as follows (Young et al., 2006):

$$\%\,Correct = \frac{N - S - D}{N}, \tag{4.15}$$

$$\%\,Error = \frac{S + D + I}{N}, \tag{4.16}$$

where N is the total number of labels in the reference transcriptions, S is the number of substitution errors, D is the number of deletion errors and I is the number of insertion errors.

Similar to the phoneme classification experiments in Chapter 3 we have used the TIMIT Corpus for all phoneme recognition experiments in this section. TIMIT is a natural choice to

compare supervised and semi-supervised training algorithms since it was manually segmented into phonemes. The front-end configuration is exactly the same as the experiments in Chapter 3.

4.3.2 Supervised Phoneme Recognition

In the first experiment we use a completely supervised method to evaluate DHDPHMMs. This allows us to compare supervised and semi-supervised methods for DHDPHMM and to determine if our semi-supervised algorithm based on approximation of the generative model works correctly. In this section, DHDPHMMs were trained only using maximum likelihood and did not use context information.

The process for training supervised models for a recognition task is exactly the same as the process we used to train DHDPHMM for the classification task in Section 3.7.4. We first extract the data for each phoneme based on the manual transcription and then we train each DHDPHMM using the corresponding time-aligned data. The recognition stage differs from the decoding stage for the classification task and involves searching through the space of all possible hypothesis guided by the language model and the acoustic evidence.

Table 4-1 shows the detailed results for both the LR DHDPHMM and HMM baseline systems. As this tables shows, the error rate decreases from 32.64% to 29.23 % (11% relative) for the full evaluation set while correct recognition rate increases from 70.72% to 74.09%. This is a major improvement relative to the baseline HMM and is consistent with classification results reported in Table 3-4.

Table 4-1. Supervised training results comparing DHDPHMM and HMM are shown.

Model	Subset	% Correct	% Sub.	% Del.	% Ins.	% Error
HMM	Core	70.72	19.98	9.30	3.36	32.64
HMM	Dev.	71.94	19.27	8.78	3.51	31.57
HMM	All	71.62	19.56	8.82	3.70	32.08
LR DHDPHMM	Core	74.09	17.95	7.96	3.79	29.71
LR DHDPHMM	Dev.	75.10	17.45	7.45	3.72	28.62
LR DHDPHMM	All	74.70	17.72	7.57	3.94	29.23

4.3.3 Semi-Supervised Phoneme Recognition

In Section 4.1.2 we introduced a semi-supervised training algorithm that approximates a generative model for semi-supervised training of DHDPHMM. We also mentioned that the algorithm needs to be initialized using a heuristic method. In this section, we investigate the performance of that algorithm and compare two different approaches for initialization:

1. Uniform segmentation (Uniform): In this approach we simply spilt observations equally between different consecutive models (essentially dividing the number of frames by the number of models).
2. Forced Alignment (CDHMM): This approach is based on using the forced alignments between HMM acoustic models trained using another speech recognizer and the utterance. These acoustic models are CD triphones and used to segment the data to initialize the algorithm. Note that it is not uncommon to use hybrid systems like this. For example deep-learning based approaches usually initialized used CDHMMs (Hinton et al., 2012).

Figure 4-2 shows the error rate on the development set for the two approaches to initialization. It can be seen that the forced alignment approach produced better results, though the differences diminish as more iterations of reestimation are used. Table 4-2 shows a comparison of the initialization methods to the HMM baseline (which uses CDHMM). We can see that initialization using CDHMM outperforms uniform initialization. Uniform initialization of DHDPHMM performs worse than the HMM baseline while CDHMM initialization of DHDPHMM reduced the error rate by 7% relative. It is also interesting to note that DHDPHMM always gave a lower substitution error rate relative to HMM. This can be attributed to better modeling capabilities of DHDPHMM. By comparing to the supervised trained model of Table 4-1 we can see, as expected, semi-supervised models outperform the supervised models.

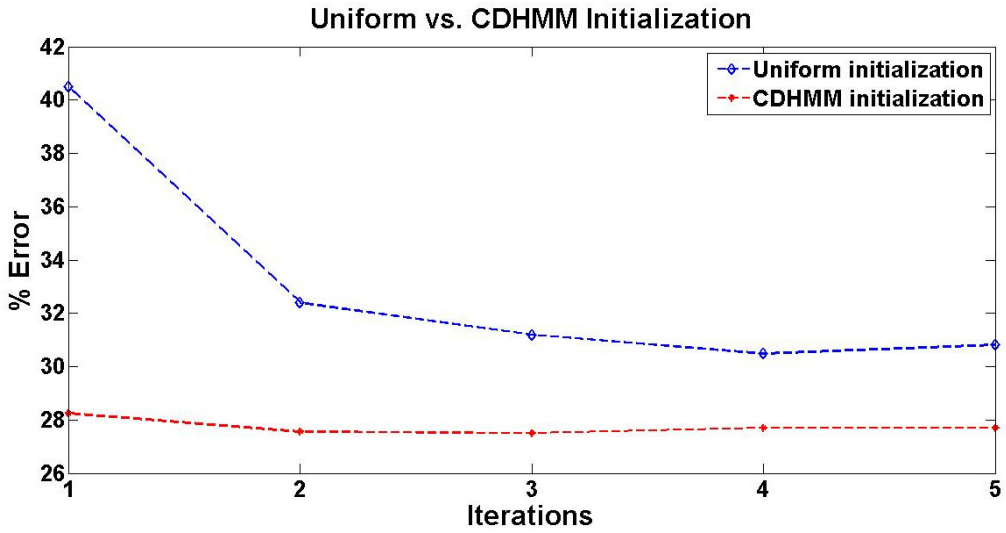


Figure 4-2. A comparison of model initialization methods for DHDPHMM is shown.

It should be emphasized that we have used an approximation algorithm which always performs worse than the exact algorithm and is sensitive to initialization. Therefore further research should be conducted to find better initialization methods or other approximations that match the generative composite DHDPHMM more closely.

4.3.4 Context Dependent Phoneme Recognition

In this section we provide experiments to compare the performance of context dependent DHDPHMM models with HMM baselines. We present both triphone DHDPHMM models and broad phonetic context (BPCC) models.

Table 4-2. A comparison of initialization methods for DHDPHMM and HMM is shown.

Model	Initial.	Subset	% Corr.	% Sub.	% Del.	% Ins.	% Err.
HMM	-	Core	72.92%	19.22%	7.86%	3.96%	31.05%
HMM	-	Dev.	74.07%	18.55%	7.39%	3.93%	29.86%
HMM	-	All	73.84%	18.73%	7.42%	4.03%	30.17%
LR DHDPHMM	Uniform	Core	71.53%	18.77%	9.70%	3.88%	32.35%
LR DHDPHMM	Uniform	Dev.	73.37%	16.78%	9.86%	3.85%	30.48%
LR DHDPHMM	Uniform	All	72.92%	17.31%	9.77%	4.08%	31.16%
LR DHDPHMM	CDHMM	Core	74.40%	17.73%	7.87%	3.42%	29.02%
LR DHDPHMM	CDHMM	Dev.	76.32%	16.91%	6.77%	3.88%	27.56%
LR DHDPHMM	CDHMM	All	75.60%	17.43%	6.97%	3.96%	28.36%

Table 4-3 compares a CD HMM baseline with both CD DHDPHMM and BPCC DHDPHMM. Both models were trained using EM after fixing the CI model structure. In other words, we have not implemented any nonparametric Bayesian model to learn context dependent models directly and CD modeling was only used to further refine the nonparametric CI models. We observe that the advantages of using DHDPHMM diminish when using CD models. The error rate decreases only 1.5% relative to the triphone HMM baseline (27.93% vs. 27.51%) while for semi-supervised CI model the error rate reduced by 7% relative to the semi-supervised CI baseline (31.05% vs. 29.02%). However, the CD systems are significantly more complex than the CI systems, containing two orders of magnitude more models. It remains a computational and algorithmic challenge to train CD models directly using DHDPHMM and nonparametric Bayesian approaches.

4.3.5 Comparisons to Other Popular Systems

Table 4-4 summarizes the results of DHDPHMM models. Table 4-5 presents a comparison of DHDPHMM to several state of the art systems. As we can see, systems can be divided into two

Table 4-3 A comparison of systems for context dependent models is shown.

Model	Subset	% Corr.	% Sub.	% Del.	% Ins.	% Error
Triphone HMM	Core	76.29%	17.87%	5.85%	4.21%	27.93%
Triphone HMM	Dev.	77.82%	16.68%	5.49%	4.22%	26.40%
Triphone HMM	All	77.67%	16.92%	5.41%	4.68%	27.01%
BPCC LR DHDPHMM	Core	73.07%	18.23%	8.70%	3.10%	30.03%
BPCC LR DHDPHMM	Dev.	75.11%	17.08%	7.81%	4.20%	29.10%
BPCC LR DHDPHMM	All	74.44%	17.56%	8.00%	4.36%	29.92%
Triphone LR DHDPHMM	Core	75.76%	17.16%	7.08%	3.27%	27.51%
Triphone LR DHDPHMM	Dev.	76.75%	16.50%	6.75%	3.49%	26.74%
Triphone LR DHDPHMM	All	76.71%	16.71%	7.01%	3.40%	26.80%

Table 4-4 A summary of DHDPHHMM results is shown.

Model	Supervised	Context Modeling	% Error	% Correct	Subset
CI LR DHDPHMM	Yes	No	29.71 %	74.09 %	Core
CI LR DHDPHMM	Yes	No	28.62 %	75.10 %	Dev.
CI LR DHDPHMM	Yes	No	29.23 %	74.70 %	All
CI LR DHDPHMM	No	No	29.02%	74.40%	Core
CI LR DHDPHMM	No	No	27.56%	76.32%	Dev.
CI LR DHDPHMM	No	No	28.36%	75.60%	All
CD LR DHDPHMM	No	Yes	27.51%	75.76%	Core
CD LR DHDPHMM	No	Yes	26.74%	76.75%	Dev.
CD LR DHDPHMM	No	Yes	26.80%	76.71%	All

groups based on their training method (discriminative or not) and context modeling. The first two rows of Table 4-5 are HMM baselines. We can see that DHDPHMM works much better than a comparable CI HMM model – the error rate drops from 31.05% for HMM to 29.02% for DHDPHMM.

The third and fourth rows show two context-dependent HMM models. We can see that CI DHDPHMM performs slightly better than the CD model in row three (CD HMM 2) but slightly worse than CD model of row four (CD HMM 3). However, CD DHDPHMM works better than all CD HMM systems presented in this table. The fact that CI DHDPHMM works better than some of the CD models is a very important result. As noted before, CI systems are much simpler than CD systems (*48* models vs. more than *4,000* models). Also, our CI models are completely nonparametric Bayesian models while our CD DHDPHMMs are EM trained models based on CI DHDPHMM models. This is one of the reasons that the gain for CD models vanishes relative to the gain for CI models. We expect if we can model CD models directly using nonparametric approaches we can restore some of this lost gain.

Our CI model also performs better than a discriminatively trained CI HMM (MMI 1 and MMI 2). However, we can see that a discriminatively trained CD HMM (row seven) gives slightly better results relative to the CD DHDPHMM model trained only using maximum likelihood. Note this system has been trained using a different discriminative framework and uses

Table 4-5. A comparison of DHDPHMM with other common systems is shown.

Model	Discrim. Training	Context Modeling	% Error	% Correct	Subset
Baseline CI-HMM	No	No	31.05%	72.92%	Core
Baseline CD-HMM	No	No	27.93%	76.26%	
CD HMM 2 (Lee and Hon, 1989)	No	Yes	30.90%	–	Core
CD HMM 3 (Young and Woodland, 1994)	No	Yes	27.70%	–	Core
HMM MMI 1 (Kapadia et al. ,1993)	Yes	No	32.50%	73.5%	Random
HMM MMI 2 / Full Cov. (Kapadia et al. ,1993)	Yes	No	30.30%	74.4%	Random
CD HMM /DM (Watanabe et al., 2010)	Yes	Yes	26.70%	–	–
Heterogeneous Class. (Halberstadt and Glass, 1998)	Yes	Yes	24.40%	–	Core
Data-driven HMM (Petrov et al., 2007)	No	Yes	26.40%	–	Core
Large Margin GMM (Sha and Saul, 2006)	Yes	No	30.10%	–	Core
CRF (Morris and Fosler-Lussier, 2008)	Yes	No	29.90%	73.2%	All
Tandem HMM (Morris and Fosler-Lussier, 2008)	Yes	Yes	30.60%	75.6%	All
CNN/CRF (Palaz et al., 2013)	Yes	Yes	29.90%	–	Core
Direct Segmental Model (Zweig, 2012)	Yes	No	33.10%	-	Core
CI HCRF – MPE initialized (Sung & Jurafsky, 2009)	Yes	No	28.30%	-	Core
CI HCRF – ML initialized (Sung & Jurafsky, 2009)	Yes	No	29.00%	-	Core
Deep Belief Network (Hinton et al., 2012)	Yes	Yes	20.00%	–	Core

a Finite State Machine (FSM) decoder (Mohri et al., 2008). Therefore, it is not a completely fair comparison. Also statistical significance test shows the difference between this system and ours is not statistically significant.

It is also important to note that two of the models that show state of art results for phoneme classification task (Table 3-4), Large Margin GMM (Sha & Saul, 2006) and Direct Segmental Model (Zweig, 2012), show much worse performance than our CI model on this recognition task.

Phoneme recognition is much more difficult than phoneme classification task and new algorithms usually perform poorly on recognition tasks initially until more elaborate training techniques are developed. In case of DHDPHMM we can see our CI models are already among the best CI models (better than discriminately trained HMMs and comparable to discriminatively trained CI HCRF). However, as discussed before, our current CD models still need further work to reach to their full potential.

From Table 4-5, we can see models based on deep learning give the state of the art results that are much better than any HMM/GMM based system. However, it should be noted that these impressive improvements are not directly a result of better modeling capabilities of HMM/DNN models but are a result of long context windows used in these systems (Pan et al., 2012). It has been shown that if we use a short window (e.g. one frame) this gain disappears. However, since HMM/GMM based models (including DHDPHMM) can't handle highly correlated observations, a comparison between HMM/DNN and HMM/GMM (including DHDPHMM) is unfair. Nevertheless, further research on the DHDPHMM/DNN system seems to be one of the next logical steps for our research.

4.4 Conclusion

In this chapter we have introduced a generative composite DHDPHMM and an approximation algorithm that allows us to train DHDPHMM (and also HDPHMM) in a semi-supervised fashion. It has been shown the resulting models perform better than both supervised DHDPHMMs and HMMs. Our results indicates our CI DHDPHMM models are among the best CI models and their performance can even be compared with some of the published CD systems.

We have also studied the improvement that can be gained by further refining our CI DHDPHMM models using traditional tied-triphone CD modeling. As expected we have found the improvements over the baseline are modest. It is expected that further improvements can be

obtained by training CD models using a nonparametric Bayesian framework. However, due to the complexity of the resulting CD system and our computational constraints, we did not pursue this approach.

We have also noted that HMM/DNN systems currently produce the state of art results for many speech recognition tasks because they combine sequence modeling capabilities of HMMs with classification capabilities of DNNs. DNNs can use highly correlated data, unlike GMMs and their nonparametric counterparts. Therefore, another promising direction is to study the performance of a hybrid DHDPHMM/DNN system. Integration of deep learning algorithms and nonparametric Bayesian modeling is an emerging area of machine learning. Two examples of such systems are the hybrid system based on DHDPHMM/DNN proposed here and hierarchical deep learning (Salakhutdinoy et al., 2013).

CHAPTER 5

ACOUSTIC UNIT AND LEXICON DISCOVERY

Recently more attention has been given to speech recognition in languages for which few resources exist. We often refer to these as less common or low resourced languages. For example, IARPA's Babel program (Harper, 2011) sponsored a competition to create a speech to text system in a mystery language in one week of time using very limited resources. Though traditional complex CD-based systems perform well when there are ample resources, it is hard to develop such systems when minimal resources exist. The high level goal of the work presented in this dissertation is to address this problem by reducing the complexity of the system.

This chapter is organized as follows. In Section 5.1, we review the motivation for this work. In Section 5.2 we review related work on lexicon discovery. In Sections 5.3 and 5.4, we introduce our approach for discovering the transducer and the lexicon respectively. Finally, in Section 5.5 experimental results are presented on segmentation, the relationship of ADUs to phonemes, lexicon discovery and the STD task.

5.1 Motivation

Modern speech recognition technology is based on decoding new acoustic observations by searching through a space defined by all possible combinations of these units. The relationship of these units to words, which is defined by a lexicon, plays an important role in this process. Accounting for the relationship between words, which is the function of a language model, is also important. To train a speech recognizer, we need speech signal data which is converted to a sequence of acoustic observations through the feature extraction process. Usually we are given a parallel word transcription (e.g. word level transcription without time alignment) with these

acoustic features. However, in the low resourced language scenario, the only available data are the acoustic observations.

The existence of a lexicon is not always guaranteed. For example, many languages do not have a suitable lexicon or even a writing system (e.g. African click languages). Even for widely used languages such as English, lexicons often have to be frequently updated because new words are constantly being added to the language. As a result, lexical resources often are a bottleneck in the process of building a speech recognizer for new languages, and construction of a lexicon is a very expensive and time-consuming process that requires a significant amount of linguistic expertise. To make matters worse, such expertise might not be readily available.

A lexicon essentially determines what sub-word units will be used in the speech recognizer. Most if not all languages can be represented in terms of a universal set of phonemes such as the International Phonetic Alphabet (Ladefoged, 1990) or Worldbet (Hieronymus, 1993). Most state of the art systems use phoneme units following what has been done in common languages such as English. However, though in theory all languages can be represented by some finite set of fundamental units, phoneme units might not be optimal for all other languages, or the specifics of these units can be an engineering problem in itself. When adding new words to the lexicon, we have to map these words into phonemes either automatically or manually. The former requires sophisticated software; the latter requires linguistic expertise. Therefore if we can derive the lexicon and these underlying units automatically from the data, we can circumvent these problems.

In this chapter, we study the problem of unsupervised discovery of acoustic units and a related problem – automatic segmentation of speech. Most approaches to automatic discovery of acoustic units (Bacchiani & Ostendorf, 1999) do this in two steps: (1) segmentation and (2) clustering. Segmentation is accomplished using a heuristic method that detects changes in energy and/or spectrum. Similar segments are then clustered using an agglomerative method such as a decision tree. Our approach is based on training a transducer that can combine both stages into

one and directly maps the observations into ADUs. Learning the transducer is completely unsupervised and we only use the acoustic data with no language knowledge or transcription.

5.2 Related Work

We first study the segmentation properties of the transducer and will show its performance is superior to other unsupervised algorithms. Next we study the relationship between ADUs and phonemes in English. In these experiments, we use the existing phoneme alignments for TIMIT to determine the relationship between manually defined phonemes and automatically discovered units. We will use two different approaches to study this relationship: (1) a confusion matrix and (2) a mapping of ADUs to phonemes. We will show there is a close relationship between ADUs and phonemes that demonstrates the discovered units are linguistically meaningful.

Next we investigate the usability of the ADUs in an STD by query task (Hazen et al., 2009). The goal of STD by query is to retrieve utterances containing the words included in the query. We will show that we can obtain state of the art results for unsupervised algorithms. Finally we propose an algorithm to discover the lexicon given a parallel word transcription. We investigate two scenarios: (1) the word alignment is given and we only need to find the best mapping between word examples and ADUs (supervised lexicon discovery), and (2) the word alignment is not given and we want to learn the mapping along with the alignment (semi-supervised lexicon discovery). We will show our proposed algorithm can give similar results to other state of the art systems reported in the literature despite the fact that our system is much simpler and discovers the lexicon independently of the acoustic units. Other systems discover acoustic units and the lexicon jointly.

5.2.1 Speech Segmentation

Speech segmentation, defined as the process of finding boundaries between various acoustic

units such as words or phonemes, is one of the fundamental processes that almost all speech recognition algorithms perform implicitly or explicitly. Semi-supervised training of acoustic models using word transcriptions and a lexicon generates speech segmentation as a byproduct. However, in this chapter we are interested in unsupervised segmentation.

Most unsupervised algorithms for speech segmentation rely on changes in the acoustic data or spectrum (Ma et al., 2005; Bacchiani and Ostendorf, 1999; Paliwal, 1990; Wang et al., 2015). In these approaches, each segment is assumed to be somewhat different from adjacent segments but different segments are not related to each other. Lee and Glass (2012) proposed a nonparametric Bayesian approach for unsupervised segmentation of speech. A Dirichlet Process Mixture (DPM) model was used. In order to obtain phoneme-like segments, a 3-state HMM was used to model each segment. A Gibbs sampler was employed to estimate the segment's boundaries.

The goal of their research is similar to ours but instead of modeling each segment using an HMM we model the segments using mixtures of Gaussians (though often only one Gaussian is needed). We also implicitly learn a bigram language model from the data that allows us to learn relationships between segments (e.g. if the previous frame belongs to the segment with label S_{n-1}, what is the probability that current frame belongs to the segment with label S_n). Our approach is similar to a speaker diarization algorithm proposed by Fox et al. (2011) where an HDPHMM was used to segment a meeting into speaker-homogenous segments. We are also using HDPHMM and DHDPHMM, but instead of segmenting only one utterance we train the HDPHMM/DHDPHMM using different utterances from different speakers. We then train a speaker independent transducer that can map the new observations into a sequence of states. The end result is that the speech is segmented into acoustically-homogenous segments defined by HDPHMM/DHDPHMM states.

5.2.2 Acoustic Unit Discovery

Classical methods for acoustic unit discovery involve segmentation and clustering. The segmentation is typically implemented using a dynamic programming method that incorporates a

heuristic stopping criterion, while clustering is implemented using a heuristic agglomerative method (Bacchiani & Ostendorf, 1999; Wang et al., 2015).

Bacchiani & Ostendorf (1999) introduced a supervised approach that assumed word alignments are given. They segmented all examples of each word subject to a length constraint. All segments of each word were then grouped together to impose a pronunciation consistency constraint. A variant of the K-MEANS algorithm was used to cluster these segments into K groups. The number of groups, K, was set a priori. The algorithm then iterates through the above steps until convergence. Our approach is completely unsupervised and does not use transcriptions or any other form of data other than the speech signal data. We learn the number of units directly from the data using a nonparametric Bayesian model and therefore do not need to use expensive model comparison techniques.

Paliwal (1990) used a similar approach of segmentation followed by clustering to discover sub-word units. The segmentation criterion was based on locating segments of data that exhibited temporal stationarity. In our approach, we also segment the utterance into stationary parts since each segment is modeled with a GMM, but our algorithm performs the segmentation implicitly as a part of acoustic unit discovery. In Paliwal (1990), clustering is performed with a variant of the K-MEANS algorithm in which the number of units should be known or pre-determined.

Singh et al. (2002) proposed a probabilistic framework to jointly estimate the acoustic units and the lexicon. However, their model requires the existence of a transcription and therefore can be regarded as a semi-supervised approach. In contrast, our model can find the acoustic units in an unsupervised manner. The algorithm needs to initialize the lexicon first and then uses the lexicon to estimate the models. The lexicon and models are iteratively reestimated. The number of units in Singh's method must be specified a priori.

Varadarajan et al. (2008) adopted an HMM state splitting algorithm to learn a transducer that maps acoustic observations into acoustic units. Though the main goals of their work are similar to ours, the algorithms are completely different. They developed a speaker dependent transducer

while our model is speaker independent. They evaluated the performance of their system by measuring the relationship between automatically discovered units and phonemes. We also report a similar metric, however, instead of learning acceptors (Mohri et al., 2008) we trained a discrete HMM to map a stream of units to stream of phonemes.

Lee and Glass (2012) proposed a nonparametric Bayesian model based on DPM that jointly segments the data and discovers the acoustical units. Our approach, however, discovers more homogenous units. We model each unit with a mixture of Gaussians while they model each unit with a 3-state HMM. Our ADU transducer also learns the relationship between different units in the form of the probability transition between different units while they do not model these relationships. Though our overall goals are similar, these goals are achieved with different strategies. We will directly compare our system to theirs in the discussion that follows.

5.2.3 Lexicon Discovery

Most of the popular approaches to discovering a lexicon assume the existence of a word transcription (Bacchiani & Ostendorf, 1999; Singh et al., 2002; Lee, 2014). Some approaches also require additional information such as time alignments of word transcriptions. In this dissertation, we have explored both scenarios.

Bacchiani and Ostendorf (1999) proposed an algorithm to jointly discover the acoustic units and the lexicon. In their algorithm, they have assumed the alignment for words are given and they learn one pronunciation for all examples of a single word. In contrast to their approach, we don't need to know the time alignment between words and acoustic observations. However we study both cases to investigate how much performance will be lost due to not knowing the alignment.

Moreover, we don't restrict the number of pronunciation variants and let the data speak for itself. We also discover the lexicon and acoustic units in two successive steps. One could argue that discovering the lexicon and acoustic units in separate steps results in a suboptimal algorithm.

However, we show that despite this disadvantage our model can produce similar results as other algorithms discussed in this section that discover these jointly.

Paliwal (1990) also proposed several methods to discover the lexicon for isolated words. These methods learn multiple pronunciations per word but in their current form can't be used for continuous speech. Fukadai et al. (1996) proposed a similar model that also needs word alignments. Singh et al. (2002) proposed an approach to estimate the lexicon along with the acoustic units in a probabilistic framework. Their approach involves initializing the lexicon with a heuristic method and then iteratively discovering the lexicon and acoustic units. Our semi-supervised method also needs to be initialized with some approximate word alignments and then iteratively reestimates the lexicon and word alignments.

Finally, Lee (2014) proposed a model that discovers the lexicon by first learning a mapping between letters in a word and acoustic units and then generating pronunciations by connecting these mappings for each word. In our approach, we also use letters to initialize our semi-supervised algorithm. However, unlike (Lee, 2014), our algorithm learns the pronunciation directly from examples and is not strongly dependent on using letters. In other words, letters have been used only to initialize the algorithm and can be replaced with other appropriate heuristic approaches if the language was not alphabetic (e.g. logograms could be used for Chinese). We also investigate the effect of adding a G2P transducer to discover a new lexicon based on lexicons discovered directly from the data.

5.3 An Unsupervised ADU Transducer

The goal of speech segmentation is to map each acoustic observation into a segment and optionally label these segments. Let's assume we have N segments already labeled with L symbols. Our goal can be expressed as mapping a string of acoustic observations to a string of labels. In speech recognition problems, observations are vectors of real numbers (instead of

symbols in text processing) and segment labels can be replaced with a vector representation that is called a posteriorgram (Zhang & Glass, 2009). A posteriorgram is a probability vector representing the posterior probability of a set of predefined symbols. Instead of using a hard decision for mapping the input string to the output string, we can provide a probability vector which can be represented by a matrix of dimension LxM where L and M are the number of distinct segment labels and input string length respectively.

A transducer specifies a binary relationship for a pair of strings (Mohri et al., 2008). Two strings are related if there is a path in the transducer that maps one string to the other. A weighted transducer also assigns a weight for each pair of strings (Mohri et al., 2008). Based on this definition our problem is to find a transducer that maps a string of acoustic features onto a string of units. It should be noted that based on this definition any HMM can be considered to be a transducer. We chose the term transducer here to emphasize the operation of converting acoustic observations into acoustic units. The problem can further divided into two sub-problems: learning a transducer and decoding string of observations into string of units (or their equivalent posteriorgram representation).

Let's assume we already knew the acoustic units (e.g. phonemes) and have trained some models for each unit (e.g. HMMs). One way to build a transducer is to connect all these HMMs together to build an ergodic network (we can also use language model information to connect phoneme HMMs together). Therefore the final transducer can be some form of ergodic HMM. However, we don't have the units and we even don't know how many units there are in the data.

In Chapters 2 and 3, we introduced two nonparametric Bayesian HMM models. HDPHMM, described in Chapter 2, is an ergodic model that can learn the number of states from the observations. DHDPHMM, described in Chapter 3, can learn different structures including ergodic structures and also can share mixture components among different states. Both of these models are good candidates to train a transducer. Since in our implementation of both models we

have used a similar interface we can easily use both and compare their performance. Next we discuss how learning and decoding problems have been solved for our transducer.

5.3.1 Learning the Transducer

We use an HDPHMM or DHDPHMM to train the transducer. The difference from the supervised phoneme training experiments discussed in Chapter 3 and the semi-supervised phoneme training experiments discussed in Chapter 4 is that we train only one HMM for all utterances in a completely unsupervised fashion. Unlike Lee (2014), we don't utilize a speech/non-speech classifier and model everything including silence with one transducer. For read speech, this does not present any problems. However, for other domains such as conversational speech, it might be a problem, and in that case we can employ a speech/non-speech classifier as well. Training is executed by sequentially presenting utterances to the HDPHMM/DHDPHMM inference algorithm and iterating using Gibbs sampling.

For our transducer, state labels (or their posteriorgrams) are the output string. Since each state is modeled by a Gaussian mixture, the segments defined by this transducer are stationary and the discovered units are sub-phonetic. However, it should be noted that this limitation can be overcome by replacing each state (e.g. mixture model) with an HMM which transforms the model into a hierarchical HMM (Fine et al., 1998). The resulting model can model dynamic segments.

5.3.2 Decoding Observations

Given a transducer and a string of observations the goal of the decoder is to find the most likely path through states of the transducer that implicitly maps the input string to the output string. This objective can be written as:

$$\arg\max_{s_1 s_2 \ldots s_M} P(s_1 s_2 \ldots s_M \mid o_1 o_2 \ldots o_N), \tag{5.1}$$

where s_1, s_2, ..., s_M represent state labels and o_1, o_2, ..., o_N represent observations. Alternatively we can also estimate the posteriorgram of the state sequence. If the goal is only to estimate the best sequence then the Viterbi algorithm (Viterbi, 1967) can be employed. If estimating the posterior is desired, the FB algorithm is preferred. Alternatively we can use other techniques such as Gibbs sampling, but these will be more computationally expensive.

The resulting transducer is the engine used to convert new acoustic observations into acoustic units. We will explore the properties of this transducer in Section 5.5 for several applications. The transducer is also the main component of lexicon learning algorithms that are discussed next.

5.4 Discovering The Lexicon

In this section, we present two algorithms to learn the lexicon from the output of our ADU transducer discussed in the previous section. It should be noted that acoustic units are discovered in a completely unsupervised manner while to discover the lexicon we need a word-level transcription of the acoustic data. We investigate three algorithms to discover the lexicon: (1) a supervised algorithm that needs alignments of words and acoustic units, (2) a semi-supervised algorithm that does not need the word alignments, and (3) a G2P algorithm (Novak et al., 2012) based on a weighted finite state transducer (Mohri et al., 2008). This latter approach is not the focus of this dissertation. The basic idea is to use the lexicon obtained by one of the direct algorithms and find a mapping between the sequence of letters and ADUs. Details of this algorithm can be found in (Novak et al., 2012). One of its advantages is that is can handle words that have not been previously seen in the training data.

5.4.1 Direct Supervised Learning

If the word alignment exists or is estimated reliably (e.g. from forced alignments generated by a high quality CDHMM system), we can extract the alignment between each word and the

stream of acoustic units produced by the ADU transducer. However, there might be many examples for each word and we need to select a handful that represents each word more accurately. There are many ways to find these representative examples. For example, we can cluster the examples and then select the centroids. In this section, we propose an algorithm that selects at most *M* examples among all instances of a given word that have the average minimum edit distance (Navarro, 2001) from other examples. The edit distance is computed using DTW. Posteriorgrams of the states are used to represent each example.

The algorithm is as follows:

1. Generate the posteriorgram representation for all utterances in the dataset using an ADU transducer.
2. Use the aligned transcription to extract all examples of each word.
3. Compute the DTW alignment between each two examples *X* and *Y* (Müller, 2007):

$$\begin{aligned} DTW(X,Y) &\triangleq C_{p^*}(X,Y) \\ &= \min\left\{C_p(X,Y) \mid p \text{ is the warping path between } X \text{ and } Y\right\}, \end{aligned} \tag{5.2}$$

   where *X* and *Y* are the two examples, *p* is the warping path that aligns *X* and *Y* and $C_p(X,Y)$ is defined as:

$$C_p(X,Y) = \sum_{i=1}^{L} c(x_{nl}, y_{ml}). \tag{5.3}$$

4. For all *k*, accumulate the distance between the k^{th} example and all other examples in the data set.
5. Select the *M* examples with a minimum average distance as representatives for that word.
6. Convert the posteriorgram of *M* examples into state labels, remove repetitions and retain the remaining *M* examples (e.g. the number of final examples for each word is less than or equal to *M*).

Note that in this formulation $c(x_{nl}, y_{ml})$ is an element of the cost matrix between X and Y (row n and column m). For our problem, we define the cost between two posteriorgram vectors as a dot product between them (Zhang & Glass, 2009):

$$c(x, y) = -\log(x \bullet y). \tag{5.4}$$

At the end, this algorithm will find at most M pronunciations for each word. This algorithm selects instances with the least average edit distance from the other training examples.

5.4.2 Direct Semi-Supervised Learning

In this section, we propose an algorithm similar to the supervised algorithm discussed in the previous section that iteratively realigns the transcriptions with the output stream of the ADU transducer. The algorithm needs to be initialized using a heuristic approach, but, as will be shown in Section 5.5.5, this initialization does not need to be accurate and can be easily generated with available resources (e.g. word transcriptions and acoustic data).

The algorithm is as follows:

1. Generate the posteriorgram representation for all utterances in the dataset using the ADU transducer.
2. Generate an approximate alignment between the words and the output stream of the ADU transducer.
3. Use the aligned transcription to extract all examples of each word.
4. Use DTW to compute the edit distance between every pair of examples using (5.2).
5. For all k, accumulate the distance between the k^{th} example and all other examples in the data set.
6. Select the M examples with minimum average distance as representatives for that word.
7. Convert the posteriorgram of the M examples into state labels, remove repetitions and retain the remaining of M examples (e.g. the number of final examples for each word is less than or equal to M).

8. Use the lexicon and associated acoustic units generated in Step 7 to build a speech recognizer.
9. Use the speech recognizer built in Step 8 to force align the transcriptions with the acoustic units.
10. Use the aligned transcriptions to extract all examples of each word.
11. If convergence is not achieved and the number of iterations is less than a specified threshold, go back to Step 4. The convergence criterion can be the WER computed on a small development set.

In Section 5.5.5, we show this simple semi-supervised algorithm will converge to the supervised algorithm in two to three iterations.

5.4.3 Discovering the Lexicon Using G2P

After discovering a lexicon using the direct supervised or semi-supervised approaches, we can use any G2P (Novak et al., 2012) algorithm to learn a mapping between the sequence of letters and the ADUs. In this work, we only used a G2P algorithm based on a weighted finite state transducer (Mohri et al., 2008). The input to the algorithm is a lexicon. G2P algorithm learns a transducer that can map any sequence of letters into sequence of ADUs. We have not optimized this algorithm and our goal was only to be able to conduct some of experiments on the TIMIT Corpus. Such an approach was required because many words in test set in TIMIT do not appear in the training data. Also, it is expected that using G2P reduces the performance. However, for TIMIT we will show that it actually improves the overall performance of the system on the open test set. In Section 5.5.5 we will discuss the characteristics of the TIMIT Corpus that enable this counterintuitive result.

5.5 Experiments

In Chapters 2 through 4, we have introduced HDPHMM and DHDPHMM and their applications in sub-word modeling. In this section, we provide some experiments that give us more insight into the segmentation properties of HDPHMM and DHDPHMM. By comparing discovered units to manually transcribed phonemes, we show that units discovered by the ADU transducer are linguistically meaningful. More specifically we show that we can predict the stream of the phonemes by only observing the stream of ADUs. Further we show that most prediction errors occur between similar phonemes (e.g. */r/* vs. */er/*) that might actually expose an underlying acoustic or pronunciation mismatch.

Next, we use the ADU transducer in unsupervised spoken term detection (STD) by query task. For this task we will show that our automatically learned units can achieve state of the art performance among unsupervised algorithms on TIMIT. We also present several experiments in which we generate the lexicon from acoustic data and parallel transcriptions using an ADU transducer. We use both TIMIT and the DARPA Resource Management Corpus (RM) (Price et al., 1988) for these experiments. We show that an ADU transducer trained on TIMIT can be used to map acoustic observations from RM into ADUs and learn a lexicon that produces a WER comparable to other automatic lexicon/unit learning algorithms.

5.5.1 Evaluation Methods

Unsupervised Segmentation

To obtain a quantitative measure of segmentation quality we compare the boundaries of discovered segments to the boundaries of manually segmented phonemes. The number of co-occurrences of segments boundaries and phoneme boundaries is called recall. The percent of declared boundaries that coincide with phoneme boundaries is called precision. A single numeric score is referred to as the F-score and defined as (Rijsbergen, 2004):

$$\text{F-score} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} . \tag{5.5}$$

We also define a similarity score (Harati et al., 2013), *S*, that measures the consistency of the segments:

$$S(s_1, s_2) = \begin{cases} s_1 = \dfrac{1}{MN} \sum\limits_{\substack{I=\{i\} \\ M=|I|}} \sum\limits_{\substack{J=\{j\}=\{j:class(j)=class(i), i \neq j\} \\ N=|J|}} \left| corr(x_i, x_j) \right|, \\ s_2 = 1 - \dfrac{1}{MN} \sum\limits_{\substack{I=\{i\} \\ M=|I|}} \sum\limits_{\substack{J=\{j\}=\{j:class(j) \neq class(i)\} \\ N=|J|}} \left| corr(x_i, x_j) \right|. \end{cases} \tag{5.6}$$

This similarity score has two main components: (1) s_1 is the in-class similarity score and is defined as the average of the correlation between different instances of segments with identical labels (e.g. two segments are similar if they are correlated); (2) s_2 is the out-of-class dissimilarity score. For example, for class *i*, the number of in-class instances is *M*. To compute s_1, we need to average over all instances of class *i* and therefore *N* is equal to *M-1*. To calculate s_2 we correlate examples in class *i* to all other data points (with classes not equal to *i*) and therefore *N* is equal to the total number of instances minus *M*. The quality of segmentation is higher when both numbers are closer to one. It should be noted that the similarity score functions much like a likelihood score; e.g. it *increases* monotonically with an increase in the number of classes. Therefore, for a meaningful comparison, the number of classes being compared for two algorithms must be the same (defined as the number of segments with same identity) or equivalently the average length of segments produced by the two algorithms should be comparable.

Relationship of ADUs to Phonemes

To compare the learned ADUs with manually labeled phonemes, we follow two approaches. First, we use a confusion matrix to visualize the relationship between phonemes from the reference transcriptions and the discovered ADUs. Second, we train discrete HMMs that map the ADUs into phonemes. We run both recognition and classification tasks on these discrete HMMs.

The definitions for classification and recognition errors are the same as the ones discussed in Chapters 3 and 4.

Spoken Term Detection by Query

TIMIT has been used to evaluate the performance of an ADU transducer based system for spoken term detection by query task. The queries have been extracted from training subset of the data and have been executed on the test section of the data. Two performance metrics have been employed to measure the performance of the system: (1) average precision of the top N hits, referred to as *P@N* (Hazen et al., 2009), and (2) average equal error rate (Hazen et al., 2009).

For each keyword we first set *N* to the number of its occurrences in the evaluation data (e.g. for example if word "age" occurs *8* times in the evaluation set then *N=8*) and then compute a list of scores for each utterance. To make the results compatible with other reported results we also assume at most one hit per utterance. We sort the scores and select the top *N*. For keyword *i*, *P@N* is computed as:

$$p@N = \frac{H_i}{N_i}, \tag{5.7}$$

where H_i is the number of hits for top N_i scores. The final *P@N* is reported as the average for all keywords.

The average equal error rate (EER) is the point on detection error tradeoff (DET) curve where the false acceptance error rate is equal to false rejection error rate. The reported EER is the average of EER for all keywords.

Lexicon Discovery

To compare the quality of the lexicon discovered using ADUs and to evaluate the algorithms presented in this chapter, we use speech recognition experiments. We compute the word error rate (WER) for both the baseline system that is trained based on a standard lexicon and systems trained using an automatically discovered lexicon. We use TIMIT and RM for experimentation. It is important to emphasis, however, that the ADU transducer used for both datasets is only trained

on the training subset of TIMIT (*3,637* utterances) in a completely unsupervised manner. We conduct several experiments for closed and open-loop systems and investigate the properties of the discovered units.

The RM Corpus (Price et al., 1988) is a *1,000*-word task that can be used for speaker independent, speaker adaption and speaker dependent experiments. The speaker independent subset of RM includes *3,990* training utterances and *1,200* evaluation utterances. The corpus includes a word-pair grammar with a perplexity of *60* (Murveit, 1991) that is used for all of the experiments in this dissertation. The front-end used for feature extraction is the same as the one used for TIMIT and was briefly discussed in Chapter 3.

5.5.2 Unsupervised Segmentation

To evaluate the nonparametric transducer, we used TIMIT because of the existence of highly accurate manual segmentations. A typical segmentation is shown in Figure 5-1 along with time-aligned phoneme labels. Segments are shown with a rectangle. The height of each rectangle, which is unique for each label, shows the corresponding label. The edges of the ADU-derived segments are seen to roughly coincide with phoneme boundaries. However, a single phoneme is often divided into one to three ADU units because ADUs are stationary while phonemes are not.

A comparison of our proposed method to other state of the art systems is shown in Table 5-1. The first row represents a system that performs unsupervised segmentation with no prior information about the number of segments for each utterance. The second row represents a system that implements a semi-supervised approach. The third row represents results of an alternate nonparametric Bayesian algorithm developed by Lee and Glass (2012). The fourth row represents the results from the ADU transducer.

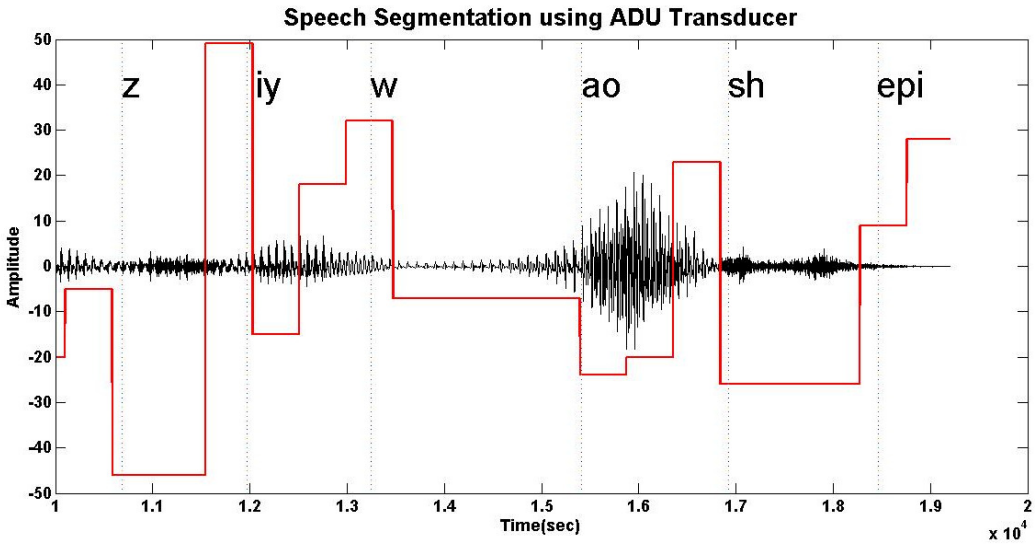


Figure 5-1. Segmentation of utterance SA1 from TIMIT using an ADU transducer is shown. Discovered units are represented by the height of the red rectangle.

Our model performs particularly well on recall, which implies that it is finding boundaries that better match the reference phoneme boundaries. The improvement in recall is over 11% even though our approach, unlike (Qiao et al., 2008) is completely unsupervised. Our precision is lower, however, which means there are slightly more false alarms. This is not unexpected since we are discovering sub-phonetic units. Using the similarity score in (5.6), we obtain a similarity score of *(0.44, 0.72)* for the manual segmentations while an ADU transducer that is constrained to produce a similar number of unique segments obtained a score of *(0.83,0.72*) (via setting the maximum number of states in HDPHMM and varying the associated hyperparameters). This similarity score is much better than the similarity score for the manual phoneme segmentations. This can be attributed to the fact that ADU segments are presenting more stationary segments versus phonemes and therefore are more consistent. Many phonemes represent sounds that have

Table 5-1. A comparison of segmentation algorithms is shown.

Algorithm	Recall	Prec.	F-score
Dusan & Rabiner (2006)	75.20	66.80	70.80
Qiao et al. (2008)	77.50	76.30	76.90
Lee & Glass (2012)	76.20	76.40	76.30
ADU Transducer	**86.51**	68.50	76.62

dynamic spectral properties therefore self-similarity for such phonemes is not as great as that for the ADUs that are inherently stationary.

5.5.3 Investigating the Relationship Between ADUs and Phonemes

It is important to explore the relationship between the ADUs and phonemes because we need to determine if the ADUs are linguistically meaningful. The first experiment involves aligning manually transcribed phonemes with ADUs. First, each utterance is passed through the transducer to generate the sequence of ADUs. Then these ADUs are aligned with manual transcriptions using timing information contained in the transcription. Finally a confusion matrix is calculated.

This confusion matrix between *48* English phonemes and *251* ADU units is shown in Figure 5-2. A general correlation between ADUs and phonemes can be observed because the diagonal region of the matrix is heavily populated. However the mapping is not consistently one to one. Some of the ADUs align with multiple phonemes. However, these phonemes are generally similar phonemes. For example, we can see ADUs that are aligned with *"sil"* (silence) can also

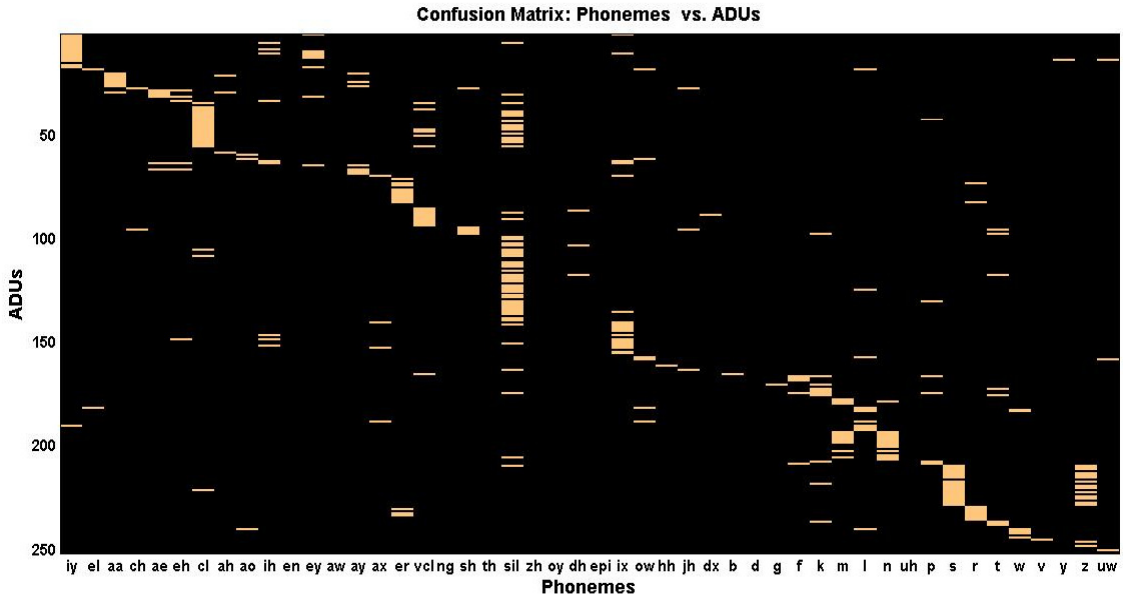


Figure 5-2. A confusion matrix that shows the relationship between the discovered ADUs and the manually transcribed phonemes is shown. For clarity only units that occurred more than 200 times are displayed.

be aligned with *"vcl"* and *"cl"* models (both *"vcl"* and *"cl"* are special types of silence). ADUs aligned with *"z"* can also be aligned with *"s"*. This is not surprising because *"z"* and *"s"* are similar acoustically and therefore confusable.

Next we train discrete HMMs that map streams of ADUs into streams of phonemes. The idea is to learn inter-ADU relationships that might also be related to their identity. Training discrete HMMs is similar to training continuous HMMs. The primary differences are (1) we use discrete emission probability models instead of Gaussian mixtures, and (2) we replace the acoustic observations (e.g. MFCC features) with ADU units (discrete symbols). In this section we conduct experiments on classification and recognition using these discrete HMMs. Our baseline here is a continuous HMM system with a single Gaussian per state trained using acoustic observations.

Classification Experiments

Table 5-2 shows the results of classification experiments for three systems: the baseline continuous HMM, HDPHMM ADUs using discrete HMMs and DHDPHMM ADUs using discrete HMMs. We can see that the HDPHMM ADU system performs slightly better than the baseline HMM system. It should be noted we used a simple continuous HMM as the baseline. This system has a complexity that is comparable to the simple discrete HMM used to learn the relationship between ADUs and phonemes. In this experiment, the goal is to show that a relationship exists between ADUs and phonemes and therefore we don't need to use a complex system such as a CD HMM. It can also be seen that HDPHMM generated ADUs are slightly better than DHDPHMM generated ADUs.

Recognition Experiments

Table 5-3 shows the result of a phoneme recognition experiment using the same discrete

Table 5-2. A comparison of phoneme classification error rates using ADUs is shown.

System	% Error
Baseline Continuous HMM	37.41
HDPHMM ADUs	35.85
DHDPHMM ADUs	37.52

Table 5-3. A comparison of phoneme recognition error rates on TIMIT using ADU streams is shown.

System	% Correct	% Error
Baseline Continuous HMM	59.71%	42.43%
HDPHMM ADUs	61.42%	43.70%
DHDPHMM ADUs	59.75%	45.53%

HMMs. We start from a stream of ADUs and find the corresponding stream of phonemes. In the recognition task we also estimate the boundaries between different phonemes. The result of this table is also consistent with Table 5-2 that represented a classification task and shows discrete HMMs trained on ADUs can match a similar HMM trained on acoustic observations. These results are fairly unique. The closest reported results were (Varadarajan et al., 2008) in which speaker dependent relationships between discovered acoustic units and phonemes are investigated on a Japanese language task. The authors used an acceptor transducer (Mohri et al., 2008) instead of a discrete HMM. In our work, we have focused on a speaker independent application instead of a speaker dependent application.

Table 5-4 shows a comparison between reference and recognized sequence of phonemes using ADU streams. Note that most of the confusions occur between similar phonemes. For example, phoneme pairs like */aw/-/ah/*, */ng/-/en/*, */d/-/t/* are among the most confused pairs.

5.5.4 Spoken Term Detection by Query

In this section we study the result of a simple unsupervised STD by query system based on

Table 5-4. A comparison of an ADU-based phoneme recognizer to the manual phoneme transcriptions is shown. Yellow shading indicates a recognition error.

1	REF	sil	hh	aw	sil	g	uh	sil	D	ix	sh	er	ix	en	sil
	HYP	sil	hh	ah	sil	g	uh	sil	D	ey	sh	er	ix	ng	sil
2	REF	sil	b	r	aw	en	ay	z	ay	sil	b	r	aw	m	ax
	HYP	sil	p	r	ow	en	ay	z	ay	sil	b	r	aa	m	ax
3	REF	sil	en	eh	v	er	hh	ae	sil	p	ix	er	ix	en	m
	HYP	sil	ix	z	ix	er	hh	ae	sil	d	ix	r	ix	en	–
4	REF	sil	ax	f	ix	sh	el	z	sil	m	iy	sil	t	aa	en
	HYP	sil	aa	f	ix	sh	el	–	sil	m	ey	sil	d	ah	ng

our ADU transducer discussed earlier. An STD system can be built based on a complex state of the art speech recognizer and work either by acoustic or text queries. However, building such a system requires all the resources needed to build a state of the art speech recognizer including a lexicon, a language model and plenty of transcribed data. For low resourced languages, this is not feasible. Our unsupervised STD by query algorithm is as follows:

1. Convert the target audio data using the ADU transducer into posteriorgrams.
2. For each query generate its posteriorgram representation using the transducer.
3. Use a subsequence DTW algorithm (Müller, 2007) to obtain a score for each utterance. To improve the robustness generate one score per utterance by averaging the scores for all examples of the given query in the training subset.
4. Compare the final score for each utterance with a threshold and return it if the score is greater than the threshold.

Table 5-5 shows the queries used in this section to assess the quality of our ADU transducer. Since some words are similar (year vs. years), we do not count confusions between two words that share the same stem. A comparison of the average *P@N* and *EER* is reported in Table 5-6 for TIMIT. The first row shows a system that utilizes a GMM to directly decode the posteriorgrams of the feature frames (Zhang and Glass, 2009). The second row shows the result of an algorithm based on a Deep Boltzmann Machine (DBM) (Zhang et al., 2012). The third row contains the

Table 5-5. A list of query terms used for the STD by query task is shown.

Query	No. Training Examples	No. Test Examples
age	3	8
warm	10	5
year	11	5
problem	22	13
artists	7	6
money	19	9
organizations	7	6
development	9	8
surface	3	8
children	18	10

Table 5-6. A comparison of unsupervised approaches to STD by query is shown.

System	P@N	EER
GMM (Zhang & Glass, 2009)	52.50%	16.40%
DBM (Zhang et al., 2012)	51.10%	14.70%
Nonparametric Bayesian (Lee & Glass, 2012)	63.00%	16.90%
Semi-Supervised English Triphones (Lee & Glass, 2012)	75.90%	11.70%
DHDPHMM ADU	56.21%	14.33%
HDPHMM ADU	61.20%	13.95%
Combined HDPHMM/DHDPHMM ADU	64.91%	11.83%

results for nonparametric Bayesian approaches by Lee and Glass (2014). The fourth row shows the result of a tied state triphone system (as mentioned above this system is trained using transcriptions and a lexicon). Rows five through seven contain the results of our ADU-based unsupervised systems.

As we can see, the combined system (row 7) performs better than every other unsupervised system in this table. Even if we ignore results of combined system, the HDPHMM ADU system in row 6 delivers the best overall *EER* for the unsupervised algorithms and the second best *P@N*. In fact, the *EER* of our combined system is getting very close to the performance of a much more complex triphone-based system that utilizes significantly more resources than the other algorithms in this table.

It is also evident that the HDPHMM transducer works better than the DHDPHMM transducer. This is consistent with the results shown in Section 5.5.3. One reason for this superiority is the fact that for HDPHMM each state is modeled with a single Gaussian and this distribution is unique to that state, while for DHDPHMM all states share a pool of Gaussians. Each state can have more than one Gaussian associated with it, and this can make some states more confusable. Table 5-7 shows some of the typical error pairs. It can be seen that for most of the cases we have partial acoustic similarity between the search query and the retrieved word (e.g. mess**age** and **age**).

Table 5-7 Error pairs for STD system.

Query	Discovered
age	mess**age**
development	fulfil**lment**
year	d**eer**
year	h**ere**
year	b**ehavior**
surface	**severe**

5.5.5 Lexicon Discovery

In this section, we present the results of both supervised and semi-supervised lexicon discovery algorithms. We refer to both of these approaches as direct algorithms since we learn the pronunciation directly from ADU transducer output. We also provide the results for a G2P-based approach that is trained from the lexicons discovered using the direct algorithms. The advantage of this approach is that it produces pronunciations for words that do not exist in the training data. We have provided results for two datasets: TIMIT and RM. The ADU transducer used in this section is only trained on the training subset of TIMIT and then used to discover the lexicon from TIMIT and RM. This allows us to evaluate the generalization capabilities of the ADU transducer.

For TIMIT there is not a widely used baseline system for a word recognition task. It is also a challenging dataset for our task since many words occur only once in the data. This makes it particularly difficult for our direct lexicon discovery approach since our estimation method depends on multiple instances of a word. When there are only a few examples estimation can be unreliable. For TIMIT, we have conducted two types of experiments:

- Closed Loop: For these experiments, we discovered the lexicon from both training and test subsets. This is a sanity check experiment and is used to prove that ADUs are meaningful units.
- Open Loop: For these experiments, we have only used the training set for discovery. Since the direct lexicon algorithm only learns the pronunciations for words it has seen during

training, we evaluate on a subset of test section named test-94 – 94 utterances selected such that every word in each utterance exists in the training set.

Note that for the G2P-based lexicon algorithm, we will also provide results for the full test set since that approach is designed to deal with unseen words.

RM is more popular for word recognition experiments. The corpus has a standard language model that makes it easier to compare to other reported results. There are approximately *1,000* unique words in RM, compared to *5,850* for TIMIT. Moreover, most words in the test set also exist in the training set which makes it a good choice for direct lexicon discovery algorithms. For RM we only have conducted open-loop experiments.

Closed-Loop Experiments on TIMIT

The first set of experiments is designed to prove that ADUs can consistently represent words. It is also designed as a sanity check for the algorithms. This set of experiments is named "closed-loop" because the lexicon is discovered from both training and testing subsets of TIMIT. However, it should be noted that other learning procedures are performed only on the training subset. The full set of utterances from TIMIT that we used in this study contains *5,850* words but only *1,353* of them occur more than *3* times in the training subset. This makes it rather a challenging dataset for discovering pronunciations since the lexicon can easily become over-trained.

Supervised Lexicon Discovery

Table 5-8 shows the results of supervised training of the lexicon for TIMIT for the closed-loop condition with a different maximum number of lexicon entries per word. The second column shows the total number pronunciations in the lexicon, which depends on *M* for the last seven entries in the table. The algorithm's performance improves when the number of pronunciations in the lexicon increases. This is the expected effect when training under closed-loop conditions because during recognition the correct pronunciation exists in the lexicon. We can see the

Table 5-8. The results of closed loop training of a lexicon for TIMIT are shown.

System	Lexicon Size	Perplexity	WER (%)
Reference Lexicon	5,850	80	24.79
Reference Lexicon	5,850	16	5.66
Supervised ADU, M=1	5,850	80	32.75
Supervised ADU, M=3	12,933	80	22.20
Supervised ADU, M=5	16,474	80	20.19
Supervised ADU, M=10	18,879	80	19.15
Supervised ADU, M=10	18,879	16	5.20
Supervised ADU, M=30	25,187	80	15.03
Supervised ADU, M=30	25,187	16	4.41

performance of the best system (row 8 in Table 5-8) is much better than a comparable CI phoneme-based system (row 1 in Table 5-8).

This experiment demonstrates that ADUs are consistent. If we can learn a proper lexicon then ADUs can potentially even outperform the phoneme-based system. It is also interesting to note that the difference in the performance fades as we use a language model with lower perplexity. This is an important detail that must be considered when comparing lexicon discovery algorithms – a fair comparison requires comparing language models with similar perplexities because it has been shown that log(WER) is linearly correlated to log(perplexity) (Klakow & Peters, 2002).

Semi-supervised Lexicon Discovery

For semi-supervised lexicon discovery, we have first trained a system using letters and then used the trained model to obtain the initial segmentations. We have used both letters and letters in context (tri-letters). The latter (e.g. *"d-o+g"*) are the counterparts of triphones. Table 5-9 presents the results for a series of experiments using various combinations of these parameters. Our semi-supervised algorithm gives results comparable to the supervised results (Table 5-8) with only a few iterations. We can see that tri-letter initialization works reasonably well and its results are very close to the supervised training algorithm. This experiment proves that we can learn the lexicon without knowing the word alignment by using a simple initialization strategy and

Table 5-9. The results for semi-supervised training of a lexicon on TIMIT are shown.

Experiment	Lexicon Size	Iteration	WER (%)
Letter Initialized	12,933	1	25.79
Letter Initialized	12,933	2	23.86
Letter Initialized	12,933	3	24.68
Tri-letter Initialized	25,187	1	16.12
Tri-letter Initialized	25,187	2	15.44

iterating through the discovery process. The last entry in this table represents performance very close to the performance of supervised training (row 8 in Table 5-8).

Open-loop Experiments on TIMIT

Closed-loop experiments proved that ADU units could be used to discover a lexicon, but we used all available instances of a word in both the training and testing subsets. As stated before, TIMIT is particularly challenging because many words only occur a few times in the corpus. Figure 5-3 shows a histogram of the number of times each word occurs in TIMIT. The median number of times a word occurs is *1*. We have limited the graph to frequencies less than *100* since there are only few words that occur more than *100* times.

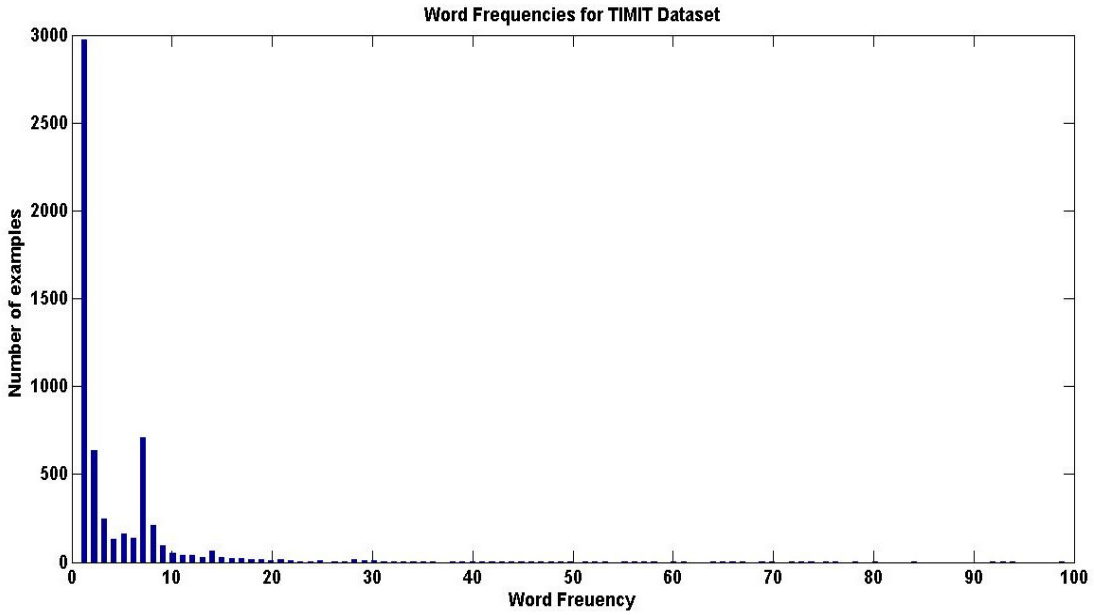


Figure 5-3. A histogram of the number of word tokens in TIMIT is shown.

As a result of this sparsity, the estimation of word pronunciations from the data is not reliable. Similarly, a closed-loop experiment can produce surprisingly good results because the discovered lexicon would be over-fit. Table 5-10 shows the result of open-loop learning of the lexicon for TIMIT dataset. For direct discovery algorithms we need to see one example of each word to learn the pronunciation. Therefore we have used the test-94 subset.

Last two rows of this table show the results for a lexicon discovered using the G2P algorithm (Novak et al., 2012) based on the directly discovered lexicon. It is apparent that the WER decreases when using the G2P algorithm. This is another example of the sparsity problem that we have seen in Figure 5-3. By learning a mapping between a sequence of letters and ADUs using the G2P algorithm, the estimated variance for each word reduces, and as a result, the WER decreases. In the next section, we use another corpus with a significantly different word frequency profile to show the problems we have encountered with TIMIT are simply a result of the sparse nature of the data.

Table 5-11 provides some examples of the errors between the reference and recognized utterances. Many of the errors occur between acoustically similar words, which is not unexpected given the sparse nature of the data.

Open-loop Experiments on RM

The RM Corpus, as opposed to TIMIT, has been used more widely in word recognition experiments (Young & Woodland, 1994) and is a better benchmark for the lexicon discovery task. Good comparative benchmarks exist (Bacchiani & Ostendorf, 1999; Singh et al., 2002).

Table 5-10. Results for open-loop training on TIMIT are shown.

Experiment	Test Subset	WER (%)
Baseline	full	24.79
Baseline	test-94	24.04
Direct Learning	test-94	42.77
G2P	test-94	37.46
G2P	full	37.03

Table 5-11. Examples of common substitution errors are shown for open-loop experiments on TIMIT.

Reference	Recognized
proceeding too slowly	proceeding curiously
be illuminating	be illuminated
my ears	my years
either a magnetic	a paramagnetic
buy these shoes than	by if you can
farm is	flow rose
the poor	the floor

Unlike TIMIT, in RM most words have several examples in the training set and as a result discovery of the lexicon is more feasible. Figure 5-4 shows a histogram of the number of times each word occurs in RM. Only words with counts less than *100* are shown. The median word frequency is *11* (compared to *1* for TIMIT) and the mean word frequency is *35* (compared to *20* for TIMIT). Therefore we expect this corpus to be a better dataset for evaluating our ability to automatically discover lexicons.

For the RM experiments we have used the same ADU transducer that was trained on the training subset of TIMIT. All utterances were first passed through the ADU transducer to obtain their posteriorgram representation. Then the lexicon was generated using the direct lexicon discovery algorithm discussed in Section 5.4. We have also generated a lexicon using G2P algorithm.

Table 5-12 shows the results obtained for RM. We can see that a CI system trained using an automatically discovered lexicon and *1* mixture component per state works better than a similar system using the reference lexicon (relatively *21%*). However, the performance is slightly worse when using more mixture components (*9.17%* vs. *11.61%*). This observation can be explained by considering the fact that ADUs are stationary units (corresponding to single Gaussian distributions). Once we increase the complexity of the system the improvement in their performance would be less than the improvement for more dynamic units such as phonemes (e.g. phonemes are less homogenous). Further, there are more ADUs than phonemes (in this case we

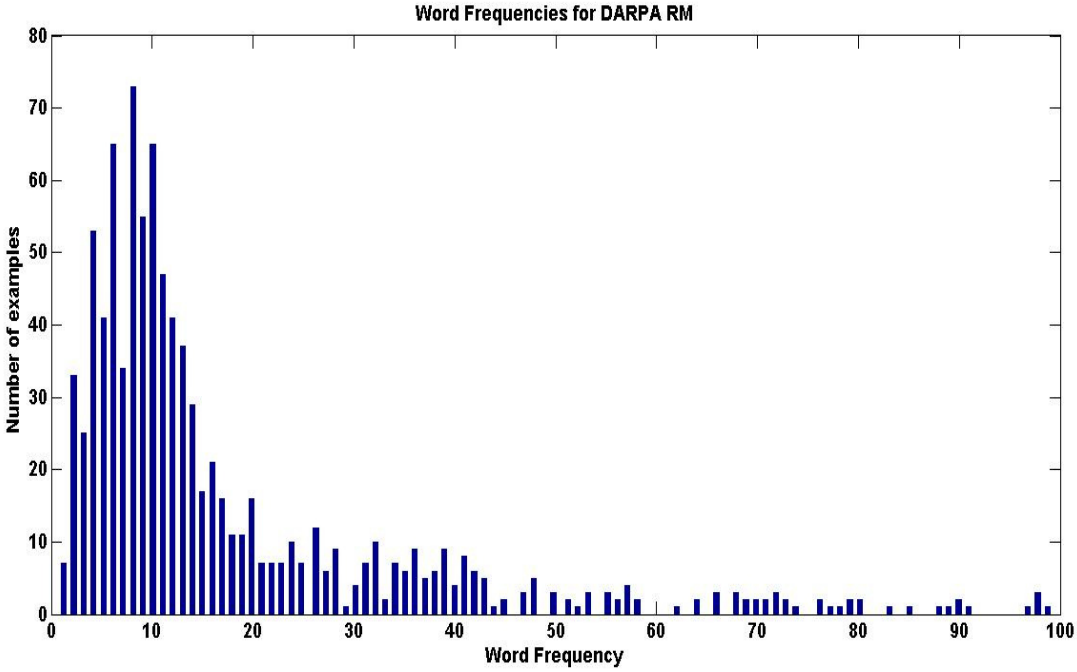


Figure 5-4. A histogram of the number of word tokens in RM is shown.

have *251* ADUs while *39* phonemes) which means we would have less data per ADU to train models. The number of parameters to be estimated increases significantly when the number of mixture components per state increases from *1* to *16*, so we expect these models to be more poorly estimated since the amount of data is fixed.

We can also see that the G2P-based system is much worse than a directly discovered lexicon. This is in contrast to the results reported for TIMIT and is a consequence of the different characteristics of the two corpora. The lexicon discovered for RM is estimated more reliably and adding a G2P reduces the performance while in the case of TIMIT the lexicon was estimated unreliably and G2P reduces the variance of the estimate.

We have also presented the results for CD-trained systems. For the baseline system we have used a phonetic decision tree (e.g. based on linguistic knowledge) to generate tied states. For the ADU-based systems we don't have linguistic knowledge (e.g. similar ADU units can be grouped). In principle it is possible to use a data-driven approach to obtain an approximation of such knowledge (e.g. clustering ADUs) but in this research we chose a simple approach based on

Table 5-12. Results for the open-looped trained lexicon are shown for RM.

Experiment	Context Modeling	Mixture No.	WER (%)
CI Baseline	CI	1	24.66
CI Baseline	CI	16	9.17
CD Baseline	CD	1	10.64
CD Baseline	CD	16	4.84
Direct algorithm	CI	1	20.34
Direct algorithm	CI	16	11.61
direct + G2P	CI	16	20.33
CD direct algorithm	CD	1	15.81
CD direct algorithm	CD	16	9.81

singleton questions for the decision tree. Instead of having questions that are categorical in nature (e.g. is the left phoneme/unit a stop?) we used each ADU as one single group with only one member and have questions such as "is the left unit *u1*?"

The result is less powerful than a tree trained by all possible questions. We can see from Table 5-12 that the gain for the CD ADU system (compare the last two rows to rows 3 and 4) is less than a CD system based on the reference lexicon and phonemes. Part of this is due to the singleton questions and part is a result of the increased number of ADUs compared to phonemes. Nevertheless, the results in Table 5-12 shows unsupervised ADUs and a semi-supervised lexicon trained based on these ADUs provides results comparable to the reference lexicon. In the next section we will compare these results with some other published results for the lexicon discovery problem.

Table 5-13 shows several examples from the lexicon discovered from the ADU transducer output. We can see the discovered pronunciations are generally consistent. For example, words "*Between*", "*Bring*" and "*Been*" start with "*u69 u13 u126 u6*", "*u69 u13 u126 u6*" and "*u66 u69 u13 u126 u6*" respectively. As we can see the first part of these words, which represent the phoneme */b/*, are almost identical with the exception of "*u66*". We can also see the same words end in a similar sequence of ADUs. This makes sense since the phonemes at the ends of the

Table 5-13. Examples of learned ADU pronunciations are shown.

Word	Phoneme Pronunciation	ADU Pronunciation
Between	b ax t w iy n	u69 u13 u126 u6 u44 u191 u49 u14 u182 u42 u68 u200 u202 u224 u130 u225 u242 u208 u34 u66 u138 u107 u210
Bring	b eh r ih ng	u69 u13 u126 u6 u213 u197 u22 u34 u138 u107 u210
Been	b ih n	u66 u69 u13 u223 u123 u79 u46 u45 u138 u107 u210
Carry	k eh r iy	u235 u78 u42 u68 u200 u202 u48 u75 u89 u133 u22 u193 u58
We	w iy	u26 u142 u208 u193 u58
March	m aa r td ch	u184 u131 u160 u26 u120 u222 u201 u70 u152 u57 u42 u204 u188 u54 u16 u65
Mars	m aa r z	u131 u160 u26 u157 u120 u197 u222 u103 u177 u4 u54 u194 u65

words are similar. Also observe that */n/* and */ng/* are confused by the ADUs. This phenomenon was also observed when comparing ADUs with phonemes in the previous section.

Comparisons with Other Systems

Table 5-14 shows the results of several competitive algorithms. Bacchiani and Ostendorf (1999), Singh et al. (2002) and Lee (2014) are representative work in this area. The first two approaches can be compared directly on RM. Lee (2014) evaluated their approach on the Jupiter Corpus (Zue et al., 2000) which consisted of spoken queries for weather information. We cannot compare directly to results by Lee (2014) because two datasets are different. Nevertheless, their results are presented in Table 5-14 and show a similar trend to ours. For example, their CI system is *19%* worse than their CI baseline while our CI system is *21%* worse than our baseline.

From this table we can see the low-complexity systems of Bacchiani and Ostendorf (1999) (rows 1 and 3) have similar performance to the ADU-based system with *1* mixture component per state (rows 5 and 8). However, their algorithm is much more complex than the ADU-based system and involves joint discovery of the lexicon and acoustic units in a supervised manner (words alignments are required). The high-complexity system (rows 2 and 4) of Bacchiani and Ostendorf (1999) performs better than other systems that use *1* mixture component per state. However, this system is based on another sophisticated algorithm – progressive refinement of a

Table 5-14. A comparison of several automatic lexicon discovery algorithms in terms of WER is shown.

System	Context Modeling	Corpus	WER (%)
CI low-complexity (1 mix) (Bacchiani and Ostendorf, 1999)	No	RM	19.70
High complexity system implicit context modeling (1 mix) (Bacchiani and Ostendorf, 1999)	Word Context	RM	11.40
CD low-complexity (1mix) (Bacchiani and Ostendorf, 1999)	Phoneme Context	RM	13.70
high-complexity system explicit context modeling (1 mix) (Bacchiani and Ostendorf, 1999)	Phoneme + Word Context	RM	9.90
CI probabilistic framework (Singh et al., 2002)	No	RM	20.00
CI phoneme baseline for (Singh et al., 2002)	No	RM	15.00
CI Nonparametric Bayesian (Lee, 2014)	No	Jupiter	17.00
CD Nonparametric Bayesian (Lee, 2014)	Phoneme Context	Jupiter	13.40
CI phoneme baseline Jupiter (Lee, 2014)	No	Jupiter	13.80
CD phoneme baseline Jupiter (Lee, 2014)	Phoneme Context	Jupiter	10.00

low-complexity system with additional iterations using the K-MEANS and Viterbi algorithms. This high-complexity system implicitly models the context (word context in this case) and therefore should be compared with context dependent systems.

Singh et al. (2002) used a slightly different baseline (e.g. semi-continuous HMMs) and a different language model with a higher perplexity. Their result is *25%* worse than their baseline while our result for *1* mixture system is *21%* better than our baseline. Their degradation in performance, however, might be a result of the fact that they have used a system with a high-perplexity language model.

Finally, the nonparametric Bayesian approach of Lee (2014) produces similar trends compared to our system. For example, their result is *19%* worse than their baseline while our result (*16* mixture) is *21%* worse than our baseline. We must emphasize that all of these systems

learned their corresponding acoustic units jointly with the lexicon on the same corpus while we intentionally introduced a mismatch to investigate the generalization performance of the ADU transducer. We also trained the lexicon separately from the ADU units. We expect we would obtain better results if we trained our ADU transducer using the same corpus.

5.6 Conclusions

In this chapter, we have investigated the segmentation properties of HDPHMM and DHDPHMM models. We have shown these nonparametric Bayesian HMMs can be used as an ADU transducer to map acoustic features into acoustic units. We have trained speaker independent transducers and shown through experimentation that ADUs discovered using TIMIT have a close relationship with phonemes in English.

We have also used the same ADU transducer for an STD by query task and obtained state of the art results for unsupervised algorithms on TIMIT. Finally we have investigated two algorithms to discover the lexicon from the output of an ADU transducer and parallel word-level transcriptions. We have shown both supervised and semi-supervised algorithms work reasonably well and the results are comparable with other reported algorithms.

One of the goals of these experiments is to demonstrate the capabilities of the discovered ADUs to generalize to completely new datasets and tasks. This is an important practical and theoretical consideration and is generally one of the goals of any useful machine learning algorithm. If ADUs discovered on TIMIT cannot be used on other English datasets with similar recording conditions (e.g. read speech) then the utility of these units would be questionable. We have shown through experimentation that an ADU transducer trained on TIMIT (which is a small dataset) generalize to RM.

Our goal in this chapter was to investigate some of the major properties of nonparametric Bayesian models discussed in previous chapters. Therefore examples provided in this chapter

(e.g. STD by query) have been approached in a relatively simple and straightforward manner. However, our results demonstrate good performance even for the relatively simple algorithms used in this dissertation. This suggests that our ADU-based units have significant untapped potential for the applications discussed in this chapter. We expect that we can improve both STD by query and lexicon discovery by incorporating more sophisticated techniques into our model.

We have also used relatively small corpora (e.g. TIMIT and RM) due to our computational limitations. Another important direction is to use more challenging corpora involving conversational speech and large vocabularies.

Finally, there are other potential applications for ADU units that we have not explored in this chapter. For example, automatic topic discovery, music retrieval and new word discovery are among the possible applications in speech and audio processing.

CHAPTER 6

CONCLUSION

In this dissertation we have investigated applications of nonparametric Bayesian approaches for some of the major problems encountered in acoustic modeling in speech recognition. Our techniques are applicable to almost any signal that has temporal structure that can be exploited. In Chapter 3 we introduced a DHDPHMM that allows sharing of mixture components. We have also derived an inference algorithm for this model and introduced several necessary extensions to model non-ergodic structures and non-emitting states. We have shown through experimentation that the proposed model outperforms HDPHMM by *14%* (*21.42%* vs. *24.40%* error rate) in problems similar to sub-word modeling. We have shown that the performance of this model is better than a discriminatively trained HMM and is similar to other state of the art algorithms for a phoneme classification task (*21.42%* vs. *24.60%*).

In Chapter 4 we have introduced a generative model for semi-supervised training of DHDPHMM. We introduced an approximation algorithm that simulates this generative model through an iterative procedure. Through experimentation we have shown that this model outperforms the supervised training algorithm (*29.02%* vs. *29.71%*). We have also introduced two approaches for context modeling. Through experimentation we have shown that CI semi-supervised DHDPHMM outperforms a baseline HMM trained using maximum likelihood or MMI (*28.36%* vs. *30.30%*). However, the gain for the CD model is not as significant. The reason is the fact that we have not approached the context modeling problem in a nonparametric Bayesian framework and have used traditional state tying approaches. In the next section we propose a model that can potentially solve this problem.

In Chapter 5 we have investigated the problems of speech segmentation and acoustic unit discovery using HDPHMM and DHDPHMM. Our approach is to train a transducer which we called ADU transducer. Through experimentation we have shown that the resulting transducer is speaker independent and can generalize to new datasets. We have also investigated the relationship between ADUs and phonemes and have shown that the discovered units are correlated with phonemes in English. We have shown that the segmentation performance of an ADU transducer is better than an unsupervised baseline (*76.62%* vs. *70.80%* F-score) and can be compared with state of art semi-supervised algorithms.

Further, we proposed a simple unsupervised STD by query algorithm based on an ADU transducer and have shown that it performs better than other unsupervised algorithms. Its EER performance can approach the performance of a complex tied-triphone system (*13.95%* for a system based on HDPHMM and system and *11.83%* for a system that combines HDPHMM and DHDPHMM systems vs. *11.70%* for triphone system).

Finally, we have proposed a semi-supervised algorithm to learn a lexicon from acoustic observations and a parallel word-level transcription. We have shown that an ADU transducer trained on TIMIT can generalize to a new dataset (e.g. RM) and a system trained using this lexicon can match or outperform other results reported in the literature (*20.33%* vs. *19.70%* for a system with *1* mixture component).

6.1 Future Work

As with any dissertation, there are many extensions or refinements of this work that can be pursued. Some of these are discussed below.

6.1.1 Nested DHDPHMM

DHDPHMM provides a framework to share mixture components within a model. Our experiments show this sharing, in addition to reducing the computational cost, improves the performance by *14%* relative to an HDPHMM. Therefore we should be able to extend this idea by defining a set of DHDPHMMs linked together using a third hierarchical structure (e.g. HDP). We can refer to such a model as a Nested DHDPHMM (NDHDPHMM). We can imagine a shared pool of mixture components for all the models in the set. We can also impose additional restrictions to force all models in the set have a similar structure. We expect this model can be useful for some of the problems mentioned in this dissertation.

It should also be noted that this nested structure could be extended to as many nested levels as necessary for a specific application. For example, this model can be used to obtain a nonparametric Bayesian context modeling algorithm if the set represents triphones with a similar central phoneme. Toward this end we can first train a CI DHDPHMM using the algorithms discussed in this dissertation and initialize the NDHDPHMM. Examples for each context will be used to train the corresponding DHDPHMM that share its components with other DHDPHMMs in NDHDPHMM. Therefore, all DHDPHMMs will be able to utilize a new observation if it is relevant to them (implicit tying).

6.1.2 A Speaker-Clustered ADU Transducer

Another application of NDHDPHMM is related to the ADU transducers introduced in Chapter 5. It is a known fact (Naito et al., 1998) that speaker-clustered systems work better than a speaker independent one if we have enough training data. A NDHDPHMM can potentially be used as collection of speaker-clustered ADU transducers where all transducers are linked together and share their mixture components. When decoding a new utterance, the system first assigns the speaker cluster and then processes the utterance using the appropriate transducer in the set. We

expect the results presented in Chapter 5 will improve moderately by using a speaker-clustered ADU transducer.

6.1.3 Different Priors

HDPHMM and DHDPHMM are both based on Dirichlet Process and hierarchical Dirichlet process priori distributions. These distributions not surprisingly impose certain constraints. For example in Chapter 3 we have shown that complexity grows slowly with the amount of data. In language modeling, other priors like Pittman-Yor (Teh, 2006) have been used instead of a Dirichlet process. Therefore it is import to examine other type of priors for models studied in this dissertation and for acoustic modeling problems.

6.1.4 HDPHMM/DHDPHMM with HMM State

In Chapter 5 we have discussed that using HDPHMM to train an ADU transducer imposes a restriction on the nature of learned ADUs by enforcing stationary units (e.g. generally phonemes are not stationary). One solution to overcome this problem is to use hierarchical HMMs (HHMMs) (Fine et al., 1998) instead of HMMs in the definition of HDPHMM. In this case each state of the HDPHMM will be replaced by a parametric HMM instead of a nonparametric Gaussian mixture and therefore each state can model non-stationary segments of the data.

6.1.5 ADU Generalization to Other Languages

We have investigated the ADU transducer for English. Specifically we have trained the transducer on TIMIT and evaluated it on both TIMIT and RM. We observed that a speaker independent transducer trained on TIMIT can generalize to a new corpus. However, it is also interesting to study the generalization to different styles of speech (e.g. conversational speech), environments (e.g. different type/level of noise) and languages. For example, can ADU units

trained on English data (possibly a much larger dataset than TIMIT) be used in Chinese spoken term detection by query task?

6.1.6 Experimentation Using Larger Corpora and More Difficult Tasks

All of the experiments in this dissertation have been conducted using relatively small corpora. TIMIT has less than *3* hours of training and RM is also on the same scale. Currently we only have used TIMIT for training our nonparametric Bayesian models. However, training DHDPHMM/HDPHMM models even using this small corpus takes between several hours to days on a medium size cluster computer where we have used both coarse and fine grain parallelism. Therefore scaling to large datasets is still an open problem and we need to explore more efficient inference algorithm based on variational methods (Blei and Jordan, 2005) instead of the current approaches based on Gibbs sampling (Gelman et al., 2004).

6.1.7 Discriminative Training and DHDPHMM/DNN

DHDPHMM already has better or comparable performance to discriminatively trained HMMs. For example error rate in phoneme classification task for DHDPHMM is *21.42%* while error rate for discriminatively trained HMMs is *24.60%*. Similarly error rate for CI DHDPHMM in a recognition task is *28.36%* while for CI HMM trained using MMI is *30.30%*. Therefore we expect to obtain moderately better results using discriminatively trained DHDPHMMs. HMM/DNN is based on predicting the state posterior probabilities for each frame (e.g. which HMM/state has generated this frame?). It is not guaranteed that a DHDPHMM/DNN works better than HMM/DNN. However, since DHDPHMM finds an optimum structure for each phoneme, it might provide DNN with further clues to classify the frame and therefore improve the overall performance.

6.1.8 More Robust Lexicon Generation

We have explored a relatively simple lexicon generation approach. Our results on well-behaved datasets like RM were comparable to other existing methods. However, we found that our performance degrades severely on datasets where the average number of examples for each word in the training set is very small. In many practical situations, we need to estimate pronunciations for many rare or even unseen words. Therefore we need a more sophisticated algorithm to train a G2P transducer from the output of an ADU transducer and parallel word transcriptions. Learning G2P transducers is an active research topic and it seems we have to leverage this ongoing research to improve our lexicon discovery algorithm. For example research on joint multigram models (Bisani and Ney, 2008) seems like a good starting point for a better lexicon discovery algorithm.

REFERENCES

Alphonso, I. (2003). *Network training for continuous speech recognition*. Mississippi State University, Oktibbeha County, Mississippi, USA.

Anderberg, M. R. (1973). *Cluster Analysis for Applications*. New York, NY, USA: Academic Press.

Bacchiani, Michael, & Ostendorf, M. (1999). Joint Lexicon, Acoustic Unit Inventory and Model Design. *Speech Communication*, 29(2-4), 99–114.

Bacchiani, M., Beaufays, F., Schalkwyk, J., Schuster, M., & Strope, B. (2008). Deploying GOOG-411: Early Lessons in Data, Measurement, and Testing. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)* (pp. 5260–5263), Las Vegas, Nevada, USA.

Bahl, L. R., Jelinek, F., & Mercer, R. L. (1983). A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 5(2), 179–190.

Baum, L. E., & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6), 1554–1563.

Beal, M., Ghahramani, Z., & Rasmussen, C. E. (2002). The Infinite Hidden Markov Model. *Proceedings of Neural Information Processing Systems* (NIPS) (pp. 577–584), Vancouver, Canada.

Bengio, Y. (2009*).* Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning,* 2(1), 1–127.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspective. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 35(8), 1798–1828.

Beulen, K., Bransch, E., & Ney, H. (1997). State Tying for Context Dependent Phoneme Models. *Proceedings of Fifth European Conference on Speech Communication and Technology* (pp. 1179–1182). Rhodes, Greece.

Bisani, M., & Ney, H. (2008). Joint-Sequence Models for Grapheme-to-Phoneme Conversion. *Speech Communication*, 50(5), 434–451.

Bishop, C. (2007). *Pattern Recognition and Machine Learning* (2nd ed., p. 738). New York, New York, USA: Springer.

Blackwell, D., & MacQueen, J. B. (1973). Ferguson Distributions Via Polya Urn Schemes. *The Annals of Statistics,* 1(2), 353–355.

Blei, D., & Jordan, M. (2005). Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1, 121–144.

Bourlard, H., & Morgan, N. (1993*). Connectionist Speech Recognition A Hybrid Approach*. New York, USA: Springer.

Bramer, M. (2007). *Principles of Data Mining*. London, UK: Springer-Verlag.

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. (1984). *Classification and Regression Trees* (1st ed., p. 368). Boca Raton, Florida, USA: Chapman and Hall/CRC.

Cappe, O., Godsill, S., & Mouline, E. (2007). An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo. *Proceedings of the IEEE*, 5(95), 899–924.

Carvalho, C., Hedibert, L., Polson, N., & Taddy, M. (2010). Particle Learning for General Mixtures. *Bayesian Analysis*, 5(4), 709–740.

Carvalho, C., Johannes, M., Hedibert, L., & Polson, N. (2010). Particle Learning and Smoothing. *Statistical Science*, 25(1), 88–106.

Clarkson, P., & Moreno, P. J. (1999). On the Use of Support Vector Machines for Phonetic Classification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 585–588). Phoenix, USA.

Davis, S., & Mermelstein, P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 357–366.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B* (Methodological), 39(1), 1–38.

Diaconis, P., Khare, K., & Saloff-Coste, L. (2010). Gibbs Sampling, Conjugate Priors and Coupling. *Sankhya A*, 72(1), 136–69.

Dusan, S., & Rabiner, L. (2006). On the Relation Between Maximum Spectral Transition Positions and Phone Boundaries. *Proceedings of INTERSPEECH* (pp. 1317–1320). Pittsburgh, Pennsylvania, USA.

Dymarski, P. (2011). *Hidden Markov Models, Theory and Applications*. InTech Open Access Publishers.

Ferguson, J. D. (1980). Variable Duration Models for Speech. *Proceedings of the Symposium on the Application of HMMs to Text and Speech* (pp. 143–179), Princeton, NJ, USA.

Fine, S., Singer, Y., & Tishby, N. (1998). The Hierarchical Hidden Markov Model: Analysis and Applications. *Machine Learning*, 32(1), 41–62.

Fink, G. A. (2008). Configuration of Hidden Markov Models from Theory to Applications. *Markov Models for Pattern Recognition* (pp. 127–136): Springer Berlin Heidelberg.

Fox, E., Sudderth, E., Jordan, M., & Willsky, A. (2011). A Sticky HDP-HMM with Application to Speaker Diarization. *The Annals of Applied Statistics*, 5(2A), 1020–1056.

Fukadai, T., Bacchiani, M., Paliwal, K., & Sagisaka, Y. (1996). Speech Recognition Based on Acoustically Derived Segment Units., *Proceedings of Fourth International Conference on Spoken Language* (ICSLP) (pp. 1077 – 1080). Philadelphia, Pennsylvania, USA.

Furui, S. (1986). Speaker-Independent Isolated Word Recognition Using Dynamic Features of Speech Spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing* , 34(1), 52 – 59.

Gales, M. J. F. (1996). *Model-Based Techniques for Noise Robust Speech Recognition*. Cambridge University, Cambridge, UK.

Ganapathiraju, A., Hamaker, J., Ordowski, M., Doddington, G., & Picone, J. (2001). Syllable-Based Large Vocabulary Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing,* 9(4), 358–366.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallet, D., Dahlgren, N., & Zue, V. (1993). TIMIT Acoustic-Phonetic Continuous Speech Corpus. The Linguistic Data Consortium Catalog. Philadelphia, Pennsylvania, USA: The Linguistic Data Consortium.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian Data Analysis* (2nd ed.). Chapman & Hall.

Ghahramani, Z. (2010). Bayesian Hidden Markov Models and Extensions. *Proceedings of the Fourteenth Conference on Computational Natural Language Learning* (pp. 56–56). Uppsala-Sweden.

Gillick, L., & Cox, S. j. (1989). Some Statistical Issues in the Comparison of Speech Recognition Algorithms. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP) (pp. 532 – 535), Glasgow, Scotland.

Gu, L., & Rose, K. (2000). Sub-State Tying in Tied Mixture Hidden Markov Models. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing* (ICASSP) (pp. II1013 – II1016). Istanbul, Turkey.

Gunawardana, A., Mahajan, M., Acero, A., & Platt, J. C. (2005). Hidden Conditional Random Fields for Phone Classification. *Proceedings of INTERSPEECH* (pp. 1117–1120), Lisbon, Portugal.

Halberstadt, A., & Glass, J. R. (1998). Heterogeneous Acoustic Measurements and Multiple Classifiers for Speech Recognition. *Ninth International Conference on Spoken Language Processing* (pp. 995–998). Sydney, Australia.

Harati, A., Picone, J., & Sobel, M. (2012). Applications of Dirichlet Process Mixtures to Speaker Adaptation. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 4321–4324). Kyoto, Japan.

Harati, A., Picone, J., & Sobel, M. (2013). Speech Segmentation Using Hierarchical Dirichlet Processes. *Proceedings of INTERSPEECH* (p. 637-641). Lyon, France.

Harati Nejad Torbati, A. H., Picone, J., & Sobel, M. (2014). A Left-to-Right HDP-HMM with HDPM Emissions. *Proceedings of the 48th Conference on Information Sciences and Systems* (pp. 1–6). Princeton, New Jersey, USA.

Harper, M. (2011). *IARPA Solicitation IARPA-BAA-11-02*. IARPA BAA.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning* (second ed.). New York City, New York, USA: Springer.

Hazen, T. J., Shen, W., & White, C. (2009). Query-By-Example Spoken Term Detection Using Phonetic Posteriorgram Templates. *Proceedings of IEEE Workshop on Speech Recognition and Understanding* (pp. 421–426). Merano, Italy.

Henter, G. E., Frean, M. R., & Kleijn, W. B. (2012). Gaussian Process Dynamical Models for Nonparametric Speech Representation and Synthesis. *Proceeding of the IEEE International Conference on Acoustics Speech and Signal Processing* (ICASSP) (pp. 4505– 4508). Kyoto, Japan.

Hieronymus, J. L. (1993). ASCII Phonetic Symbols for the World's Languages: Worldbet. *Journal of the International Phonetic Association*, 23.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohammed, A., Jaitly, N., Senior, A., et al. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 29(6), 83–97.

Huang, X., Acero, A., & Hon, H.-W. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm and System Development* (p. 1008). Upper Saddle River, New Jersey, USA: Prentice Hall.

Ishii, J., Tonomura, M., & Matsunaga, S. (1996). Speaker Adaptation Using Tree Structured Shared-State HMMs. *Fourth International Conference on Spoken Language* (pp. 1149–1152). Philadelphia, Pennsylvania, USA.

Ishwaran, H., & Zarepour, M. (2002). Exact and Approximate Sum Representations for the Dirichlet Process. *Canadian Journal of Statistics*, 30(2), 269–283.

Jelinek, Fred. (1997). *Statistical Methods for Speech Recognition* (p. 305). Boston, Massachusetts, USA: The MIT Press.

Jelinek, Frederick, & Mercer, R. L. (1980). Interpolated Estimation of Markov Source Parameters from Sparse Data. *Proceedings of Workshop Pattern Recognition in Practice* (pp. 381–397). Amsterdam, The Netherlands.

Juang, B.-H., & Rabiner, L. (1991*).* Hidden Markov Models for Speech Recognition. *Technometrics*, 33(3), 251–272.

Jun, J. (1995). *Perceptual and Articulatory Factors in Place Assimilation: An Optimality Theoretic Approach*. University of California, Los Angeles ,USA.

Kadane, J. B., & Lazar, N. A. (2004). Methods and Criteria for Model Selection. *Journal of the American Statistical Association*, 99(465), 279–290.

Kapadia, S., Valtchev, V., & Young, S. (1993). MMI Training for Continuous Phoneme Recognition on the TIMIT Database. *Proceeding of the IEEE International Conference on Acoustics Speech and Signal Processing* (ICASSP) (pp. 491 – 494). Minneapolis, MN, USA.

Klakow, D., & Peters, J. (2002). Testing the Correlation of Word Error Rate and Perplexity. *Speech Communication*, 38(1-2), 19–28.

Kneser, R., & Ney, H. (1995.). Improved Backing-off for M-gram Language Modeling. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP) (Vol. 1, pp. 181–184), Detroit, MI, USA.

Ladefoged, P. (1990). The Revised International Phonetic Alphabet. *Language*, 66(3), 550–552.

Ladefoged, P., & Johnson, K. (1993). *A Course in Phonetics* (3rd ed.). USA: Harcourt College Publication, .

Lamel, L., & Gauvain, J.-L. (1993). High Performance Speaker-Independent Phone Recognition Using CDHMM. *In Proceeding of Eurospeech*. (pp. 121–124). Berlin, Germany.

Lee, C. (2014). *Discovering Linguistic Structures in Speech: Models and Applications*. Massachusetts Institute of Technology.

Lee, C., & Glass, J. (2012). A Nonparametric Bayesian Approach to Acoustic Model Discovery. *Proceedings of the Association for Computational Linguistics* (pp. 40–49). Jeju, Republic of Korea.

Lee, K.-F. (1989). Context-Independent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(4), 599– 609.

Lee, K.-F. (1990*).* Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition. *IEEE Transactions on Acoustics, Speech and Language Processing,* 38(4), 599 – 609.

Lee, K.-F., & Hon, H.-W. (1989). Speaker-Independent Phone Recognition Using Hidden Markov Models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11), 1641–1648.

Lefèvre, F. (2003). Nonparametric Probability Estimation for HMM-Based Automatic Speech Recognition. *Computer Speech & Language*, 17(2-3), 113–136.

Levinson, S., Rabiner, L. R., & Sondhi, M. M. (1983). An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition. *Bell System Technical Journal*, 62(4), 1035–1074.

Ma, B., Li, H., & Lee, C.-H. (2005). An Acoustic Segment Modeling Approach to Automatic Language Identification. *Proceedings of INTERSPEECH* (pp. 2829–2832). Lisbin, Portugal.

Ma, J., & Schwartz, R. (2008). *Unsupervised versus Supervised Training of Acoustic Models*. *Proceedings of INTERSPEECH* (pp. 2374–2377). Brisbane, Australia:

MacEachern, S. N. (1999). Dependent Nonparametric Processes. *ASA Proceedings of the Section on Bayesian Statistical Science* (pp. 50–55). Alexandria, Virginia, USA.

McLachlan, G., & Thriyambakam, K. (2008). *The EM Algorithm and Extensions* (p. 400). Hoboken, New Jersey, USA: Wiley-Interscience.

Meignier, S., Bonastre, J.-F., & Igouney, S. (2001). E-HMM Approach for Learning and Adapting Sound Models for Speaker Indexing. *Odyssey Speaker Language Recognition Workshop* (pp. 175–180). Crete, Greece.

Mohammed, A., Dahl, G., & Hinton, G. (2009). Deep Belief Networks for Phone Recognition. *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications,* Vancouver, Canada.

Mohri, M., Pereira, F., & Riley, M. (2008). Speech Recognition with Weighted Finite-State Transducers. *Springer Handbook on Speech Processing and Speech Communication* (pp. 559–584), New York, USA, Springer.

Morgan, N., & Bourlard, H. (1995). Continuous Speech Recognition. *IEEE Signal Processing Magazine,* 12(3), 24 – 42.

Morris, J., & Fosler-Lussier, E. (2008). Conditional Random Fields for Integrating Local Discriminative Classifiers. *IEEE Transactions on Acoustics, Speech and Language Processing*, 16(3), 617 – 628.

Murveit, H., Butzberger, J., & Weintraub, M. (1991). Speech Recognition in SRI's Resource Management and ATIS Systems. *DARPA Speech and Natural Language Workshop* (pp. 94 –100).

Müller, M. (2007). Dynamic Time Warping. Information Retrieval for Music and Motion. *Information Retrieval for Music and Motion* (pp. 69–82). Secaucus, NJ, USA: Springer.

Naito, M., Deng, L., & Sagisaka, Y. (1998). Speaker Clustering for Speech Recognition Using the Parameters Characterizing Vocal-Tract Dimensions. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 981 – 984). Seattle, Washington, USA.

Navarro, G. (2001). A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, 33(1), 31–88.

Novak, J. R., Minematsu, N., & Hirose, K. (2012). WFST-Based Grapheme-to-Phoneme Conversion: *Open Source Tools for Alignment, Model-Building and Decoding. International Workshop on Finite State Methods and Natural Language Processing* (pp. 45–49), Donostia, Spain.

OpenMP Architecture Review Board (2008). *Application Program Interface Version 3.0*.

Palaz, D., Collobert, R., & Magimai-Doss, M. (2013). End-to-end Phoneme Sequence Recognition using Convolutional Neural Networks. *NIPS Deep Learning Workshop*, Lake Tahoe, USA.

Paliwal, K. (1990). Lexicon-Building Methods for an Acoustic Sub-Word Based Speech Recognizer. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 729–732). Albuquerque, New Mexico, USA.

Pan, J., Liu, C., Wang, Z., Hu, Y., & Jiang, H. (2012). Investigation of Deep Neural Networks (DNN) for Large Vocabulary Continuous Speech Recognition: Why DNN Surpasses GMMs in Acoustic Modeling. *Proceedings of Chinese Spoken Language Processing* (ISCSLP), 2012 8th International Symposium on (pp. 301 – 305). Kowloon, China.

Park, H., & Yoo, C. D. (2011). Gaussian Process Dynamical Models for Phoneme Classification. *NIPS 2011 Workshop on Bayesian Nonparametrics: Hope or Hype?,* Granada, Spain.

Petrov, S., Pauls, A., & Klein, D. (2007). Learning Structured Models for Phone Recognition. *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 897–905) , Prague, Czech.

Picone, J. (1990). Continuous Speech Recognition Using Hidden Markov Models. *IEEE Acoustics, Speech, and Signal Processing Magazine*, 7(3), 26–41.

Pitman, J. (1993). *Probability*. New York, New York, USA: Springer-Verlag.

Pitman, J. (1996). Random Discrete Distributions Invariant under Size-Biased Permutation. *Advances in Applied Probability*, 25(2), 525–539.

Pitman, J., & Yor, M. (1997). The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *The Annals of Probability*, 25(2), 855–900.

Price, P., Fisher, W., Bernstein, J., & Pallett, D. (1988). The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition. *Proceedings of the IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 651–654 vol. 1). New York, New York, USA.

Qiao, Y., Shimomura, N., & Minematsu, N. (2008). Unsupervised Optimal Phoneme Segmentation: Objectives, Algorithms and Comparisons. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 3989–3992). Las Vegas, Nevada, USA.

Quintana, F. A., & Tam, W. (1996). Bayesian Estimation of Beta-binomial Models by Simulating Posterior Densities. *Journal of the Chilean Statistical Society*, 13(1-2), 43–56.

Rabiner, L. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), 257–286.

Rasmussen, C. E. (2000). The Infinite Gaussian Mixture Model. *Proceedings in Advances in Neural Information Processing Systems* (NIPS) (pp. 554–560). Denver, Colorado, USA.

Rasmussen, C. E., & Ghahramani, Z. (2001). Occam's Razor. *Proceedings of Advances in Neural Information Processing Systems* (NIPS) (pp. 294 –300), Vancouver, Canada.

Rath, S., Povey, D., Vesely, K., & Cernocky, J. (2013). Improved Feature Processing for Deep Neural Networks. *Proceedings of INTERSPEECH* (pp. 109–113). Lyon, France.

Rijsbergen, C. J. Van. (2004). *The Geometry of Information Retrieval*. Cambridge University Press.

Rodriguez, A. (2011). On-Line Learning for the Infinite Hidden Markov. *Communications in Statistics: Simulation and Computation*, 40(6), 879–893.

Sainath, T. N., Kingsbury, B., & Ramabhadran, B. (2012). Auto-Encoder Bottleneck Features Using Deep Belief Networks. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 4153–4156), Kyoto, Japan.

Salakhutdinov, R., Tenenbaum, J. B., & Torralba, A. (2013*).* Learning with Hierarchical-Deep Models*. IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol. 35, pp. 1958–1971).

Schwartz, R., Chow, Y., Roucos, S., Kimball, O., Krasner, M., & Makhoul, J. (1985). Improved Hidden Markov Modeling of Phonemes for Continuous Speech Recognition. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing* (ICASSP) (pp. 1205 – 1208), San Diego, California, USA.

Seide, F., Li, G., & Yu, D. (2011). Conversational Speech Transcription Using Context-Dependent Deep Neural Networks. *Proceedings of INTERSPEECH* (pp. 437–440). Florence, Italy.

Sethuraman, J. (1994). A Constructive Definition of Dirichlet Priors. *Statistica Sinica*, 4(2), 639–650.

Sha, F., & Saul, L. K. (2006). Large Margin Gaussian Mixture Modeling for Phonetic Classification and Recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP) (pp. 265–268). Toulouse, France.

Shang, L. (2009.). Nonparametric Discriminant HMM and Application to Facial Expression Recognition. *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2090–2096). Miami, FL, USA.

Singh, R., Raj, B., & Stern, R. M. (2002). Automatic Generation of Subword Units for Speech Recognition Systems. *IEEE Transactions on Speech and Audio Processing*, 10(2), 89–99.

Steinberg, J. (2013). *A Comparative Analysis of Bayesian Nonparametric Inference Algorithms for Acoustic Modeling in Speech Recognition*. Temple University, Philadelphia, USA.

Steinberg, J., Harati, A., & Picone, J. (2013). A Comparative Analysis of Bayesian Nonparametric Inference Algorithms for Acoustic Modeling in Speech Recognition. *Proceedings of International Joint Conferences on Computer, Information, Systems Sciences, & Engineering* (pp. 1–5). Bridgeport, Connecticut, USA: Springer-Verlag.

Sudderth, E. (2006). *Graphical Models for Visual Object Recognition and Tracking*. Massachusetts Institute of Technology, Cambridge, MA, USA.

Sung, Y.-H., & Jurafsky, D. (2009). Hidden Conditional Random Fields for Phone Recognition. *IEEE Workshop on Speech Recognition and Understanding* (pp. 107 – 112), Merano, Italy.

Teh, Y., & Jordan, M. (2010). Hierarchical Bayesian Nonparametric Models with Applications. In S. W. Hjort, C. Holmes, P. Mueller (Ed.), *Bayesian Nonparametrics: Principles and Practice* (pp. 158–207). Cambridge-UK: Cambridge University Press.

Teh, Y., Jordan, M., Beal, M., & Blei, D. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(47), 1566–1581.

Teh, Y.-W. (2006). A Hierarchical Bayesian Language Model Based on Pitman-Yor Processes. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (pp. 985–992).

Teh, Y.-W. (2010). Dirichlet Process. *Encyclopedia of Machine Learning* (pp. 280–287). USA: Springer.

Tomlinson, G., & Escobar, M. (1999). *Analysis of Densities. University of Toronto,* Toronto, Canada.

Varadarajan, B., Khudanpur, S., & Dupoux, E. (2008). Unsupervised Learning of Acoustic Sub-word Units. *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics Short Papers* (pp. 165–168). Columbus,OH, USA.

Viterbi, A. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2), 260–269.

Wang, H., Tan, L., Leung, C.-C., Ma, B., & Haizhou, L. (2015). Acoustic Segment Modeling with Spectral Clustering Methods. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(2), 264–277.

Wang, P., Domeniconi, C., & Laskey, K. B. (2010). Nonparametric Bayesian Clustering Ensembles. *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases* (pp. 435–450), Springer Berlin Heidelberg.

Watanabe, S., Hori, T., McDermott, E., & Nakamura, A. (2010). A Discriminative Model for Continuous speech Recognition Based on Weighted Finite State Transducers. *Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing* (ICASSP), (pp. 4922 – 4925). Dallas, TX, USA.

Watanabe, S., Minami, Y., Nakamura, A., & Ueda, N. (2003). Application of Variational Bayesian Estimation and Clustering to Acoustic Model Adaptation. *Proceeding of the IEEE Conference on Acoustics, Speech, and Signal Processing* (ICASSP) (pp. I–568 – I–571), Hong-Kong, Hong-Kong.

Wolfowitz, J. (1942). Additive Partition Functions and a Class of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 13(3), 247–279.

Wood, F., & Teh, Y.-W. (2009). A Hierarchical Nonparametric Bayesian Approach to Statistical Language model domain adaptation. *International Conference on Artificial Intelligence and Statistics* (pp. 607–614), Florida USA.

Wooters, C., & Huijbregts, M. (2007). The ICSI RT07s Speaker Diarization System. *Multimodal Technologies for Perception of Humans* (p. pp 509–519). Springer.

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., et al. (2006). The *HTK Book* (p. 384). Cambridge, UK.

Young, S., & Woodland, P. C. (1994). State Clustering in HMM-Based Continuous Speech Recognition. *Computer Speech & Language*, 8(4), 369–383.

Yu, D., & Deng, L. (2010). Deep-Structured Hidden Conditional Random Fields for Phonetic Recognition. *Proceeding of INTERSPEECH* (pp. 2986–2989).

Zhai, C., & Lafferty, J. (2004). A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Transaction of Information Systems*, 22(2), 179–214.

Zhang, Y., & Glass, J. R. (2009). Unsupervised Spoken Keyword Spotting via Segmental DTW on Gaussian Posteriorgrams. *Proceeding of IEEE Workshop on Automatic Speech Recognition & Understanding* (pp. 398–403). Merano, Italy.

Zhang, Y., Salakhutdinov, R., Chang, H.-A., & Glass, J. (2012). Resource Configurable Spoken Query Detection using Deep Boltzmann Machines. *Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing (*ICASSP) (pp. 5161–5164). Kyoto, Japan.

Zipf, G. (1932). *Selective Studies and the Principle of Relative Frequency in Language*. MIT Press.

Zue, V., Seneff, S., Glass, J. R., Polifroni, J., Pao, C., Hazen, T. J., & Hetheringto, L. (2000). Jupiter: A Telephone-Based Conversational Interface for Weather Information. *IEEE Transactions on Speech and Audio Processing*, 8(1), 85–96.

Zweig, G. (2012). Classification and Recognition with Direct Segment Models. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing* (ICASSP) (pp. 4161 – 4164).