Towards A Compact Speech Recognizer: Subspace Distribution Clustering Hidden Markov Model

Brian Kan-Wing Mak

B.Sc. (Eng.)., Electrical Engineering, University of Hong Kong, Hong Kong, 1983

M.S., Computer Science, University of California, Santa Barbara, USA, 1989

A dissertation submitted to the faculty of the
Oregon Graduate Institute of Science and Technology
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Computer Science and Engineering

April 1998

The dissertation "Towards A Compact Speech Recognizer: Subspace Distribution Clustering Hidden Markov Model" by Brian Kan-Wing Mak has been examined and approved by the following Examination Committee:

Etienne Barnard
Associate Professor, CSE Department
Oregon Graduate Institute
Thesis Research Adviser

Enrico Bocchieri
Principal Member of Technical Staff
AT&T Labs – Research
Thesis Research Adviser

Ronald Cole
Professor, CSE Department
Oregon Graduate Institute

Hynek Hermansky
Associate Professor, EE Department
Oregon Graduate Institute

Dedication

To
my fiancée Man-Yin Tsang,
my family,
and those with a curious mind.

Acknowledgements

Anyone who has obtained a Ph.D. degree will probably agree that while the degree is rewarding, the life of Ph.D. study is hell — whoever wants a second Ph.D. degree must be either a lunatic or a moron! I even coin the term TIPS (Thesis In Preparation Syndrome) to describe the cycles of ups and downs, frustrations and eureka-type of joys during the life of Ph.D. study. To thrive in the study, if not to strive for the degree, great mentors are a great blessing. Borrowing the remark from Isaac Newton, "If I have seen further than you, it is by standing upon the shoulders of giants," I am fortunate to have the shoulders of two great advisors to stand upon: Etienne Barnard, Associate Professor of the Department of Computer Science and Engineering of Oregon Graduate Institute of Science and Technology (OGI); and Enrico Bocchieri, Principal Member of Technical Staff of AT&T Labs.

I still remember my first meeting with Etienne and Enrico. In the spring of 1994, I started looking for a thesis advisor. In my first "formal" meeting with Etienne, he said (paraphrased), "I think we have some interests in common, and we should make a good team." It was a very encouraging remark for a first-year Ph.D. student. By the summer of 1996, I went to AT&T Labs on an internship. When I first met Enrico in his office, he mentioned that before hiring me, he read my personal webpage and found that my work on phone clustering could be of use in his new project. Only then I was convinced that some people really read my personal webpage! Many thanks to both of them, for their invaluable advice, wise guidance, and insightful questions. I am also impressed with their depth of knowledge and their humility in treating students.

This thesis grew out of a project Enrico suggested during my summer internship at AT&T Labs, and is finished during my research at the Labs while pursuing my Ph.D. degree. This work would not have happened if not for the efforts of several people: Etienne who recommended me to the internship program; Roberto Pieraccini who forwarded my internship application to his colleague, Enrico; Enrico who recruited me; Ron Cole, director of CSLU (Center for Spoken Language Understanding) of OGI, Larry Rabiner, director of SIPS (Speech and Image Processing Services) Labs of AT&T Labs, David Roe and Candy Kamm, the previous and current head of Department HA6154000 of SIPS Labs, who smoothed out all administrative red tape to make this collaboration between

Contents

List of Tables

List of Figures

Abstract

Towards A Compact Speech Recognizer:
Subspace Distribution Clustering
Hidden Markov Model

Brian Kan-Wing Mak

Supervisors: Etienne Barnard and Enrico Bocchieri

After decades of research in speech recognition, the technology is finally entering into the commercial market. A significant challenge is to downsize research laboratory recognizers so that they can be used on platforms with less computational power: Most contemporary laboratory recognizers require too much memory to run, and are too slow for mass applications. This thesis addresses the problem by greatly reducing the number of parameters in the acoustic models. We focus on more compact acoustic models because they constitute a major component of any speech recognizers, and the computation of their likelihoods consumes 50–70% of total recognition time for many typical tasks.

The main contribution of this thesis is the formulation of a new acoustic modeling method which we call *subspace distribution clustering hidden Markov modeling* (SDCHMM). The theory of SDCHMM is based on tying continuous density hidden Markov models (CDHMMs) at a new finer sub-phonetic unit, namely the subspace distribution. Two methods are presented to implement the SDCHMMs. The first implementation requires training a set of intermediate CDHMMs followed by model conversion in which the distributions from the CDHMMs are projected onto orthogonal subspaces, and similar subspace distributions are then tied over *all* states and *all* acoustic models in each subspace. By

exploiting the combinatorial effect of subspace distribution encoding, all original full-space distributions can be represented by combinations of a small number of subspace distribution prototypes. Consequently, there is a great reduction in the number of model parameters, and thus substantial savings in memory and computation. Furthermore, we demonstrate in the second implementation method that, given prior knowledge of the tying structure of the subspace distributions, SDCHMMs can be trained directly from much less data. This renders SDCHMM very attractive in the practical implementation of acoustic models, speaker-specific training, and speaker/environment adaptation.

Evaluation on the ATIS (Airline Travel Information System) task shows that in comparison to a CDHMM system, a SDCHMM system achieves 7- to 18-fold reduction in memory required for acoustic models, runs 30–60% faster, and can be trained with 10–20 times less data, without any loss of recognition accuracy.

Chapter 1

Introduction

Five decades of interdisciplinary research in widely different areas such as linguistics, psychoacoustics, signal processing, computer science, pattern recognition, and information theory, has greatly advanced the state of the art in automatic speech recognition (ASR). The ASR technology has evolved progressively from recognizing a few hundred isolated words with speaker dependency in the 70s, tens to hundreds of connected words without speaker dependency in the 80s, to speaker-independent large-vocabulary continuous speech at present. The capabilities of some state-of-the-art recognizers over a wide range of tasks are summarized in Figure 1.1 [12, 28, 34, 54, 87]. In the figure, the difficulty of a recognition task (from the perspective of a recognizer) is measured by the perplexity of its language model, whereas the performance of a recognizer is gauged by its word error rate (WER). The perplexity of a language model is the average number of words that may follow another word [69]; and WER is the percentage of words which are wrongly recognized[1]. Although these recognizers are task-specific, and they usually only operate well in the domains and under the conditions (channel, signal-to-noise ratio, accent, etc.) they are trained for, they already represent a mature technology ready for large-scale deployment.

The advancement of ASR technology can be attributed to the following factors:

- the success of modeling acoustics using the stochastic hidden Markov models (HMMs)

- the success of statistical language modeling which produces simple but powerful language models

---

[1]See Section 3.1.2 for more details.

40

Word Error Rate (%)

8

6

4

2

0

telephone
conversation
(Switchboard)

read news
(NAB)

isolated
English letters

read database
query (RM)

(spontaneous)
airline reservation
(ATIS)

digit strings

11   25   50   100   200

Perplexity
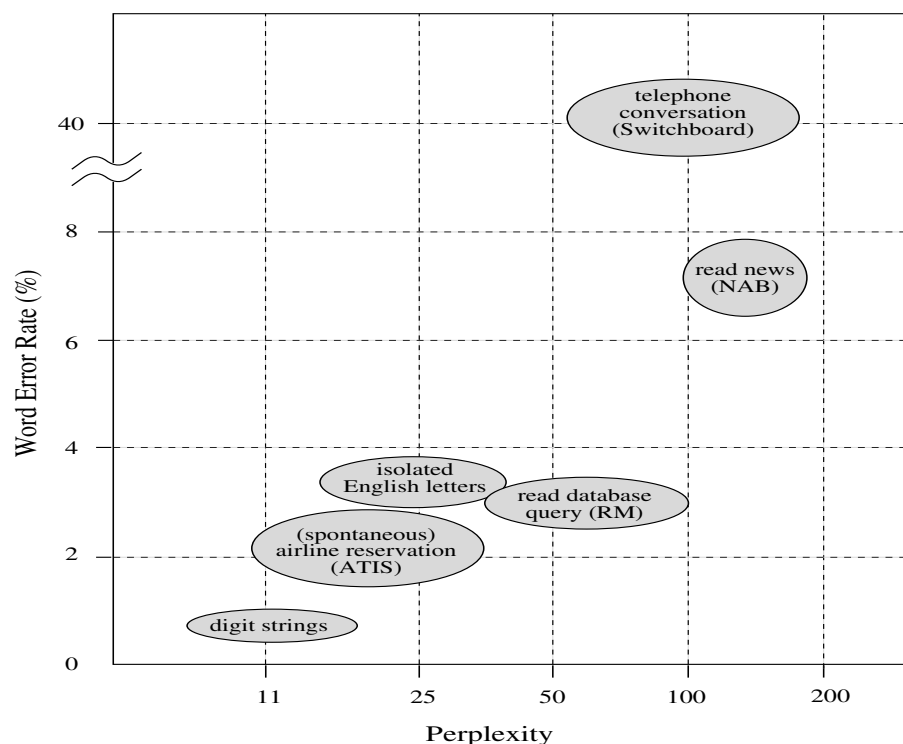
Figure 1.1: Capability of state-of-the-art automatic speech recognizers

- the use of dynamic programming algorithms together with pruning techniques in efficiently searching the vast solution space;

- the availability of large domain-specific speech and text corpora which allow training of more complex and accurate acoustic models as well as language models

- the blessing of Moore's law[2] for progress in semiconductor technology — laboratory workstations today are as powerful as a supercomputer less than two decades ago. Without fast processors and large on-board memory, large-vocabulary continuous speech recognition is simply impossible.

---

[2] In 1965, Gordon Moore identified that the logic density of silicon integrated circuits has been doubling every eighteen months with proportional decreases in cost. The statement has held true since then.

1.1 The Problem: Too Many Parameters

Despite the long desire to use speech — often the most natural and efficient modality humans use to communicate — for human-machine interaction, the ASR technology has not prevailed, and the promise of a ubiquitous speech user interface has yet to be fulfilled. What is hidden behind the impressive ASR results in Figure 1.1 is the tremendous computational cost of many state-of-the-art recognizers: To arrive at the low WERs, recognizers are running at one to two orders of magnitude slower than real time, requiring high-end research workstations equipped with hundreds of megabytes (MB) of memory. A significant challenge is to adjust laboratory recognizers so that they may be deployed on more affordable machines of lower processing power and smaller memory size *without* losing accuracy. Techniques exist to reduce memory requirement alone, for example, by using simpler but less accurate models, or through data compression [73]. There are also techniques to speed up computation alone: for example, by simply exercising more vigorous pruning schemes, by computing state likelihoods only from a small subset of the most relevant state probability density distributions [6, 8, 41, 63, 79], or by fast-match techniques [21]. However, these techniques are usually done at the expense of recognition accuracy; in the case of computation speedup, more memory is usually required. In order to achieve faster computation speed and smaller memory footprint without sacrificing accuracy — three seemingly conflicting goals — each system component (acoustic models, language model, search engine, knowledge database, etc.) should be subject to careful scrutiny.

Analysis of the execution profiles of speech recognizers of various vocabulary sizes (except the very large vocabulary[3]) reveals that roughly 50–70% of the overall recognition time is spent in computing state likelihoods of the acoustic models. The result is not surprising; in their craving for cranking out every bit of recognition accuracy, laboratory speech recognizers are building highly complex acoustic models with a huge number of

---

[3]For vocabulary size of more than tens of thousands of words, computation of state likelihoods amounts to about 25–30% of total runtime. Nonetheless, most ASR applications that are conceivably ready for deployment in the near future will have smaller vocabularies and will be the main focus in this thesis.

parameters to capture the fine phonetic details. For example, let us look at two state-of-the-art recognizers which were among the top three systems in the ARPA evaluation of ATIS (Airline Travel Information System) [25] in 1994. AT&T's general-purpose ATIS recognizer [9] contains more than 6 million parameters in its acoustic models requiring more than 24MB of memory space and consequently runs at 7 times real time to obtain a WER of 5.2% on the ATIS task on an SGI O_2 machine (MIPS R10000 CPU, 195 MHZ, 2GB shared memory). Similarly CMU's Sphinx II large-vocabulary recognizer [28] requires more than 40MB of memory to represent its acoustic models which have over 10 million parameters, and obtains a WER of 5.1% on the same task using an 175MHz Alphastation at the speed of 9 times real time[4]. In general, a large model parameter space leads to the following problems:

- larger memory requirement

- slower recognition

- requiring more training data

- requiring more data for speaker/environment adaptation.

Thus, if we can reduce the number of free parameters in acoustic models — the basic component of any speech recognizer — both the memory and the speed problem will be addressed at the same time. In addition, if the goal of less training data can be achieved, productivity can be improved as well since recognition systems can then be trained in a shorter time using less memory. In this thesis, we propose a more efficient acoustic modeling methodology to arrive at a more compact recognizer.

1.2 Proposed Solution: It Is Time to Share More!

The most common approach to reducing the number of parameters in acoustic models is parameter tying: Similar structures are discovered among the acoustic models,

---

[4]This is the recognition speed before the recent implementation of efficient search algorithms as described in [73]. The efficient search later increases the speed to 1.6 times real time.

and they are then tied together to share the same value. With the (limited) amount of training data on hand, parameter tying allows more complex acoustic models to be estimated reliably while the number of model parameters will not grow unchecked. In the past, the technique of parameter tying has been applied successfully at various granularities: Phones (generalized biphones/triphones [48], context-independent phones [49]), states (tied-state HMM [32, 88]), observation distributions (tied-mixture/semi-continuous HMM [4, 29, 80]), and feature parameters [85] have all been tied. For example, of the two aforementioned systems, AT&T's ATIS recognizer is a tied-state system, whereas CMU's Sphinx II employs semi-continuous HMMs. The technology trend is to tie acoustic models at finer and finer details so as to maintain good resolution among models as much as possible. In this thesis, we propose to push the technique to an even finer subphonetic unit — subspace distributions — in the context of hidden Markov modeling[5]. Subspace distributions are the projections of the full-space distributions of an HMM in lower dimensional spaces. The hypothesis is that speech sounds are more alike in some acoustic subspaces than in the full acoustic full space. We call our novel HMM formulation *"subspace distribution clustering hidden Markov modeling"* (SDCHMM).

Subspace distribution clustering hidden Markov models (SDCHMMs) are derived from already existing continuous density hidden Markov models (CDHMMs) without requiring any extra training data nor re-training. The distributions of CDHMMs are projected onto orthogonal subspaces (or streams[6]), and similar subspace distributions are then tied into a small number of distribution prototypes (or codewords) over *all* states and *all* acoustic models in each subspace. By exploiting the combinatorial effect of subspace distribution encoding, all original full-space distributions can be closely approximated by some combinations of a small number of subspace distribution prototypes. Consequently, there is a great reduction in the number of model parameters, and thus substantial savings in memory and computation. This renders SDCHMM very attractive in practical implementation

---

[5]Since most state-of-the-art speech recognition systems are HMM-based, we thus only consider our thesis in such context. Recently artificial neural networks (ANN) have been applied to ASR with some success [2, 11, 76], but HMMs remain the dominant technology.

[6]In this thesis, the two terms, "subspace" and "stream" are used interchangeably to mean a feature space of dimension smaller than that of the full feature space. "Subspace" is clearer mathematically, but "stream" is more common in the speech recognition community.

6

of acoustic models. Furthermore, we demonstrate that given *a priori* knowledge of the tying structure of the subspace distributions in SDCHMMs, SDCHMMs can be trained directly from speech data without going through intermediate CDHMMs. Because of the great reduction of parameters in SDCHMMs, such SDCHMM training requires much less training data. It will therefore be of great importance to speaker-specific training and speaker/environment adaptation.

From the perspective of quantization, one may consider SDCHMM as an approximation to the highly accurate CDHMM, achieving great data compression by subspace distribution quantization. From the perspective of hidden Markov modeling, SDCHMM unifies the theory of CDHMM which employs full-space state probability density distributions and the feature-parameter-tying HMM [84, 85] which is generated by scalar quantization of the distributions. SDCHMM combines the accuracy of CDHMM with the compactness of feature-parameter-tying HMM. In this aspect, it is interesting to compare this work with a similar approach called "split vector quantization" [44, 65] that has been successfully applied to high-quality, low-bit rate speech coding for years. In speech coding, it is known that (full) vector quantization (VQ) results in smaller quantization distortion than scalar quantization at any given bit rate [35]. However, to attain the required high quality in practical telecommunication, full VQ suffers from training, memory, and computation problems much like those of our current complex ASR systems. Split VQ overcomes the complexity problem of full VQ by splitting the speech vectors into sub-vectors of lower dimensions and quantizing the sub-vectors in their subspaces.

1.3 Thesis Summary and Outline

In this thesis, we present the theory of subspace distribution clustering hidden Markov modeling. The development of the theory and its implementation is done using Gaussian distributions with diagonal covariances, though it can be applied more generally. Before the implementation of the new models, the following two basic issues are answered:

- how to define the subspaces?

- how to tie the subspace distributions?

We suggest a simple but coherent definition for streams of any dimension: The streams comprise the most correlated features. We devise a modified k-means Gaussian clustering scheme using the Bhattacharyya distance as the distance measure between Gaussian distributions [17]. Finally, two implementation methods for the SDCHMM are studied in detail: model conversion from CDHMMs and direct SDCHMM training.

Throughout the thesis, the ATIS recognition task is used as the test-bed for evaluating the SDCHMMs. In summary, the performance of a set of 20-stream SDCHMMs converted from the context-dependent CDHMMs of the baseline AT&T's ATIS system with only 64 subspace Gaussian prototypes per stream epitomizes the power of SDCHMM:

- It is slightly more accurate than the baseline CDHMM system (WER of 5.0% versus 5.2%).

- It runs at twice the speed of the original system (3.5 versus 7.0 times real time).

- The acoustic models take up 1.8MB of memory compared with the original 24MB — a 13-fold reduction.

- If we have *a priori* knowledge of the tying structure of its subspace Gaussians and the mixture weights (borrowed from the original CDHMMs), it can be trained directly from scratch with as little as 8.3 minutes of speech with *no* loss of accuracy. (For comparison, the original CDHMM system is trained with 36 hours of speech.)

The organization of this dissertation is as follows.

In Chapter 2, the current technology of acoustic modeling is reviewed. After an introduction to hidden Markov modeling, choices of speech modeling units are discussed, leading to the necessity of the technique of parameter tying. Since many variations are used, we will discuss the most typical acoustic modeling techniques rather than specific implementations.

In Chapter 3, we describe the baseline AT&T ATIS recognition system, the performance of which is used as a benchmark throughout this thesis. All aspects of the system such as signal processing, training procedure, language modeling, and search algorithm

will be discussed. In addition, its accuracy, computation time, and memory requirement will be given.

Chapter 4 is the main part of the thesis as it will present the theory of SDCHMM in detail, and explain why it is preferred by comparing with other modeling methodologies that are described in Chapter 2.

Chapter 5 presents the first implementation method of SDCHMMs: model conversion from CDHMMs. Here, we propose a coherent definition for the streams and a Gaussian clustering algorithm to convert CDHMMs to SDCHMMs, which are then evaluated on the ATIS task. The effect of different numbers of streams and different amounts of tying will be studied and evaluated on three metrics: accuracy, computation time, and memory requirement.

In Chapter 6, we present a brief analysis on the phonetic-acoustic nature of the tying structure of SDCHMMs. That is, we examine whether the tying structure among different phonemes agrees with what we know from phonetics.

Chapter 7 describes the second implementation method of SDCHMMs — direct SD-CHMM training. First, we expand the theory of SDCHMM by presenting the reestimation formulas of its various quantities. Then the SDCHMM training scheme is developed. Finally, by progressively reducing the amount of ATIS training data by half each time, we study the training data requirement for SDCHMMs (which have many fewer model parameters than the original CDHMMs).

Finally, in Chapter 8, we summarize our findings of using SDCHMM for automatic speech recognition, and our contributions in this thesis. We also suggest some directions for future development of SDCHMM, especially in the area of speaker or environment adaptation.

Chapter 2

Review of Acoustic Modeling Using Hidden Markov Model

Before we may call upon the large inventory of classification methods to recognize speech, we first have to build a mathematical model for each speech sound from its acoustic signal so that the acoustic models describe the sounds as "closely" as possible (according to some cost function, such as likelihood). A mathematical model requires a definition of the input observation space, and a model form. In this chapter, we first look at some characteristics of speech which will help guide our choice of a mathematical model for speech. Then we will discuss some choices of speech units commonly used for acoustic modeling, leading to the need of parameter tying.

2.1 Speech Characteristics

Speech is produced through complex coordination among articulators in our vocal tract such as vocal cords, jaw bones, tongue, lips, etc. Although the dimensions, shapes and dynamic behaviors of the articulators vary from one person to another, they move in well coordinated ways, governed by the laws of physics. As a result, speech exhibits the following characteristics:

SC-I: Speech is a time-varying signal. A speech model must describe the temporal behavior of the acoustics.

SC-II: Speech is not memoryless. Each speech sound is produced by a sequence of well-known movements of articulators. A speech model thus has to capture the sequential

nature of speech.

SC-III: As a mechanical system, our speech production system cannot change abruptly from one configuration to another, resulting in a "quasi-stationary" speech signal at intervals when its articulators stay at relatively stable positions during the course of the speech.

SC-IV: Since the articulators move smoothly during the production of speech, successive acoustic samples are highly correlated, and there is much redundancy overall. This suggests that a more succinct representation of the raw speech signal is possible.

SC-V: The realization of the same speech sound varies from person to person, and from time to time even with the same person. Thus a speech model must allow for such variabilities.

2.2   Selection of Input Speech Space and Speech Model

Besides accounting for the forgoing speech characteristics, the choice of acoustic representation and modeling methodology depends largely on the recognition paradigm. As of date, automatic speech recognition (ASR) has been best tackled in the framework provided by statistical pattern recognition (SC-V). In the following, we will describe only the most common acoustic representation and modeling technique used in this paradigm.

2.2.1   Cepstral Input

Spectral representations have been found to be adequate for speech. In practice, short-term spectral analysis is usually applied over a window of 20–30ms of speech (SC-III) at about every 10ms (SC-I). The spectrum (envelope) is then encoded succinctly by a vector of, say, 12 cepstral coefficients [72] (SC-IV). Due to the findings from psychoacoustical studies which show that humans do not perceive frequencies greater than 1kHz in a linear scale but instead in a logarithmic scale, the cepstral coefficients are more commonly expressed as mel-frequency [82] cepstral coefficients (MFCC) or perceptual linear predictive (PLP) coefficients [24] in the Bark scale [90].
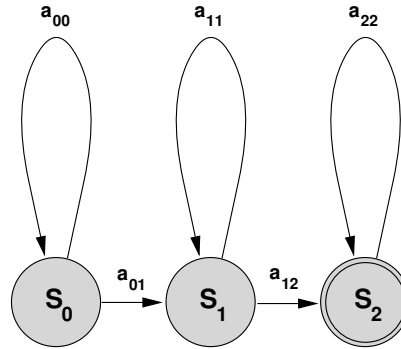
Figure 2.1: A first-order 3-state left-to-right hidden Markov model (where a_{ij} denotes the transition probability from state i to state j)

2.2.2 Hidden Markov Model

Most of the aforementioned speech characteristics can be captured by a probabilistic finite-state machine called the hidden Markov model (HMM). Figure 2.1 shows a first-order 3-state left-to-right HMM, most commonly used in ASR. The left-to-right HMM has a set of states with one designated as the (leftmost) starting state and one the (rightmost) ending state, representing the beginning and the ending of a speech sound. A state roughly corresponds to a quasi-stationary region in the speech sound (SC-III) while state transitions correspond to temporal movement of the speech signal (SC-I). In a left-to-right HMM, only left-to-right state transitions are allowed so as to capture the sequential nature of speech (SC-II). When a state transition occurs, an acoustic observation is emitted. In various formulations, an emitted observation has been associated with the transition arc, the source state or the destination state. Here we associate an observation with the destination state. Both transitions and observation emissions are probabilistic and they model the temporal variability and acoustic variability of speech respectively (SC-V). The hidden nature of the model is due to a doubly embedded stochastic process: Only the stochastic process which emits acoustic events at the states is directly observable and the other stochastic process (state transition) which controls state occupancy is hidden.

Figure 2.2 depicts a basic configuration of continuous density HMM-based acoustic models. In the figure, each phonetic unit is modeled separately by one left-to-right HMM

12

which consists of three states, and the state observation probability distribution is esti-
mated as a mixture density with two Gaussian components.
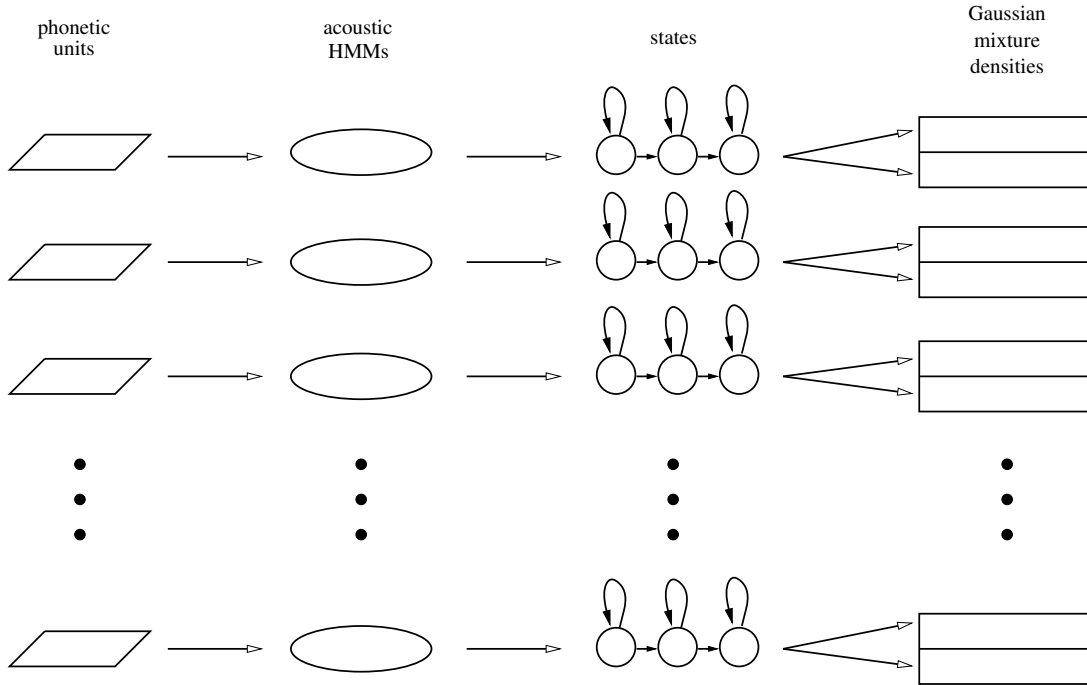
Figure 2.2: Basic configuration of acoustic models using hidden Markov modeling

First-order HMM Assumptions

There are two major assumptions in the foregoing discussion:

1. State-dependency of transition. For simplicity and computational tractability, a first-order HMM is used, so that state transitions only depend on the current state and not on the past state history nor the future states. Further, as a side effect, the assumption also leads to an unrealistic exponential state duration distribution.

2. State-dependency of observation. An acoustic observation depends only on its emitting state, and not on neighboring states nor previous frames of observation. Consequently, states actually become stationary (not just quasi-stationary) and the observations in a given state can thus be assumed to be independently and identically

distributed.

HMM Variants

Obviously the two assumptions violate the fact that neighboring acoustic events are highly correlated. To alleviate the shortcomings, many modifications have been proposed recently which mainly differ in the extent to which they relax the assumptions:

- The simplest way to incorporate temporal or contextual information without modifying the HMM formulation is to add dynamic features onto the input feature vector [18].

- Higher-order HMMs have been used to include more of the past state history. Because of the big increase in computational complexity, only second-order HMMs have been tried, but with limited success [58].

- State-dependent correlations between successive frames are explicitly modeled by conditioning the observation probability jointly on both the current state and the preceding observation [64]. Because of the huge increase in the number of estimation parameters, approximation is used to express the joint conditional probabilities in terms of individual conditional probabilities in bigram-constrained HMMs [83] (using observation bigrams), or using an extended logarithmic pool [40] .

- Hybrid models of Artificial Neural Networks and HMMs incorporate contextual information simply by using the current feature frame together with neighboring feature frames [11].

- Hybrid models of Recurrent Neural Networks and HMMs explicitly estimate the posterior phone probability of the current frame conditional on both the state and *all* its previous frames more effectively [76].

- The state-dependent acoustic trajectory is modeled by an autoregressive process in autoregressive HMMs [37, 68], and in trended HMMs with a linear polynomial regression function of time [14].

14

- Segmental HMMs such as the stochastic trajectory model [20] and stochastic segment model [62] relax both assumptions and explicitly model acoustic trajectories of variable durations as well as the duration.

A detailed description of HMMs is outside the scope of this thesis and interested readers are referred to [70].

2.2.3   Our Choice of HMM for Acoustic Modeling

For simplicity and computational tractability, we employ the first-order HMMs with cepstral coefficients augmented by their first- and second-order time derivatives as the speech features. They are most commonly used and are found to be about as accurate as the theoretically more sound variants. Their simplicity and explicit parameterization also make them amenable to a wide range of techniques that perform fast state-likelihood computation [6, 8, 41, 63, 79] and reduce parameters through parameter tying (see Section 2.4).

In particular, the continuous density HMM (CDHMM) is used as generally it is found to be highly accurate. For fast computation and trainability, the continuous state observation probability density distributions are estimated as mixture Gaussian densities with diagonal covariances. In theory, the mixture Gaussian density function can approximate, arbitrarily closely, any finite continuous density function.

Finally, we would also like to point out some minor issues in our implementations of HMMs:

- Precise estimation of the transition probabilities is found unnecessary, and each transition arc is simply assigned an equal probability of *1/fan-out*, where *fan-out* is the number of arcs coming out of the source state [10, 89].

- No explicit duration modeling is performed as it is not found necessary.

- Our HMMs are trained using the simpler segmental *k*-means algorithm [38] which has been found to work as well as the conventional Baum-Welch algorithm [3] ([45, 71, 89]).

2.3 Speech Unit to Model

Although it may be most natural to model each word individually, the approach is impractical in most recognition tasks, except those of small vocabularies and isolated words. In most cases, there are not sufficient training data for each word. One separate model for each word also requires large storage space and long decoding time. Thus, it is generally necessary to model sub-word units so that training, storage, and computational resources can be shared. The sub-word speech units are shared among the pronunciations of a lexicon, and can avoid repetitions of likelihood computation of the models in, for example, fast lexical tree search. Chronologically, ASR has progressed with smaller and smaller units of speech exhibiting finer and finer acoustic details:

- Multi-phone units: syllables, demisyllables, and diphones. They encapsulate co-articulatory effects between phones. The major problem in the past was the large number (over 1000 demisyllables, 2500 diphones, and 20,000 syllables) of these units. With larger speech corpora nowadays, they are worth reconsidering[1].

- Context-independent (CI) phones are monophones. Each phone is modeled by the same statistics, irrespective of its surrounding acoustic contexts.

- Context-dependent (CD) phones are phones uttered in a specific context. For instance, left/right biphones are phones on the right/left of another phone; triphones assume that acoustic realization depends only on the immediately preceding and following phones.

- Sub-phonetic units are components of a phonetic HMM such as states or distributions. They are motivated mainly for efficient acoustic modeling and may not be associated with a phonetic meaning.

---

[1]As a matter of fact, there is already renewed interest in using syllables as the modeling units [22, 27, 33].

2.4 Parameter Tying

The history of acoustic modeling is guided by the need to strike a balance between two conflicting goals for acoustic models: trainability and resolution. That is, the acoustic models should contain enough fine acoustic details so that different models can be resolved during decoding; but too many details generally result in too many model parameters, and reduce the robustness of model parameters when estimated from limited amounts of training data (and the amount of training data is always limited as the complexity of acoustic models grows). Moreover, a large number of model parameters leads to larger memory size, slower recognition, and more difficult speaker/environment adaptation. In the past, the technique of parameter tying has been applied successfully to obtain such a balance by reducing the number of parameters in acoustic models at various granularities. Let us look at some of the most typical HMM tying schemes.

Monophone HMM: A monophone ties all allophones of a base phone. The small number (less than 100) of monophones results in CI systems that are simple to implement and fast to run. However, the accuracy of CI systems is usually modest.

Generalized context-dependent phone HMM: Generalized triphone HMMs [47] are most commonly used. The large number of triphones (about 125,000 in English, assuming all possible combinations of three phones) makes it almost impossible to model each of them equally reliably. Fortunately many contexts of the *same* base phone are similarly realized and may be clustered to much fewer generalized triphones. Generalized triphones derived using a phonetic decision tree have the additional benefit of capturing "unseen" triphones [48].

Tied-state HMM (TSHMM): Since co-articulatory effects are more prominent at the onset and ending of a phone than at its center, they are better categorized at local HMM states than over the whole HMM phone as in generalized triphones. Therefore TSHMM clusters the *corresponding* HMM states of the *same* base phone [32, 88]. Young and Woodland reported a 5-fold reduction in the number of states in their TSHMM systems [88].

Tied-mixture HMM/semi-continuous HMM (SCHMM): SCHMM is a generalization of the discrete HMM in which prototypes are continuous Gaussian distributions with diagonal covariances instead of just mean vectors [4, 29, 80]. Consequently, an SCHMM enjoys fast computation of state likelihoods but reduces the quantization errors of discrete HMMs, with a negligible increase in model parameters. Moreover, the SCHMM is a special case of distribution tying in which all Gaussian components of all HMMs are clustered to a common set of Gaussian prototypes.

Feature-parameter-tying HMM (FPTHMM): In speech coding, it is found that for the same bit rate, full-space vector quantization (VQ) is always more efficient than scalar quantization (SQ); nonetheless, SQ can represent each 1-dimensional feature in very few scalar codewords and thus requires much smaller storage for its codebooks. In the same spirit, FPTHMM [84, 85] clusters Gaussian components of a continuous density HMM (CDHMM) with diagonal covariances in each dimension into very few 1-dimensional Gaussian prototypes[2]. When combined with SQ of input features, likelihoods due to all shared scalar Gaussian prototypes can be pre-computed and stored in a look-up table. Subsequent (full-space) state observation likelihoods can then be computed more efficiently without any divisions or multiplications [78].

Several tying schemes may be cascaded together in a HMM system at the same time to achieve the best balance between model complexity and model-estimation robustness as exemplified in Figure 2.3. In the figure, the objects in each tying level, namely HMMs, states, and Gaussian distributions, are organized as a pool of shared objects. An object in a tying level may be shared by several objects in the previous level (to its left). Not only are the tying structures more compact to store, they also avoid evaluating the same object twice during recognition. Perhaps the best example is the hidden Markov network of [77] which ties allophones, states, distributions and feature parameters. Alternatively, several tying schemes may also be combined or merged. For example, genones [15] or state-clustered tied-mixture HMMs [61] divide all states into classes and only tie the mixtures

---

[2]Speech recognition systems are only concerned with the storage space for the distributions. As an analogy, transmission bit rate in speech coding is equivalent to the memory requirements in ASR for encoding the full-space distributions by the scalar distribution prototypes of each dimension.

18
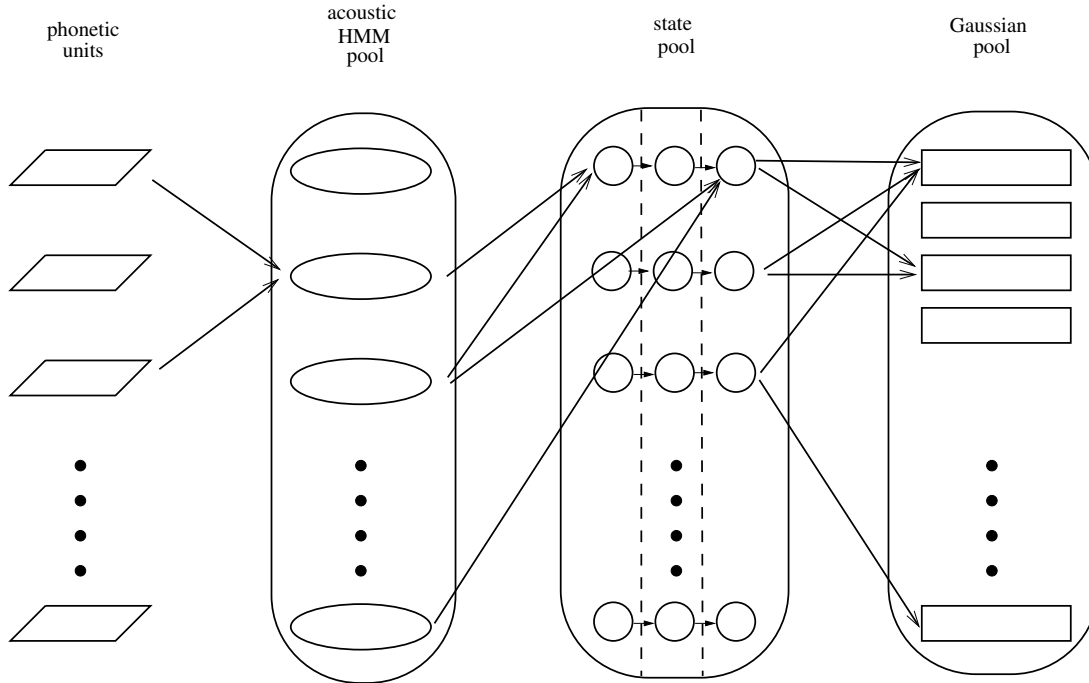
Figure 2.3: Various tying schemes of acoustic models using hidden Markov modeling

within each class with the aim to enhance model resolution.

In this thesis, we try to push the technology of parameter tying to a finer sub-phonetic level, namely that of subspace distributions. We call our new HMM derivative, the *subspace distribution clustering hidden Markov model* (SDCHMM). It may be considered a generalization of distribution tying and feature-parameter tying techniques. Before we introduce our SDCHMM in Chapter 4, we first present in the next chapter the baseline ASR system and the benchmark test we will use throughout this thesis to demonstrate the efficacy of SDCHMMs.

Chapter 3

AT&T's Baseline ATIS Recognizer

Throughout this dissertation, our novel *subspace distribution clustering hidden Markov model* (SDCHMM) is evaluated on the ARPA-ATIS task. The task is chosen for two reasons: Firstly, it represents a commercially viable application for automatic speech recognition technology in the near future; secondly, and most importantly, we were able to employ AT&T Labs' state-of-the-art ATIS recognizer. Clearly, it can be trivial to introduce improvements on a mediocre recognizer. On the other hand, if we can show significant enhancements to a highly fine-tuned and accurate recognizer, there is good reason to see SDCHMM as a promising alternative to the current acoustic modeling techniques.

In this chapter, we describe in details the ATIS task, various components of AT&T's baseline ATIS recognizer with emphasis on its acoustic models, as well as its benchmark performance against which our SDCHMM performance will be gauged in the later chapters.

3.1 The ATIS task

The Air Travel Information System (ATIS) task was initiated in 1989 in response to the call from the ARPA Spoken Language Systems Program for development of speech recognition and natural language research, using spontaneous and goal-directed speech rather than read speech. An ATIS system allows users to speak naturally to inquire about air travel information stored as a relational database which is derived from the Official Airline Guide.

3.1.1   ATIS Corpora

Several ATIS corpora[1] known as ATIS0 [23], ATIS1[2], ATIS2 [25], and ATIS3 [13] were collected over the years by several institutions: AT&T, BBN, CMU, MIT, NIST, SRI, and TI. To date, the corpora contain nearly 25,000 utterances with a vocabulary size of 1,536 words. The query database includes information on 23,457 air flights for 46 cities and 52 airports in the United States and Canada. A set of 981 utterances were set aside for the 1994 ARPA ATIS evaluation. This official test set consists of 131 different subject-scenario combinations spoken by 24 different subjects. Below are some example utterances from the test set:

``I would like to find a flight from Charlotte to Las Vegas that makes a stop in Saint Louis.''

``I would like to return from Chicago around seven p m to Kansas City.''

``What are the prices for the flights on Wednesday evening.''

``Please show which flight serves dinner.''

``Show me all the morning flights from Philadelphia to Fort Worth.''

``What does fare code M mean?''

3.1.2   ATIS Evaluation Tests

There are three types of official evaluations [66]:

SPREC: SPontaneous speech RECognition evaluation tests only the capability of the speech recognizer component in terms of word error rate (WER). That is, for each test utterance, the decoded string from a recognizer is compared (by a string alignment software using dynamic programming algorithm [26]) with the known orthographic transcription of the utterance, and the number of substituted words, deleted words, as well as inserted words are counted. The WER is computed as,

---

[1]The ATIS corpora are now maintained and distributed by the Linguistic Data Consortium. Consult its website at http://www.ldc.upenn.edu/ldc/catalog/html/speech_html/atis.html for more information about the corpora and how to obtain them.

[2]ATIS1 is, however, never published.

$$\text{WER} = \%\text{substitutions} + \%\text{deletions} + \%\text{insertions}.$$

NL: <u>N</u>atural <u>L</u>anguage understanding evaluation tests only the understanding component of a recognition system given the textual transcription of an utterance, and is measured by the correctly answered queries.

SLS: <u>S</u>poken <u>L</u>anguage under<u>S</u>tanding evaluation tests the performance of the whole system (*both* the speech recognizer and the natural language understanding modules).

3.2   The Baseline Recognizer

In the 1994 ARPA–ATIS evaluation [66], AT&T's ATIS System had the best NL performance, answering 94.1% of the queries correctly, and was among the three second best systems in the SPREC test with a WER of 3.5%. Since this thesis deals only with acoustic modeling, efficacy of which is best measured by speech recognition alone (i.e. the SPREC test), we restrict ourselves here to describe only its speech-recognition component [10]. Refer [52] for its NL understanding component. In addition, to allow faster research turnaround time, we adopt in this thesis a baseline system configuration which, in terms of computation time and memory requirement, is less costly but more realistic than the 1994 evaluation system's.

3.2.1   Signal Processing

The recognizer frontend is based on mel-frequency cepstral analysis of input speech sampled at 16kHz. The DC bias is removed and the speech is pre-emphasized. At a frame rate of 100Hz, 31 mel-frequency energy components are computed from a filter bank by performing an FFT on a *frame* of 20ms of speech. The energies are converted to 12 mel-frequency cepstral coefficients (MFCCs) by cosine transform. *Cepstral mean subtraction* is then performed using the average MFCCs per utterance. Finally a speech feature vector for one frame is composed from 39 components: 12 MFCCs and normalized power, and their first- and second-order time derivatives computed as follows:

$$x[t] \;\; = \;\; \text{normalized MFCC or power}$$

$$\Delta x[t] \;=\; 2x[t+2] + x[t+1] - x[t-1] - 2x[t-2]$$

$$\Delta\Delta x[t] \;=\; \Delta x[t+1] - \Delta x[t-1] \; .$$

3.2.2 Lexicon

ATIS is a medium-vocabulary speech recognition task. There are 1,536 words in the lexicon with one pronunciation for each word. A set of 45 phones as shown in Table 3.1 is used in the lexicon.

Table 3.1: ATIS Phones (Phone-like Units)

PHONE	EXAMPLE	PHONE	EXAMPLE	PHONE	EXAMPLE
aa	father	ae	bat	ah	above
ao	bought	aw	now	ax	the
axr	diner	ay	bye	b	ban
ch	church	d	dad	dh	they
dx	rider	eh	bet	el	bottle
en	button	er	bird	ey	fake
f	four	g	gag	hh	hay
ih	bit	ix	roses	iy	beat
jh	judge	k	kick	l	lot
m	man	n	not	ng	ping
nx	any	ow	boat	oy	boy
p	pan	r	red	s	sad
sh*	she	t	tell	th	thief
uh	book	uw	boot	v	very
w	wet	y	yet	z	zoo

(* The phone "zh" as in "measure" is folded into the phone "sh".)

Lexical Structure

An important factor in decoding efficiency is the organization of lexicon, or the lexical structure. When all the words (or phrases) are independently represented by a sequence of phonemes, the lexical structure is said to be *linear*. Recently, a more compact structure, the lexical (phonetic) tree [1, 59, 73] has been found to greatly improve search speed in large vocabulary speech recognition. In the lexical tree, words in the lexicon are arranged

in a tree structure so that those with the same prefix (in their phonetic pronunciations) share the same part of the tree. Not only does the lexical tree greatly compress the lexicon, but it also avoids evaluating the same phonetic models twice.

With the 1,536 words in our lexicon, there are only 1,502 distinct pronunciations. Counts of common prefixes of different lengths are shown in Table 3.2. From the table, the ratio of the number of root HMMs to the total number of words is 2.9% for a CI tree and 25.6% for a CD tree. Thus, the use of lexical tree will be more effective in a CI system than a CD system. As a result, the baseline CI recognizer employs a lexical tree and the baseline CD recognizer uses a linear lexicon (see also Section 3.2.6).

Table 3.2: ATIS: Count of common prefixes of different lengths among the words in the lexicon

PREFIX LENGTH	COUNT
1	43
2	385
3	900
4	1185
5	1330
6	1403
7	1453
8	1482
9	1490
10	1495
11	1500
12	1501
13	1502
14	1502

3.2.3   Acoustic Modeling

Both context-independent (CI) and context-dependent (CD) phone modeling are explored. The base phones include the 45 phones in Table 3.1 and three noise models. The CI system models each base phone, while the CD system models intra-word triphones, word-beginning right-context biphones, word-ending left-context biphones, and backs off to

---

Algorithm 1: Baseline CDHMM training via segmental k-means algorithm

---

Goal: To train CDHMM acoustic models using mixture Gaussian densities; each density has a maximum of M components with diagonal covariances.

Step 1. Segment all training data into HMM states by supervised Viterbi algorithm.

Step 2. For triphone modeling only, tie the corresponding states of triphones of the same base phone using the *ensemble merging algorithm* (as described in Algorithm 2).

Step 3. Estimate the mixture Gaussian density of each state by k-means clustering using all the training data alloted to the (possibly tied) state. The maximum number of mixture components is fixed to M, and the mixture weights are set to the proportion of data frames assigned to a component.

Step 4. Repeat Step 1 – 3 until the models converge.

---

the appropriate biphones or monophones when there is not enough training data for the context (fewer than 1,000 speech frames in this implementation). Each phone model is a 3-state left-to-right continuous-density hidden Markov model (CDHMM), as shown in Figure 2.1, with the exception of one noise model which has only one state[3]. State observation distributions are estimated as continuous mixture Gaussian densities with diagonal covariances.

The CDHMM acoustic models are trained via the *segmental k-means training algorithm* [38] as described in Algorithm 1. The average frame likelihood of the training utterances is computed to test for model convergence. All acoustic models are speaker-independent and gender-independent[4].

3.2.4 Ensemble Merging Algorithm for State Tying

Triphone states are tied by the *ensemble merging algorithm* [9] shown in Algorithm 2. It is a bottom-up agglomerative clustering algorithm in which two states are tied if the tying results in minimum increase in total ensemble distortion. The distortion $\mathcal{D}_i$ of a

---

[3]In an attempt to model noises of different characteristics, three noise models of different complexities are used. The 1-state noise model is meant to model short noises, while the other two 3-state noise models are to capture longer noises.

[4]In ARPA's 1994 ATIS evaluation, AT&T's system interpolated gender-dependent acoustic models with gender-independent acoustic models.

Algorithm 2: Ensemble merging algorithm for state tying

Goal: To tie a set of HMM states using an Euclidean distortion measure until each tied state has at least N feature vectors.

Distortion Measure: The distortion of an ensemble of vectors is defined as the sum of Euclidean distances between each vector and the ensemble mean vector. Each component of a vector is normalized to unit variance before distortion computation.

Step 1. Select the HMM state S_i, the ensemble containing the fewest feature vectors. If the ensemble has more than N feature vectors, stop.

Step 2. Find another state S_j, which when merged with the state S_i, will give the smallest increase in total distortion.

Step 3. Merge the two states, S_i and S_j.

Step 4. Repeat Step 1 – 3.

Gaussian ensemble G_i with n_i D-dimensional feature vectors $\boldsymbol{x}$ of mean $\boldsymbol{\mu}_i$ and variance $\boldsymbol{\sigma}_i^2$ is defined as

$$
\mathcal{D}_i \;\overset{\text{def}}{=}\; \sum_{\boldsymbol{x} \in G_i} |\boldsymbol{x} - \boldsymbol{\mu}_i|^2 \tag{3.1}
$$

$$
= \; n_i \sum_{f=1}^{D} \sigma_{i,f}^2. \tag{3.2}
$$

When two ensembles G_i and G_j are merged, the combined variance $\sigma_f'^2$ of each dimension f becomes

$$
\sigma_f'^2 \;=\; \frac{n_i(\sigma_{i,f}^2 + \mu_{i,f}^2) + n_j(\sigma_{j,f}^2 + \mu_{j,f}^2)}{n_i + n_j} - \left(\frac{n_i \mu_{i,f} + n_j \mu_{j,f}}{n_i + n_j} \right)^2, \tag{3.3}
$$

and the distortion increase is given by

$$
\Delta \mathcal{D} \;=\; \frac{n_i n_j}{n_i + n_j} \sum_{f=1}^{D} (\mu_{i,f} - \mu_{j,f})^2. \tag{3.4}
$$

Equation (3.4) provides a more efficient implementation of the algorithm than a direct distortion computation using Equation (3.3). It also shows that the Euclidean distortion measure depends only on the first-order statistics of the ensemble distributions.

In addition, to avoid an otherwise $O(n^3)$ complexity — $O(n^2)$ to compute the distortion increase between any two Gaussians in each iteration and $O(n)$ iterations when a small number of prototypes are required — Algorithm 2 introduces the heuristic Step–1: At each iteration, the Gaussian corresponding to the smallest training ensemble must be merged. As a result, the complexity is reduced to $O(n^2)$.

3.2.5   Language Modeling

In the domain-specific ATIS corpus, sequences of coherent words which are similar in semantics or syntax occur frequently. Two sorts of such word sequences are identified: *word classes*, and *compound words* or *verbal forms*. For example, a word class can be the set of flight numbers, the set of airport names, the set of arrival times, etc.; whereas phrases like "thank you", "I'd like to", or "how much" are three different compound words or verbal forms. The AT&T ATIS recognition system manually defines 13 word classes and about 100 compound words or verbal forms. Bigrams of these word sequences are modeled[5] using a second-order Variable N-gram Stochastic Automaton (VNSA) [75]. Modeling word classes instead of their individual words has an additional benefit of enhanced robustness due to more training examples.

An example of a simple bigram language model of three tokens "a", "b", and "c", and its VNSA implementation are shown in Figure 3.1. An ϵ-transition in a VNSA serves two purposes:

- It reduces the word history during a traversal of the network and backs off to a language model of lower complexity. For instance, in Figure 3.1(a), the bigram $P(b|a)$ does not exist. The VNSA in Figure 3.1(b) approximates $P(b|a)$ with the unigram $P(b)$ weighted by η_a via the path $a \to \epsilon \to b$. i.e. $P(b|a) = \eta_a \cdot P(b)$.

- It introduces non-determinism to the decoding procedure. For instance, the string "cba" can be decoded as "cba", "cϵba", "cbϵa", or "ϵcϵbϵa", etc. The string with the highest probability is chosen.

---

[5]In ARPA's 1994 ATIS evaluation, AT&T's system employed trigrams of word sequences realized by a third-order VNSA [74].
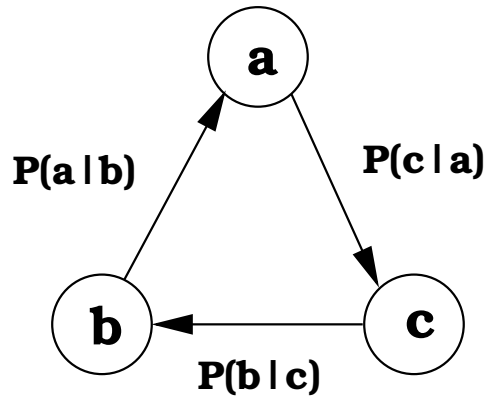
(a) Bigram language model
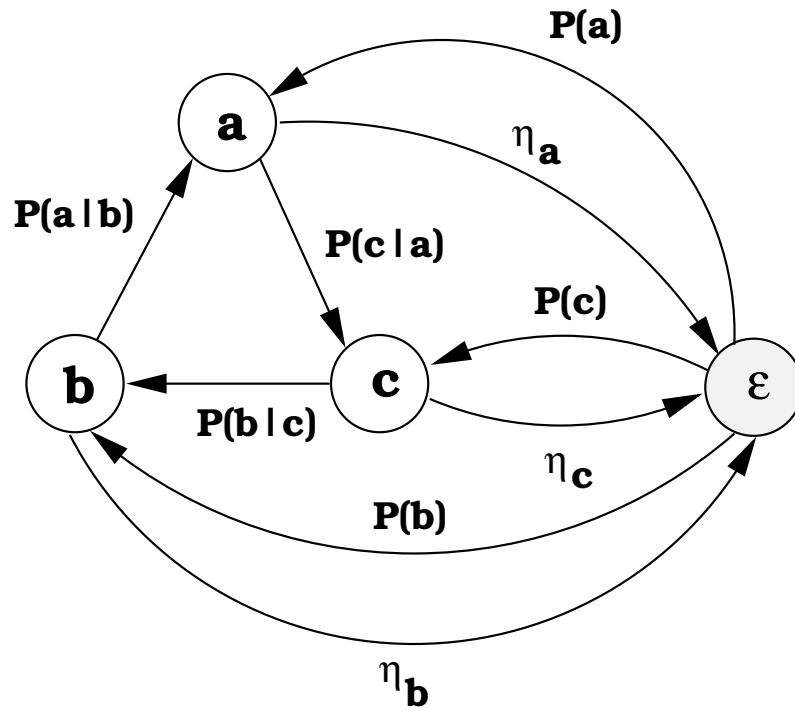
(b) VNSA implementation

Figure 3.1: An example of a (second-order) VNSA implementation of a simple bigram language model

28

The values of the η's are estimated by minimizing the perplexity of the language model using a separate held-out dataset [75].

3.2.6   Decoding

Viterbi beam search [55, 56, 60] is used. Viterbi search is a dynamic programming algorithm [26] which finds the most likely sequence of states of HMMs for a given sequence of observations. To reduce the otherwise immense search space, Viterbi beam search prunes those states with log likelihoods less than that of the best path (at that moment) by a preset threshold called *beam-width*. It has been found that the beam-width can greatly be reduced with no loss in recognition accuracy [7]; further decrease in the beam-width will trade off accuracy for speed and memory.

Further computation efficiency is obtained by evaluating the state likelihood due only to the most likely mixture component of the state (see also Section 4.3.2). It has been verified empirically that the technique does not result in any loss of recognition accuracy [5, 16, 59, 79].

As in common practice, our decoder also ensures no Gaussian, state, or model likelihoods are evaluated twice for a given frame of speech. (This factor, together with the high ratio of number of root HMMs to number of words in the lexicon (25.6%) for the CD lexical tree, render the use of lexical tree for the CD system unnecessary.)

3.3   Baseline Performance

All benchmark tests are run on ARPA's official 1994 ATIS evaluation test set. The baseline testing conditions and performance for both the CI and the CD systems are summarized in Table 3.3. Notice that all the triphones appearing in the ATIS corpora are modeled in the CD system. Due to the scarcity of training data for some triphones, the CD models are trained with an addition of 8,000 WSJ utterances[6].

The CI system is run on a low-end SGI machine comparable to a 166MHz Pentium

---

[6]The WSJ corpus consists primarily of read speech with texts drawn from a machine-readable corpus of Wall Street Journal news text [67].

PC. The CD system is run on a high-end SGI machine which is about three times the speed of the low-end machine and is comparable to a 200MHz Pentium Pro PC. The machines are chosen to reflect a realistic performance on currently available desktop PCs. Clearly the performance due to one specific choice of beam-width in Table 3.3 can be misleading. Figure 3.2 shows a more complete picture of the performance when the beam-width varies from 70–170 in the CI system and 110–190 in the CD system. The baseline operating points are chosen to be close to the asymptotic accuracies attained at reasonable recognition time.

Table 3.3: ATIS: Testing conditions and performance of the baseline CI/CD systems

CONDITION/PERFORMANCE	CI SYSTEM	CD SYSTEM
#Test Sentences	981 (1994 ARPA-ATIS evaluation set)	
Vocabulary	1,536 words	
Language Model	word-sequence bigram (perplexity $\approx$ 20)	
#Training Utterances	~12,000 ATIS	~20,000 ATIS + ~8,000 WSJ
#HMMs	48	9,769
#States	142	3,916 (tied)
Max. #Mixtures per State	16	20
#Gaussians (39-dimensional)	2,254	76,154
#Acoustic Parameters	178,066	6,016,166
Search	one-pass Viterbi beam search	
Lexical Structure	lexical tree	linear lexicon
Beam-Width	100	170
CPU	150MHz MIPS R4400	195MHZ MIPS R10000
Word Error Rate	9.4%	5.2%
Time (x real-time)	1.93	7.06
HMM Memory Usage	0.71MB	24MB
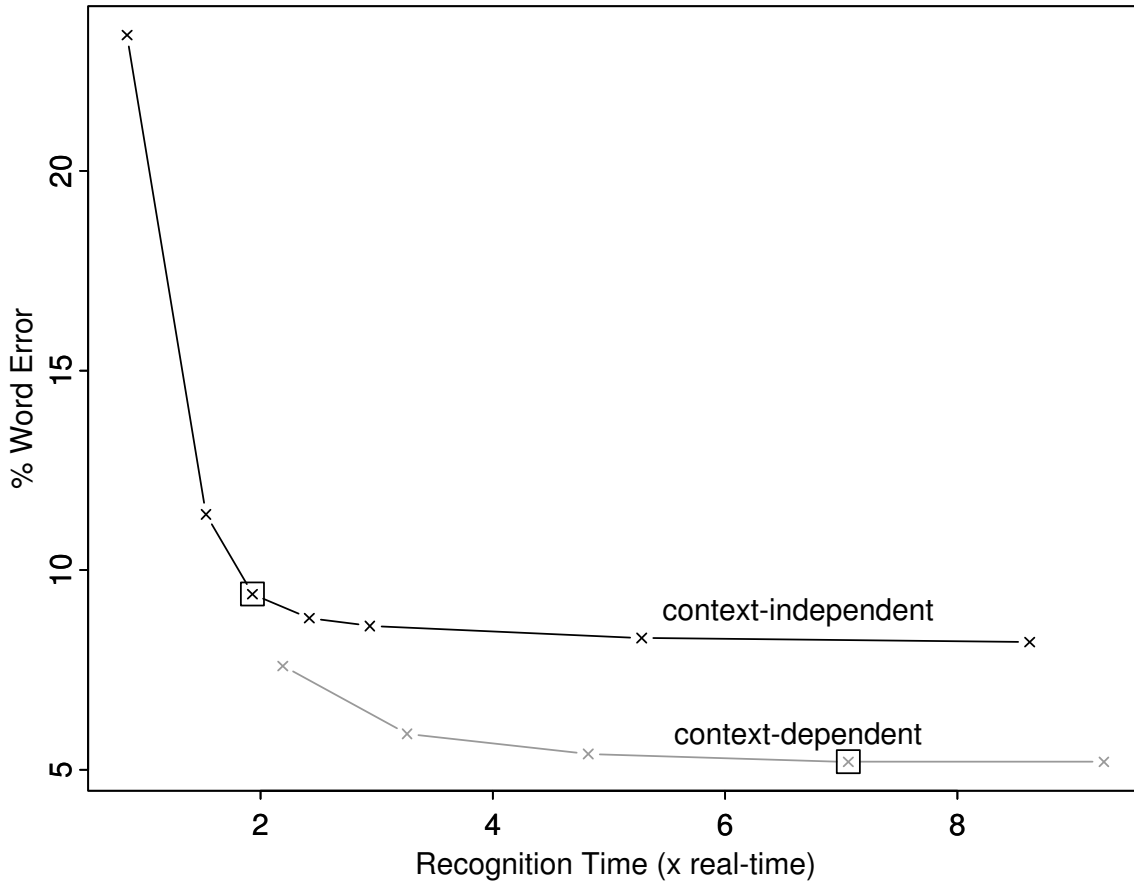
Figure 3.2: (the baseline performance of Table 3.3 are marked with squares)

It can be seen that even with this medium-vocabulary domain-specific task, the more accurate CD system is still about an order of magnitude away from real-time performance on a high-end PC. Execution profiles of the systems show that 50–70% of the total run-time is spent on computing state likelihoods of the acoustic models. The acoustic models alone consume more than 24MB of memory; and when combined with the language models and the software code, the whole recognition system can easily exceed 40MB of memory. Tasks with larger vocabularies will require more complex acoustic and language models and thus even more memory. Therefore, a large reduction in the memory size of the acoustic models is desirable. Furthermore, the baseline systems require a large amount of training data: The CI and the CD systems are trained with 18 and 36 hours of speech

respectively. One major cause for all these (computation, memory, and training) costs is the large set of model parameters, the reduction of which is the main theme of this thesis.

Chapter 4

Subspace Distribution Clustering Hidden Markov Model (SDCHMM)

We pointed out in Chapter 1 that many of the state-of-the-art speech recognizers are too big (consuming a lot of memory) and too slow to run in desktop personal computers that most people can afford. We attribute part of the problem to their complex acoustic models, encompassing millions of model parameters. To reduce the large number of model parameters without compromising recognition performance, we apply the proven technology of parameter tying.

4.1 Theory of SDCHMM

The theory of SDCHMM is derived from that of the continuous density hidden Markov model (CDHMM). Let us first consider a set of CDHMMs (possibly with tied states) in which state-observation distributions are estimated as mixture Gaussian densities with M components and diagonal covariances. Later we will extend the theory by considering CDHMMs with other types of mixture densities.

Using the following notations (where, as usual, bold-faced quantities represent vectors):

O : an observation vector of dimension D

$P(O)$: state output probability given observation O

           (with subscripts of CDHMM or SDCHMM when the context requires clarity)

c_m : weight of the m-th mixture component

$\boldsymbol{\mu}_m$: mean vector of the m-th mixture component

$\boldsymbol{\sigma}_m^2$: variance vector of the m-th mixture component

$\mathcal{N}(\cdot)$: Gaussian probability density function (pdf)

the state observation probability is given by

$$P_{CDHMM}(\boldsymbol{O}) \;=\; \sum_{m=1}^{M} c_m \; \mathcal{N}(\boldsymbol{O}; \boldsymbol{\mu}_m, \boldsymbol{\sigma}_m^2), \quad \sum_{m=1}^{M} c_m = 1. \tag{4.1}$$

The key observation is that a Gaussian with diagonal covariance can be expressed as a product of subspace Gaussians where the subspaces (or streams) are orthogonal and together span the original full feature vector space. Formally, let us denote the full vector space of dimension D by $\mathcal{R}^D$ with an orthonormal basis, which are composed of the column vectors of the $D \times D$ identity matrix. $\mathcal{R}^D$ is decomposed into K orthogonal subspaces $\mathcal{R}^{d_k}$ of dimension d_k, $1 \leq k \leq K$, with the following conditions:

Condition 1:

$$\sum_{k=1}^{K} d_k = D \tag{4.2}$$

Condition 2:

$$\mathcal{R}^{d_i} \cap \mathcal{R}^{d_j} = \emptyset, \;\; 1 \leq i \neq j \leq K. \tag{4.3}$$

Condition 3: The basis of each subspace is composed of a subset of the basis vectors of the full vector space.

Each of the original full-space Gaussians is projected onto each of the K subspaces to obtain K subspace Gaussians of dimension d_k, $1 \leq k \leq K$, with diagonal covariances. That is, Equation (4.1) can be rewritten as

$$P_{CDHMM}(\boldsymbol{O}) \;=\; \sum_{m=1}^{M} c_m \left(\prod_{k=1}^{K} \mathcal{N}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}_{mk}^2) \right) \tag{4.4}$$

where

$\boldsymbol{O}_k$: projection of observation $\boldsymbol{O}$ onto the k-th subspace

$\boldsymbol{\mu}_{mk}$: projection of mean vector of the m-th component onto the k-th subspace

$\boldsymbol{\sigma}_{mk}^2$: projection of variance vector of the m-th component onto the k-th subspace

For each stream, we treat the subspace Gaussians as the basic modeling unit, and tie them across *all* states of *all* CDHMM acoustic models. Hence, the state observation probability in Equation (4.4) is modified as

$$P_{SDCHMM}(\boldsymbol{O}) \;=\; \sum_{m=1}^{M} c_m \left(\prod_{k=1}^{K} \mathcal{N}^{tied}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}_{mk}^2) \right). \tag{4.5}$$

The ensuing HMM will be called the *subspace distribution clustering hidden Markov model* (SDCHMM). Figure 4.1 shows an extension of the various HMM tying schemes of Figure 2.3 to include SDCHMMs. There are 4 streams in the example.

4.1.1 Generalization

The foregoing SDCHMM formulation can be generalized to any mixture density insofar as the component pdf $\mathcal{F}(\boldsymbol{O})$ can be expressed as a product of subspace pdf's of the same functional form. That is,

$$\mathcal{F}(\boldsymbol{O}) = \prod_{k=1}^{K} \mathcal{F}(\boldsymbol{O}_k) \;, \tag{4.6}$$

provided that the three conditions on the subspaces mentioned above are satisfied.

An obvious candidate for this functional is a Gaussian pdf with block-diagonal covariance. A matrix, A, is said to be in block-diagonal form if A can be partitioned into square submatrices such that all non-diagonal square submatrices are null (or zero). That is, a block-diagonal matrix A has the form of

$$A = \begin{pmatrix} A_{11} & (0) & \cdots & \cdots & (0) \\ (0) & A_{22} & \cdots & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (0) & (0) & \cdots & A_{kk} & (0) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (0) & (0) & \cdots & \cdots & A_{KK} \end{pmatrix} \tag{4.7}$$

where $A_{kk}, \; 1 \le k \le K$, are square matrices.

A block-diagonal matrix has the following two useful properties in our context:

phonetic
units

acoustic
HMM
pool

state
pool

Gaussian
pool

stream #1: subspace Gaussian pool

stream #2: subspace Gaussian pool

stream #3: subspace Gaussian pool

stream #4: subspace Gaussian pool
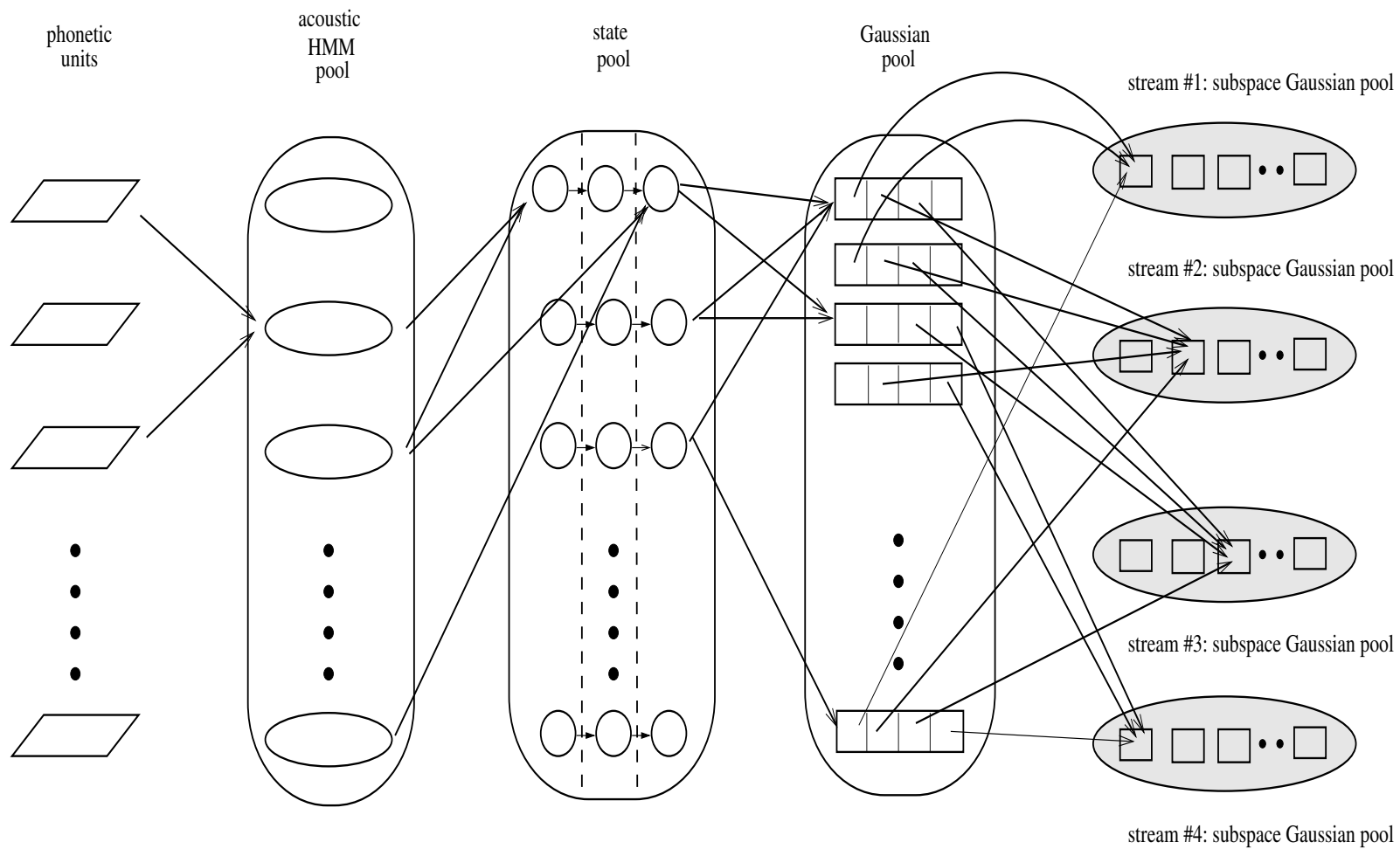
Figure 4.1: Subspace distribution clustering hidden Markov models with 4 streams

Property 1: The determinant of a block-diagonal matrix is equal to the product of the determinants of its diagonal submatrices. That is,

$$|A| = \prod_{k=1}^{K} |A_{kk}|. \tag{4.8}$$

Property 2: The inverse of a block-diagonal matrix is another block-diagonal matrix where the constituent diagonal submatrices are the inverses of the original diagonal submatrices. That is,

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} & (0) & \cdots & \cdots & (0) \\ (0) & A_{22}^{-1} & \cdots & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (0) & (0) & \cdots & A_{kk}^{-1} & (0) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ (0) & (0) & \cdots & \cdots & A_{KK}^{-1} \end{pmatrix}. \tag{4.9}$$

Thus a Gaussian with block-diagonal covariance can also be expressed as a product of subspace Gaussians with *full* covariance as

$$\mathcal{N}(\boldsymbol{O}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\Sigma}_{mk}), \tag{4.10}$$

where $\boldsymbol{\Sigma}_m$ is the block-diagonal covariance matrix of the full-space Gaussian, and $\boldsymbol{\Sigma}_{mk}$ is the full covariance matrix of the k-th stream subspace Gaussian. The rest of SDCHMM theory then applies as before.

While other pdf functionals or Gaussians with block-diagonal covariances appear intriguing, they have not been widely studied (except, e.g., [89]) in automatic speech recognition. To keep our focus on the main issue of SDCHMM in this thesis, we investigate only SDCHMMs based on CDHMMs with mixture Gaussian densities and diagonal covariances.

4.2   Distribution Clustering

In practice, the proposed SDCHMM as in Equation (4.5) can be obtained by clustering or quantizing the subspace Gaussians of CDHMMs in each stream. That is, to derive

K-stream SDCHMMs from a set of CDHMMs in which there are originally a total of N full-space Gaussian distributions, the subspace Gaussians in each stream are clustered into a small set of L subspace Gaussian prototypes (or codewords)

$$\mathcal{N}^{quantized}(\boldsymbol{O}_k; \boldsymbol{\mu}_{lk}, \boldsymbol{\sigma}_{lk}^2), \quad 1 \leq l \leq L, \quad 1 \leq k \leq K$$

where $L \ll N$. Each original subspace Gaussian is then "approximated" by its nearest subspace Gaussian prototype

$$\mathcal{N}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}_{mk}^2) \approx \mathcal{N}^{quantized}(\boldsymbol{O}_k; \boldsymbol{\mu}_{lk}, \boldsymbol{\sigma}_{lk}^2)$$

with l being given by

$$l = \operatorname*{argmin}_{1 \leq q \leq L} dist\left(\mathcal{N}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}_{mk}^2), \ \mathcal{N}^{quantized}(\boldsymbol{O}_k; \boldsymbol{\mu}_{qk}, \boldsymbol{\sigma}_{qk}^2)\right)$$

where $dist(\cdot)$ measures the distance between two Gaussian distributions.

In this respect, SDCHMMs can be considered as an approximation to the conventional CDHMMs. Since it has been proved by years of research that CDHMM is a good model for speech recognition, a carefully-designed approximation to the CDHMM formulation — SDCHMM — should, in principle, also deliver high performance.

In general, since quantization in lower dimensions results in smaller quantization error (see Appendix A for a formal proof), more streams should be adopted in SDCHMMs in order to maintain the performance of their parent CDHMMs.

Two distribution clustering algorithms are discussed in length and evaluated in Chapter 5.

4.3   Why Are SDCHMMs Good?

If the subspace distributions are properly clustered, all original full-space distributions can be represented by some combinations of a small number of subspace distribution prototypes with small quantization errors. The combinatorial effect of subspace distribution encoding can be very powerful: For instance, a 20-stream SDCHMM system with as few

as 2 subspace distribution prototypes per stream can represent $2^{20} = 1,048,576$ different full-space distributions. Of course, in reality, more prototypes are required to ensure small quantization errors. This can be achieved with more streams or more prototypes per stream.

4.3.1 Savings in Model Parameters and Memory

Table 4.1 computes the number of model parameters in discrete HMMs (DHMMs), semi-continuous HMMs (SCHMMs), CDHMMs and SDCHMMs. Each stream of DHMMs is described by discrete VQ prototypes and state observation histograms. SCHMMs are similar to DHMMs except that the prototypes are now continuous Gaussians. CDHMMs parameters comprise the mixture weights and the means and variances of the mixture Gaussian densities. SDCHMMs parameters consist of mixture weights, continuous subspace Gaussian prototypes for each stream, and the encoding indices (or pointers) between the original full-space Gaussians and the subspace Gaussian prototypes. For large systems, the number of HMM states S is relatively large (typically several thousands), and any terms involving S dominate the sums in the first row of Table 4.1. The second row of the table gives the approximated number of model parameters in such large systems. Hence, in terms of the number of model parameters, DHMMs are dominated by their state observation histograms, SCHMMs by their mixture weights, CDHMMs by their Gaussians and SDCHMMs by their subspace Gaussian encoding indices.

Table 4.1: Number of model parameters in various types of HMM (S = #states, M = #mixtures per state, L = #codewords per stream, K = #streams and D = feature dimension)

DISCRETE HMM	SEMI-CONTINUOUS HMM	CONTINUOUS DENSITY HMM	SUBSPACE DISTRIBUTION CLUSTERING HMM
SLK+LD	SLK+L(2D)	SM+SM(2D)	SM+L(2D)+SMK
$\approx$SLK	$\approx$SLK	$\approx$SM(2D)	$\approx$SMK

For DHMMs and SCHMMs, the number of streams is usually 3 or 4 [28, 50], and the number of prototypes is about 256–1024; while CDHMMs typically have 20–30 mixture

components per state density. For example, in large system (with large value of S), if $M = 20, L = 256, K = 4$, and $D = 39$, the number of model parameters in DHMMs, SCHMMs, and CDHMMs will be $1024S, 1024S$, and $1560S$ respectively. Thus CDHMMs generally require more memory than DHMMs and SCHMMs. Since most of the memory consumed by CDHMMs is used to store the Gaussian parameters, if SDCHMMs can approximate CDHMMs with few subspace Gaussian prototypes, substantial memory savings can be achieved. For example, using the same figures for M and K, the number of parameters in SDCHMMs will be $80S$. However, as will be seen in the next chapter, more streams are required in SDCHMMs to obtain similar performance to CDHMMs, with a typical value of 20 streams. Even with 20 streams, the SDCHMMs will have only $400S$ model parameters — less than half of those of DHMMs and SCHMMs.

Notice that there is a trade-off between memory savings and performance of SDCH-MMs. To reduce memory requirement (which is mostly used to store the subspace Gaussian encoding indices), fewer streams are desirable. However, fewer streams mean higher stream dimensionality which means greater quantization errors and poorer performance for a given number of prototypes (see Appendix A).

4.3.2 Savings in Computation

As mentioned in Section 3.2.6, the state likelihood can be approximated by the likelihood of the most likely mixture component of the state with no loss of overall recognition accuracy. That is, the summation operator of Equation (4.5) is replaced by the max operator as

$$P(\boldsymbol{O}) = \max_{1 \leq m \leq M} \left(c_m \prod_{k=1}^{K} \mathcal{N}^{tied}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}^2_{mk}) \right), \tag{4.11}$$

which becomes

$$\log(P(\boldsymbol{O})) = \max_{1 \leq m \leq M} \left(\log(c_m) + \sum_{k=1}^{K} \log \left(\mathcal{N}^{tied}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}^2_{mk}) \right) \right). \tag{4.12}$$

in the logarithmic domain.

The computational efficiency of SDCHMMs comes from the fact that, since a small number of the subspace Gaussians are shared by a large number of full-space Gaussian

components, all these subspace Gaussian log likelihoods can be pre-computed once and only once at the beginning of every frame, and their values are stored in lookup tables. During Viterbi decoding [86], the state log likelihood computation of Equation (4.12) is reduced to a summation of K pre-computed subspace Gaussian log likelihoods and the mixture weight for each component of the state. Obviously, the wider the beam-width, the greater the savings in state likelihood computation (as there will be more active states).

Again, there is a trade-off between computational savings and the performance of SDCHMMs. To increase the likelihood computation efficiency, fewer streams are desirable as fewer additions are then needed to compute the full-space state likelihoods. However, fewer streams will lead to poorer performance unless more prototypes are used.

4.4 Comparison with Semi-Continuous HMM

At first glance, SDCHMMs may appear similar to SCHMMs [4, 29, 80]: Both methods divide the feature space into streams, and tie subspace distributions across all states of all HMMs. However, close scrutiny shows that K-stream SCHMMs compute the state likelihood differently as

$$P_{SCHMM}(\boldsymbol{O}) = \prod_{k=1}^{K} \left(\sum_{m=1}^{M} c_{mk} \, \mathcal{N}^{tied}(\boldsymbol{O}_k; \boldsymbol{\mu}_{mk}, \boldsymbol{\sigma}^2_{mk}) \right), \quad \sum_{m=1}^{M} c_{mk} = 1. \qquad (4.13)$$

where c_{mk} is the weight of the m-th mixture component in the k-th stream.

Comparing Equation (4.13) with Equation (4.5), one finds two differences:

- There is a switch between the product operator $(\prod)$ and summation operator $(\sum)$ in the two equations.

- In an SCHMM state, each of the K subspace Gaussians is associated with its own mixture weight c_{mk}, whereas one mixture weight c_m is shared among all the K subspace Gaussians of a SDCHMM state.

Both differences arise from the fact that SCHMMs assume stream independence in the *global* feature space, whereas SDCHMMs assume stream independence in the *local* feature space — an assumption inherited from CDHMMs with mixture Gaussian densities and

diagonal covariances. That is, for each state, SCHMMs estimate one mixture Gaussian density from each of the streams *independently*, and then combine the subspace Gaussian likelihoods by assuming again independent streams. However, the assumption of feature independence between the streams commonly used in speech recognition is hardly justified. SDCHMMs therefore start with CDHMMs using the full feature speech vectors without assuming any feature independence. The correlation between features at each state is well modeled by a mixture Gaussian density. The implicit assumption of stream independence in the local feature space results directly from using Gaussians with diagonal covariances in the pdf estimation process. Theoretically, a mixture Gaussian density with diagonal covariances may model *any* distribution, should there be sufficient Gaussians and ample training data. An implication of the difference in the scope of the assumptions is the number of streams required: The SCHMM favors fewer streams of higher dimensions, so that correlation among more features can be modeled and there will be fewer mixture weights; on the contrary, SDCHMM favors more streams of lower dimensions so that quantization of the subspace Gaussians of CDHMMs will give smaller quantization errors and more accurate models (Table 4.1).

We will consequently test the following hypothesis in Chapter 5: By deriving our SDCHMMs from the more accurate CDHMMs with the less stringent assumption of stream independence in the local feature space, the SDCHMMs may be equally accurate even with fewer model parameters.

Another difference between SDCHMM and SCHMM not readily observed from Equations (4.5) and (4.13) is that SCHMM requires each state to have the same number of mixture components equal to the number of distribution prototypes while SDCHMM does not. As a result, SDCHMM usually has many fewer mixture components per state, and thus has the following advantages over SCHMM:

- Fewer components mean fewer mixture weights which then take less memory space.

- Fewer components are involved in state likelihood computation which then takes less CPU time.

4.5   Comparison with Feature-Parameter-Tying HMM

The feature-parameter-tying HMM (FPTHMM) [84, 85] turns out to be a special case of our SDCHMM when the number of streams is set to the size of the feature vector; i.e. $K = D$. In a sense, the FPTHMM is the scalar quantization (SQ) version of our SDCHMM. In Section 4.3.1, it is shown that the main storage cost of SDCHMMs is incurred by the subspace Gaussian encoding indices which grow in proportion with the number of streams. Similarly Equation (4.12) indicates that the computation cost of the state likelihood is directly proportional to the number of streams once all subspace Gaussian likelihoods are pre-computed. Thus, although SQ of the subspace Gaussians in FPTHMMs has the advantage of simplicity and generally requiring fewer prototypes, it will need more storage space and more computation time than SDCHMMs with $K < D$. The difference will be more conspicuous for large systems.

The definition of streams, the number of streams, and the number of prototypes, all affect the system performance of automatic speech recognition as measured in terms of memory size, computation time, and recognition accuracy. Their interactions will be investigated in the next chapter where a practical implementation of SDCHMMs from CDHMMs is proposed and evaluated on the ATIS recognition task.

Chapter 5

Implementation of SDCHMMs (I): Model Conversion from Continuous Density HMMs (CDHMMs)

The formulation of the subspace distribution clustering hidden Markov model (SDCHMM) as of Equation (4.5) of Chapter 4 suggests that SDCHMMs may be implemented in the following two steps:

1. Train continuous density hidden Markov models (CDHMMs) for all the phonetic units (possibly with tied states), wherein state observation distributions are estimated as mixture Gaussian densities with diagonal covariances.

2. Convert the CDHMMs to SDCHMMs by tying the subspace Gaussians in each stream as shown in Figure 5.1.

Since the training of CDHMMs is well covered in the literature [31, 69], we will not repeat it here. Instead, when we discuss the reestimation of SDCHMM parameters in Chapter 7, we will review the reestimation of CDHMM parameters as well. In this chapter, we assume that a set of (well-trained) CDHMMs is given, and we focus only on the conversion of the CDHMMs to SDCHMMs.

Tying of subspace Gaussians consists of splitting the full speech feature vector space into disjoint subspaces, projecting mixture Gaussians of CDHMMs onto these subspaces, and then clustering the subspace Gaussians into a small number of Gaussian prototypes in each subspace. In the following, we investigate various streams definitions and distribution

43

44
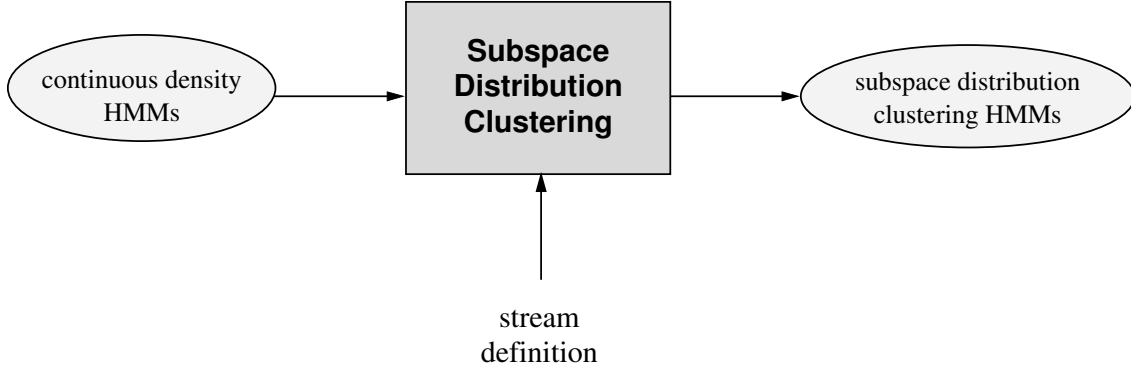
Figure 5.1: Conversion of CDHMMs to SDCHMMs

clustering algorithms to tie subspace Gaussians. The subsequent model conversion schemes are simple and fast. Yet the converted SDCHMMs are as accurate as the original CDHMMs but run faster and consume less memory. In addition, no re-training of the converted SDCHMMs is found necessary.

5.1 Issue I: Stream Definition

To derive K-stream SDCHMMs, we first have to partition the feature set Ω^D with D features into K disjoint feature subsets Ω^{d_k} with d_k features, $1 \leq k \leq K$. Formally, let $\mathcal{P}_K^D$ be such a partition, then

$$\mathcal{P}_K^D = \left\{ \Omega^{d_k} : \sum_{k=1}^{K} d_k = D \ \ \text{and} \ \Omega^{d_k} \cap \Omega^{d_j} = \emptyset \ \text{for} \ 1 \leq k \neq j \leq K \right\} \ . \tag{5.1}$$

The number of all possible partitions $N_{\mathcal{P}}$ can be found as follows. If the dimensions of all partitions, d_k, $1 \leq k \leq K$, are distinct, then $N_{\mathcal{P}}$ is given by

$$N_{\mathcal{P}} = \frac{D!}{d_1! d_2! \cdots d_K!} \ . \tag{5.2}$$

On the other hand, if all partitions are of the same dimension, i.e. $d_1 = d_2 = \cdots = d_K = d$ then

$$N_{\mathcal{P}} = \frac{D!}{(d!)^K K!} \ . \tag{5.3}$$

In general, let there be K_1 partitions with different dimensions, and K_2 partitions with the same dimension such that $K_1 + K_2 = K$. That is, d_k, $1 \leq k \leq K_1$, are distinct, but $d_{K_1+1} = d_{K_1+2} = \cdots = d_K = d$. Then

$$N_{\mathcal{P}} = \frac{D!}{d_1! d_2! \cdots d_{K_1}! (d!)^{K_2} K_2!} \quad .$$

(5.4)

The partition $\mathcal{P}_K^D$ is optimal if subsequent tying of subspace Gaussians in the feature subspaces of the partition results in minimal total quantization error for a pre-determined number of prototypes and clustering algorithm. In general, the clustering problem cannot be solved analytically, and is tackled numerically using iterative procedures. Since the total number of possible partitions is usually very large, it is not feasible to determine the optimal partition by numerically computing the quantization errors due to all possible candidates. Thus some heuristic approach has to be used to obtain a reasonable partition.

5.1.1 Common Streams

Our speech input comprises 39 features: 12 MFCCs, normalized power, and their first- and second-order time derivatives. Based on commonly-used streams in discrete HMM and semi-continuous HMM, the following "common" definitions of streams are explored:

1–stream definition: all features in one stream.

$$\boxed{12MFCC + 12\Delta MFCC + 12\Delta^2 MFCC + e + \Delta e + \Delta^2 e}$$

4–stream definition: cepstra, their first derivatives, their second derivatives, and power terms in four separate streams.

$$\boxed{12MFCC}$$
$$\boxed{12\Delta MFCC}$$
$$\boxed{12\Delta^2 MFCC}$$
$$\boxed{e + \Delta e + \Delta^2 e}$$

13–stream definition: twelve streams of each cepstral coefficient together with its derivatives, and one stream of the power terms.

$$12 * \boxed{MFCC + \Delta MFCC + \Delta^2 MFCC}$$
$$\boxed{e + \Delta e + \Delta^2 e}$$

39–stream definition: each 1-dimensional feature is put into one stream.

The main heuristic here is to put conceptually similar features together in one stream. The 1-stream and 39-stream definitions are included for reference. Note that 1-stream SDCHMMs are identical with the original CDHMMs, and 39-stream SDCHMMs are the same as feature-parameter-tying HMMs (FPTHMMs).

5.1.2 Correlated-Feature Streams

We adopt the heuristic that correlated features, by definition, should tend to cluster in a similar manner, and require each stream to have the most correlated features. Intuitively this criterion should result in smaller distortions for the clustered subspace Gaussians. For example, let us consider two features F_1 and F_2, having (scalar) data clustered around a, b, c, d and p, q, r, s with the same covariance respectively. If their means are fully correlated, they will give rise to four 2-dimensional Gaussians: $G_1(a, p)$, $G_2(b, q)$, $G_3(c, r)$, and $G_4(d, s)$, and their means lie on a straight line as shown in Figure 5.2(a). A clustering algorithm using an Euclidean distortion measure will produce two clusters, $\{G_1, G_2\}$ and $\{G_3, G_4\}$. Now let us assume that the F_1 and F_2 data are fixed but that their means are not correlated as described in Figure 5.2(b), forming the Gaussians $G_1(a, p)$, $G_2(b, r)$, $G_3(c, q)$ and $G_4(d, s)$. Although subsequent clustering produces the same two clusters, $\{G_1, G_2\}$ and $\{G_3, G_4\}$ as before, the distortion of the resulting clusters in this case is much larger. The reason is that according to Equation (3.4), the increase in the Euclidean distortion on clustering two Gaussians is proportional to the Euclidean distance between their means; it is clear that the distance between the means of G_1 and G_2, or G_3 and G_4 are larger in Figure 5.2(b) than that in Figure 5.2(a).

Strictly speaking, the correlation should be computed from Gaussian mean vectors of the given CDHMMs to deduce the stream definitions. In this work, however, in order to obtain one single stream definition for all ATIS experiments, we compute the correlations
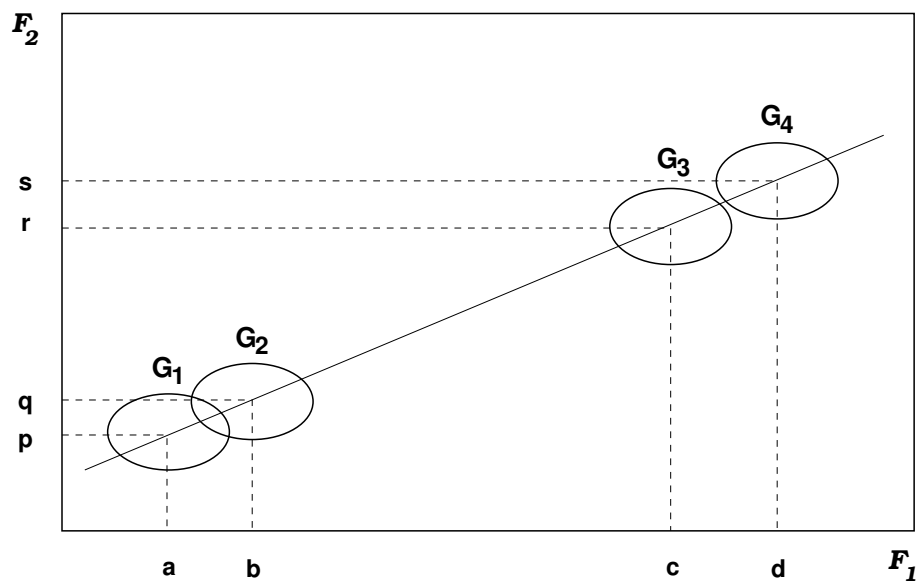
(a) Correlated features
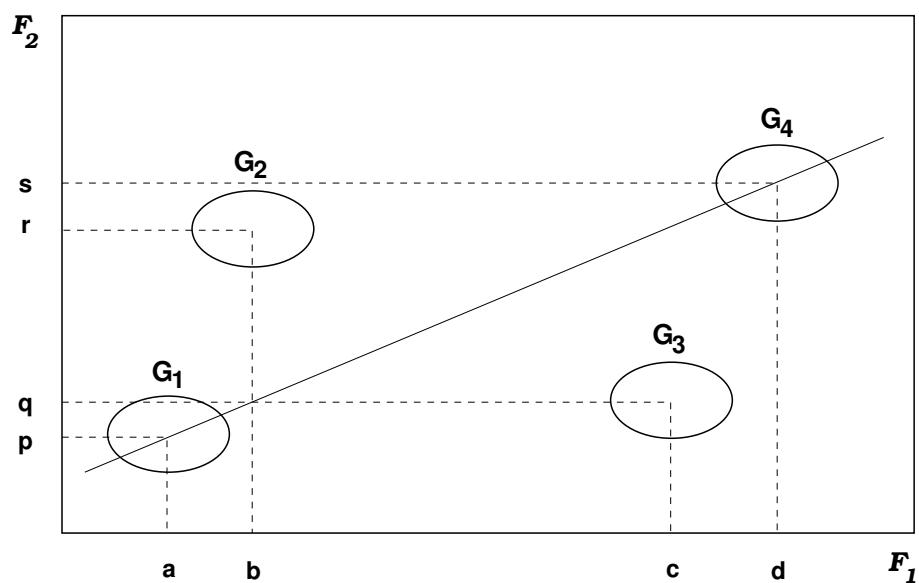
(b) Uncorrelated features

Figure 5.2: Effect of correlated and uncorrelated features on clustering

from feature frames of 1,000 ATIS training utterances instead. In preliminary experiments, we found little difference between stream definitions derived by these two methods. Note also that, although the features are assumed uncorrelated locally within each Gaussian distribution, during clustering of the subspace Gaussians, it is the global feature correlation that matters.

This definition has the additional benefit of providing a single coherent definition for *any* arbitrary number of streams of *any* dimension.

Multiple Correlation Measure

The correlation ρ_{ij} between two variables is commonly measured by Pearson's moment product correlation coefficient

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j}, \tag{5.5}$$

where σ_i and σ_j are the standard deviations of the i-th and j-th variables respectively, and σ_{ij} is the square root of their covariance. Nevertheless, multiple correlation measures among three or more variables are less studied. In the statistics literature, multiple correlation is usually reduced to a binary correlation as follows: One variable is identified as the *criterion* variable and the rest as the *predictor* variables. A single *derived* variable is computed from a linear combination of the predictor variables and the binary correlation between the derived variable and the criterion variable is taken as the multiple correlation among the variables. One way to determine the combination weights of the derived variable is to compute what are called the *beta weights* so as to maximize the resulting binary correlation [39].

However, in our context, a multiple correlation measure that emphasizes mutual correlations among all variables at the same time is more desirable. In this thesis, we propose a new definition of a multiple correlation coefficient R defined as

$$R \stackrel{\text{def}}{=} 1 - \text{determinant of correlation matrix of the variables.} \tag{5.6}$$

That is, the multiple correlation coefficient R among k variables is,

$$R = 1 - \begin{vmatrix} 1 & \rho_{12} & \rho_{13} & \cdots & \rho_{1k} \\ \rho_{21} & 1 & \rho_{23} & \cdots & \rho_{2k} \\ \rho_{31} & \rho_{32} & 1 & \cdots & \rho_{3k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{k1} & \rho_{k2} & \rho_{k3} & \cdots & 1 \end{vmatrix} \tag{5.7}$$

In particular, when there are only two variables,

$$R = 1 - \begin{vmatrix} 1 & \rho_{ij} \\ \rho_{ji} & 1 \end{vmatrix} = \rho_{ij}^2 \ .$$

Hence, in the case when there are only two variables, R equals the square of the moment product correlation coefficient.

Since the correlation matrix is symmetric, its determinant is equal to the product of its eigenvalues. Therefore,

$$R = 1 - \lambda_1 \lambda_2 \cdots \lambda_k \tag{5.8}$$

where λ_j is the j-th eigenvalue of the correlation matrix. Equation (5.8) gives a geometrical interpretation to the multiple correlation measure. When all the variables are highly correlated, the correlation matrix corresponds to an elongated ellipsoid with most eigenvalues except one being small, giving a small value for their product and thus a high value of R. When the variables are less correlated, the matrix is more spherical, giving a higher value for the eigenvalue product and smaller value of R. It can also easily be shown that R has the following desirable properties of a correlation measure:

- $0 \leq R \leq 1$

- when all variables are correlated, i.e. $\forall i, j, \ \rho_{ij} = 1, R = 1$

- when all variables are uncorrelated, i.e. $\forall i, j, \ \rho_{ij} = 0, R = 0$.

Algorithm 3:   Selection of the most correlated-feature streams (of the same dimension)

Goal: Given D features, define K n-dimensional streams with $D = nK$.

Step 1. Compute the multiple correlation coefficient among *any* set of n features according to Equation (5.7). (There are totally $C(D, n)$ coefficients.)

Step 2. Sort the multiple correlation coefficients in descending order, each tagged by an n-feature tuple indicating the features it computes from.

Step 3. Starting from the top, an n-feature tuple is moved from the sorted list to the "solution list" if *none* of its features already appear in any feature tuples of the solution list.

Step 4. Repeat Step 3 until all features appear in the solution list.

Step 5. The feature tuples in the "solution list" are the K-stream definition.

Derivation of Streams

Practically, we apply a greedy algorithm [26] to obtain streams in which the features are most correlated, as depicted in Algorithm 3. It is simple to modify the algorithm in cases when the number of features D is not a multiple of the number of streams K. For instance, one may require $(K - 1)$ streams to have the same dimension, and have this dimension as large as possible. This is accomplished by setting n to either $\lfloor D/K \rfloor$ or $\lceil D/K \rceil$ so that $(D - n(K - 1))$ is positive and is minimized, and then proceeding with Algorithm 3 to find the first $(K - 1)$ streams. The leftover $(D - n(K - 1))$ features are then put together as the last stream. Since the streams are restricted to have the same dimension except at most one of them, the computation of multiple correlation coefficients involves only determinants of any $n \times n$ matrices obtained by deleting any $(D - n)$ rows and the corresponding columns from the $D \times D$ feature correlation matrix — which needs to be computed once. As a result, the algorithm is efficient.

Table 5.1 shows the definitions of 13 and 20 correlated-feature streams generated by Algorithm 3 using 1,000 utterances from the ATIS training corpus. From the 20-stream definition, $MFCC$ and $\Delta^2 MFCC$ are found mostly correlated. Moreover, many streams of the 13-stream definition are feature supersets of streams from the 20-stream definition.

Table 5.1: ATIS: Definitions of correlated-feature streams

(a) 13-stream definition

STREAM	FEATURES
1	c_1, c_7, $\Delta\Delta c_7$
2	c_2, c_8, $\Delta\Delta c_8$
3	c_3, c_9, $\Delta\Delta c_9$
4	c_4, $\Delta\Delta c_4$, Δe
5	c_5, $\Delta\Delta c_2$, $\Delta\Delta c_5$
6	c_6, $\Delta\Delta c_6$, $\Delta\Delta e$
7	c_{10}, c_{12}, $\Delta\Delta c_{12}$
8	c_{11}, $\Delta\Delta c_1$, $\Delta\Delta c_{11}$
9	Δc_1, Δc_7, Δc_9
10	Δc_2, Δc_3, Δc_5
11	Δc_4, Δc_6, e
12	Δc_8, Δc_{10}, Δc_{12}
13	Δc_{11}, $\Delta\Delta c_3$, $\Delta\Delta c_{10}$

(b) 20-stream definition

STREAM	FEATURES
1	c_1, $\Delta\Delta c_1$
2	c_2, $\Delta\Delta c_2$
3	c_3, $\Delta\Delta c_3$
4	c_4, $\Delta\Delta c_4$
5	c_5, $\Delta\Delta c_5$
6	c_6, $\Delta\Delta c_6$
7	c_7, $\Delta\Delta c_7$
8	c_8, $\Delta\Delta c_8$
9	c_9, $\Delta\Delta c_9$
10	c_{10}, $\Delta\Delta c_{10}$
11	c_{11}, $\Delta\Delta c_{11}$
12	c_{12}, $\Delta\Delta c_{12}$
13	Δc_1, Δc_7
14	Δc_2, Δc_6
15	Δc_3, Δc_5
16	Δc_4, e
17	Δc_8, Δc_9
18	Δc_{10}, Δc_{11}
19	Δe, $\Delta\Delta e$
20	Δc_{12}

5.2   Issue II: Subspace Gaussian Clustering

Two very different clustering schemes are investigated: The bottom-up agglomerative clustering algorithm used previously for tying HMM states in Chapter 3, and a top-down modified k-means clustering algorithm.

5.2.1   Agglomerative Gaussian Clustering Algorithm

The *ensemble merging algorithm* for state tying described in Algorithm 2 of Section 3.2.3 can be applied without modification to cluster subspace Gaussians in each stream instead of HMM states. It is a bottom-up agglomerative clustering scheme in which two subspace Gaussians are merged if they result in minimum increase in distortion (scatter). The

algorithm has a complexity of $O(n^2)$.

5.2.2 Modified k-means Gaussian Clustering Algorithm

Algorithm 4 shows a newly devised $O(JLn)$ modified k-means clustering algorithm which derives L subspace Gaussian prototypes in J iterations without using any heuristics. With $JL \ll n$ for large acoustic models, the linearity in n implies improved efficiency (over the ensemble merging algorithm).

Algorithm 4:   Modified k-means Gaussians clustering algorithm

---

Goal: To derive K-stream SDCHMMs with L subspace Gaussian prototypes per stream.

Step 1. *Initialization:* First train a 1-stream Gaussian mixture model with L components. Project each of the L Gaussian components onto the K subspaces according to the given K-stream specification. The resultant KL subspace Gaussians will be used as initial subspace Gaussian prototypes.

Step 2. Similarly project each Gaussian pdf in the original CDHMMs onto the K subspaces.

Step 3. For each stream, repeat Step 4 & 5 until some convergence criterion is met.

Step 4. *Membership*: Associate each subspace Gaussian of CDHMMs with its nearest prototype as determined by their Bhattacharyya distance.

Step 5. *Update*: Merge all subspace Gaussians which share the same nearest prototype to become the new subspace Gaussian prototypes.

To compute the distance between two Gaussians during distribution clustering, we adopt the classification-based Bhattacharyya distance, which is defined as

$$D_{bhat} = \frac{1}{8}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \left[\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right]^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \frac{1}{2}\ln\frac{\left|\frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}\right|}{\sqrt{|\boldsymbol{\Sigma}_1||\boldsymbol{\Sigma}_2|}} \qquad (5.9)$$

where, $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, $i = 1, 2$, are the means and covariances of the two Gaussians [17]. The Bhattacharyya distance has been used in several speech-related tasks [42, 54, 57], leading to good results. The Bhattacharyya distance captures both the first- and the second-order statistics, and is expected to give better clustering results than the Euclidean distortion measure of Equation (3.4), which makes use of only the first-order statistics.

To initiate the iterative k-means clustering procedure for the conversion of CDHMMs to K-stream SDCHMMs with L subspace Gaussian prototypes per stream, we first train a Gaussian mixture model with L components using 1,000 ATIS training utterances. The L Gaussians are split into L subspace Gaussians for each stream, which are then used as seeds for clustering. If no training data is available, one may, for example, randomly pick L subspace Gaussians from the CDHMMs to start the clustering procedure.

5.3 ATIS Recognition Evaluation

AT&T's context-independent (CI) and context-dependent (CD) baseline ATIS recognizers of Section 3.2 are utilized for our SDCHMM evaluation. All components of the baseline recognizers are kept intact, except that their acoustic models are converted from CDHMMs to SDCHMMs. The testing conditions are exactly the same as those described in Table 3.3. All subspace Gaussian log-likelihoods are pre-computed at the beginning of each frame, and their values are stored in tables in contiguous memory[1]. In addition, for implementation and system simplicity, all streams are tied to the same number of subspace Gaussian prototypes in all our SDCHMMs.

5.3.1 Evaluation of Stream Definitions and Clustering Algorithms

With the two types of stream definitions of Section 5.1 and the two clustering algorithms of Section 5.2, four different combinations of stream definitions and clustering algorithms are tested using 13 streams:

- common stream definition + ensemble merging

- common stream definition + modified k-means Gaussian clustering

- correlated-feature stream definition + ensemble merging

- correlated-feature stream definition + modified k-means Gaussian clustering.

---

[1]We have also tried to compute the subspace Gaussian log-likelihoods on the fly during decoding, but unless when there are more than 512 prototypes per stream, pre-computation of the log-likelihoods always entails faster recognition.

54

Thirteen streams are chosen because both the common stream definition and the correlated-feature stream definition readily apply. Each stream consists of exactly three features, and is tied to 8–256 subspace Gaussian prototypes. Each of the ensuing 13-stream SDCHMM systems is then tested on the 1994 ATIS evaluation dataset.

Figure 5.3(a) and (b) show incremental improvements in recognition performance when correlated-feature streams and/or the modified k-means Gaussian clustering algorithm are used. The incremental improvement due to either correlated-feature streams or the modified k-means Gaussian clustering algorithm alone is similar in the case of CI models. In the case of CD models, most of the gain in accuracy comes from the modified k-means Gaussian clustering algorithm. Nonetheless, the improvements are observed with both CI and CD models at almost all levels of quantization — various numbers of subspace Gaussian prototypes. This shows that by bringing more knowledge into play — correlation in the correlated-feature stream definition and second-order statistics in the modified k-means Gaussian clustering algorithm, better subspace Gaussian tying is achieved. In particular, the improvement is more pronounced with fewer prototypes. For example, for the CD SDCHMMs, WER with 8 subspace Gaussian prototypes per stream drops 24% compared with 10% drop with the use of 256 prototypes. This is desirable as fewer prototypes usually translate into faster recognition.

Henceforth, all experiments are run with SDCHMMs derived using the modified k-means Gaussian clustering algorithm with correlated-feature streams except for the 4-stream SDCHMMs which are derived with the common 4-stream definition.

5.3.2 Evaluation of SDCHMMs

I. Recognition Accuracy

For 1, 4, 13, 20, and 39 correlated-feature streams, the modified k-means Gaussian clustering algorithm is run, in each case, to obtain CI SDCHMMs with 8–256 subspace Gaussian prototypes per stream, or CD SDCHMMs with 2–256 subspace Gaussian prototypes per stream. Figure 5.4 shows their recognition accuracies in terms of word error rate (WER).

In general, WER decreases with more streams and more prototypes as expected, since more streams of smaller dimensions should result in smaller distortions when the subspace
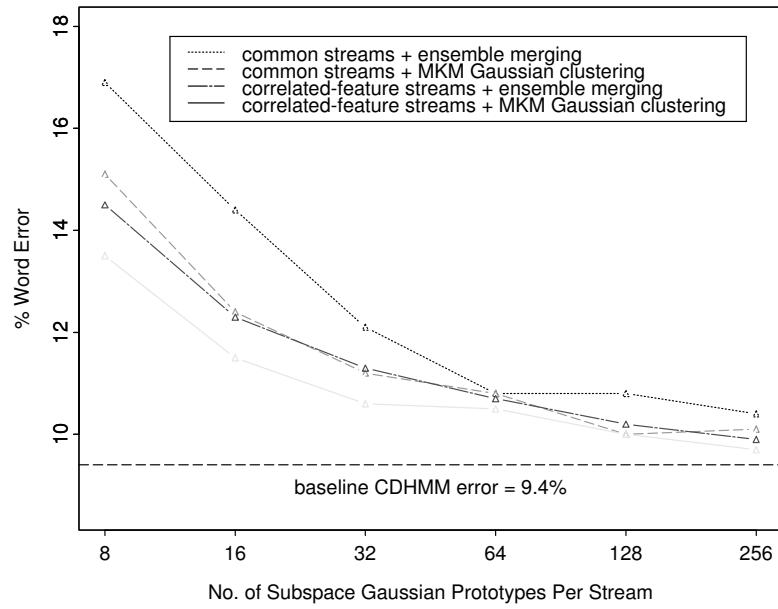
(a) Context-independent models
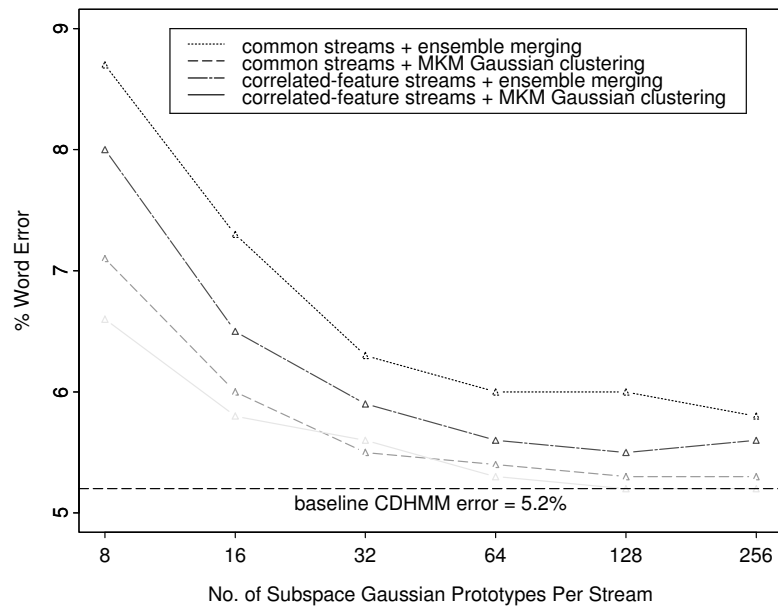
(b) Context-dependent models

Figure 5.3: ATIS: Recognition accuracy of 13-stream SDCHMMs with various stream definitions and clustering schemes
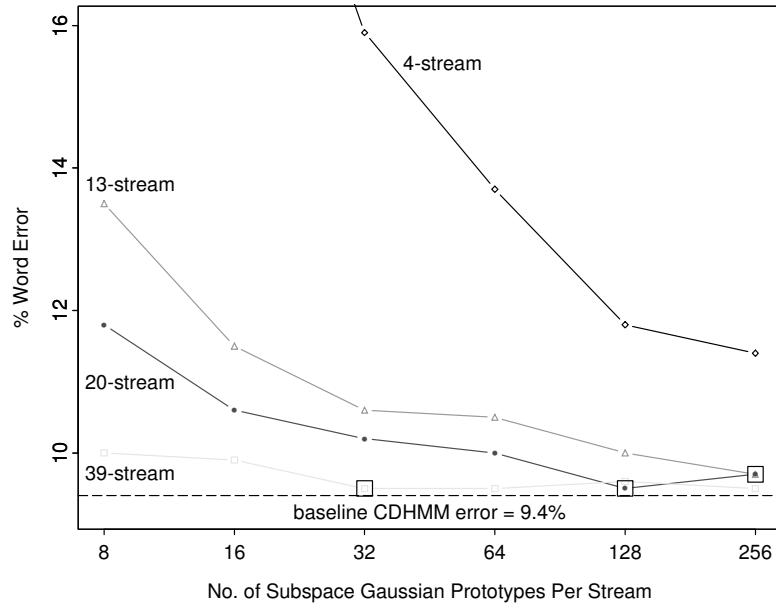
(a) Context-independent models
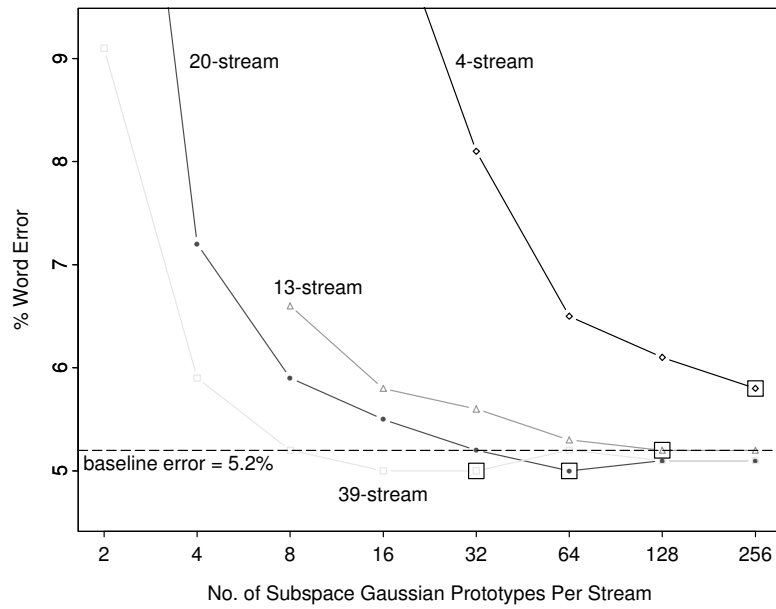
(b) Context-dependent models

Figure 5.4: ATIS: Effect of number of streams and subspace Gaussian prototypes on SDCHMM recognition accuracy (the best systems of Table 5.2 are marked with squares)

Gaussians are quantized, and more prototypes should give smaller quantization errors. In other words, when the same number of prototypes is used, SDCHMMs with more streams are more accurate; or, SDCHMMs with more streams can achieve the same accuracy with fewer prototypes than SDCHMMs with fewer streams. For example, 39-stream CD SDCHMMs obtain the best WER of 5.0% with 16 subspace Gaussian prototypes, while 20-stream CD SDCHMMs require 64 prototypes, and 13-stream CD SDCHMMs reach their best WER of 5.2% with at least 128 prototypes.

However, for CD models, the WER actually increases for 20- and 39-stream SDCHMMs after 128 and 64 prototypes respectively. This suggests that some of the original CD CDHMMs may not be well trained (even with the addition of some 8,000 WSJ training utterances. See Section 3.3). Subspace Gaussian tying may help improve these poorly trained models by interpolating them with better-trained models, or by pooling together more training data for them.

The best CI SDCHMMs (with 20 streams and 128 prototypes, or 39 streams and 32 prototypes) compare well with the baseline CI CDHMMs (9.5% vs. 9.4%, a relative 1% increase in WER), and the best CD SDCHMMs (with 20 streams and 64 prototypes, or 39 streams and 16 prototypes) actually outperform the baseline CD CDHMMs (5.0% vs. 5.2%, a relative 4% reduction in WER).

Finally, Figure 5.4(b) shows that even with two subspace Gaussian prototypes, or one bit of information per stream, a 39-stream CD SDCHMM system can still achieve a WER of only 9.1%. This is not too surprising when one realizes that this SDCHMM system can, in principle, represent 2^{39} or about half a trillion distinct full-space Gaussians, and there are only about 76,154 Gaussians in the original CDHMM system — about one-tenth of one-millionth of the representables. From another perspective, it also suggests high redundancy in current CDHMMs, and that SDCHMMs are more efficient in representing the model parameter space.

58

II. Recognition Speed

The corresponding total recognition times of the SDCHMM systems of Figure 5.4 are presented in Figure 5.5 relative to real-time performance. The relationships between recognition speed and the number of prototypes are generally parabolas that curve upwards. The longer recognition time at the two ends of the parabolic curves are due to two very different effects:

- In general, more prototypes simply require more computation for the subspace Gaussian log-likelihoods.

- Fewer prototypes lead to poorer SDCHMMs (due to larger quantization errors) with less discriminating power and more active states during a Viterbi search using the same beam-width, and thus more computation. Figure 5.6 shows the corresponding number of active states. Notice the high correlation [2] between the number of active states in Figure 5.6 and WERs of Figure 5.4.

The first effect is weaker in the CD SDCHMM system than in the CI SDCHMM system. It is because that there are many more active states during decoding in the CD system than in the CI system — about 10 times more from Figure 5.6. With the large number of active states in the CD system, the pre-computation of subspace Gaussian log-likelihoods represents a small proportion of the total computation time. Consequently, the speed of the CD SDCHMM system is insensitive to the first effect.

The impact of the number of streams on recognition speed is complicated by the above two effects, but in general, more streams means more additions in the computation of state log-likelihoods (Equation (4.12)) and hence longer recognition time. In addition, another complication arises from the software implementation. For the same number of subspace Gaussian prototypes per stream, the total number of function calls to compute their log likelihoods increases with the number of streams. Each function call adds overhead due to invocation of the call, and initialization of various data structures.

---

[2] The relationship between the number of active states during decoding and final recognition accuracy should not be overstated. At best, the number of active states may be used to explain the poorer performance of the converted models (compared with their original ones) but *not* to predict the latter.
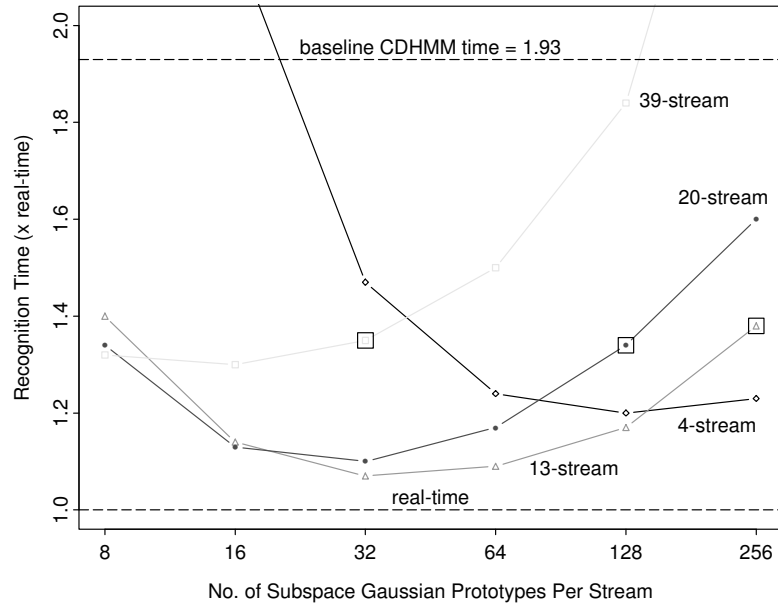
(a) Context-independent models
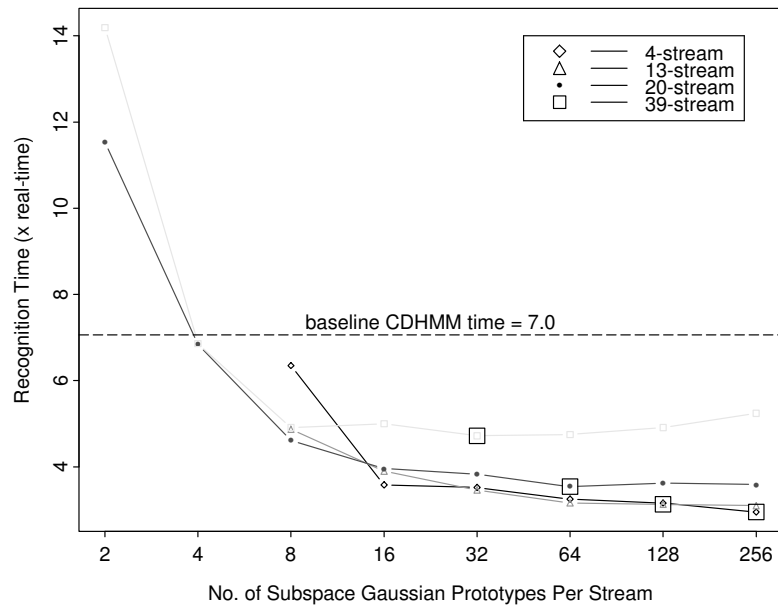
(b) Context-dependent models

Figure 5.5: ATIS: Effect of number of streams and subspace Gaussian prototypes on SDCHMM recognition speed (the best systems of Table 5.2 are marked with squares)

60
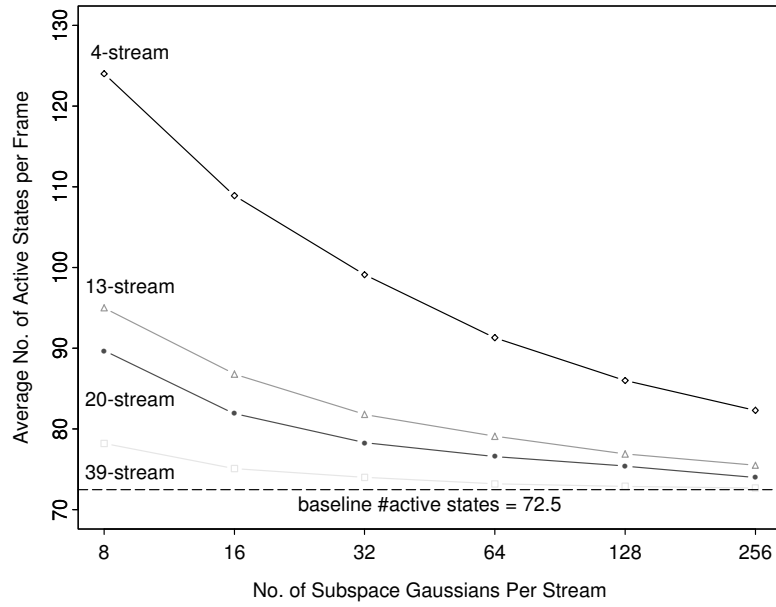
(a) Context-independent models
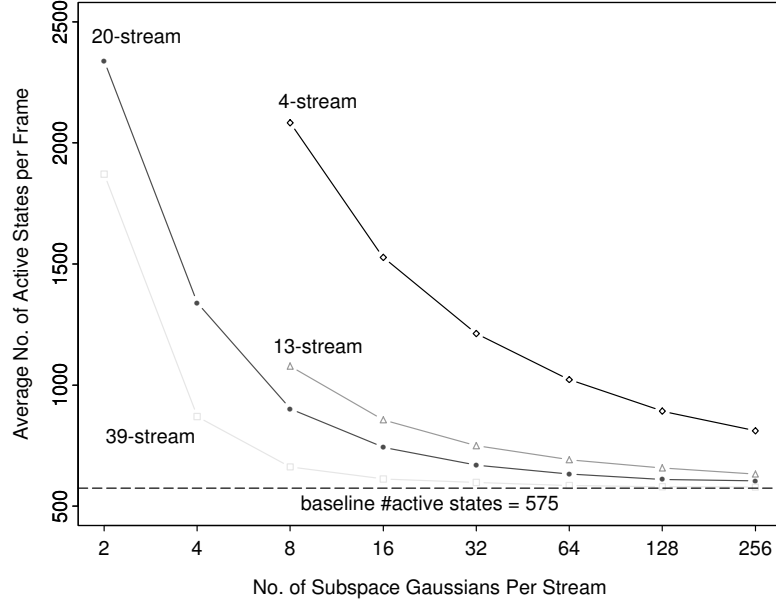
(b) Context-dependent models

Figure 5.6: ATIS: Number of active states during decoding

5.3.3 Summary of Best Results

From the discussion above, there is a trade-off between recognition accuracy and recognition speed by adjusting the number of streams and the number of prototypes. By overlaying Figure 5.5 onto Figure 5.4, the best SDCHMM recognition performance with various numbers of streams are determined and summarized in Table 5.2. The table also lists comparative results for the baseline CDHMMs (or 1-stream SDCHMMs):

TIME : recognition time

PR : reduction in the number of model parameters. The figures in parentheses include also the encoding indices/pointers when original full-space Gaussians are mapped to their subspace Gaussians — one index/pointer per subspace Gaussian per stream.

MS : memory savings, assuming 4-byte floats for mixture weights, Gaussian means, and variances, and 1-byte indices for encoding subspace Gaussians.

Table 5.2: ATIS: Summary of the best results (K = #streams, n = #subspace Gaussian prototypes per stream, CI = context independent, CD = context dependent, WER = word error rate (%), TIME is relative to that of the baseline system, PR = parameter reduction, and MS = memory savings. For PR, figures in parentheses take into account the mappings of subspace Gaussians to the full-space Gaussians. For MS, 1-byte mappings are assumed.)

CI/CD	K	n	WER	TIME	PR	MS
CI	1	2254	9.4	1.00	1	1
CI	13	256	9.7	0.72	8 (3.5)	6.1
CI	20	128	9.5	0.70	15 (3.1)	7.6
CI	39	32	9.5	0.70	38 (1.9)	6.7
CD	1	76154	5.2	1.00	1	1
CD	4	256	5.8	0.42	63 (15)	35
CD	13	128	5.2	0.44	70 (5.6)	18
CD	20	64	5.0	0.50	74 (3.8)	13
CD	39	32	5.0	0.67	77 (2.0)	7.3

The CD SDCHMMs perform better than the CI SDCHMMs when compared with their respective baseline systems. The CD SDCHMMs require fewer prototypes but give relatively better accuracies, higher computation efficiency, greater memory savings and larger reduction in model parameters. The most plausible explanation is that the CI models are less complex and robustly trained due to the large amount of available training data. Further tying of CI model parameters renders over-smoothing of the parameters. As a result, more prototypes are required to maintain acceptable quantization errors. On the contrary, the CD SDCHMMs are highly complex and model all triphone contexts observed in the data, but modeling the rare triphones has always been a problem. The baseline CDHMMs resort to state-tying to alleviate the problem. Obviously, results of Table 5.2 suggest that some triphones are still not well trained, and further tying at the smaller sub-phonetic unit of subspace Gaussians can effectively reduce the model parameter space to obtain more robust models. Nevertheless, it is still amazing to see that the 76,154 Gaussians of the baseline CDHMMs can be represented by 32 — 128 subspace Gaussians per stream.

Thirteen, 20 or 39 streams all work well in both CD or CI systems, but their impacts on savings in computation, memory, model parameters and accuracy are quite different. For the CI systems, 13- to 39-stream SDCHMMs all give similar performance in terms of accuracy, speed and memory requirement. The only difference lies in their number of model parameters: 39-stream SDCHMMs (with 1-dimensional scalar streams) have the fewest model parameters if one does not count the subspace Gaussian encoding parameters, thanks to the efficiency of scalar quantization which requires fewer prototypes. However, once we include the encoding parameters, 39-stream SDCHMMs require more model parameters than SDCHMMs with fewer streams because they consume one encoding parameter per stream for each subspace Gaussian. On the other hand, since there are many more distributions and HMM state evaluations in CD systems than in CI systems, the greater sharing of Gaussian parameters in CD SDCHMMs entails greater savings in computation, memory, and model parameters.

Various statistical significance tests from NIST (National Institute of Standards and Technology) are run on the performance differences among the recognition systems of

Table 5.2, and their results are presented in Appendix C. Most of the tests indicate no significant difference among the various CI (CD) systems. The only test that indicates a difference actually finds the SDCHMM systems more accurate.

Operating Curves

The foregoing discussion that is based on Viterbi decoding using one particular beam-width can be biased. Figure 5.7 studies the effect of beam-width on various SDCHMM systems of Table 5.2 with their operating curves. An operating curve shows the speed and accuracy of a recognizer at varying beam-width.

The asymptotic performances of CI SDCHMMs are basically the same as those of their parent CI CDHMMs, while CD SDCHMMs outperform CD CDHMMs asymptotically. In addition, the SDCHMM curves always lie to the left of the CDHMM curve on each graph; thus SDCHMM systems are always faster. Similarly, operating curves of SDCHMMs with fewer streams also lie to the left of SDCHMMs with more streams though they may saturate sooner with poorer accuracies (for example, compare the operating curves of 20-stream and 39-stream CI SDCHMMs, or those of 13-stream and 20-stream CD SDCHMMs). The best compromise seems to come from 20-stream SDCHMM systems.

5.4 Summary and Discussion

We show that by properly projecting mixture Gaussians of accurate CDHMMs onto subspaces using the correlated-feature stream definition and carefully tying the ensuing subspace Gaussians by the modified k-means Gaussian clustering algorithm, accurate SDCHMMs can be converted from CDHMMs. The correlated-feature stream definition, though not guaranteed optimal, gives reasonably good results. The resulting SDCHMMs fulfill the promises of faster computation and less memory requirement. For example, compared with the baseline CDHMM system, the best CI SDCHMM system saves the total computation time by 30% and obtain an 8-fold reduction in HMM memory with an insignificant (absolute) 0.1% drop in accuracy. Similarly, the best CD SDCHMM system outperforms the baseline CDHMM system by 0.2% in absolute accuracy, yet it runs twice as fast with
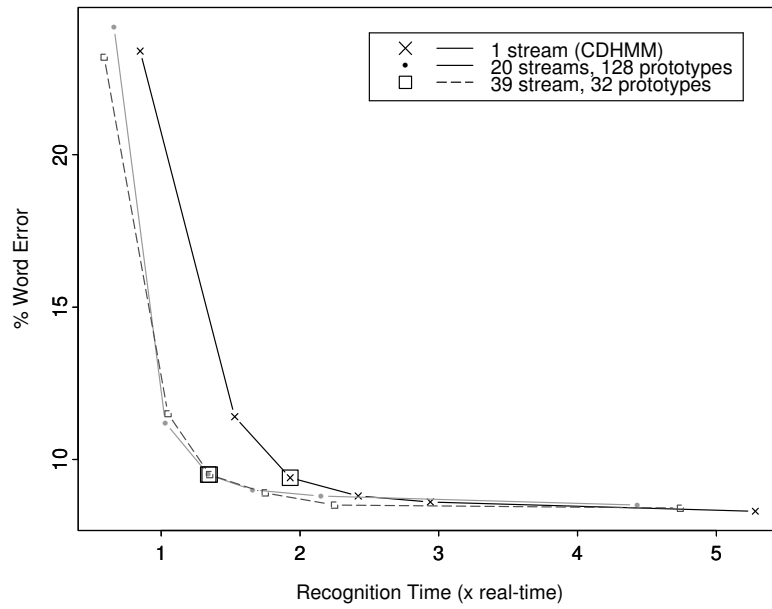
(a) Context-independent models
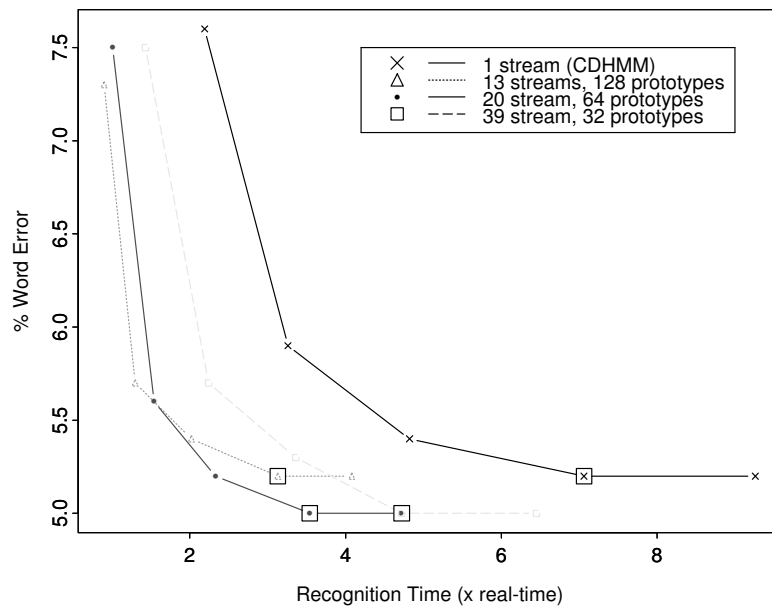
(b) Context-dependent models

Figure 5.7: ATIS: Operating curves of SDCHMMs (the best systems of Table 5.2 are marked with squares)

a 13-fold reduction in memory used for acoustic models.

The model conversion is fast, since both the correlated-feature stream definition and the modified k-means clustering scheme are algorithmically simple. For instance, on an SGI machine (195MHZ MIPS R10000), conversion of CI CDHMMs to SDCHMMs can be accomplished in less than a minute, while the conversion of CD models takes a few minutes to an hour, depending on the extent of quantization[3].

As a side effect of subspace Gaussian tying, some full-space Gaussians will become identical after the tying process. Table 5.3 shows the number of distinct full-space Gaussians after subspace Gaussian tying. Notice that even with 32 prototypes per stream, there is very little compression (less than 1%) of full-space Gaussians using 13, 20 or 39 streams. This shows the efficacy of SDCHMMs: The distinctiveness of the Gaussians is well maintained despite the great compression in the subspaces.

Table 5.3: ATIS: Number of distinct full-space Gaussians after subspace Gaussian tying in context-dependent SDCHMMs

#PROTOTYPES	#STREAMS			
	4	13	20	39
2	16	4658	32141	72662
4	256	47312	72834	76075
8	1919	68606	75809	76140
16	7505	74664	76085	76149
32	18937	75810	76132	76150
64	34643	76057	76148	76150
128	53530	76113	76147	76150
256	64635	76137	76150	76150

The CD SDCHMMs show greater relative improvements than the CI SDCHMMs probably due to the higher degree of redundancy and decreased robustness of the original CD CDHMMs. One may thus postulate that SDCHMMs may be more effective with larger acoustic models.

[3]The computation of the modified k-means Gaussian clustering algorithm is proportional to the number of Gaussians. The ratio of numbers of Gaussians in the CD and CI systems is 76154 : 2254 = 34 : 1.

The 39-stream SDCHMMs in our case are equivalent to feature-parameter-tying HMMs (FPTHMMs). Our results show that, although they give the highest compression of subspace Gaussians, they do not give the greatest reduction in computation time and memory size. The major shortcomings of FPTHMMs — SDCHMMs with scalar streams — are:

- When the original full-space Gaussians are encoded by the subspace Gaussian prototypes, more mapping indices/pointers are required.

- During decoding, the computation of each state log-likelihood involves additions of more subspace Gaussian log-likelihoods.

The impact of the number of streams on accuracy, computation time, and memory size is complicated. All things considered, 13 and 20 streams seem to be better choices.

Re-training of the converted CI SDCHMMs has also been studied, and no significant improvement is observed. Since the converted CD SDCHMMs already surpass the baseline performance, based on our experience with re-training the CI SDCHMMs, we are not surprised that re-training will not improve the CD SDCHMMs. On the other hand, with the great reduction of Gaussian parameters (mixture weights, Gaussian means, and variances) by one to two orders of magnitude, one should expect SDCHMMs to be trained from scratch with much less training data than their parent CDHMMs. Direct training of SDCHMMs will be pursued in Chapter 7.

Chapter 6

Analysis of the Subspace Distribution Tying Structure

In the last chapter, continuous density HMMs (CDHMMs) were converted to subspace distribution clustering HMMs (SDCHMMs) by projecting the Gaussians of CDHMMs onto subspaces and clustering the subspace Gaussians in each stream into a small set of prototypes. The Gaussian clustering process is fully automatic, utilizing only acoustic information from the data. Yet recognition results on the ATIS task (Table 5.2) show that, for instance, SDCHMMs with 20 streams and 64 subspace Gaussians prototypes per stream are adequate to represent the original context-dependent CDHMMs containing 76,154 full-space Gaussians — a reduction of Gaussian parameters (means and variances) by a factor of more than 1,000. Such efficient tying suggests that the original Gaussians were highly redundant. It is therefore interesting to "see" *how* acoustics in terms of the subspace Gaussians are similarly realized by *which* speech units. We refer to the tying information among subspace Gaussians of SDCHMMs together with the mappings between them and the full-space Gaussians of CDHMMs as the *subspace Gaussian tying structure* (SGTS), or generally *subspace distribution tying structure* (SDTS) when the type of distribution is immaterial for the discussion.

With the huge number of combinations of phonetic units, HMM states, and Gaussian components in the SDCHMMs, it will be very hard to visualize the whole subspace Gaussian tying structure in a single picture. In the following, we present a simple quantitative analysis of the number of subspace Gaussians shared by the corresponding HMM states of any pair of phones. We hope that the analysis will shed some light on the acoustic-phonetic

68

nature of speech.

6.1 SDCHMMs to Analyze

In order to generate some readable visual plots of the subspace Gaussian tying structure between some pair of phones, we employ a less complex SDCHMM system. To do that, we first re-train context-independent (CI) CDHMMs with about 4,000 ATIS training utterances using the *segmental k-means training algorithm* [38]. The CI CDHMMs have the same HMM configuration as the baseline system except that there are only four Gaussian mixture components per state. Twenty-stream CI SDCHMMs are then derived from the CDHMMs by the model conversion scheme as explained in Chapter 5 requiring 64 prototypes per stream. The resulting CDHMMs and SDCHMMs have recognition WERs of 12.2% and 12.6% respectively[1]. The SGTS of the 20-stream CI SDCHMMs is then analyzed.

6.2 Methodology

For the corresponding states of any two phonetic SDCHMMs with the same number of HMM states, which are modeled as mixture Gaussian densities, the constituent subspace Gaussians of their full-space Gaussians are compared. Specifically, for each stream, the number of common subspace Gaussians at the corresponding states of the two SDCHMMs are counted *irrespective* to which mixture components the subspace Gaussians come from. The procedure may be expressed in pseudo-code as follows:

    for each pair of phones (P, Q) with the same number of states

        for each state

        {

            $num\_common\_subgaussian = 0$

            for each stream

---

[1]These results are worse than the baseline CI models of Table 3.3. It is mainly due to their reduced model complexity: 4 mixtures vs. 16 mixtures per density in the baseline models.

```
            {
                    P.list = subspace Gaussians from all mixture components of
                            phone P in this state projected onto this stream
                    Q.list = subspace Gaussians from all mixture components of
                            phone Q in this state projected onto this stream
                    num_common_subgaussian += Common_Subgaussian(P.list, Q.list)
            }
            print(num_common_subgaussian)
        }
    Common_Subgaussian(list₁, list₂)
    {
            find the number of common subspace Gaussians between list₁ and list₂
    }
```

Since each subspace Gaussian may be represented by its prototype index, a 20-stream full-space Gaussian can be represented by a tuple of 20 prototype indices, one for each stream. For example, the 4-mixture density of the third state of the phones "s" and "z" are represented as:

$\{$ < 2, 4, 3, 2, 9, 46, 2, 52, 2, 2, 33, 13, 46, 37, 13, 21, 46, 60, 42, 2 >,

< 0, 24, 31, 34, 28, 2, 28, 35, 46, 37, 46, 46, 33, 46, 37, 46, 46, 48, 24, 21 >,

< 2, 24, 12, 24, 2, 46, 24, 16, 13, 21, 47, 12, 46, 46, 46, 46, 2, 48, 28, 2 >,

< 4, 37, 12, 25, 34, 46, 4, 52, 31, 21, 16, 25, 12, 51, 44, 24, 5, 25, 12, 4 > $\}$

and

$\{$ < 46, 4, 47, 2, 47, 46, 13, 52, 2, 2, 33, 46, 46, 41, 13, 21, 46, 13, 24, 2 >,

< 0, 24, 31, 34, 28, 12, 28, 35, 27, 37, 46, 12, 33, 46, 37, 21, 46, 48, 13, 21 >,

< 46, 24, 46, 24, 2, 46, 24, 16, 13, 21, 47, 52, 33, 46, 46, 46, 2, 57, 28, 2 >,

< 0, 4, 46, 44, 28, 13, 47, 37, 25, 1, 5, 4, 25, 51, 35, 21, 5, 25, 25, 25 > $\}$

respectively. Thus to determine the number of common subspace Gaussians in the twelfth stream of the third state of "s" and "z", the two lists {13, 46, 12, 25} and {46, 12, 52,

4} are compared and the result is two. Note that the order of the indices is ignored. The computation is repeated for every stream, and the counts are accumulated for each state.

6.3 Results

Since all the 45 ATIS phones (excluding the three noise models) shown in Table 3.1 have three states, the number of common subspace Gaussians between any pairs of the 45 phones can be computed for each of their three states. The results are tabulated in Table B.1(a)–(c) in Appendix B as confusion matrices, one for each state.

The phones are further divided into two major categories: 18 vowels and 27 consonants[2]. Histograms of counts of the number of common subspace Gaussians between any two phones within each category and across the two categories are shown in Figure 6.1 together with some of their statistics.

In addition, Figure 6.2(a)–(c) provides a visualization of the SGTS between three pairs of phones belonging to various phonetic categories:

- vowel-vowel pair: "ae" and "eh"

- consonant-consonant pair: "s" and "z"

- consonant-vowel pair: "t" and "iy".

In each of the three figures, the abscissas are stream indices ranging from 1 to 20, while the ordinates are the subspace Gaussian prototype indices for each stream. For each stream of the 4-mixture Gaussians of a state, the subspace Gaussian prototype indices of the first phone in the pair are represented by the four letters "a", "b", "c", and "d". Subspace Gaussians symbolized by the same letter belong to the same full-space Gaussian component. Thus if one connects all the letter "a"'s together across the 20 streams, one obtains the "trajectory" of a full-space Gaussian encoded by its subspace Gaussian prototypes. On the other hand, the four subspace Gaussian prototype indices of the second phone in the pair of the same stream are represented indiscriminately by square boxes. A

---

[2]The vowels are: aa, ae, ah, ao, aw, ax, axr, ay, eh, er, ey, ih, ix, iy, ow, oy, uh, and uw; the consonants are: b, ch, d, dh, dx, el, en, f, g, hh, jh, k, l, m, n, ng, nx, p, r, s, sh, t, th, v, w, y, and z.

(a) Between any two vowels

(b) Between any two consonants

(c) Between a vowel and a consonant

(d) Between any two phones
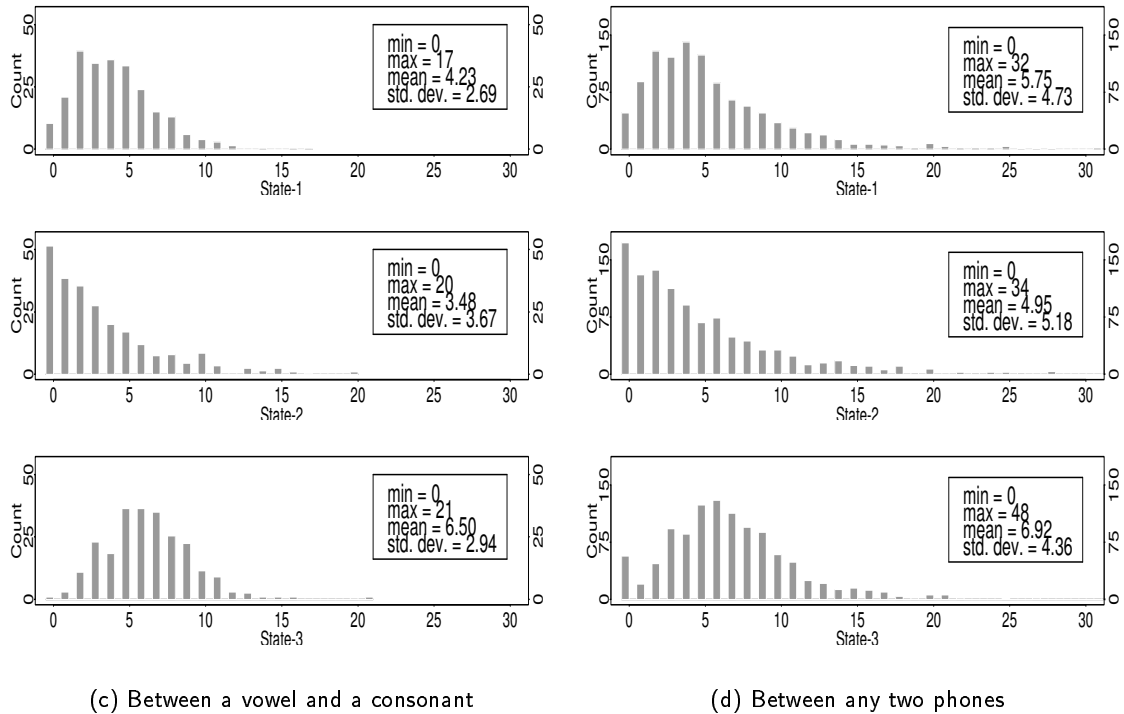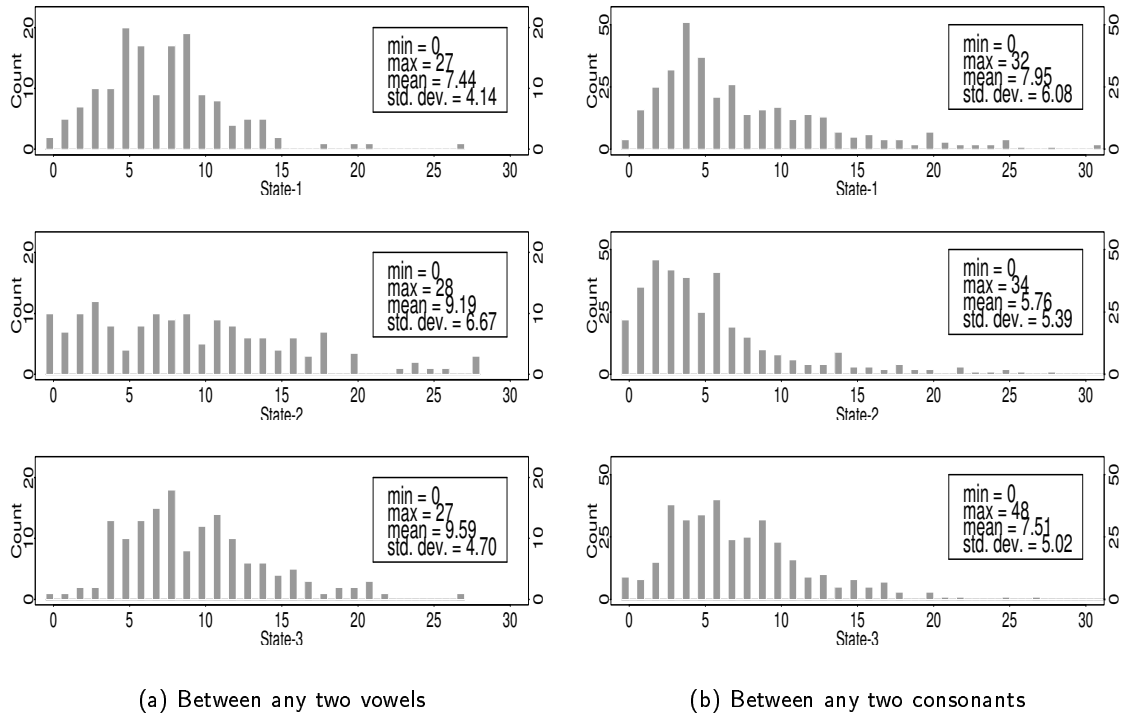
Figure 6.1: ATIS: Counts of the number of common subspace Gaussians between phones of different broad categories

match of subspace Gaussians between the two phones occurs when any of the four letters is "captured" by a box. (Due to the low resolution on the ordinate, only when a letter is right in the middle of a square box is there a match.) From Table B.1, the number of matches in the three figures, from the first state to the third state are:

- between "ae" and "eh": 21, 26, 27

- between "s" and "z": 25, 28, 48

- between "t" and "iy": 0, 0, 5.

6.4 Discussion

The figures in Table B.1 should be compared with the expected number of common subspace Gaussians between two SDCHMM states should the match occur by pure chance. The problem may be re-phrased in the following abstraction:

> Given a box of N balls numbered 1 to N, and two bags, each having a capacity of m balls, a ball is randomly picked from the box and put into a bag, and a ball of the same number is put back into the box from the stock. The procedure is repeated $2m$ times until both bags are full. Now compare the two sets of m balls in the two bags, and determine the expected number of matches.

Note that the number of matches does not follow a simple binomial distribution because the match ignores the position of the m balls in a bag. In our case, $N = 64$ and $m = 4$, and the expected number of matches is found to be 0.24 per stream. Thus the expected number of common subspace Gaussians between two 20-stream SDCHMM states is $20 \times 0.24 = 4.8$ should the match occur by chance.

By comparing the expected number of matches of 4.8 and the figures shown in Table B.1, Figure 6.1, and Figure 6.2, we have the following observations:

- The extent of sharing of subspace Gaussians splits along broad phonetic categories (vowels and consonants; and within consonants, along sub-categories of fricatives,

plosives, nasals and approximants [43]). That is, there is more sharing of subspace Gaussians between two vowels or two consonants than between a vowel and a consonant; and, within consonants, there is more sharing between two fricatives, two plosives, etc. The effect is most obvious from Figure 6.2 in which vowel pair "ae"– "eh" and consonant pair "s"–"z" have 25–60% of their subspace Gaussians shared in all three states; whereas there is basically no sharing between the consonant-vowel pair "t"–"iy".

- In the mid-states, where the coarticulatory effect is weak and the identity of a phone is better preserved, there is much less sharing of subspace Gaussians between vowel-consonant pairs, while vowel-vowel pairs exhibit more sharing. In fact, the average number (3.48) of common subspace Gaussians between vowel-consonant pairs is well below the expected value of 4.8. The histogram for the case of vowel-vowel pairs is also more uniform than that of consonant-consonant pairs. This may be attributed by the more gradual differences in the articulations of the vowels. In contrast, the articulations of different categories of consonants are very different (c.f. nasals vs. plosives).

- On average, there is more sharing between two vowels than between two consonants. This again confirms the greater resemblance between vowels.

All the observations are well in accord with our phonetic knowledge about the phones. The analysis provides some understanding of the efficiency of subspace distribution clustering hidden Markov modeling in encoding the phonetic information.
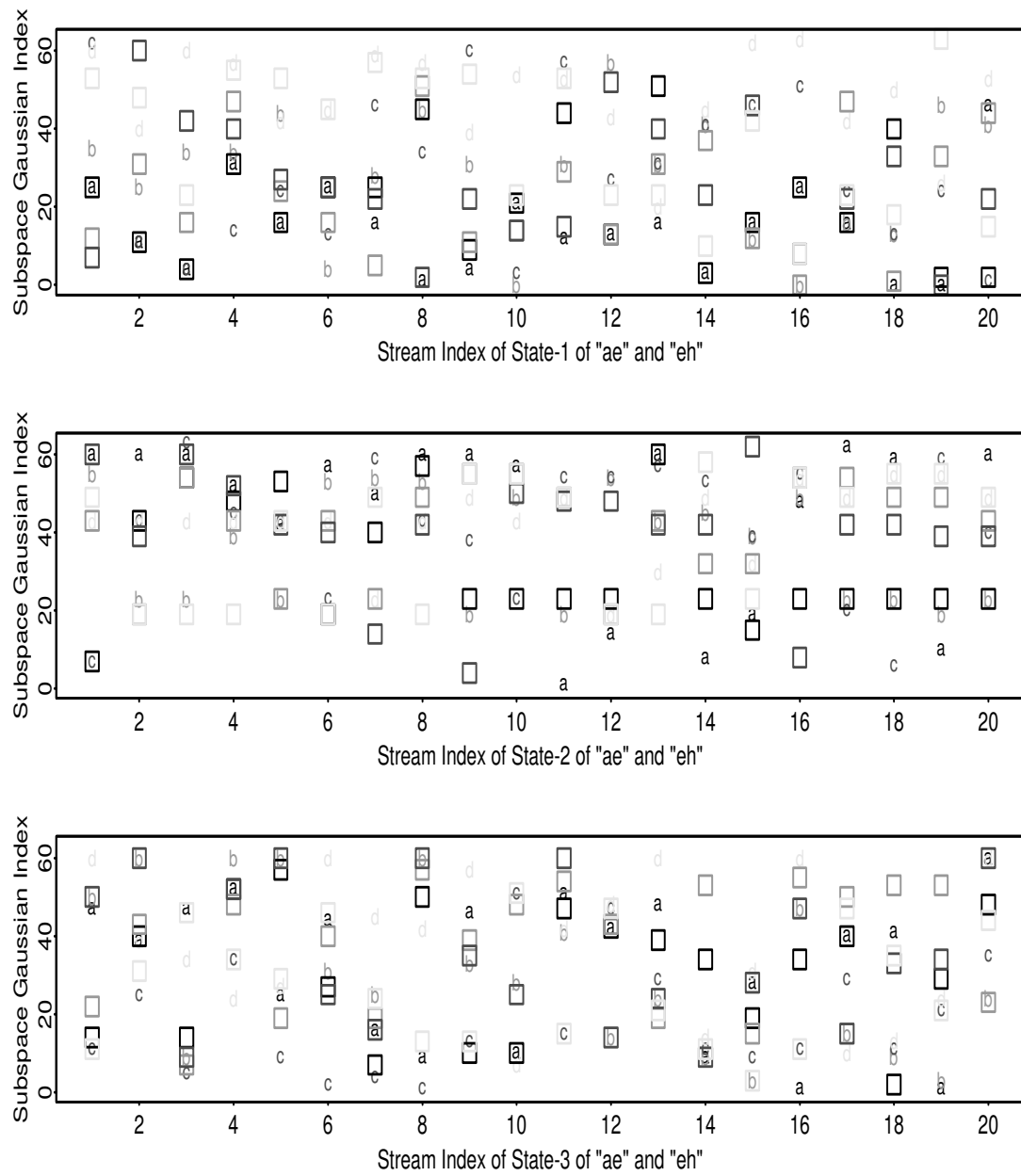
Figure 6.2: Subspace Gaussian tying structure (a) between "ae" and "eh" (number of matches from the 1st to the 3rd state are 21, 26, 27)
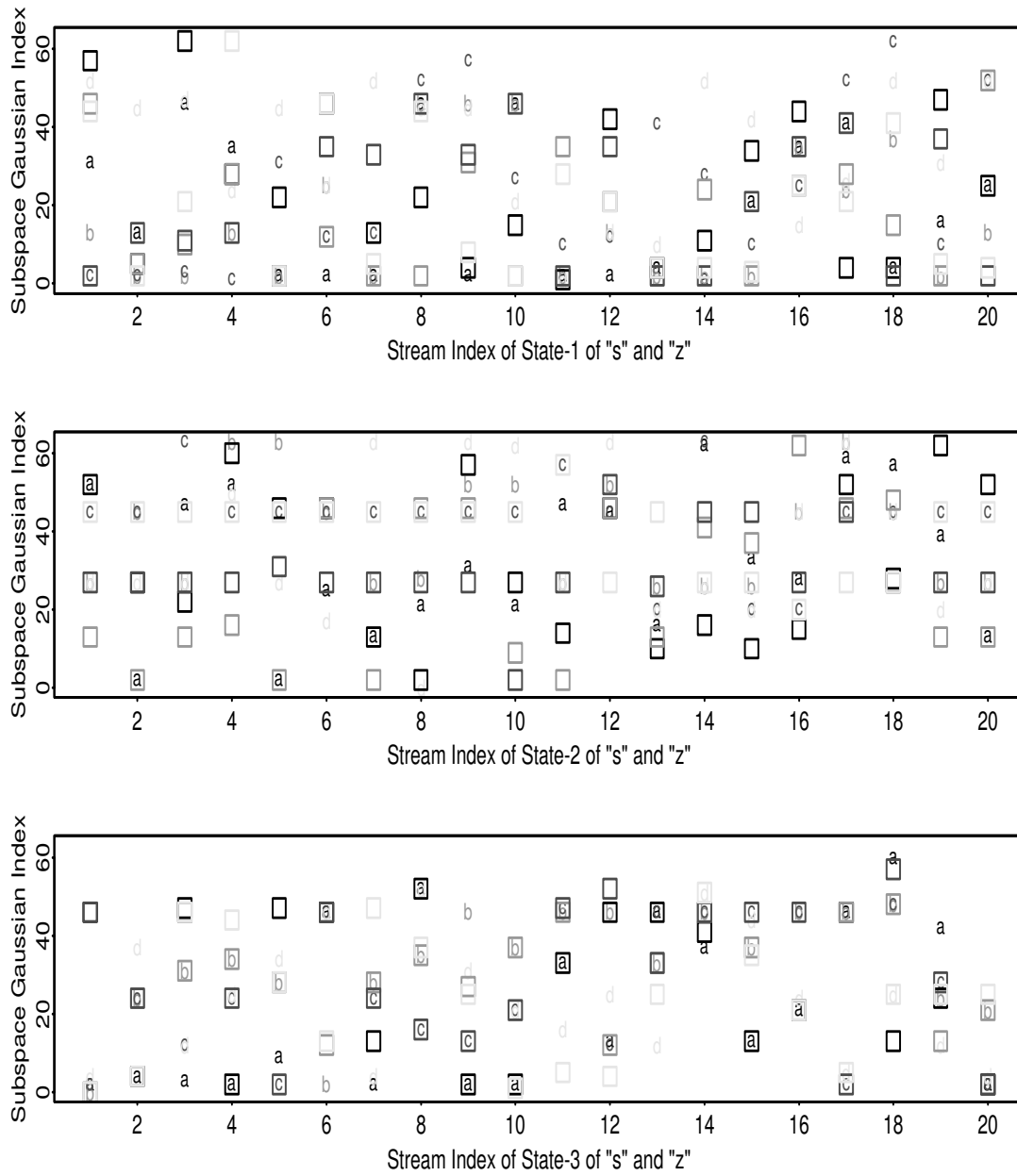
Figure 6.2: Subspace Gaussian tying structure (b) between "s" and "z" (number of matches from the 1st to the 3rd state are 25, 28, 48)

76
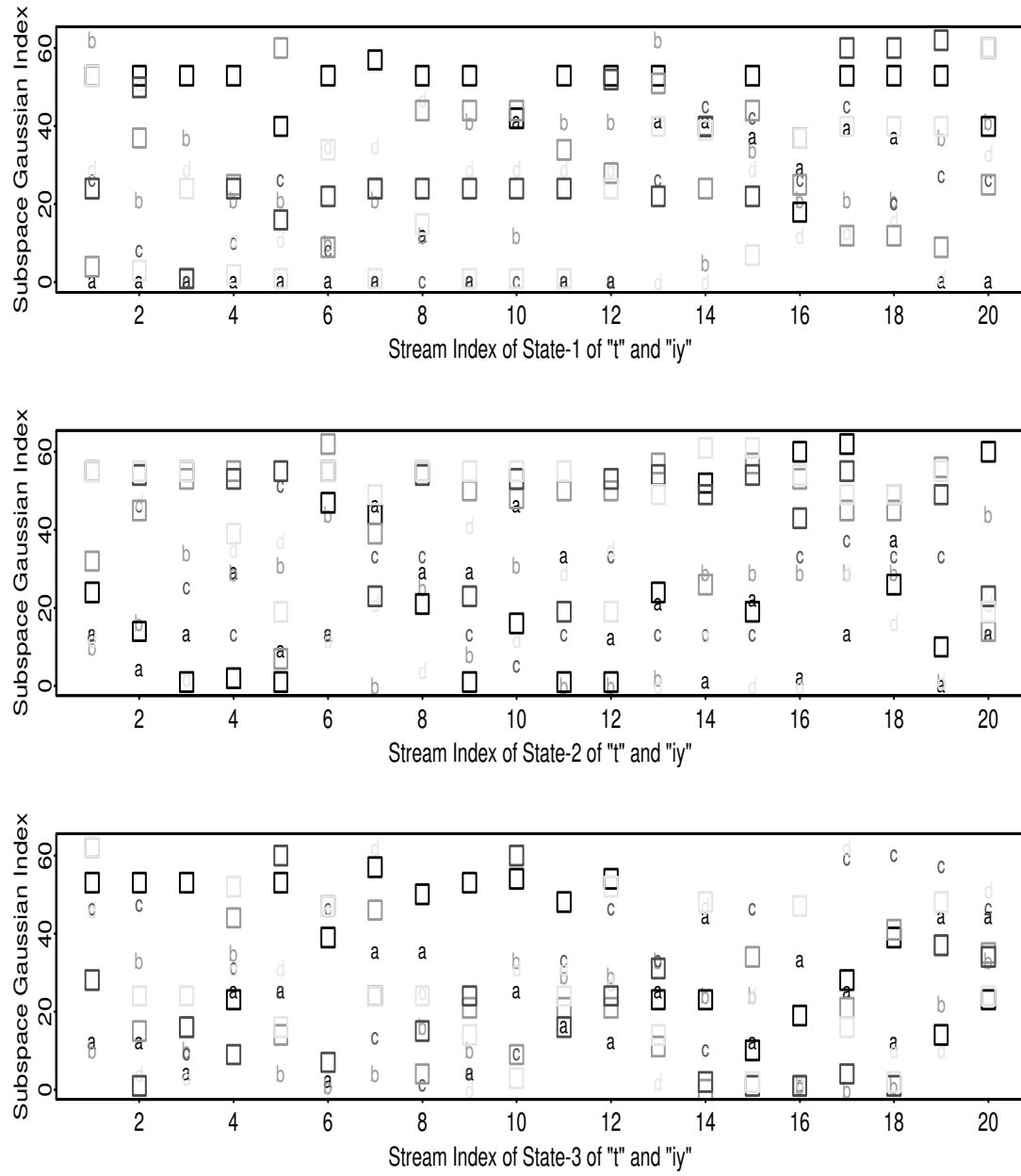
Figure 6.2: Subspace Gaussian tying structure (c) between "t" and "iy" (number of matches from the 1st to the 3rd state are 0, 0, 5)

Chapter 7

Implementation of SDCHMMs (II): Direct SDCHMM Training

Using the model conversion technique explained in Chapter 5, subspace distribution clustering hidden Markov models (SDCHMMs) can be trained from raw speech data in an indirect training scheme as shown in Figure 7.1(a).

Model conversion is simple and runs quickly, and the ensuing SDCHMMs fulfill three promises of the technique of parameter tying:

- fewer model parameters

- faster computation of model likelihoods

- no loss in recognition accuracy (and possibly some gain).

However, since the scheme requires intermediate CDHMMs, it requires an amount of training data as large as CDHMM training does, and does not take advantage of the fewer model parameters in SDCHMMs. Recognition performance of SDCHMMs in Table 5.2 indicates that, if the subspace Gaussian tying structure (SGTS)[1] is ignored, SDCHMMs have many fewer model parameters (mixture weights, Gaussian means, and variances) — by one to two orders of magnitude — than their parent CDHMMs. Thus, if we have *a priori* knowledge of the SGTS, or in general, the subspace distribution tying structure (SDTS), one should be able to train SDCHMMs directly from much less speech data as schematically

---

[1]SGTS is defined in Chapter 6 as the tying information among subspace Gaussians of SDCHMMs together with the mappings between them and the full-space Gaussians of CDHMMs.

78
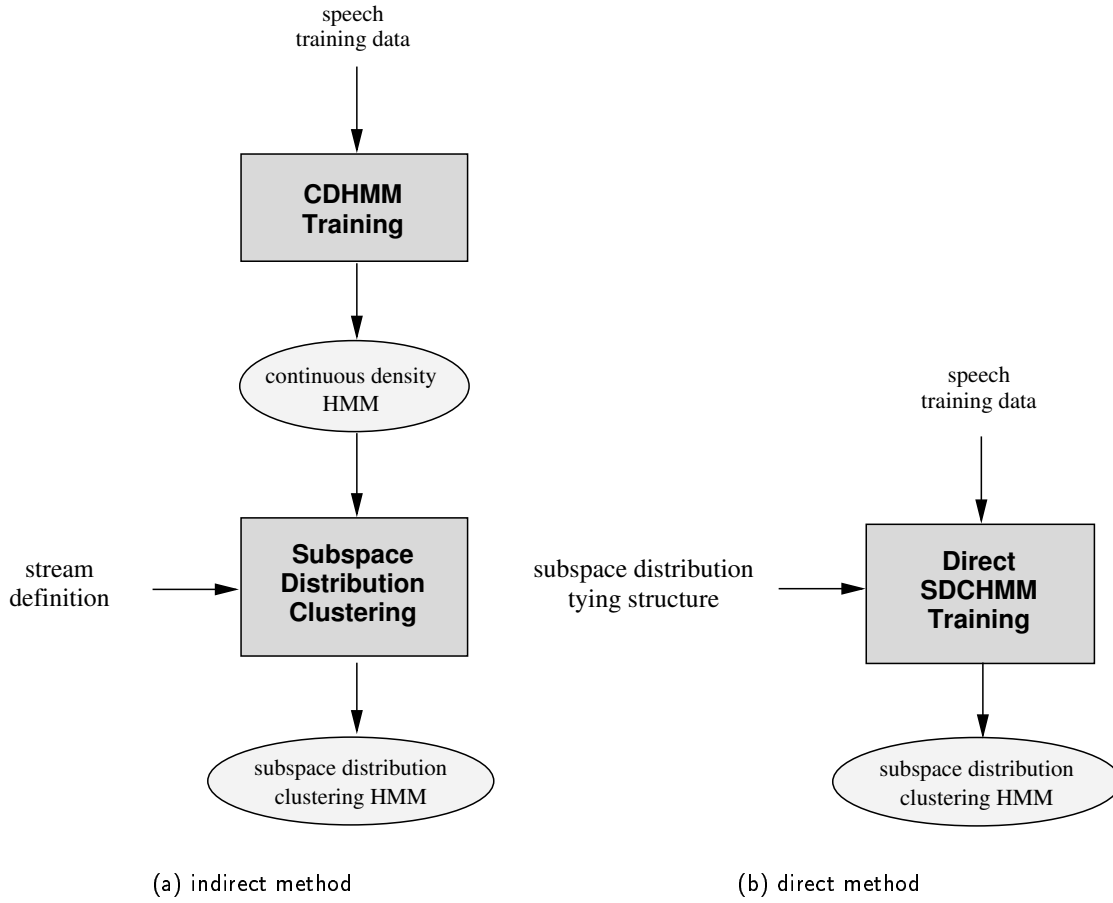
Figure 7.1: SDCHMM training schemes

shown in Figure 7.1(b). That an SGTS imposes a great constraint on the HMM config-
uration may have one wonder why a presumed SGTS necessarily leads to good acoustic
models. However, the acoustic-phonetic analysis of SGTS in Chapter 6 suggests that the
SGTS is not arbitrary; it efficiently represents the inherent inter-relationship among the
phones. The presumption of an SGTS should therefore be considered as a utilization of
phonetic knowledge in designing our acoustic models, resulting in fewer model parameters
and theoretically requiring less training data.

In this chapter, we first review the reestimation formulas of CDHMM parameters, and
then extend them to the reestimation of SDCHMM parameters. It will be shown that
SDCHMM reestimation is just a special case of CDHMM reestimation where statistics

are gathered in a way dictated by the SGTS. Then, in a set of experiments wherein ATIS training data are progressively halved each time, the data requirement for directly training SDCHMMs from scratch is investigated, and is compared with that for training CDHMMs.

7.1 A Review of Maximum Likelihood Estimation of CDHMM Parameters Using the EM Algorithm (with Single Observation Sequence)

An N-state HMM λ is defined by three sets of parameters:

- initial-state probabilities $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_N]$

- state-transition probability matrix $\boldsymbol{a} = \{a_{ij}\}, 1 \leq i, j \leq N$

- state observation pdf's $\boldsymbol{b} = [b_1, b_2, \ldots, b_N]$.

Given an observation sequence of T frames of a speech unit, $\boldsymbol{O} = \boldsymbol{o}_1 \boldsymbol{o}_2 \cdots \boldsymbol{o}_T$ (where $\boldsymbol{o}_t$ is the observation vector at time t), and a pre-defined HMM configuration for the unit, the estimation problem is to find an HMM $\lambda \equiv (\boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b})$ such that a score function $\mathcal{C}(\boldsymbol{O}, \lambda)$ is maximized. Cast as an optimization problem, the estimation is most commonly solved by the iterative *Baum-Welch* (BW) algorithm [3], a specific case of the *Expectation–Maximization* (EM) algorithm [30] with the log likelihood as the score function. That is,

$$\mathcal{C}(\boldsymbol{O}, \lambda) \stackrel{\text{def}}{=} \log P(\boldsymbol{O} \mid \lambda)$$

In each iteration of the EM algorithm, the current model parameters $\lambda \equiv (\boldsymbol{\pi}, \boldsymbol{a}, \boldsymbol{b})$ are reestimated to $\hat{\lambda} \equiv (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{b}})$ which maximizes the score function. This is done sequentially in two steps: an E-step and an M-step.

7.1.1 E-step

An auxiliary function, commonly called the Q function, for the expected cost of the new model conditional on the observation and the old model is constructed as follows:

$$Q(\lambda, \hat{\lambda}) \stackrel{\text{def}}{=} \mathcal{E}_{\boldsymbol{q}} \left[\log P(\boldsymbol{O}, \boldsymbol{q} \mid \hat{\lambda}) \mid \boldsymbol{O}, \lambda \right] \tag{7.1}$$

$$= \sum_{\boldsymbol{q}} P(\boldsymbol{q} \mid \boldsymbol{O}, \lambda) \cdot \log P(\boldsymbol{O}, \boldsymbol{q} \mid \hat{\lambda}) . \tag{7.2}$$

The expectation $\mathcal{E}[\cdot]$ is evaluated over all possible state sequences $\boldsymbol{q} = q_1 q_2 \cdots q_T$, which is the hidden data or the random variable of the stochastic Markov process.

The theory of EM algorithm guarantees a monotonic improvement in the value of the Q function after each iteration of the E-step and M-step. The final result is a locally, if not globally, optimal model.

7.1.2   M-step

Since

$$\log P(\boldsymbol{O}, \boldsymbol{q} \mid \hat{\lambda}) = \log \left(\hat{\pi}_{q_1} \hat{b}_{q_1}(\boldsymbol{o}_1) \hat{a}_{q_1 q_2} \hat{b}_{q_2}(\boldsymbol{o}_2) \cdots \hat{a}_{q_{T-1} q_T} \hat{b}_{q_T}(\boldsymbol{o}_T) \right) \tag{7.3}$$

$$= \log(\hat{\pi}_{q_1}) + \sum_{t=1}^{T-1} \log(\hat{a}_{q_t q_{t+1}}) + \sum_{t=1}^{T} \log(\hat{b}_{q_t}(\boldsymbol{o}_t)) , \tag{7.4}$$

Equation (7.2) may be split into a sum of three independent Q functions:

$$Q(\lambda, \hat{\lambda}) = Q_{\boldsymbol{\pi}}(\lambda, \hat{\boldsymbol{\pi}}) + \sum_{i=1}^{N} Q_{\boldsymbol{a}_i}(\lambda, \hat{\boldsymbol{a}}_i) + \sum_{i=1}^{N} Q_{b_i}(\lambda, \hat{b}_i) \tag{7.5}$$

where $\hat{\boldsymbol{a}}_i = [\hat{a}_{i1}, \hat{a}_{i2}, \ldots, \hat{a}_{iN}]$, and

$$Q_{\boldsymbol{\pi}}(\lambda, \hat{\boldsymbol{\pi}}) = \sum_{i=1}^{N} P(q_1 = i \mid \boldsymbol{O}, \lambda) \cdot \log(\hat{\pi}_i) \tag{7.6}$$

$$Q_{\boldsymbol{a}_i}(\lambda, \hat{\boldsymbol{a}}_i) = \sum_{j=1}^{N} \sum_{t=1}^{T-1} P(q_t = i, q_{t+1} = j \mid \boldsymbol{O}, \lambda) \cdot \log(\hat{a}_{ij}) \tag{7.7}$$

$$Q_{b_i}(\lambda, \hat{b}_i) = \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}, \lambda) \cdot \log(\hat{b}_i(\boldsymbol{o}_t)) . \tag{7.8}$$

Maximization of $Q(\lambda, \hat{\lambda})$ can then be done by maximizing the three independent Q functions separately, since each involves a different set of optimization variables. By taking the first derivative of each of the Equations (7.6)–(7.8) and using appropriate Lagrange multipliers, the maximal value of $\hat{\lambda}$ is

$$\hat{\pi}_i = \gamma_1^{\lambda}(i) \tag{7.9}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t^{\lambda}(i, j)}{\sum_{t=1}^{T-1} \gamma_t^{\lambda}(i)} , \tag{7.10}$$

where

$$\gamma_t^{\lambda}(i) \stackrel{\text{def}}{=} P(q_t = i \mid \boldsymbol{O}, \lambda) \tag{7.11}$$

is the probability of being in state i at time t, and

$$\xi_t^\lambda(i,j) \stackrel{\text{def}}{=} P(q_t = i, q_{t+1} = j \mid \boldsymbol{O}, \lambda) \tag{7.12}$$

is the probability of being in state i at time t and state j at time $t+1$, given the model λ and the observation sequence $\boldsymbol{O}$. The likelihood functions $\gamma(\cdot)$ and $\xi(\cdot)$ can be efficiently computed by the *forward-backward* algorithm [69].

The reestimation formula of $\boldsymbol{b}$ depends on the functional form of the state observation pdf. Here, we will consider only the two cases when the state output distribution is either a single Gaussian distribution or a mixture Gaussian density.

Case I: Single Gaussian Output Distribution

That is, $b_i(\boldsymbol{o}_t) = \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$. Then

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{t=1}^{T} \gamma_t^\lambda(i)\, \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t^\lambda(i)} \tag{7.13}$$

$$\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{t=1}^{T} \gamma_t^\lambda(i)\, (\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i)'}{\sum_{t=1}^{T} \gamma_t^\lambda(i)} \tag{7.14}$$

where $(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i)'$ is the transpose of $(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i)$.

Conceptually, the reestimate of the mean (covariance) of a Gaussian of state i is the normalized sum of its observations (cross product of deviations from mean observations) weighted by their probability of being in state i.

Case II: Mixture Gaussian Output Distribution

That is, $b_i(\boldsymbol{o}_t) = \sum_{m=1}^{M} c_{im}\, \mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}), \quad \sum_{m=1}^{M} c_{im} = 1$.

Since an HMM state with a mixture density is equivalent to a multi-state HMM with single-mixture densities [36], the reestimates of $\boldsymbol{b}$ are similar to those of Case I except that the quantity $\gamma_t^\lambda(i)$ is modified as $\gamma_t^\lambda(i,m)$ which is the probability of being in state i and the m-th mixture component at time t, given the model λ and the observation sequence $\boldsymbol{O}$. Hence,

$$\hat{c}_{im} = \frac{\sum_{t=1}^{T} \gamma_t^\lambda(i,m)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t^\lambda(i,m)} \tag{7.15}$$

$$\hat{\boldsymbol{\mu}}_{im} = \frac{\sum_{t=1}^{T} \gamma_t^\lambda(i,m)\, \boldsymbol{o}_t}{\sum_{t=1}^{T} \gamma_t^\lambda(i,m)} \tag{7.16}$$

$$\hat{\boldsymbol{\Sigma}}_{im} = \frac{\sum_{t=1}^{T} \gamma_t^{\lambda}(i, m) \, (\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i)(\boldsymbol{o}_t - \hat{\boldsymbol{\mu}}_i)'}{\sum_{t=1}^{T} \gamma_t^{\lambda}(i, m)} \; . \tag{7.17}$$

7.1.3 Viterbi Training

A simple variant of the Baum-Welch training algorithm, called the Viterbi training algorithm, is commonly used in practice. It is simpler and faster, and by many reports as effective as the Baum-Welch algorithm. Essentially, instead of considering all possible state segmentations to decide the updates as in the Baum-Welch algorithm, Viterbi training uses only the most likely state sequence. That is, Viterbi training modifies the Q function as follows:

$$Q(\lambda, \hat{\lambda}) \stackrel{\text{def}}{=} P(\boldsymbol{q}_{max} \mid \boldsymbol{O}, \lambda) \cdot \log P(\boldsymbol{O}, \boldsymbol{q}_{max} \mid \hat{\lambda}) \tag{7.18}$$

where

$$\boldsymbol{q}_{max} = \underset{\boldsymbol{q}}{\operatorname{argmax}} \; P(\boldsymbol{O}, \boldsymbol{q} \mid \lambda) \; . \tag{7.19}$$

All the reestimation formulas of $\boldsymbol{\pi}$, $\boldsymbol{a}$, and $\boldsymbol{b}$ for the Baum-Welch method shown above can be readily adapted for Viterbi training by simply changing the definitions of the likelihood functions $\gamma(\cdot)$ and $\xi(\cdot)$ as follows:

$$\gamma_t^{\lambda}(i) = \begin{cases} 1 \text{ if } q_t = i \\ 0 \text{ otherwise} \end{cases}$$

$$\xi_t^{\lambda}(i, j) = \begin{cases} 1 \text{ if } q_t = i \text{ and } q_{t+1} = j \\ 0 \text{ otherwise} \end{cases} .$$

7.2 Extension to Maximum Likelihood Estimation of SD-CHMM Parameters

SDCHMM parameters may be estimated using the EM algorithm in much the same way as CDHMM parameters are estimated above. In fact, the additional constraints imposed by the SDTS only alter the way in which statistics are gathered from the observations in the estimation of distribution parameters. Moreover, since the SDTS concerns *all* acoustic

models, the main difference between CDHMM estimation and SDCHMM estimation is that while each CDHMM may be estimated in isolation, *all* SDCHMMs have to be estimated at the same time.

In the following, let us denote the whole set of SDCHMMs of all speech units by Λ, and augment each observation sequence $\boldsymbol{O}$ and model parameters $\boldsymbol{\pi}$, $\boldsymbol{a}_i$, and b_i to $\boldsymbol{O}^\lambda$, $\boldsymbol{\pi}^\lambda$, $\boldsymbol{a}_i^\lambda$, and b_i^λ respectively to make their model dependency explicit. The new Q function is modified as:

$$Q(\Lambda, \hat{\Lambda}) \stackrel{\text{def}}{=} \mathcal{E}_{\boldsymbol{q}} \left[\log \prod_{\lambda \in \Lambda} P(\boldsymbol{O}^\lambda, \boldsymbol{q} \mid \hat{\lambda}) \,\middle|\, \boldsymbol{O}^\Lambda, \Lambda \right] \tag{7.20}$$

$$= \sum_{\lambda \in \Lambda} \sum_{\boldsymbol{q}} P(\boldsymbol{q} \mid \boldsymbol{O}^\lambda, \lambda) \cdot \log P(\boldsymbol{O}^\lambda, \boldsymbol{q} \mid \hat{\lambda}) \tag{7.21}$$

$$= \sum_{\lambda \in \Lambda} Q_{\boldsymbol{\pi}}(\lambda, \hat{\boldsymbol{\pi}}^\lambda) + \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} Q_{\boldsymbol{a}_i}(\lambda, \hat{\boldsymbol{a}}_i^\lambda) + \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} Q_{b_i}(\lambda, \hat{b}_i^\lambda) \,. \tag{7.22}$$

where $Q_{\boldsymbol{\pi}}(\lambda, \hat{\boldsymbol{\pi}}^\lambda)$, $Q_{\boldsymbol{a}_i}(\lambda, \hat{\boldsymbol{a}}_i^\lambda)$ and $Q_{b_i}(\lambda, \hat{b}_i^\lambda)$ are defined as in Equations (7.6)–(7.8).

7.2.1   Reestimation of $\boldsymbol{\pi}$ and $\boldsymbol{a}$ in SDCHMM

It is clear that from the theory of SDCHMM (Equation (4.5)) that only the state observation pdf $b_i^\lambda(\cdot)$ of the CDHMM is modified, while the definitions of the initial-state probabilities $\boldsymbol{\pi}$ and state-transition probabilities $\boldsymbol{a}$ are kept intact. Hence, $\boldsymbol{\pi}$ and $\boldsymbol{a}$ can still be estimated separately for each SDCHMM, and their reestimation formulas remain the same as those of the conventional CDHMM given by Equations (7.9) and (7.10) respectively.

7.2.2   Reestimation of b in SDCHMM

According to the theory of SDCHMM, the state observation pdf $b_i^\lambda(\cdot)$ of state i of a K-stream SDCHMM λ is assumed to be a mixture density with M components $b_{im}^\lambda(\cdot)$ and mixture weights c_{im}, $1 \leq m \leq M$, such that $b_{im}^\lambda(\cdot)$ is a product of K subspace pdf's $b_{imk}^\lambda(\cdot)$, $1 \leq k \leq K$, of the same functional form. That is,

$$b_i^\lambda(\boldsymbol{o}_t^\lambda) = \sum_{m=1}^{M} c_{im} b_{im}^\lambda(\boldsymbol{o}_t^\lambda), \quad \sum_{m=1}^{M} c_{im} = 1 \tag{7.23}$$

$$= \sum_{m=1}^{M} \left(c_{im} \prod_{k=1}^{K} b_{imk}^{\lambda}(\boldsymbol{o}_{tk}^{\lambda}) \right) \tag{7.24}$$

where $b_{imk}^{\lambda}(\cdot)$ and $\boldsymbol{o}_{tk}^{\lambda}$ are the projections of $b_{im}^{\lambda}(\cdot)$ and $\boldsymbol{o}_{t}^{\lambda}$ onto the k-th feature subspace respectively.

The reestimation formula for the mixture weights c_{im} is the same as in the case of CDHMM given by Equation (7.15) since it does not depend on the functional form of the component distribution. For the reestimation of component distribution, again, only a single Gaussian distribution and a mixture Gaussian density are considered.

Let us denote the whole set of state output distributions of all models by $\boldsymbol{B}$. From Equation (7.22), the Q function for $\boldsymbol{B}$ is given by

$$Q_{\boldsymbol{B}}(\Lambda, \hat{\boldsymbol{B}}) = \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} Q_{b_i}(\lambda, \hat{b}_i^{\lambda}) \ . \tag{7.25}$$

Case I: Single Gaussian Output Distribution

Let us first look at the special case when there is only one Gaussian in the mixture density. Equation (7.24) may then be simplified to

$$b_i^{\lambda}(\boldsymbol{o}_t^{\lambda}) = \prod_{k=1}^{K} b_{ik}^{\lambda}(\boldsymbol{o}_{tk}^{\lambda}) \tag{7.26}$$

by dropping the mixture weight of unity and the mixture component subscript m. Substituting Equation (7.8) and Equation (7.26) into Equation (7.25), we have

$$
\begin{aligned}
Q_{\boldsymbol{B}}(\Lambda, \hat{\boldsymbol{B}}) &= \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^{\lambda}, \lambda) \cdot \log(\hat{b}_i^{\lambda}(\boldsymbol{o}_t^{\lambda})) \\
&= \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^{\lambda}, \lambda) \cdot \left(\log \prod_{k=1}^{K} \hat{b}_{ik}^{\lambda}(\boldsymbol{o}_{tk}^{\lambda}) \right) \\
&= \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^{\lambda}, \lambda) \cdot \left(\sum_{k=1}^{K} \log(\hat{b}_{ik}^{\lambda}(\boldsymbol{o}_{tk}^{\lambda})) \right) \\
&= \sum_{k=1}^{K} \left(\sum_{\lambda \in \Lambda} \sum_{i=1}^{N} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^{\lambda}, \lambda) \cdot \log(\hat{b}_{ik}^{\lambda}(\boldsymbol{o}_{tk}^{\lambda})) \right) \\
&\equiv \sum_{k=1}^{K} Q_{\boldsymbol{B}_k}(\Lambda, \hat{\boldsymbol{B}}_k) \tag{7.27}
\end{aligned}
$$

where

$$Q_{\boldsymbol{B}_k}(\Lambda, \hat{\boldsymbol{B}}_k) = \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^\lambda, \lambda) \cdot \log(\hat{b}_{ik}^\lambda(\boldsymbol{o}_{tk}^\lambda)) \; . \qquad (7.28)$$

As the streams are assumed independent in the *local* acoustic space, each $Q_{\boldsymbol{B}_k}(\Lambda, \hat{\boldsymbol{B}}_k)$ can be maximized independently.

Now suppose there are L_k subspace pdf prototypes $h_{kl}(\cdot)$, $1 \le l \le L_k$, in the k-th stream of the set of K-stream SDCHMMs Λ, $1 \le k \le K$. Each subspace pdf, say, $b_{ik}^\lambda(\cdot)$ in stream k of state i, is tied to one of the subspace pdf prototypes of the stream, say, $h_{kl}(\cdot)$, $1 \le l \le L_k$. That is, $\forall \lambda \in \Lambda$, $\forall i \in [1, N]$, $\forall k \in [1, K]$, $\exists l \in [1, L_k]$ such that $b_{ik}^\lambda(\cdot) \equiv h_{kl}(\cdot)$. Then the reestimation of $b_{ik}^\lambda(\cdot)$ becomes the reestimation of $h_{kl}(\cdot)$ and may be expressed verbally as follows:

reestimation of the parameters of pdf $h_{kl}(\cdot)$	=	reestimation of the pdf parameters as in conventional CDHMM, *but* the statistics are gathered from all frames belonging to **all** $b_{ik}^\lambda(\cdot) \equiv h_{kl}(\cdot)$ over **all** states and **all** models.

Thus the Q function for the subspace pdf's in the k-th stream of Equation (7.28) can be rewritten as:

$$Q_{\boldsymbol{B}_k}(\Lambda, \hat{\boldsymbol{B}}_k) = \sum_{\lambda \in \Lambda} \sum_{i=1}^{N} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^\lambda, \lambda) \cdot \log(\hat{h}_{kl}(\boldsymbol{o}_{tk}^\lambda)) \quad \text{such that} \quad b_{ik}^\lambda(\cdot) \equiv h_{kl}(\cdot)$$

$$= \sum_{l=1}^{L_k} \left(\sum_{\lambda \in \Lambda_{i \; : \; b_{ik}^\lambda \equiv h_{kl}}} \sum_{t=1}^{T} P(q_t = i \mid \boldsymbol{O}^\lambda, \lambda) \cdot \log(\hat{h}_{kl}(\boldsymbol{o}_{tk}^\lambda)) \right) \; . \qquad (7.29)$$

In particular if the pdf's are Gaussians, that is,

$$h_{kl}(\boldsymbol{o}_{tk}) = N(\boldsymbol{o}_{tk}; \boldsymbol{\mu}_{kl}, \boldsymbol{\Sigma}_{kl})$$

then the new model is

$$\hat{\boldsymbol{\mu}}_{kl} = \frac{\sum_{\lambda \in \Lambda} \sum_{i \; : \; b_{ik}^\lambda \equiv h_{kl}} \sum_{t=1}^{T} \gamma_t^\lambda(i) \cdot \boldsymbol{o}_{tk}^\lambda}{\sum_{\lambda \in \Lambda} \sum_{i \; : \; b_{ik}^\lambda \equiv h_{kl}} \sum_{t=1}^{T} \gamma_t^\lambda(i)} \qquad (7.30)$$

$$\hat{\boldsymbol{\Sigma}}_{kl} = \frac{\sum_{\lambda \in \Lambda} \sum_{i \; : \; b_{ik}^\lambda \equiv h_{kl}} \sum_{t=1}^{T} \gamma_t^\lambda(i)(\boldsymbol{o}_{tk}^\lambda - \hat{\boldsymbol{\mu}}_{kl})(\boldsymbol{o}_{tk}^\lambda - \hat{\boldsymbol{\mu}}_{kl})'}{\sum_{\lambda \in \Lambda} \sum_{i \; : \; b_{ik}^\lambda \equiv h_{kl}} \sum_{t=1}^{T} \gamma_t^\lambda(i)} \; . \qquad (7.31)$$

Case II: Mixture Gaussian Output Distribution

Again extending the reestimation of $\boldsymbol{b}$ to the general case of mixture densities can simply be done by taking into account all mixture components, and substituting $\gamma_t^\lambda(i, m)$ for $\gamma_t^\lambda(i)$ in Equations (7.30) and (7.31) as follows:

$$\hat{\boldsymbol{\mu}}_{kl} = \frac{\sum_{\lambda \in \Lambda} \sum_{i,m \,:\, b_{imk}^\lambda \equiv h_{kl}} \sum_{t=1}^T \gamma_t^\lambda(i, m) \cdot \boldsymbol{o}_{tk}^\lambda}{\sum_{\lambda \in \Lambda} \sum_{i,m \,:\, b_{imk}^\lambda \equiv h_{kl}} \sum_{t=1}^T \gamma_t^\lambda(i, m)} \qquad (7.32)$$

$$\hat{\boldsymbol{\Sigma}}_{kl} = \frac{\sum_{\lambda \in \Lambda} \sum_{i,m \,:\, b_{imk}^\lambda \equiv h_{kl}} \sum_{t=1}^T \gamma_t^\lambda(i, m)(\boldsymbol{o}_{tk}^\lambda - \hat{\boldsymbol{\mu}}_{kl})(\boldsymbol{o}_{tk}^\lambda - \hat{\boldsymbol{\mu}}_{kl})'}{\sum_{\lambda \in \Lambda} \sum_{i,m \,:\, b_{imk}^\lambda \equiv h_{kl}} \sum_{t=1}^T \gamma_t^\lambda(i, m)} \;. \qquad (7.33)$$

7.2.3 Remarks

Although the reestimation formulas of SDCHMMs look much like those of semi-continuous HMMs (SCHMMs) (or tied-mixture HMM), there are several important differences:

- While all SCHMM states share the same set of subspace Gaussians, the tying of subspace Gaussians among the SDCHMM states is governed by the SDTS and is generally not the same for all the states. This requires the modifier $\sum_{i,m \,:\, b_{imk}^\lambda \equiv h_{kl}}$, in the reestimation formulas of SDCHMM.

- Since streams in an SCHMM are assumed globally independent, the model parameters of each stream are estimated independently. Thus, for a K-stream SCHMM, there will be K different sets of $\boldsymbol{\pi}$, $\boldsymbol{a}$, state density mixture weights and distribution prototypes. On the other hand, streams are locally independent in an SDCHMM. Except for the K different sets of distribution prototypes, one for each stream, a K-stream SDCHMM has only one set of other model parameters ($\boldsymbol{\pi}$, $\boldsymbol{a}$, and state density mixture weights).

- Similarly, the likelihood functions $\gamma(\cdot)$ and $\xi(\cdot)$ are computed only once for all streams of an SDCHMM, whereas an SCHMM requires their computation separately for each stream.

7.3 Evaluation of Direct SDCHMM Training

In this section, we study the following problem:

> If the "perfect" subspace Gaussian tying structure for the acoustic models of a
> task is known, how much training data is required to directly train SDCHMMs
> for the task?

An SGTS used for direct SDCHMM training of a task is considered "perfect" if it is
obtained through model conversion of CDHMMs to SDCHMMs of the same task. On the
other hand, an SGTS is said to be "imperfect" for SDCHMM training if it is obtained
from model conversion of CDHMMs of a different task, or CDHMMs of the same task but
acquired in a different environment (ambient noise, channel, gender, etc.). The perfect
SGTS is employed in this thesis to study the upper bound for the effectiveness of direct
SDCHMM training when the exact SGTS for a task is known.

The ATIS task is again chosen for evaluating the direct SDCHMM training scheme.
Both context-independent (CI) and context-dependent (CD) SDCHMMs will be trained
and evaluated. Nonetheless, more emphasis is put on the CI models simply because the
simpler and fewer CI models allow us to train and test many CDHMMs and SDCHMMs
of various complexities in a manageable amount of time. Moreover, CI modeling tends
to be more stable as there is usually ample coverage of training data for the CI phones.
In contrast, CD modeling requires delicate fine-tuning effort to obtain a good balance
between training data and model accuracy, which may complicate our main research goal
here.

Speech is converted into 39-dimensional feature vectors as described in Section 3.2.1.
Each phone model is a 3-state left-to-right HMM with the exception of one noise model
which has only one state. The testing conditions (test dataset, vocabulary, pronunciation
models, language models, decoding algorithm, and beam-width) are exactly the same as
those described in Table 3.3. All these have been used consistently throughout this thesis.

Lastly, the number of streams is fixed to 20 for all SDCHMMs trained below. This
follows from the conclusion in Chapter 5 which suggests that 20 streams give a good
balance between accuracy, computation time, and model memory on the ATIS task.

7.3.1 Methodology

To evaluate the effectiveness of direct SDCHMM training, its training data requirement is compared with that for CDHMM training. The evaluation procedure consists of the following basic steps:

Step 1. Generate N data subsets $\mathcal{S}_i$, $1 \leq i \leq N$, from all the given training data by progressively cutting the data in half. That is, the amount of data in $\mathcal{S}_{i+1}$ is half of that in $\mathcal{S}_i$.

Step 2. Train CDHMM acoustic models with *all* available training data in $\mathcal{S}_1$.

Step 3. Convert the CDHMMs to SDCHMMs as described in Chapter 5 (Figure 7.1(a)).

Step 4. Deduce the subspace distribution tying structure from the converted SDCHMMs.

Step 5. For each data subset $(\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots, \mathcal{S}_N)$, repeat Steps 6 and 7.

Step 6. Train CDHMMs and adjust (lower) the number of components in each state mixture density to obtain the best CDHMMs with the reduced amount of training data.

Step 7. Train SDCHMM acoustic models using the direct SDCHMM training scheme as shown in Figure 7.1(b) with the SDTS obtained in Step 4.

Step 8. Compare the recognition performance of all CDHMMs and SDCHMMs obtained in the above steps.

7.3.2 Preparation of Training Datasets

A collection of 16,896 utterances from the ATIS-2 and ATIS-3 corpora (see Section 3.1), which were acquired at five sites (BBN, CMU, MIT, NIST, and SRI), are employed in this study. They are divided into 16 datasets of roughly 1,000 utterances each, denoted as S1, S2, S3, ..., to S16, so that data from the five sites are spread out into each dataset as evenly as possible. The 100 longest utterances from S16 are selected for bootstrapping HMMs and this set is denoted as dataset A. Other smaller datasets are derived as follows:

Table 7.1: ATIS: Training datasets (∗ Datasets are phonetically labeled by the baseline ATIS recognizers. † Figures are averages.)

DATASET	#FRAMES	DURATION (min.)	DESCRIPTION
X	13,000,205	2,167	baseline CD CDHMMs training data
Y	6,444,959	1,074	baseline CI CDHMMs training data
Test	545,642	91	981 (1994 ARPA's official) test utterances
S1–16	8,883,240	1,480	16,896 utterances
S1–4	2,140,470	357	4,226 utterances
S1–2	1,080,650	180	2,114 utterances
S1	527,599	88	1,055 utterances
S0	249,565	42	500 utterances from subset S1
A∗	101,309	17	100 utterances from subset S16
B∗	49,616	8.3	50 utterances from subset A
C∗	27,811	4.6	25 utterances from subset B
D∗	12,421	2.1	12 utterances from subset C
E1–E10∗	7,758†	1.29†	15 utterances from subset S15
F1–F10∗	2,702†	0.45†	5 utterances from subset S15

- dataset S0 contains 500 utterances from dataset S1

- dataset B contains 50 utterances from dataset A

- dataset C contains 25 utterances from dataset B

- dataset D contains 12 utterances from dataset C

- E-sets comprise 10 datasets denoted as E1, E2, ..., E10, and each contains 15 utterances from dataset S15, three from each of the five collecting sites.

- F-sets comprise 10 datasets denoted as F1, F2, ..., F10, which are sub-sampled from the corresponding E-sets such that each contains five utterances, one from each of the five collecting sites.

All the various datasets are summarized in Table 7.1. Datasets S5 to S14 are not used at all in this study. Datasets A, B, C, D, the E-sets, and the F-sets are all phonetically labeled. This is done by aligning each utterance with its transcription through Viterbi

decoding using the baseline CI-CDHMM (CD-CDHMM) ATIS recognizers of Chapter 3 when CI (CD) SDCHMMs are trained.

7.3.3   Hybrid Viterbi/Baum-Welch Training Procedure

In this evaluation, we adopt a combination of Viterbi training (VT) and Baum-Welch (BW) reestimation to train all acoustic models, with an additional step of segmental k-means (SKM) training for CDHMM training. The hybrid VT/BW training procedure takes advantage of the simplicity of Viterbi training and the accuracy of Baum-Welch. The procedures for training CDHMMs and SDCHMMs are schematically depicted in Figure 7.2, and the details are described in Algorithm 5 and Algorithm 6.

The training procedures for CDHMMs and SDCHMMs are very similar, but the following differences are worthy of notice:

- While each CDHMM phone may be trained in isolation (using Viterbi training), all SDCHMM phones must be estimated at the same time since all of them contribute to the statistics of the subspace Gaussian prototypes.

- SDCHMM training does not need the segmental k-means algorithm to derive the required model complexity of M mixtures per state because the complexities of all models are defined in the given SGTS.

- The *asymmetry* in the SGTS is crucial to successful training of the SDCHMM. If we initialize all components of an M-mixture CDHMM in the same way (as SDCHMMs are initialized in Algorithm 6), they will remain identical after training. The resulting M-mixture CDHMM is functionally no different from a 1-mixture CDHMM. On the other hand, in the case of SDCHMM initialization, since it is impossible for all mixture components of all phones to tie to the prototypes in exactly the same way, each subspace Gaussian prototype will not receive the same set of observations after initialization.

Finally, since strictly left-to-right 3-state HMMs are used, all initial-state probabilities are zero, except those of the first states which have total probability of unity. That is, for all

(small set of)
labelled data

uniform state
segmentation

CDHMM
initialization

phone-level
Baum-Welch

bootstrapped
1-mixture
CDHMMs

segmental
k-means

phone-level
Baum-Welch

all training
data

M-mixture
CDHMMs

phone-level
segmentation

phone-level
Baum-Welch

(a) CDHMM training

(small set of)
labelled data

uniform state
segmentation

a priori knowledge of
subspace distribution
tying structure

SDCHMM
initialization

phone-level
Baum-Welch

bootstrapped
M-mixture
SDCHMMs

phone-level
segmentation

all training
data

phone-level
Baum-Welch

M-mixture
SDCHMMs

(b) SDCHMM training

Figure 7.2: Hybrid Viterbi/Baum-Welch training procedure

Algorithm 5: Hybrid Viterbi/Baum-Welch training algorithm for estimating CDHMM

Goal: To train CDHMM acoustic models. Each state output distribution is a mixture Gaussian density with M components and diagonal covariances.

Step 1. *Uniform State Segmentation:* Segment the small set of phonetically labeled bootstrap data into HMM states evenly. That is, if a labeled phone has T frames of speech, and is modeled by an N-state CDHMM, each state will have T/N frames.

Step 2. *CDHMM Initialization:* A 1-mixture CDHMM is initialized for each phone with the state-segmented data.

Step 3. *Phone-Level Baum-Welch (BW) Reestimation:* Each initial CDHMM is reestimated with several BW iterations until the model converges to obtain the bootstrapped 1-mixture CDHMMs. The boundaries of the labeled phones are kept intact during the reestimation.

Step 4. *Segmental k-means (SKM) training:* Segment *all* training data with the bootstrapped 1-mixture CDHMMs using Viterbi segmentation so that each frame of speech is labeled with an HMM state. Then for each state, cluster all the speech vectors of the state into a maximum of M ensembles using the k-means clustering method to obtain the required model complexity of M-mixture densities.

Step 5. *Phone-Level Baum-Welch Reestimation:* Refine the M-mixture CDHMMs with more BW iterations. Again phone boundaries determined from the Viterbi segmentation is fixed during the reestimation.

Step 6. *Phone-Level Viterbi Segmentation:* Phonetically re-segment *all* training data with the most current M-mixture CDHMMs. No state segmentation is required.

Step 7. Perform phone-level Baum-Welch reestimation.

Step 8. Repeat Steps 6 and 7 until the models converge.

---

models, $\pi_1 = 1.0$ and $\pi_2 = \pi_3 = 0.0$. We further simplify our training procedures by fixing all state-transition probabilities to 0.5 as many researchers have found that in practice, an estimated state-transition matrix makes no difference in recognition performance [46].

7.3.4 Experiment I: Effectiveness of Direct SDCHMM Training

We first check, for the same amount of training data, whether SDCHMMs trained by the direct SDCHMM training algorithm achieve the same recognition performance as that of the SDCHMMs converted from CDHMMs. Only CI models are trained in this experiment,

---

Algorithm 6: Hybrid Viterbi/Baum-Welch training algorithm for estimating SDCHMMs

---

Goal: To train SDCHMM acoustic models with a given subspace Gaussian tying structure (SGTS). Each state output distribution is a mixture Gaussian density with M components and diagonal covariances.

Step 1. *Uniform State Segmentation:* Segment the small set of phonetically labeled bootstrap data into HMM states evenly. That is, if a labeled phone has T frames of speech, and is modeled by an N-state SDCHMM, each state will have T/N frames.

Step 2. *SDCHMM Initialization:* Initialize all states of all phones with the uniformly-segmented state frames. Each frame is assumed to contribute equally to each Gaussian component of its M-mixture SDCHMM state — that is, the probability of a frame being generated by each Gaussian component is $1/M$.

Step 3. *Phone-Level Baum-Welch (BW) Reestimation:* All initial SDCHMMs are reestimated with several BW iterations until the models converge to obtain the bootstrapped M-mixture SDCHMMs. The boundaries of the labeled phones are kept intact during the reestimation.

Step 4. *Phone-Level Viterbi Segmentation:* Phonetically re-segment *all* training data with the most current M-mixture SDCHMMs so that each frame of speech is labeled with an HMM (phone). No state segmentation is required.

Step 5. *Phone-Level Baum-Welch Reestimation:* Refine the M-mixture SDCHMMs with more BW iterations. Again phone boundaries determined from the Viterbi segmentation is fixed during the reestimation.

Step 6. Repeat Steps 4 and 5 until the models converge.

---

and the SGTS from the converted SDCHMMs is used for direct SDCHMM training. The number of VT cycles and BW iterations at various stages of the training procedure are determined empirically when the expected log likelihoods of the models converge.

(A) Procedure

1. **Training of CDHMMs:** CDHMMs are trained with the dataset S1–4 (meaning a combination of S1, S2, S3, and S4). Following the CDHMM training procedure of Algorithm 5, the phonetically labeled data of dataset A is used to bootstrap a 1-mixture CDHMM for each of the 48 (CI) monophones. Five BW iterations are run after model initialization. Using the bootstrapped 1-mixture CDHMMs and all the

training utterances in S1–4, CDHMMs with 16-mixture or 32-mixture densities are obtained with one iteration of segmental k-means training. The models are then reestimated with 20 BW iterations. Lastly, one cycle of the hybrid VT/BW training with 10 BW iterations gives the final models. The number of Gaussians in the 16-mixture and 32-mixture CDHMMs are 2,143 and 4,086 respectively.

2. **Derivation of SGTS:** The 16-mixture and 32-mixture CDHMMs trained with S1–4 are converted to 20-stream SDCHMMs with 16, 32, 64, and 128 subspace Gaussian prototypes per stream. Recognition on the ATIS test data determines the best SDCHMMs in each case of model complexity: 128 prototypes for the 16-mixture SDCHMMs and 64 prototypes for the 32-mixture SDCHMMs. SGTS's are derived from the best sets of 16-mixture and 32-mixture SDCHMMs and are denoted as CI-SGTS-M16-n128 and CI-SGTS-M32-n64 respectively.

3. **Training of SDCHMMs:** Two sets of SDCHMMs are trained with the dataset S1–4 using each of the two SGTS's derived above. Following the SDCHMM training procedure of Algorithm 6, 20-stream SDCHMMs with the given SGTS are initialized with the bootstrap data from dataset A. Five BW iterations are run to get the bootstrapped SDCHMMs. All the training data are then phonetically re-labeled by Viterbi alignment, and another 5 BW iterations give us the final SDCHMMs.

(B) Result and Discussion

The recognition results of the three sets of models:

- CI CDHMMs trained from the dataset S1–4

- CI SDCHMMs converted from the CDHMMs (converted SDCHMMs)

- CI SDCHMMs directly trained from the dataset S1–4 using the SGTS of the converted SDCHMMs (trained SDCHMMs)

on the ATIS test data are shown in Table 7.2.

The new 16-mixture CI CDHMMs actually perform slightly better than the baseline 16-mixture CI CDHMMs (9.0% vs. 9.4% (see Table 3.3)), though they are trained with

Table 7.2: ATIS: Comparison of recognition accuracies among CI CDHMMs, CI SDCHMMs converted from the CDHMMs, and CI SDCHMMs estimated by direct SDCHMM training using the SGTS of the converted SDCHMMs

#MIXTURES PER STATE	TOTAL #GAUSSIAN COMPONENTS	#PROTOTYPES PER STREAM	WORD ERROR RATE (%)		
			CDHMM	CONVERTED SDCHMM	TRAINED SDCHMM
16	2143	128	9.0	9.5	9.3
32	4086	64	8.5	8.7	8.7

only ~4,000 utterances (while the baseline system is trained with ~12,000 utterances). The baseline system is trained using the standard SKM algorithm, while the new CDHMMs are trained by the new hybrid VT/BW algorithm. This shows that the VT/BW training algorithm works well.

Secondly, even though the baseline 16-mixture CDHMMs and the new 16-mixture CDHMMs are trained in very different ways, the 20-stream SDCHMMs converted from both sets of CDHMMs have exactly the same recognition performance (as judged by their word error rates (Table 3.3 vs. Table 7.2). This suggests that the model conversion scheme to create SDCHMMs from CDHMMs is robust.

Thirdly and most importantly, the SDCHMMs trained from scratch using our novel direct SDCHMM training algorithm perform as well as the converted SDCHMMs. The result demonstrates the effectiveness of direct SDCHMM training. In fact, the average Bhattacharyya distances per prototype between the converted SDCHMMs and the trained SDCHMMs for the two cases of CI-SGTS-M16-n128 and CI-SGTS-M32-n64 are 0.038 and 0.014 respectively. These distances translate to high Bhattacharyya errors[2] of 48.1% and 49.3% respectively, suggesting that the two sets of subspace Gaussians in the converted SDCHMMs and the trained SDCHMMs are very similar. Thus if one is only given the SGTS and the training data of a set of converted SDCHMMs, the SDCHMMs can be "recovered" by our direct SDCHMM training algorithm to a fair degree of approximation.

---

[2]Bhattacharyya error is defined here as $0.5 \exp(-Bhattacharyya\ distance) \times 100\%$. As a reference, should any two Gaussians be identical, their Bhattacharyya distance will be zero, giving a Bhattacharyya error of 50%.

7.3.5 Experiment II: Data Requirement for Training Context-Independent SDCHMM

Once the effectiveness of direct SDCHMM training is established, we go a step further to investigate how many training data are required. In this experiment, the data requirement for training CI SDCHMMs is compared to that for training CI CDHMMs using the methodology described in Section 7.3.1.

(A) Procedure

The same procedure for training CDHMMs as in Experiment I is repeated with five training datasets: A only, S0 only, S1 only, S1–2, and S1–4. Dataset A is used to bootstrap all models. The maximum number of mixtures[3] in each state density is also varied from one to 32 in powers of two.

Similarly, the same SDCHMM training procedure of Experiment I is repeated with the five datasets. In addition, we also train SDCHMMs with the smaller datasets: B only, C only, and D only. These latter SDCHMMs are only bootstrapped with the training data under study in each case (and *not* with dataset A) using three BW iterations. It is found that no more VT/BW cycles are needed as the models already converge after bootstrapping.

(B) Result and Discussion

The recognition accuracies of all CDHMMs and SDCHMMs trained above are shown in Figure 7.3. In addition, Table 7.3 presents the model complexities, in terms of the total number of Gaussians, of all the CDHMMs.

As the model complexity decreases, the accuracy or resolution power of HMMs is compromised. This may be caused by limited amount of training data, or by hard-limiting the number of mixtures in each state density. The effect is clearly observed in Table 7.3 and Figure 7.3: When the model complexity (measured in terms of the number of Gaussians) is reduced, the recognition accuracy drops. The recognition performance of all CDHMMs with different number of mixtures falls off when they are presented with fewer than 197 minutes of training speech (dataset S1–2). In contrast, the recognition performance of the 20-stream SDCHMMs trained with CI-SGTS-M16-n128 or CI-SGTS-M32-n64 using

---

[3]Note that the final number of mixtures in a density produced by the segmental k-means algorithm (Algorithm 1) can be fewer than what the user specifies, when there are too few training data in the state.

Table 7.3: ATIS: Number of Gaussians in CDHMMs trained with different datasets and various numbers of mixtures per state

#MIXTURES PER STATE	TRAINING DATASET				
	A	S0	S1	S1-2	S1-4
1	142	142	142	142	142
2	257	273	280	283	283
4	452	516	535	559	563
8	735	927	1022	1077	1117
16	1019	1563	1863	2050	2143
32	1249	2268	3150	3734	4086
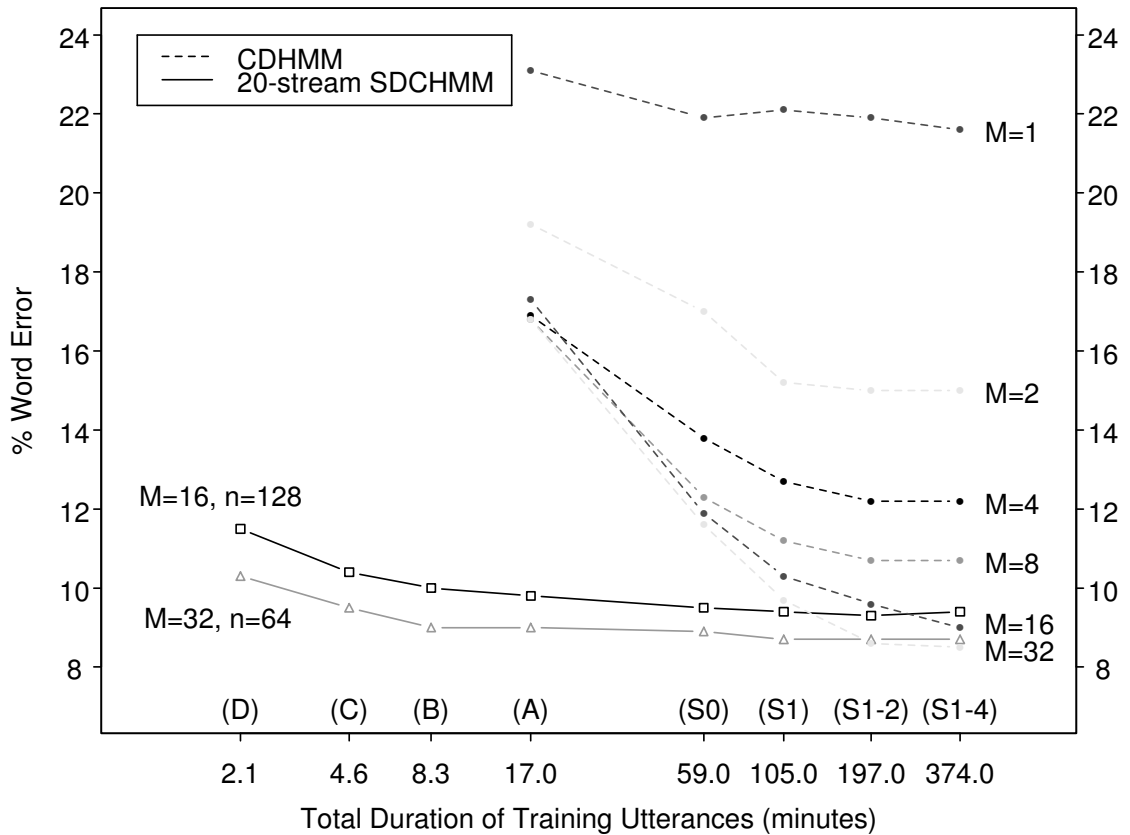
Figure 7.3: ATIS: Comparison between the amount of training data required for CDHMM training and direct SDCHMM training (M = #mixtures and n = #subspace Gaussian prototypes per stream)

the direct SDCHMM training algorithm does not start to fall significantly until there is less than 8.3 minutes of training speech (dataset B). Moreover, the performance of these two sets of SDCHMMs, trained with only 8.3 minutes of speech, is unmatched by any CDHMMs (with the same or simpler model complexity) trained with less than 197 minutes of speech in this study. This is a roughly 20-fold reduction in the amount of training data for SDCHMMs. The result should be attributed to the fewer model parameters (mixture weights, Gaussian means, and variances) of SDCHMMs — the ratios of the number of model parameters in the two SDCHMMs to that in their parent CDHMMs are 1:14 (for CI-SGTS-M16-n128) and 1:36 (for CI-SGTS-M32-n64).

Furthermore, as the amount of training data is reduced, the performance of SDCHMMs degrades gracefully whereas the performance of CDHMMs drops sharply. For example, when the amount of training data is pared down from 374 minutes (dataset S1–4) to 17 minutes (dataset A) the word error rates (WERs) of the 16-mixture and 32-mixture CDHMMs increases by almost 100%. On the other hand, the WER of the corresponding SDCHMMs trained using CI-SGTS-M16-n128 and CI-SGTS-M32-n64 drops by only ∼20% when the amount of training data is slashed from 374 minutes (dataset S1–4) to 2.1 minutes (dataset D). At first sight, this does not seem to be possible: For instance, when the 32-mixture SDCHMMs are trained with CI-SGTS-M32-n64 and the dataset D, there are only 12421 frames of speech to train the 4,086 Gaussians of the 48 monophones. That is, on average, there are about only 259 training frames per phone or three training frames per Gaussian! Even worse is the fact that some phones are rare, or do not even appear in the small training dataset D as shown in the frame distribution over the phones in Figure 7.4(a). For example, phones "hh" and "oy" do not occur in dataset D, and consonants like "el", "g", "jh", "nx", "th", and "uh" are rare. However, if one looks at the frame distribution over the 64 subspace Gaussians of each stream of the SDCHMMs in Figure 7.4(b), one should be convinced that there are ample estimation data for most of the subspace Gaussians (194 frames on average), and there is full coverage for all of them. Thus the efficient sharing of Gaussian parameters in the SDCHMMs plays an equally important role in reducing the training data requirements.

Another benefit of the greatly reduced number of model parameters in SDCHMMs is that the direct SDCHMM training procedure is simpler and faster than the CDHMM training scheme. For datasets larger than S0, direct SDCHMM training requires only one VT/BW pass on the bootstrapped models, while CDHMM training requires an extra SKM/BW pass. On smaller datasets, A to D, the models converge even faster — within
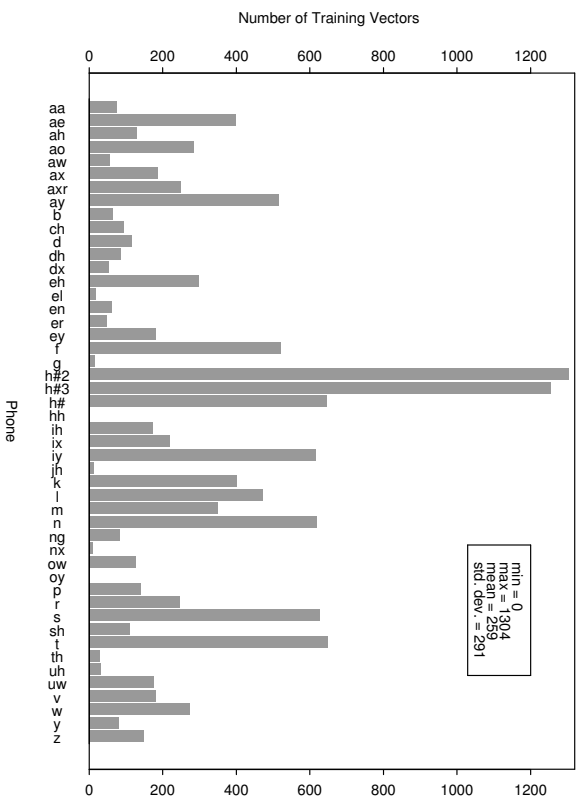
(a) Over the phones
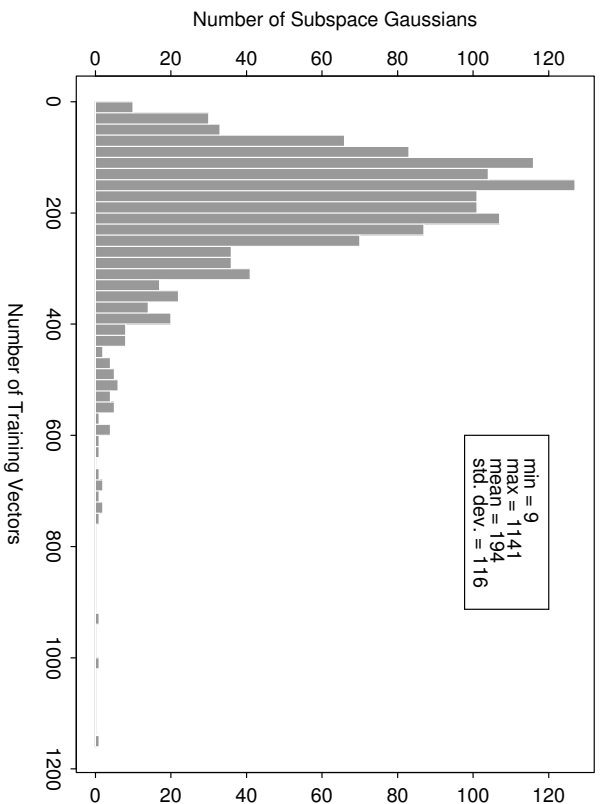
(b) Over the subspace Gaussians

Figure 7.4: Frame distribution of training dataset D (2.1 minutes, 12421 frames of speech)
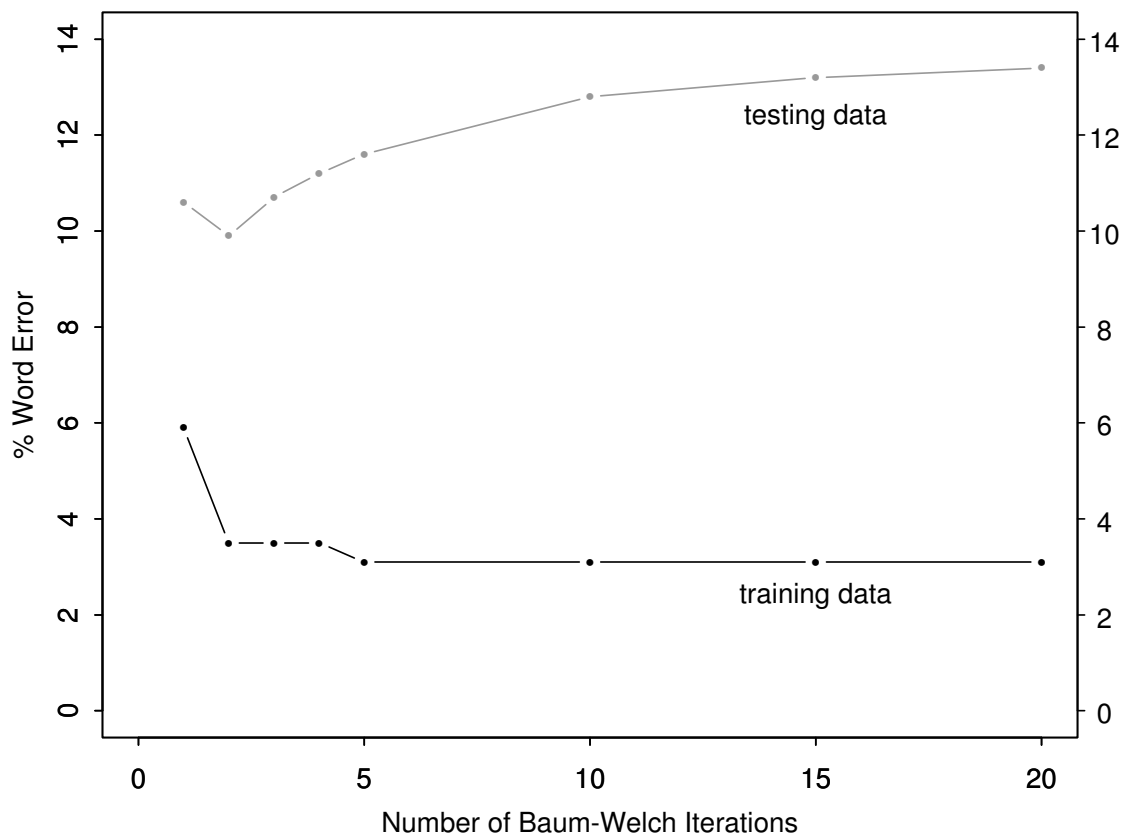
Figure 7.5: ATIS: Over-training with small amount of training data (dataset D, 2.1 minutes of speech)

the BW iterations during bootstrapping. SDCHMM training also goes through fewer BW iterations on the VT/BW pass than CDHMM training. However, there is one important caveat on training SDCHMMs with a small amount of data: We observe that they can easily be over-trained. For example, Figure 7.5 shows the WERs on both the training and testing data after each BW iteration during the SDCHMM training using CI-SGTS-M16-n128 and dataset D. The training flattens quickly after five BW iterations but over-training occurs after two BW iterations. Thus in practice, we need to stop training with a cross-validation technique.

7.3.6 Experiment III: Performance Variability with Little Training Data

(A) Procedure

When the amount of training data is small, the effect of random sampling of training data may become important. To check the performance variability of SDCHMM training with little training data, we repeat the SDCHMM training procedure of Experiment II with 20 even smaller datasets: E1 only, E2 only, ..., E10 only, F1 only, F2 only, ..., and F10 only. Each of the E-sets contains 15 utterances, and each of the F-sets contains five utterances, with durations ranging from 13.35 seconds to 97.82 seconds of speech. Both CI-SGTS-M16-n128 and CI-SGTS-M32-n64 are tried. We find that with these very small datasets, only one BW iteration after SDCHMM initialization is enough for model convergence.

(B) Result and Discussion

Figure 7.6 shows the scatter plots of the recognition accuracies of SDCHMMs trained with each of the two SGTS's over each of the 20 datasets. Superimposed on each scatter plot is a cubic B-spline fit generated by the statistical software S-PLUS [81]. The performance of the CI-SGTS-M32-n64 SDCHMMs degrades more slowly than that of the CI-SGTS-M16-n128 SDCHMMs when the amount of training data decreases. This is clearly due to the fact that there are even fewer model parameters and more sharing among the subspace Gaussians of the CI-SGTS-M32-n64 SDCHMMs. Nonetheless, it is observed that the 20 individual recognition results for each set of SDCHMMs fit well into the curve-fitting spline with only small fluctuations. Combining these results with those of Experiment II, we see a consistent trend that SDCHMMs can be trained with many fewer data over different samples of training sets.

7.3.7 Experiment IV: Data Requirement for Training Context-Dependent SDCHMM

Since Experiment II already shows that context-independent SDCHMMs require much less training data than CDHMMs, we next investigate if context-dependent (CD) SDCHMMs also require little training data. Thus, only CD SDCHMMs are trained.

(A) Procedure

As mentioned before, CD modeling requires more fine tuning to control the phonetic coverage (e.g. through using other parameter tying techniques such as state tying). In order
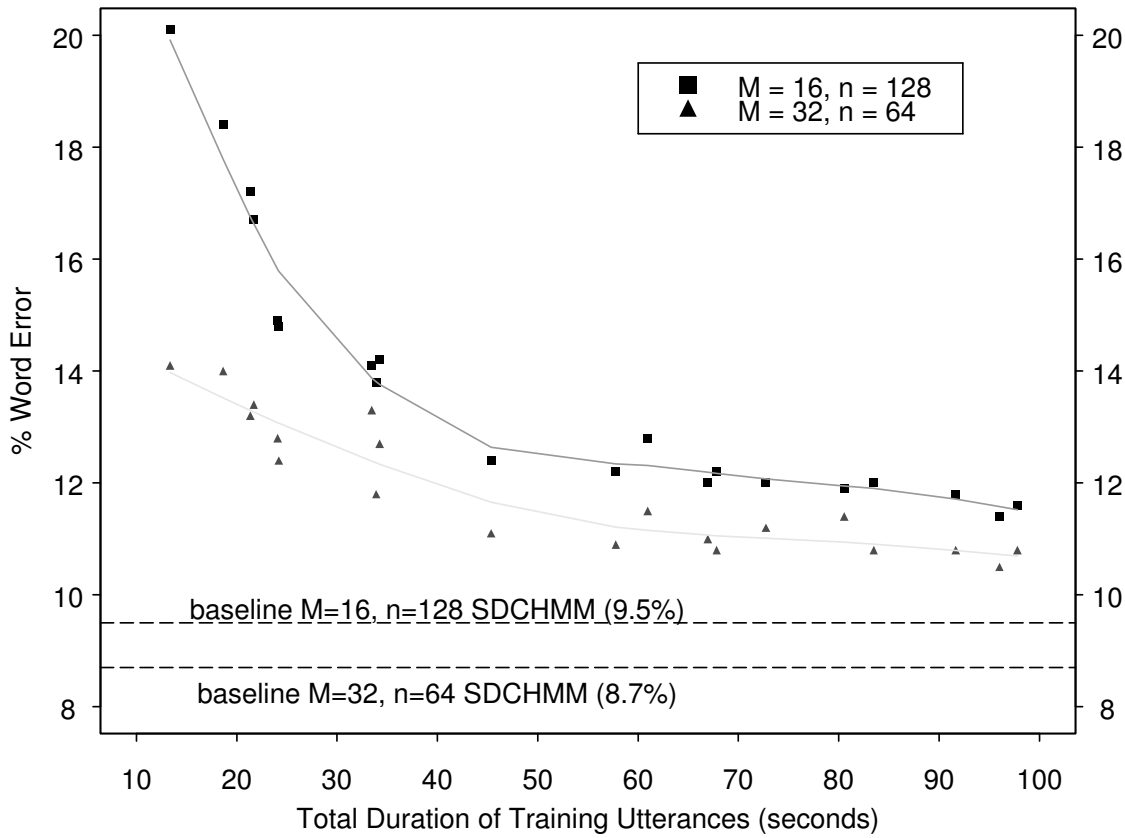
Figure 7.6: ATIS: Variability with few training data (M = #mixtures and n = #subspace Gaussian prototypes per stream)

not to let other factors possibly complicate our main research goal here, we start from the baseline context-dependent ATIS CDHMMs of Chapter 3. The subspace Gaussian tying structure, denoted as CD-SGTS-M20-n64, is extracted from the 20-stream SDCH-MMs converted from this baseline CD CDHMMs, which have 20-mixtures and 64 subspace Gaussian prototypes per stream. This SGTS is used for all CD SDCHMM training in this experiment. We also have all training data phonetically labeled by the baseline CD CDHMMs. To save training computation, subsequent SDCHMM training will *not* re-segment any training data.

The exact training procedure is as follows: For datasets no smaller than S0 (i.e. S0 only, S1 only, S1–2, S1–4, S1–8, and S1–16), CD SDCHMMs are again initialized with the CD-SGTS-M20-n64 using the phonetically transcribed dataset A as described in Algorithm 6. Then it is found that one BW iteration is enough to get the bootstrapped CD SDCHMMs. The bootstrapped models are reestimated by running the BW training algorithm on the

training data under study. Again one BW iteration is enough for the models to converge. On the other hand, for the smaller datasets (i.e. A only, B only, C only, and D only), the bootstrapping — uniform state segmentation followed by SDCHMM initialization and one BW iteration — alone is found to be sufficient as further BW reestimation gives no further improvement on the models' likelihoods.

(B) Result and Discussion

The middle curve of Figure 7.7 shows the recognition performance of the resulting CD SDCHMMs. While we may expect a large performance degradation with little training data (since the CD models are more complex than the CI models), that the asymptotic performance does not meet the baseline performance and occurs with more than 735 minutes of speech (dataset S1–8) is a big disappointment.

One possible explanation may be the insufficient coverage of the triphones being modeled in the smaller training datasets. In the baseline system, all triphones appearing in all of the ATIS corpora are modeled; there are altogether 9,769 of them. However, due to insufficient coverage for some triphones, an additional 8,000 utterances from the Wall Street Journal corpus are employed to provide the coverage. To check our conjecture, we find out the number of triphones that are not covered in each training dataset, and the result is overlaid onto Figure 7.7 (the top curve). When the amount of training data is less than 59 minutes (dataset S0), the triphones coverage is only about 5% (in the smallest dataset D) — 30% (in dataset S0); the low coverage seems to cause the irregular performance of the trained CD SDCHMMs. Even with all data from S1–16, about 8% of the triphones are unrepresented. This may explain the gap between the asymptotic performance of the trained CD SDCHMMs, and that of the baseline (converted) CD SDCHMMs (WERs of 5.5% vs. 5.0%).

In addition, we have the following two observations about training CD SDCHMM:

- Although there is inadequate triphone coverage with a limited amount of training data, there is still high coverage of the subspace Gaussians of the CD SDCHMMs (full coverage in all our experiments).

- When a speech unit is not observed in the training data, the main effect on SDCHMM training is that the mixture weights of its SDCHMM will not be learned — they stay at their initial values of $1/M$ (where M is the number of Gaussian mixtures in the state density) and are not reestimated in subsequent VT/BW training cycles.
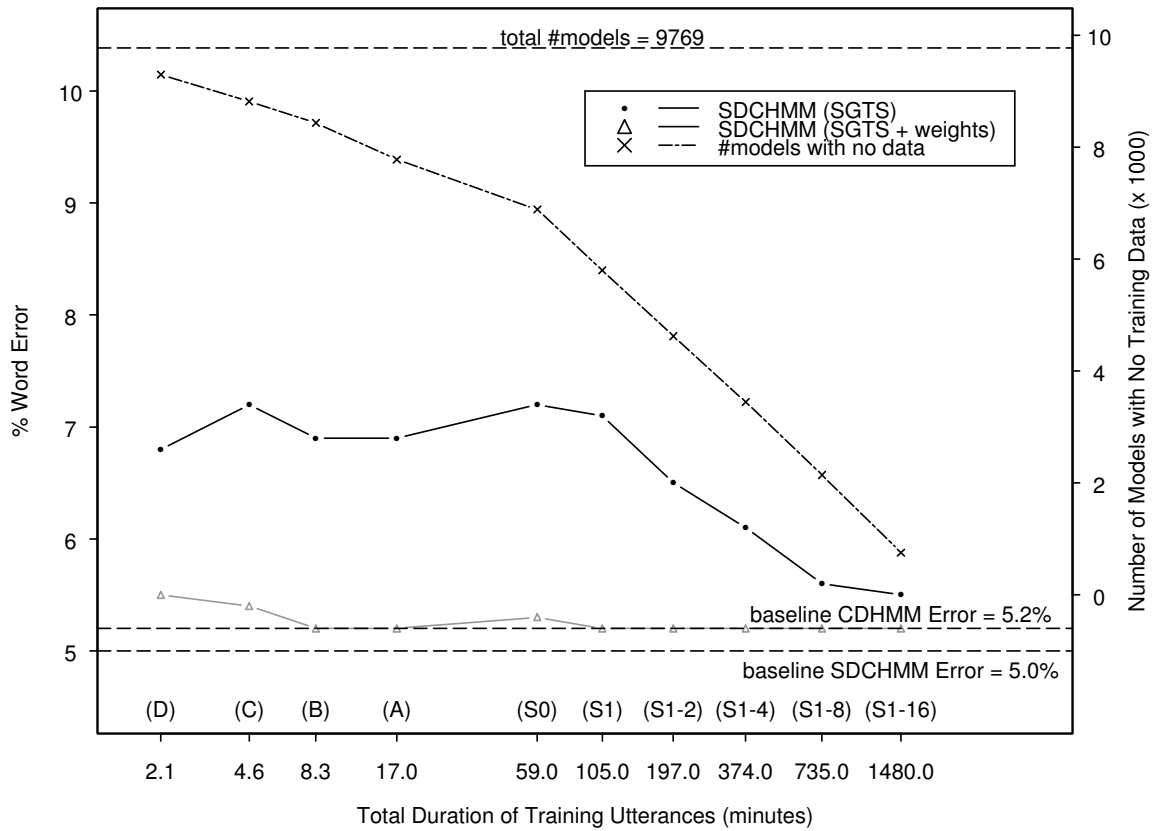
Figure 7.7: ATIS: Data requirement for CD SDCHMM training

Hence, to confirm our conjecture that the poor performance of CD SDCHMM training is due to poor triphone coverage in the given training data, we repeat the experiment by borrowing the mixture weights from the baseline CD SDCHMMs, and by fixing them during direct SDCHMM training. For the small datasets A, B, C, and D, two to five BW iterations are now required, whereas only one BW iteration after bootstrapping is still adequate for larger datasets. The result is presented in the bottom curve on Figure 7.7. By incorporating additional *a priori* knowledge of the mixture weight (on top of the SGTS, CD-SGTS-M20-n64), the CD SDCHMMs (which have a model complexity of 76,154 Gaussians), can now be trained from as little as 8.3 minutes of speech (dataset B) with no degradation in performance when compared with the baseline CD CDHMMs, even when only 14% of the triphones are observed in the training data.

Table 7.4: Comparing data requirements for SDCHMM training and CDHMM training (M = #mixtures per state, N = total #Gaussian components, n = #subspace Gaussian prototypes per stream, T = amount of training data in minutes, and WER = word error rate (%))

20-stream SDCHMM					CDHMM			
M	N	n	T(min)	WER	M	N	T(min)	WER
16	2143	128	2.1	11.5	32	2268	42	11.6
			4.6	10.4	16	1863	88	10.3
			8.3	10.0	16	1863	88	10.3
			17	9.8	32	3150	88	9.7
32	4086	64	2.1	10.3	16	1863	42	10.3
			4.6	9.5	16	1863	180	9.6
			8.3	9.0	32	3734	180	8.6
			17	9.0	32	3734	180	8.6

7.4 Summary and Discussion

In this chapter, we successfully train SDCHMMs directly from much less data without training intermediate CDHMMs. For example, Table 7.4 compares the performance of CI SDCHMMs thus trained with CI CDHMMs of the smallest possible model complexity that can be trained with the least amount of speech and give similar recognition accuracy. It can be seen that the amount of data required for direct SDCHMM training is about 10 — 20 times less than that for CDHMM training. Such great reduction in the amount of training data is attributed to the many fewer model parameters in SDCHMMs as well as to the efficacious tying of subspace Gaussians among the models. While the fewer model parameters, in theory, require less estimation data, should the tying of subspace Gaussians not be efficacious, SDCHMM training would have required even sampling of the phones in the training data. However our experiments show that even when many phones are under-represented in the training data (Figure 7.4(a) or Figure 7.7), there is still a good coverage of the subspace Gaussians (Figure 7.4(b)); hence, good estimation of SDCHMMs is still possible.

When the amount of training data is small (say, less than 8 minutes of speech on the ATIS task), the performance of the ensuing SDCHMMs degrades gracefully. However, over-training readily occurs in this case. In this study, we exhaustively search for the best

BW iteration to stop using the test data. In practice, cross-validation using unseen data should be employed.

Direct SDCHMM training requires *a priori* knowledge of, at least, the subspace Gaussian tying structure. Although in our experiments, the tying structure is derived from an existing recognizer on the same task, our results are still significant. One possible application is speaker enrollment — using a speaker-independent SGTS to train speaker-specific SDCHMMs with little enrollment data.

Results of Experiment IV also suggest that if more *a priori* information is available, even less training data may be sufficient. For instance, we may also incorporate the mixture weights and/or Gaussian variances in addition to the SGTS from the converted SDCHMMs (from which the SGTS is derived), and fix them during SDCHMM training. This may be found useful in speaker (environment) adaptation.

Of course, we still need one set of CDHMMs from which to derive the SGTS for SDCHMM training. It will be interesting to investigate if the SGTS is task independent so that one may deduce a "generic" SGTS from a very accurate CDHMMs and apply it to SDCHMM training in other tasks.

Chapter 8

Conclusions and Future Work

This thesis addresses the problem of high computational cost (in both time and space) of contemporary speech recognizers by greatly reducing the number of parameters in their acoustic models. We choose to tackle the problem by making more compact acoustic models because they constitute a major component of any speech recognizer, and computing their likelihoods takes up 50–70% of total recognition time for many typical tasks (other than very large vocabulary recognition). We start with a set of continuous density hidden Markov models (CDHMMs) using mixture Gaussian densities with diagonal covariances, which are currently the most accurate models for speech recognition. Then, by exercising the technique of parameter tying at a finer sub-phonetic level, namely that of subspace distributions, we arrive at a set of more compact models which we call the *subspace distribution clustering hidden Markov models* (SDCHMMs).

While it can be trivial to reduce the computational cost of a speech recognizer at the expense of its accuracy, it is much harder to increase its speed and reduce its memory footprint while retaining its accuracy at the same time. Since the problem is mainly attributed to the large number of model parameters, we employ the proven technique of parameter tying to reduce the redundancy in CDHMMs — and thus create a more efficient representation of the acoustic models (SDCHMMs). The technique of parameter tying has the additional benefit of reducing the amount of data required to train the new SDCHMMs.

In this thesis, we have given a full account of the theory and the implementation of SDCHMM. Through a series of training and recognition experiments on the ATIS task, we demonstrate that in comparison to the CDHMMs, the new SDCHMMs have the following advantages:

- They reduce the number of Gaussian parameters (means and variances) by as much as three orders of magnitude.

108

- They reduce the total number of model parameters (mixture weights, Gaussian means, and variances) by 20–80 times.

- They achieve 7- to 18-fold decrease in memory size.

- They run 30–60% faster.

- They can be directly estimated with 10–20 times less training data.

All these are achieved with no loss in recognition accuracy.

8.1   Contributions

The most significant contribution of this thesis is the formulation of a new acoustic modeling method which we call *subspace distribution clustering hidden Markov modeling* (SDCHMM). The theory of SDCHMM is formulated as a simple derivative of CDHMM based on tying subspace distributions from a set of conventional CDHMMs. Two methods are presented to implement the SDCHMMs as shown in Figure 8.1: One requires training intermediate CDHMMs and the other requires *a priori* knowledge of the *subspace distribution tying structure* (SDTS).

Working with the implementation and evaluation of SDCHMM, this thesis contains further contributions as follows:

New Unit of Parameter Tying

We show that tying at an even finer sub-phonetic unit, the subspace distribution, is possible. The hypothesis is that speech sounds are more alike in some acoustic subspaces than in the full acoustic space. An analysis of the ensuing SDCHMMs shows that similar phoneme pairs indeed share more subspace Gaussians than non-similar phoneme pairs. Empirically we also show that fewer subspace Gaussian prototypes are required in SDCHMMs with more streams.

Generalization of CDHMM with Locally Independent Streams

We generalize the formulation of CDHMM by introducing the notion of *locally* independent streams. By splitting the local acoustic space of each distribution of the CDHMMs into disjoint subspaces (or streams), and tying the subspace distributions, SDCHMMs maintain the accuracy of CDHMMs by retaining the essential complexity but reducing redundancy
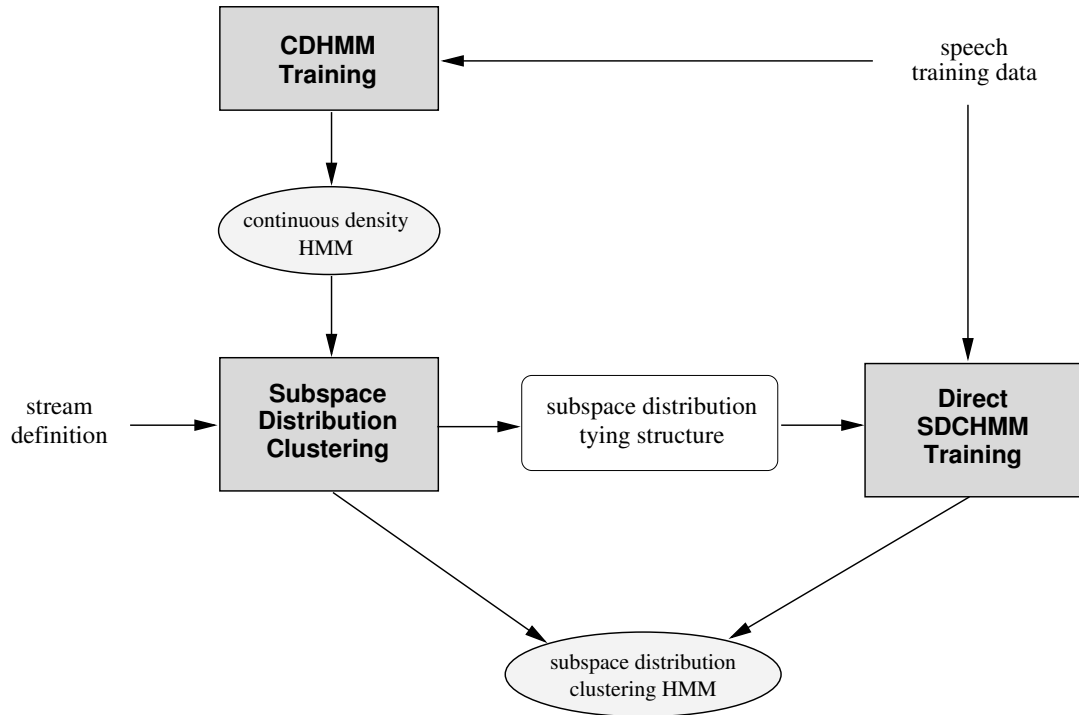
Figure 8.1: Two methods of training SDCHMMs

in the latter through efficient representation of the acoustic space. While the use of streams is not new, streams are usually assumed globally independent — an obviously wrong assumption — in other HMM derivatives (for example, discrete HMM or semi-continuous HMM). Locally independent streams in SDCHMM come naturally from the theory of CDHMM with diagonal covariances; from this perspective, they are *not* an assumption at all.

Generalization of CDHMM and FPTHMM

The SDCHMM provides a full spectrum of HMMs with variable number of streams, with the CDHMM at one end of the spectrum utilizing full-space distributions and the feature-parameter-tying HMM (FPTHMM) at the other extreme utilizing scalar distributions. One may pick the optimal SDCHMMs according to the relevant system requirements and configuration (recognition accuracy, processing power, and RAM/cache space).

Direct Training of Parameter-Tying HMM from Speech

Parameter-tying HMMs are usually created in two steps: The parent HMMs are first trained from scratch, and the parameters of interest are then tied in a separate procedure. While it is common and straightforward to re-train the resulting HMMs (wherein some model parameters are tied), we introduce the novel idea of treating the subspace distribution tying structure as part of the HMM architecture to facilitate training SDCHMMs directly from speech data without intermediate CDHMMs. As a result, we demonstrate that SDCHMMs can be trained from much less data than CDHMMs.

8.2   Future Work

In this thesis, we have presented a complete description of the theory of SDCHMMs and we have proposed a definition of streams, a novel Gaussian clustering algorithm, and two methods of SDCHMM implementation. The results of their evaluation on the ATIS task are very encouraging. Nonetheless, more experiments on tasks of different perplexities and recording conditions should be done to confirm the current findings. In addition, techniques which deal with current HMMs should be reconsidered in the context of SDCHMMs to make use of its compactness and its SDTS. The following are some interesting topics we will be pursuing in the near future.

Model Improvement

Currently we adopt a definition of streams, which uses the heuristic that correlated features tend to cluster in a similar manner. It is obtained by a greedy algorithm and it works reasonably well in this study. Nevertheless, it will be preferable to have a more formal definition of streams without the use of any heuristics.

The importance of stream definition will become more evident when we investigate other functionals for the mixture component distribution. One plausible candidate is a Gaussian distribution with block-diagonal covariance. In the past, Gaussian distributions with diagonal covariance were the most popular choice because of their trainability and computational efficiency. However, it is claimed in [53] that better acoustic models are obtained with explicit modeling of correlations among cepstral parameters. Though Gaussian distributions with full covariance are able to model such correlations, they are too costly for likelihood computations, for storage, and for training. Hence, Gaussian distributions with block-diagonal covariance, which only model the most important correlations

explicitly seem to be a good compromise. For example, for the 39-dimensional feature vector in our recognition system, its diagonal covariance involves 39 variances, and its full covariance has $39 \times 39 = 1521$ parameters. A 13-stream block-diagonal covariance of the features with 3 features per stream will only have $3 \times 3 \times 13 = 117$ parameters — only three times the number of parameters in a diagonal covariance. In fact, [89] used such a distribution with two streams: One for the static features and the other for the dynamic features. In the context of our SDCHMM, more streams are preferable, and that will also mean a much smaller increase in distribution parameters when compared with the use of diagonal covariance.

Hybrid Speaker-Dependent Training/Speaker Adaptation (SDT/SA)

One of the most exciting findings in this thesis is the small data requirement for training SDCHMMs. Though it is generally agreed that speaker-dependent (SD) models perform better than speaker-independent (SI) models, speaker-dependent training is greatly hampered by the lack of speaker-specific training data. However, in Chapter 7, we show that speaker-independent SDCHMMs can be trained with as little as 8 minutes of ATIS speech. This opens up new possibilities for training speaker-dependent SDCHMMs with little data.

The idea is to reduce the amount of data required for training speaker-dependent SDCHMMs through the incorporation of *a priori* knowledge of a speaker-independent SDTS, which can be derived from a large amount of training data. The underlying assumption is that the SDTS is speaker independent, or approximately speaker independent. From another perspective, the procedure can also be considered as a speaker adaptation procedure — adapting the model parameters of speaker-independent SDCHMMs while keeping the speaker-independent SDTS intact. Generalizing this hybrid SDT/SA approach, one may employ the following schemes in training speaker-dependent SDCHMMs, progressively using more *a priori* information from the speaker-independent models:

- SD data + SI SDTS $\rightarrow$ SD SDCHMMs

- SD data + SI SDTS + SI mixture-weights $\rightarrow$ SD SDCHMMs

- SD data + SI SDTS + SI mixture-weights + SI variances $\rightarrow$ SD SDCHMMs.

One may further enhance the robustness of the SD SDCHMMs through interpolation with the SI SDCHMMs.

Speaker Adaptation

The SDTS derived from SDCHMMs may be used in speaker adaptation of non-SDCHMMs as well. There are two common approaches for speaker adaptation: The Bayesian learning approach and transformation-based approach, best exemplified by the MAP [19] and the MLLR [51] techniques respectively. With either approach, when very little speaker-specific data is available, one is generally required to put the model parameters (usually only the Gaussian means) into equivalence classes to share the scarce resources. Since our analysis in Chapter 6 suggests that the SDTS is phonetically plausible, one may use the SDTS to define the equivalence classes. For instance, if one starts with CDHMMs, one may derive the equivalence classes from an SDTS obtained by converting the CDHMMs to SDCHMMs with an appropriate number of subspace distribution prototypes per stream, depending on the amount of adaptation data — fewer prototypes when there are fewer data.

A similar approach may be applied for adaptation to other environmental factors (such as noise, channel, etc.) as well.

Task-Independent Model Bootstrapping

It will be also interesting to see if the SDTS derived from one speech corpus can be applied to another corpus, particularly when the latter is acquired under similar recording conditions. If this is the case, one may obtain a set of bootstrapped SDCHMMs for a new task quickly with few training data using a well-trained SDTS. Thereafter, one has the option to continue SDCHMM training, or convert the set of SDCHMMs to CDHMMs for further training.

8.3   Final Remarks

At the outset, our aim was to derive more compact acoustic models. In conclusion, we obtain two reduced upper bounds on the acoustic models of ATIS:

- A reduced upper bound on the minimum number of Gaussian parameters. Table 5.2 suggests that for the 39-dimensional speech features (12 MFCCs and normalized energy, and their first- and second-order derivatives) we use for the ATIS task, 32–128 Gaussian prototypes per stream are adequate with 13–39 streams.

- A reduced upper bound on the minimum data requirement for training HMMs. In conventional training of continuous density HMMs, each HMM is trained independently, requiring a large amount of training data. The analysis in Chapter 6 suggests that the subspace Gaussian tying structure can capture the inter-relationship among the phones. By making use of this prior knowledge in the direct SDCHMM training scheme of Chapter 7, the whole set of HMMs can be trained simultaneously with about eight minutes of ATIS speech.

It is our belief that more compact acoustic models are possible, and we hope that this thesis sheds some light in this direction.

Bibliography

[1] G. Antoniol, F. Brugnara, M. Cettolo, and M. Federico. "Language Model Representation for Beam-Search Decoding". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 588–591, 1995.

[2] E. Barnard, R.A. Cole, M. Fanty, and P. Vermeulen. "Real-World Speech Recognition with Neural Networks". In J. Alspector, R. Goodman, and T.X. Brown, editors, *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications 2*, pages 186–193. Lawrence Erlbaum Associates, Publishers, 1995.

[3] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". *Annals of Mathematical Statistics*, 41:164–171, 1970.

[4] J.R. Bellegarda and D. Nahamoo. "Tied Mixture Continuous Parameter Modeling for Speech Recognition". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(12):2033–2045, December 1990.

[5] P. Beyerlein. "Fast Log-Likelihood Computation for Mixture Densities in a High-Dimensional Feature Space". In *Proceedings of the International Conference on Spoken Language Processing*, volume 1, pages 271–274, 1994.

[6] P. Beyerlein and M. Ullrich. "Hamming Distance Approximation for a Fast Log-Likelihood Computation for Mixture Densities". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 2, pages 1083–1086, 1995.

[7] E. Bocchieri. "A Study of the Beam-Search Algorithm for Large Vocabulary Continuous Speech Recognition and Methods for Improved Efficiency". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 3, pages 1521–1523, 1993.

[8] E. Bocchieri. "Vector Quantization for the Efficient Computation of Continuous Density Likelihoods". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 692–695, 1993.

[9] E. Bocchieri and G. Riccardi. "State Tying of Triphone HMM's for the 1994 AT&T ARPA ATIS Recognizer". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 2, pages 1499–1502, 1995.

[10] E. Bocchieri, G. Riccardi, and J. Anantharaman. "The 1994 AT&T ATIS CHRONUS Recognizer". In *Proceedings of ARPA Spoken Language Systems Technology Workshop*, pages 265–268. Morgan Kaufmann Publishers, 1995.

[11] H. Bourlard and N. Morgan. "Hybrid Connectionist Models for Continuous Speech Recognition". In C.H. Lee, F.K. Soong, and K.K. Paliwal, editors, *Automatic Speech and Speaker Recognition (Advanced Topics)*, chapter 11, pages 259–283. Kluwer Academic Publishers, 1996.

[12] M. Cohen, Z. Rivlin, and H. Bratt. "Speech Recognition in the ATIS Domain Using Multiple Knowledge Sources". In *Proceedings of ARPA Spoken Language Systems Technology Workshop*, pages 261–264. Morgan Kaufmann Publishers, 1995.

[13] D. Dahl et al. "Expanding the Scope of the ATIS Task: The ATIS-3 Corpus". In *Proceedings of ARPA Human Language Technology Workshop*. Morgan Kaufmann Publishers, 1994.

[14] L. Deng. "A Generalized Hidden Markov Model with State-Conditioned Trend Functions of Time for the Speech Signal". *Signal Processing*, 27(1):65–78, January 1992.

[15] V. Digalakis and H. Murveit. "Genones: Optimizing the Degree of Tying in a Large Vocabulary HMM-based Speech Recognizer". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 537–540, 1994.

[16] J. Fritsch, I. Rogina, T. Sloboda, and A. Waibel. "Speeding Up the Score Computation of HMM Speech Recognizers with the Bucket Voronoi Intersection Algorithm". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 2, pages 1091–1094, 1995.

[17] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, Inc., 2nd edition, 1990.

116

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

[19] Jean-Luc Gauvain and C.H. Lee. "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains". *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, April 1994.

[20] Y. Gong. "Stochastic Trajectory Modeling and Sentence Searching for Continuous Speech Recognition". *IEEE Transactions on Speech and Audio Processing*, 5(1):33–44, January 1997.

[21] P.S. Gopalakrishnan and L.R. Bahl. "Fast Match Techniques". In C.H. Lee, F.K. Soong, and K.K. Paliwal, editors, *Automatic Speech and Speaker Recognition (Advanced Topics)*, chapter 17, pages 413–428. Kluwer Academic Publishers, 1996.

[22] S. Greenberg. "Understanding Speech Understanding Towards a Unified Theory of Speech Perception". In W.A. Ainsworth and S. Greenberg, editors, *Proceedings of the ESCA Tutorial and Advanced Research Workshop on the Auditory Basis of Speech Perception*, pages 1–8. Keele University, UK, 1996.

[23] C.T. Hemphill, J.J. Godfrey, and G.R. Doddington. "The ATIS Spoken Language Systems Pilot Corpus". In *Proceedings of the DARPA Speech and Natural Language Workshop*. Morgan Kaufmann Publishers, 1990.

[24] H. Hermansky. "Perceptual Linear Predictive (PLP) Analysis of Speech". *Journal of Acoustical Society of America*, 87(4):1738–1752, April 1990.

[25] L. Hirschman et al. "Multi-Site Data Collection and Evaluation in Spoken Language Understanding". In *Proceedings of ARPA Human Language Technology Workshop*. Morgan Kaufmann Publishers, 1993.

[26] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithm*. Computer Science Press, 1978.

[27] Z. Hu, J. Schalkwyk, E. Barnard, and R. Cole. "Speech Recognition Using Syllable-Like Units". In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 1117–1120, 1996.

[28] X. Huang et al. "The SPHINX-II Speech Recognition System: An Overview". *Journal of Computer Speech and Language*, 7(2):137–148, April 1993.

[29] X. Huang and M.A. Jack. "Semi-continuous Hidden Markov Models for Speech Signals". *Journal of Computer Speech and Language*, 3(3):239–251, July 1989.

[30] X.D. Huang, Y. Ariki, and M.A. Jack. "Fundamentals of Pattern Recognition". In *Hidden Markov Models for Speech Recognition*, chapter 2, pages 10–51. Edinburgh University Press, 1990.

[31] X.D. Huang, Y. Ariki, and M.A. Jack. *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, 1990.

[32] M. Hwang. "Shared Distribution Hidden Markov Models for Speech Recognition". *IEEE Transactions on Speech and Audio Processing*, 1(4):414–420, October 1993.

[33] R.J. Jones, S. Downey, and J. S. Mason. "Continuous Speech Recognition Using Syllables". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 3, pages 1171–1174, 1997.

[34] B.H. Juang, W. Chou, and C.H. Lee. "Minimum Classification Error Rate Methods for Speech Recognition". *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265, May 1997.

[35] B.H. Juang, D.Y. Gray, and A.H. Gray, Jr. "Distortion Performance of Vector Quantization for LPC Voice Coding". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 30(2):307–309, April 1982.

[36] B.H. Juang, S.E. Levinson, and M.M. Sondhi. "Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains". *IEEE Transactions on Information Theory*, 32(2):307–309, March 1986.

[37] B.H. Juang and L.R. Rabiner. "Mixture Autoregressive Hidden Markov Models for Speech Signals". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(6):1404–1413, December 1985.

[38] B.H. Juang and L.R. Rabiner. "A Segmental K-means Algorithm for Estimating Parameters of Hidden Markov Models". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(9):1639–1641, September 1990.

[39] S.K. Kachigan. *Multivariate Statistical Analysis (A Conceptual Introduction)*. Radius Press, 1991.

[40] N.S. Kim and C.K. Un. "Frame-Correlated Hidden Markov Model Based on Extended Logarithmic Pool". *IEEE Transactions on Speech and Audio Processing*, 5(2):149–160, March 1997.

118

[41] Y. Komori, M. Yamada, H. Yamamoto, and Y. Ohora. "An Efficient Output Probability Computation for Continuous HMM Using Rough and Detail Models". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 2, pages 1087–1090, 1995.

[42] T. Kosaka and S. Sagayama. "Tree-structured Speaker Clustering for Fast Speaker Adaptation". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 245–248, 1994.

[43] P. Ladefoged. *A Course in Phonetics*. Harcourt Brace Jovanovich College Publishers, 3rd edition, 1993.

[44] K.W. Law and C.F. Chan. "Split-Dimension Vector Quantization of Parcor Coefficients for Low Bit Rate Speech Coding". *IEEE Transactions on Speech and Audio Processing*, 2(3):443–446, July 1994.

[45] C.H. Lee. "Acoustic Modeling of Subword Units for Speech Recognition". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 721–724, 1990.

[46] C.H. Lee, C.H. Lin, and B.H. Juang. "A Study on Speaker Adaptation of the Parameters of Continuous Density Hidden Markov Models". *IEEE Transactions on Signal Processing*, 39(4):806–814, April 1991.

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

[48] K.F. Lee, S. Hayamizu, H.W. Hon, C. Huang, J. Swartz, and R. Weide. "Allophone Clustering for Continuous Speech Recognition". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 749–752, 1990.

[49] K.F. Lee and H.W. Hon. "Large-Vocabulary Speaker-Independent Continuous Speech Recognition Using HMM". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 123–126, 1988.

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

[51] C.J. Leggetter and P.C. Woodland. "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models". *Journal of Computer Speech and Language*, 9(2):171–185, April 1995.

[52] E. Levin and R. Pieraccini. "CHRONUS, The Next Generation". In *Proceedings of ARPA Spoken Language Systems Technology Workshop*, pages 269–271. Morgan Kaufmann Publishers, 1995.

[53] A. Ljolje. "The Importance of Cepstral Parameter Correlations in Speech Recognition". *Journal of Computer Speech and Language*, 8(3):223–232, July 1994.

[54] P.C. Loizou and A.S. Spanias. "High-Performance Alphabet Recognition". *IEEE Transactions on Speech and Audio Processing*, 4(6):430–445, November 1996.

[55] B. Lowerre and R. Reddy. "The Harpy Speech Understanding System". In *Trends in Speech Recognition*, pages 340–360. Prentice Hall, 1980.

[56] B.T. Lowerre. *Dynamic Speaker Adaptation in the Harpy Speech Recognition System*. PhD thesis, Department of Computer Science, Carnegie Mellon University, April 1976.

[57] B. Mak and E. Barnard. "Phone Clustering Using the Bhattacharyya Distance". In *Proceedings of the International Conference on Spoken Language Processing*, volume 4, pages 2005–2008, 1996.

[58] J.F. Mari, J.P. Haton, and A. Kriouile. "Automatic Word Recognition Based on Second-Order Hidden Markov Models". *IEEE Transactions on Speech and Audio Processing*, 5(1):22–25, January 1997.

[59] H. Ney, R. Haeb-Umbach, B. Tran, and M. Oerder. "Improvements in Beam Search for 10000-Word Continuous Speech Recognition". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 9–12, 1992.

[60] H. Ney, D. Mergel, A. Noll, and A. Paeseler. "A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 833–836, 1987.

[61] L. Nguyen et al. "The 1994 BBN/BYBLOS Speech Recognition System". In *Proceedings of ARPA Spoken Language Systems Technology Workshop*, pages 77–81. Morgan Kaufmann Publishers, 1995.

[62] M. Ostendorf and S. Roukos. "A Stochastic Segment Model for Phoneme-Based Continuous Speech Recognition". *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(12):1857–1869, December 1989.

[63] M. Padmanabhan, D. Nahamoo L.R. Bahl, and P. de Souza. "Decision-Tree Based Quantization of the Feature Space of a Speech Recognizer". In *Proceedings of the European Conference on Speech Communication and Technology*, pages 147–150, 1997.

[64] K.K. Paliwal. "Use of Temporal Correlation Between Successive Frames in a Hidden Markov Model Based Speech Recognizer". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 215–218, 1993.

[65] K.K. Paliwal and B.S. Atal. "Efficient Vector Quantization of LPC Parameters". *IEEE Transactions on Speech and Audio Processing*, 1(1):3–14, January 1993.

[66] D.S. Pallett et al. "1994 Benchmark Tests for the ARPA Spoken Language Program". In *Proceedings of ARPA Human Language Technology Workshop*, pages 5–36. Morgan Kaufmann Publishers, 1995.

[67] D.B. Paul and J.M. Baker. "The Design for the Wall Street Journal-based CSR Corpus". In *Proceedings of the International Conference on Spoken Language Processing*, volume 2, pages 899–902, 1992.

[68] A.B. Poritz. "Linear Predictive Hidden Markov Models and the Speech Signal". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 1291–1294, 1982.

[69] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[70] L.R. Rabiner and B.H. Juang. "An Introduction to Hidden Markov Models". *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[71] L.R. Rabiner, B.H. Juang, S.E. Levinson, and M.M. Sondhi. "Recognition of Isolated Digits Using Hidden Markov Models with Continuous Mixture Densities". *AT&T Technical Journal*, 64(6):1211–1233, July-August 1985.

[72] L.R. Rabiner and R.W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, 1978.

[73] M.K. Ravishankar. *"Efficient Algorithms for Speech Recognition"*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1996.

[74] G. Riccardi, E. Bocchieri, and R. Pieraccini. "Non-deterministic Stochastic Language Models for Speech Recognition". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 237–240, 1995.

[75] G. Riccardi, R. Pieraccini, and E. Bocchieri. "Stochastic Automata for Language Modeling". *Journal of Computer Speech and Language*, 10(4):265–293, October 1996.

[76] T. Robinson, M. Hochberg, and S. Renals. "The Use of Recurrent Neural Networks in Continuous Speech Recognition". In C.H. Lee, F.K. Soong, and K.K. Paliwal, editors, *Automatic Speech and Speaker Recognition (Advanced Topics)*, chapter 10, pages 233–258. Kluwer Academic Publishers, 1996.

[77] S. Sagayama. "Hidden Markov Network for Precise and Robust Acoustic Modeling". In C.H. Lee, F.K. Soong, and K.K. Paliwal, editors, *Automatic Speech and Speaker Recognition (Advanced Topics)*, chapter 7, pages 159–184. Kluwer Academic Publishers, 1996.

[78] S. Sagayama and S. Takahashi. "On the Use of Scalar Quantization for Fast HMM Computation". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 213–216, 1995.

[79] F. Seide. "Fast Likelihood Computation for Continuous-Mixture Densities Using a Tree-Based Nearest Neighbor Search". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 2, pages 1079–1082, 1995.

[80] E. Singer and R.P. Lippmann. "A Speech Recognizer Using Radial Basis Function Neural Networks in an HMM Framework". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 629–632, 1992.

[81] StatSci, a Division of MathSoft, Inc. *S-PLUS Guide to Statistics and Mathematical Analysis*, pages 6–52. StatSci, Seattle, Washington, 3.2 edition, 1993.

[82] S.S. Stevens and J. Volkmann. "The Relation of Pitch of Frequency: A Revised Scale". *American Journal of Psychology*, 53:329–353, 1940.

[83] S. Takahashi, T. Matsuoka, Y. Minami, and K. Shikano. "Phoneme HMMs Constrained by Frame Correlations". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 219–222, 1993.

[84] S. Takahashi and S. Sagayama. "Effects of Variance Tying for Four-Level Tied Structure Phone Models". In *Proceedings of ASI Conference*, volume 1-Q-23, pages 141–142, 1995.

[85] S. Takahashi and S. Sagayama. "Four-Level Tied-Structure for Efficient Representation of Acoustic Modeling". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 520–523, 1995.

[86] A.J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm". *IEEE Transactions on Information Theory*, 13:260–269, April 1967.

[87] P.C. Woodland, C.J. Leggetter, J.J. Odell, V. Valtchev, and S.J. Young. "The Development of the 1994 HTK Large Vocabulary Speech Recognition System". In *Proceedings of ARPA Spoken Language Systems Technology Workshop*, pages 104–109. Morgan Kaufmann Publishers, 1995.

[88] S.J. Young and P.C. Woodland. "The Use of State Tying in Continuous Speech Recognition". In *Proceedings of the European Conference on Speech Communication and Technology*, volume 3, pages 2203–2206, 1993.

[89] Yunxin Zhao. "A Speaker-Independent Continuous Speech Recognition System Using Continuous Mixture Gaussian Density HMM of Phoneme-Sized Units". *IEEE Transactions on Speech and Audio Processing*, 1(3):345–361, July 1993.

[90] E. Zwicker, G. Flottorp, and S.S. Stevens. "Critical Bandwidth in Loudness Summation". *Journal of Acoustical Society of America*, 29:548–557, 1957.

Appendix A

Smaller Quantization Error in Lower Dimensions

In this Appendix, we want to show that for a set of Gaussian ensembles with diagonal covariance matrices, the Gaussian quantization error is always smaller when quantization is done in lower dimensions.

Let us consider three 2-dimensional Gaussians G_i with n_i vectors and mean (x_i, y_i), $1 \leq i \leq 3$, and diagonal covariances as shown in Figure A.1. Let us denote the distortion of a Gaussian ensemble in the 2-dimensional space as $\mathcal{D}_{xy}$, and that in the F_x and F_y dimension by $\mathcal{D}_x$ and $\mathcal{D}_y$ respectively. We further assume a distortion measure with the following additivity property:

Additivity Property:

$$\mathcal{D}_{xy} = \mathcal{D}_x + \mathcal{D}_y \tag{A.1}$$

That is, the distortion of a full-space Gaussian (with diagonal covariance) is the sum of the distortions of its independent subspace Gaussians. As an example, the Euclidean distortion measure defined in Equation (3.2) satisfies this property.

Now let us compute the increase in distortion when two of the three Gaussians are clustered in 2-dimensional or 1-dimensional space. (Clearly, by the definition of the distortion, the total distortion of the Gaussians before clustering is the same, regardless whether it is computed in 2-dimensional or 1-dimensional space.)
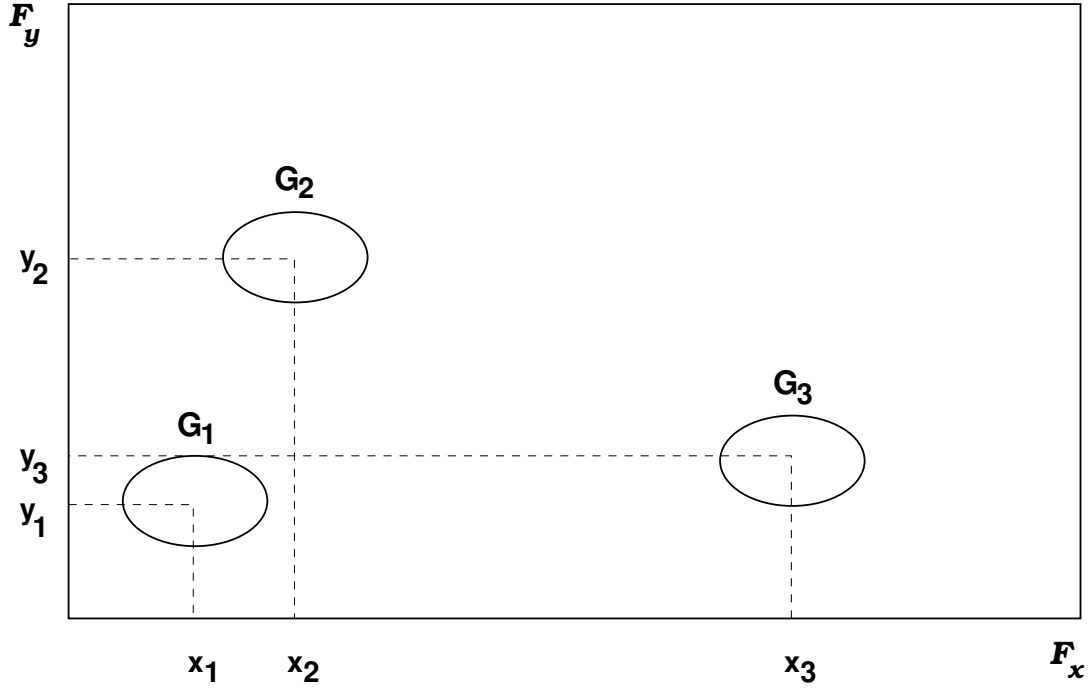
Figure A.1: Smaller quantization error in lower dimensions

Gaussian Clustering in 2-Dimensional Space

Without loss of generality, let us assume that among the three Gaussian pairs: $\{G_1, G_2\}$, $\{G_1, G_3\}$, and $\{G_2, G_3\}$, clustering G_{i_0} and G_{j_0}, $1 \leq i_0 \neq j_0 \leq 3$, gives the minimum increase in distortion. That is,

$$\Delta \mathcal{D}_{xy}(i_0, j_0) \leq \Delta \mathcal{D}_{xy}(i, j), \quad 1 \leq i \neq j \leq 3 \tag{A.2}$$

Gaussian Clustering in 1-Dimensional Space

Now, if the 2-dimensional Gaussians are projected onto each of the orthogonal 1-dimensional subspaces F_x and F_y, we obtain three 1-dimensional Gaussians in each of the subspaces: G_{1x}, G_{2x}, and G_{3x} in F_x, and G_{1y}, G_{2y}, and G_{3y} in F_y. In the subspace F_x, let us assume that G_{i_1} and G_{j_1}, $1 \leq i_1 \neq j_1 \leq 3$, cluster together with the minimum increase in distortion. Similarly, in the subspace F_y, G_{i_2} and G_{i_2}, $1 \leq i_2 \neq j_2 \leq 3$, cluster together with the minimum increase in distortion. That is,

$$\Delta \mathcal{D}_x(i_1, j_1) \leq \Delta \mathcal{D}_x(i, j), \quad 1 \leq i \neq j \leq 3 \tag{A.3}$$

and

$$\Delta \mathcal{D}_y(i_2, j_2) \leq \Delta \mathcal{D}_y(i, j), \quad 1 \leq i \neq j \leq 3 \ . \tag{A.4}$$

Hence, we have

$$\Delta\mathcal{D}_x(i_1, j_1) \leq \Delta\mathcal{D}_x(i_0, j_0) \tag{A.5}$$

and

$$\Delta\mathcal{D}_y(i_2, j_2) \leq \Delta\mathcal{D}_y(i_0, j_0) \ . \tag{A.6}$$

Adding Equations (A.5) and (A.6), we have

$$
\begin{aligned}
& \Delta\mathcal{D}_x(i_1, j_1) + \Delta\mathcal{D}_y(i_2, j_2) \\
\leq\ & \Delta\mathcal{D}_x(i_0, j_0) + \Delta\mathcal{D}_y(i_0, j_0) \\
=\ & \Delta\mathcal{D}_{xy}(i_0, j_0) \ . \qquad\qquad (\textit{by the additivity property})
\end{aligned}
$$

It is straightforward to extend the proof to Gaussians of higher dimensions.

Note that the fact that smaller quantization errors are obtained in lower dimensions does not contradict the general claim that vector quantization (VQ) is more efficient than scalar quantization (SQ) (or quantization in lower dimension). The former only implies that for the *same* number of prototypes per stream, the quantization error is smaller when there are more streams of lower dimensions. However, if we measure the quantization efficiency in terms of the number of coding bits, the smaller quantization errors obtained with K streams is achieved at the expense of K times the number of bits required by VQ. In other words, it is generally found that for the *same* number of coding bits, quantization in the full space achieves smaller quantization errors than quantization in the subspaces. For example, for a 10-dimensional vector space, if we use one bit to encode the full space in VQ, there are only two 10-dimensional prototypes to represent the full space, giving rise to enormous quantization errors. However, even one bit per feature in SQ can effectively represent $2^{10} = 1,024$ different full-space prototypes with much smaller errors. On the other hand, it is more efficient to use ten bits to directly encode the full space with VQ than SQ, since VQ does not make the assumption of feature independence and thus produces better prototypes with less quantization errors. One major shortcoming of VQ is its larger memory requirement: For the same example, encoding the 10-dimensional space with 10-bit VQ requires a codebook of 1,024 10-dimensional vectors, whereas a 10-bit SQ codebook contains only $2^1 \times 10 = 20$ scalars which is equivalent to two 10-dimensional vectors in memory size.

Appendix B

Count of Common Subspace Gaussians between Phones

Table B.1: ATIS: Number of common subspace Gaussians between any two phones (a) 1st state

	aa	ae	ah	ao	aw	ax	axr	ay	eh	er	ey	ih	ix	iy	ow	oy	uh	uw	b	ch	d	dh	dx	el	en	f	g	hh	jh	k	l	m	n	ng	nx	p	r	s	sh	t	th	v	w	y	z
aa	80	8	9	15	8	5	9	8	8	12	8	4	10	2	11	9	9	5	1	3	2	2	4	5	8	5	4	6	7	6	11	3	4	1	3	6	3	5	7	3	6	1	4	5	2
ae	8	80	1	8	6	8	8	7	21	7	11	13	13	3	7	6	3	4	6	8	5	4	7	4	11	5	8	11	7	6	2	2	2	3	4	7	8	6	8	6	3	2	5	7	3
ah	9	1	80	2	9	6	5	14	4	6	9	6	9	5	5	10	8	8	3	3	1	3	4	5	5	0	3	2	0	2	2	10	1	2	4	1	3	2	0	1	1	4	8	4	2
ao	15	8	2	80	3	7	13	5	4	6	5	4	9	2	10	2	2	5	3	4	3	2	5	2	4	5	4	5	6	9	12	2	1	3	0	6	8	5	6	8	6	2	7	10	5
aw	8	6	9	3	80	3	3	13	4	6	6	1	2	5	15	4	7	0	2	0	2	0	1	7	6	0	1	5	2	1	5	4	2	1	2	0	2	1	0	0	1	0	1	5	0
ax	5	8	6	7	3	80	8	3	10	6	4	14	9	5	11	1	11	5	2	3	4	6	6	7	7	1	4	5	5	4	6	7	6	7	8	3	9	4	5	9	4	5	7	8	7
axr	9	8	5	13	3	8	80	3	8	20	6	8	11	4	8	6	5	10	2	2	2	3	1	7	8	3	2	3	2	6	7	3	2	5	2	2	16	5	6	1	3	2	6	8	4
ay	8	7	14	5	13	3	3	80	5	6	5	3	3	9	9	11	10	1	2	5	2	3	4	5	2	2	8	1	1	1	6	4	4	2	2	1	5	4	3	2	2	2	2	4	3
eh	8	21	4	4	4	10	8	5	80	11	14	12	14	10	5	4	3	5	4	4	3	4	2	11	8	4	5	4	3	3	4	0	2	9	3	5	2	9	5	3	4	2	5	5	5
er	12	7	6	6	6	6	20	6	11	80	12	8	13	7	10	6	9	9	3	2	3	5	4	5	5	4	3	9	1	3	11	5	3	5	1	6	12	6	4	1	2	1	3	6	8
ey	8	11	9	5	6	4	6	5	14	12	80	7	9	9	11	9	5	5	2	4	1	1	2	9	14	4	4	3	6	0	4	5	4	5	8	2	2	6	3	1	3	2	4	7	4
ih	4	13	6	4	1	14	8	5	12	8	7	80	27	9	5	1	12	18	6	4	5	6	5	8	8	3	2	3	5	3	9	7	7	5	10	4	8	8	5	4	3	3	6	10	7
ix	10	13	9	9	2	9	11	3	14	13	9	27	80	7	6	0	9	14	2	4	2	4	3	10	10	1	3	9	4	2	8	5	7	3	7	1	5	4	2	2	1	5	6	9	4
iy	2	3	5	2	5	5	4	9	10	7	9	9	7	80	7	6	8	9	4	6	1	4	2	11	4	4	3	6	5	7	5	12	5	2	3	4	4	0	1	1	4	17	5		
ow	11	7	5	10	15	11	8	9	5	10	11	5	6	7	80	10	6	3	3	4	3	3	7	10	5	2	4	4	5	3	6	5	4	7	6	3	8	0	3	4	4	2	6	7	6
oy	9	6	10	2	4	1	6	11	4	6	9	1	0	6	10	60	8	2	3	1	2	0	4	6	2	0	1	2	0	1	7	2	4	7	3	0	4	1	3	1	0	2	2	6	2
uh	9	3	8	2	7	11	5	10	3	9	5	12	9	8	6	8	80	6	3	2	3	6	5	10	6	1	2	4	3	3	6	8	6	5	5	3	3	6	2	3	2	5	4	2	
uw	5	4	8	5	0	5	10	1	5	9	5	18	14	9	3	2	6	80	5	8	5	7	9	7	6	3	3	8	2	6	5	3	6	4	9	2	6	3	8	2	7	5	6	5	4
b	1	6	3	3	2	2	2	4	3	2	6	2	4	3	3	3	3	5	80	11	25	18	9	4	0	23	21	8	14	17	2	4	3	5	1	20	1	2	4	17	18	13	13	6	4
ch	3	8	3	4	0	3	2	5	4	2	4	4	4	6	4	1	2	8	11	80	21	15	13	4	2	13	20	1	26	14	2	1	2	6	5	10	4	8	16	25	9	7	10	8	7
d	2	5	1	3	2	4	2	2	3	3	1	5	2	1	3	2	3	5	25	21	80	21	16	1	2	10	28	4	19	22	7	4	4	4	3	12	3	4	7	32	15	11	14	3	6
dh	2	4	3	2	0	6	3	3	4	5	1	6	4	4	3	0	6	9	18	15	21	80	12	6	3	11	20	4	10	16	4	3	5	4	1	12	9	6	11	20	24	13	10	10	7
dx	4	7	4	5	1	6	1	4	2	4	2	5	3	2	7	4	5	7	9	13	16	12	80	7	3	2	20	3	10	13	5	4	10	5	19	4	7	3	7	13	1	20	9	5	3
el	5	4	5	2	7	7	7	5	11	5	9	8	10	11	10	6	10	6	4	4	1	6	7	80	9	2	3	5	2	7	4	8	2	4	3	2	2	6	7	6	6				
en	8	11	5	4	6	7	8	5	8	5	14	8	10	4	5	2	6	4	0	2	2	3	3	9	80	3	4	4	4	1	5	5	15	9	14	0	5	4	4	2	2	5	4	8	1
f	5	5	0	5	0	1	3	2	4	4	4	3	1	4	2	0	1	5	23	13	10	11	2	2	3	80	7	11	16	12	1	4	0	5	3	25	5	11	13	16	31	8	12	4	5
g	4	8	3	4	1	4	2	2	5	3	4	2	3	3	4	1	2	5	21	20	28	20	20	3	4	7	80	5	18	23	3	5	9	4	8	10	7	5	22	12	15	13	8	9	
hh	6	11	2	5	5	5	3	8	4	9	3	3	9	5	4	2	4	3	8	1	4	4	3	3	4	11	5	80	5	3	6	6	3	4	4	6	4	7	11	4	12	6	5	10	11
jh	7	7	0	6	2	5	2	1	3	1	6	5	4	4	5	0	3	8	14	26	19	10	10	5	4	16	18	5	80	9	5	3	4	7	5	11	3	6	14	15	12	9	11	3	7
k	6	6	2	9	1	4	6	1	3	3	0	3	2	2	3	1	3	2	17	14	22	16	13	2	1	12	23	9	80	2	2	5	2	6	24	8	4	7	31	16	10	14	4	7	
l	11	2	2	12	5	6	7	1	4	11	4	9	8	6	6	7	6	6	2	2	7	4	5	5	5	1	3	6	5	2	80	1	5	4	6	3	8	5	4	3	4	3	3	10	9
m	3	2	10	2	4	7	3	6	0	5	5	7	5	7	5	2	8	5	4	1	4	3	4	2	5	4	5	6	3	2	1	80	12	8	13	2	2	3	0	1	4	8	3	9	6
n	4	2	1	1	2	6	2	4	2	3	4	7	7	5	4	4	6	3	3	2	4	5	10	7	15	0	9	3	4	5	5	12	80	13	20	3	7	1	7	2	5	9	5	7	5
ng	1	3	2	3	1	7	5	4	9	5	5	3	12	7	5	6	5	9	5	6	4	4	5	4	9	5	4	7	2	4	8	13	80	10	3	4	2	3	4	1	7	5	11	4	
nx	3	4	4	0	2	8	2	2	3	1	8	10	7	5	6	3	5	9	1	5	3	1	19	8	14	3	8	4	5	6	6	13	20	10	80	3	3	4	4	4	1	12	6	5	7
p	6	7	1	6	0	3	2	2	5	6	2	4	1	2	3	0	5	2	20	10	12	12	4	2	0	25	10	6	11	24	3	2	3	3	3	80	5	4	11	17	18	10	12	4	1
r	3	8	3	8	2	9	16	1	2	12	2	8	5	3	8	4	3	6	1	4	3	9	7	4	5	5	7	4	3	8	8	2	7	4	3	5	80	6	8	4	6	6	7	7	6
s	5	6	2	5	1	4	5	5	9	6	6	8	4	4	0	1	3	3	2	8	4	6	3	2	4	11	7	7	6	4	5	3	1	2	4	4	6	80	12	3	13	2	2	4	25
sh	7	8	0	6	0	5	6	4	5	4	3	5	2	4	3	0	3	6	4	16	7	11	7	3	4	13	5	11	14	7	4	0	7	3	4	11	8	12	80	10	14	4	10	5	12
t	3	6	1	8	0	9	1	3	3	1	1	4	2	0	4	1	2	2	17	25	32	20	13	2	2	16	22	4	15	31	3	1	2	4	4	17	4	3	10	80	17	9	13	5	4
th	6	3	1	6	1	4	3	2	4	2	3	3	1	1	4	0	3	7	18	9	15	24	1	2	2	31	12	12	12	16	4	4	5	1	1	18	6	13	14	17	80	4	12	7	10
v	1	2	4	2	0	5	2	2	2	1	2	3	5	1	2	2	2	5	13	7	11	13	20	6	5	8	15	6	9	10	3	8	9	7	12	10	6	2	4	9	4	80	8	5	9
w	4	5	8	7	1	7	6	2	5	3	4	6	6	4	6	2	5	6	13	10	14	10	9	7	4	12	13	5	11	14	3	3	5	5	6	12	7	2	10	13	12	8	80	9	5
y	5	7	4	10	5	8	8	4	5	6	7	10	9	17	7	6	4	5	6	8	3	10	5	6	8	4	8	10	5	4	10	9	7	11	5	4	7	4	5	5	7	5	9	80	5
z	2	3	2	5	0	7	4	3	5	8	4	7	4	5	6	2	2	4	4	7	6	7	3	6	1	5	9	11	7	7	9	6	5	4	7	1	6	25	12	4	10	9	5	5	80

Table B.1: ATIS: Number of common subspace Gaussians between any two phones (b) 2nd state

	aa	ae	ah	ao	aw	ax	axr	ay	eh	er	ey	ih	ix	iy	ow	oy	uh	uw	b	ch	d	dh	dx	el	en	f	g	hh	jh	k	l	m	n	ng	nx	p	r	s	sh	t	th	v	w	y	z
aa	80	14	2	28	24	11	11	7	18	15	16	2	4	13	14	0	0	9	0	2	0	0	0	10	16	0	0	6	0	0	8	1	10	6	1	0	3	2	5	0	1	0	1	11	4
ae	14	80	3	15	13	11	13	18	26	18	17	4	10	16	8	1	0	11	0	4	0	3	1	6	10	3	1	8	2	0	1	2	6	7	2	0	4	3	4	0	2	2	2	8	2
ah	2	3	80	3	3	7	3	6	4	0	1	8	5	2	6	3	17	2	3	4	8	4	5	6	0	1	3	5	6	7	3	6	7	3	3	5	8	0	2	4	1	0	7	2	2
ao	28	15	3	80	18	5	16	8	16	18	12	6	9	11	11	2	0	4	0	2	0	3	3	8	14	4	0	6	0	0	11	2	7	8	2	0	5	3	4	1	3	3	2	13	3
aw	24	13	3	18	80	5	10	12	6	7	12	6	7	7	25	3	0	9	2	4	0	0	1	13	6	3	0	5	1	0	10	5	5	9	0	1	4	3	6	1	1	1	2	10	6
ax	11	11	7	5	5	80	3	6	8	8	17	11	14	15	11	0	11	16	1	3	5	4	1	9	10	1	1	2	5	2	3	3	10	8	4	4	9	1	2	4	1	0	2	13	5
axr	11	13	3	16	10	3	80	3	14	28	12	7	4	14	9	2	3	8	0	1	1	1	2	3	9	2	1	4	3	0	7	5	6	6	1	2	11	5	5	1	3	3	3	6	4
ay	7	18	6	8	12	6	3	80	18	10	6	8	5	4	9	9	1	7	1	1	0	3	3	10	7	2	0	4	1	0	5	1	3	4	2	3	6	3	3	1	0	1	2	5	1
eh	18	26	4	16	6	8	14	18	80	23	20	7	3	12	8	3	1	14	0	2	0	2	3	3	11	0	0	8	0	0	1	1	8	4	3	1	1	2	1	0	0	1	1	11	0
er	15	18	0	18	7	8	28	10	23	80	20	9	9	13	7	2	0	15	0	2	0	2	0	3	14	4	0	4	2	0	5	0	6	8	0	0	10	2	5	1	1	3	1	10	2
ey	16	17	1	12	12	17	12	6	20	20	80	9	6	24	20	4	1	18	0	2	0	1	0	8	13	3	0	2	0	0	4	2	10	10	0	0	1	0	1	0	1	0	1	15	0
ih	2	4	8	6	6	11	7	8	7	9	9	80	28	9	7	0	13	12	0	2	3	3	4	7	6	0	3	5	4	5	7	5	6	5	4	5	8	0	1	3	2	1	5	6	4
ix	4	10	5	9	7	14	4	5	3	9	6	28	80	8	10	2	12	13	0	1	4	4	3	6	10	2	3	2	3	8	6	2	7	10	3	2	16	0	3	7	5	0	1	5	0
iy	13	16	2	11	7	15	14	4	12	13	24	9	8	80	12	1	3	10	0	4	0	5	2	10	15	2	1	5	6	1	1	4	14	9	0	0	1	5	8	0	2	4	4	20	2
ow	14	8	6	11	25	11	9	9	8	7	20	7	10	12	80	0	2	16	0	3	1	4	1	15	9	1	1	1	1	10	10	2	6	10	0	1	2	1	5	0	2	1	4	17	2
oy	0	1	3	2	3	0	2	9	3	2	4	0	2	1	0	20	1	2	1	3	0	0	2	2	0	0	0	0	1	1	1	0	0	1	3	2	2	0	2	0	1	1	1	2	1
uh	0	0	17	0	0	11	3	1	1	0	1	13	12	3	2	1	80	4	4	2	6	7	3	7	1	0	5	2	5	9	3	11	9	2	5	7	8	0	0	9	4	3	7	3	1
uw	9	11	2	4	9	16	8	7	14	15	18	12	13	10	16	2	4	80	0	0	2	4	2	13	20	0	0	1	2	3	5	2	15	11	4	3	4	0	0	2	2	0	1	15	2
b	0	0	3	0	2	1	0	1	0	0	0	0	0	0	0	1	4	0	80	2	12	17	9	1	0	2	19	2	4	8	2	6	3	2	5	13	2	1	0	7	5	15	10	2	2
ch	2	4	4	2	4	3	1	1	2	2	2	2	1	4	3	3	2	0	2	80	1	6	4	1	4	10	0	8	16	1	3	8	3	8	5	7	3	14	23	5	14	9	2	1	14
d	0	0	8	0	0	5	1	0	0	0	0	3	4	0	1	0	6	2	12	1	80	18	17	5	2	0	22	2	8	13	7	2	2	0	6	6	4	0	0	19	7	4	7	2	4
dh	0	3	4	3	0	4	1	3	2	2	1	3	4	5	4	0	7	4	17	6	18	80	14	5	6	1	12	3	9	6	4	9	2	3	8	6	2	2	4	10	10	13	7	4	4
dx	0	1	5	3	1	1	2	3	3	0	0	4	3	2	1	2	3	2	9	4	17	14	80	5	3	3	18	5	9	10	3	8	5	3	14	6	3	3	2	18	6	7	9	1	3
el	10	6	6	8	13	9	3	10	3	3	8	7	6	10	15	2	7	13	1	1	5	5	5	80	7	1	1	3	3	2	15	6	6	7	6	1	5	2	2	1	0	4	5	12	4
en	16	10	0	14	6	10	9	7	11	14	13	6	10	15	9	0	1	20	0	4	2	6	3	7	80	4	1	4	1	1	6	9	34	25	3	0	4	5	9	1	4	2	0	6	4
f	0	3	1	4	3	1	2	2	0	4	3	0	2	2	1	0	0	0	2	10	0	1	3	1	4	80	1	7	4	3	1	1	3	2	0	6	4	20	16	2	25	4	3	1	6
g	0	1	3	0	0	1	1	0	0	0	0	3	3	1	1	0	5	0	19	0	22	12	18	1	1	1	80	3	7	22	5	1	4	2	5	9	9	1	0	16	8	5	7	0	1
hh	6	8	5	6	5	2	4	4	8	4	2	5	2	5	1	0	2	1	2	8	2	3	5	3	4	7	3	80	7	2	4	8	5	4	3	3	5	10	8	4	8	7	1	0	14
jh	0	2	6	0	1	5	3	1	0	2	0	4	3	6	1	1	5	2	4	16	8	9	9	3	1	4	7	7	80	3	2	3	3	6	11	10	4	7	6	14	8	6	4	2	14
k	0	0	7	0	0	2	0	0	0	0	0	5	8	1	10	1	9	3	8	1	13	6	10	2	1	3	22	2	3	80	6	2	3	0	6	18	5	3	1	20	6	6	4	0	1
l	8	1	3	11	10	3	7	5	1	5	4	7	6	1	10	1	3	5	2	3	7	4	3	15	6	1	5	4	2	6	80	3	3	4	4	1	7	3	3	2	2	3	6	2	5
m	1	2	6	2	5	3	5	1	1	0	2	5	2	4	2	0	11	2	6	8	2	9	8	6	9	1	1	8	3	2	3	80	12	11	8	6	2	3	2	0	6	7	7	3	6
n	10	6	7	7	5	10	6	3	8	6	10	6	7	14	6	0	9	15	3	3	2	2	5	6	34	3	4	5	3	3	3	12	80	26	13	4	3	5	6	6	1	6	1	5	4
ng	6	7	3	8	9	8	6	4	4	8	10	5	10	9	10	1	2	11	2	8	0	3	3	7	25	2	2	4	6	0	4	11	26	80	6	2	2	4	6	2	2	1	4	6	4
nx	1	2	3	2	0	4	1	2	3	0	0	4	3	0	0	3	5	4	5	5	6	8	14	6	3	0	5	3	11	6	4	8	13	6	80	11	4	3	1	6	1	7	5	6	6
p	0	0	5	0	1	4	2	3	1	0	0	5	2	0	1	2	7	3	13	7	6	6	6	1	0	6	9	3	10	18	1	6	4	2	11	80	2	3	0	15	14	8	4	2	3
r	3	4	8	5	4	9	11	6	1	10	1	8	16	1	2	2	8	4	2	3	4	2	3	5	4	4	9	5	4	5	7	2	3	2	4	2	80	2	2	4	6	0	5	3	3
s	2	3	0	3	3	1	5	3	2	2	0	0	0	5	1	0	0	0	1	14	0	2	3	2	5	20	1	10	7	3	3	3	5	4	3	3	2	80	24	1	11	3	1	0	28
sh	5	4	2	4	6	2	5	3	1	5	1	1	3	8	5	2	0	0	0	23	0	4	2	2	9	16	0	8	6	1	3	2	6	6	1	0	2	24	80	0	11	6	2	2	22
t	0	0	4	1	1	4	1	1	0	1	0	3	7	0	0	0	9	2	7	5	19	10	18	1	1	2	16	4	14	20	2	0	6	2	6	15	4	1	0	80	6	5	4	0	5
th	1	2	1	3	1	1	3	0	0	1	1	2	5	2	2	1	4	2	5	14	7	10	6	0	4	25	8	8	8	6	2	6	1	2	1	14	6	11	11	6	80	8	4	2	11
v	0	2	0	3	1	0	3	1	0	3	0	1	0	4	1	1	3	0	15	9	4	13	7	4	2	4	5	7	6	6	3	7	6	1	7	8	0	3	6	5	8	80	6	3	6
w	1	2	7	2	2	2	3	2	1	1	1	5	1	4	4	1	7	1	10	2	7	7	9	5	0	3	7	1	4	4	6	7	1	4	5	4	5	1	2	4	4	6	80	10	1
y	11	8	2	13	10	13	6	5	11	10	15	6	5	20	17	2	3	15	2	1	2	4	1	12	6	1	0	0	2	0	2	3	5	6	6	2	3	0	2	0	2	3	10	80	2
z	4	2	2	3	6	5	4	1	0	2	0	4	0	2	2	1	1	2	2	14	4	4	3	4	4	6	1	14	14	1	5	6	4	4	6	3	3	28	22	5	11	6	1	2	80

Table B.1: ATIS: Number of common subspace Gaussians between any two phones (c) 3rd state

	aa	ae	ah	ao	aw	ax	axr	ay	eh	er	ey	ih	ix	iy	ow	oy	uh	uw	b	ch	d	dh	dx	el	en	f	g	hh	jh	k	l	m	n	ng	nx	p	r	s	sh	t	th	v	w	y	z
aa	80	20	8	22	19	8	10	10	15	7	10	7	8	6	15	3	6	4	8	7	6	5	5	6	7	7	5	6	6	5	7	8	6	5	2	9	9	5	6	3	9	4	6	5	6
ae	20	80	11	17	11	13	10	12	27	7	10	11	12	7	17	5	9	5	4	5	8	6	11	8	8	6	8	9	8	5	7	7	3	6	9	7	13	8	5	6	7	3	9	3	6
ah	8	11	80	9	13	20	9	6	9	11	4	8	13	6	13	6	11	7	3	8	3	9	7	5	3	5	9	7	5	4	5	2	3	9	8	7	8	7	5	3	7	3	5	3	5
ao	22	17	9	80	16	8	11	9	16	4	5	4	6	2	11	4	7	5	5	9	5	6	5	7	4	8	6	7	8	7	15	6	6	4	7	11	8	3	1	4	4	1	11	5	3
aw	19	11	13	16	80	5	8	6	8	6	6	5	5	4	12	0	8	4	10	2	3	5	7	10	6	4	4	4	10	5	13	6	4	5	6	7	11	2	1	5	9	5	16	5	2
ax	8	13	20	8	5	80	4	7	7	11	14	13	11	16	12	7	10	15	4	7	7	5	8	5	7	6	8	2	5	8	4	9	11	12	7	9	9	2	6	6	5	9	5	2	3
axr	10	10	9	11	8	4	80	4	3	16	7	8	12	4	8	1	9	11	9	4	7	7	4	3	5	6	5	2	4	8	6	7	3	4	4	21	7	4	10	8	2	9	3	6	
ay	10	12	6	9	6	7	4	80	11	8	19	11	15	17	7	16	6	10	11	9	9	10	7	11	5	4	10	8	10	7	11	4	3	4	10	5	11	8	8	5	4	5	4	6	10
eh	15	27	9	16	8	7	3	11	80	10	8	11	8	7	7	6	6	2	2	8	6	3	5	7	5	4	6	7	2	3	9	5	6	5	6	3	8	5	5	5	6	2	10	7	6
er	7	7	11	4	6	11	16	8	10	80	8	14	14	6	9	4	12	9	9	11	7	9	6	2	4	5	6	9	5	10	5	8	7	8	7	8	8	21	8	9	7	11	6	3	5	7
ey	10	10	4	5	6	14	7	19	8	8	80	12	10	21	10	5	5	8	3	9	3	6	7	6	9	4	9	6	3	10	8	8	14	5	7	7	6	7	9	8	5	2	6	10	
ih	7	11	8	4	5	13	8	11	11	14	12	80	21	13	7	7	18	14	3	11	6	9	8	6	7	3	11	6	12	6	12	10	8	6	10	6	7	9	10	4	7	11	3	6	7
ix	8	12	13	6	5	11	12	15	8	14	10	21	80	8	10	14	10	21	6	9	7	7	8	6	9	2	7	8	7	1	8	7	13	9	7	6	15	10	12	7	4	6	3	4	6
iy	6	7	6	2	4	16	4	17	7	6	21	13	8	80	11	8	8	12	8	7	3	9	7	5	6	2	8	7	5	7	5	8	16	10	9	12	5	6	3	6	5	3	5	5	14	4
ow	15	17	13	11	12	12	8	7	7	9	10	7	10	11	80	4	12	12	8	9	7	6	3	6	12	5	5	7	2	9	10	8	8	9	5	8	9	7	5	7	6	5	7	0	7
oy	3	5	6	4	0	7	1	16	6	4	5	7	14	8	4	60	4	5	6	5	6	6	11	8	7	2	8	3	3	3	2	5	5	5	9	4	8	4	6	1	2	3	3	7	3
uh	6	9	11	7	8	10	9	6	6	12	5	18	10	8	12	4	80	14	6	13	5	7	3	5	10	8	6	4	9	6	10	7	13	9	6	11	7	6	9	9	7	7	4	0	5
uw	4	5	7	5	4	15	11	10	2	9	8	14	21	12	12	5	14	80	6	6	5	3	3	11	4	9	2	6	6	8	7	20	8	10	3	11	3	10	5	1	5	4	5	3	
b	8	4	3	5	10	4	9	11	2	9	3	3	6	8	8	6	6	6	80	7	14	13	10	6	8	9	3	9	6	11	8	3	12	5	10	7	4	10	9	12	4	2	8		
ch	7	5	8	9	2	7	4	9	8	11	9	11	9	7	9	5	13	6	7	80	9	12	3	2	5	13	10	3	20	15	8	3	9	4	4	11	4	18	20	15	7	6	3	0	22
d	6	8	3	5	3	7	7	9	6	7	3	6	7	3	7	6	5	8	14	9	80	18	10	3	5	11	16	2	11	9	6	12	13	13	6	3	6	9	13	12	14	11	6	3	9
dh	5	6	9	6	5	5	7	10	3	9	6	9	7	9	6	6	7	5	13	12	18	80	17	3	6	6	14	9	10	8	6	12	9	7	15	5	7	10	9	8	13	10	6	3	9
dx	5	11	7	5	7	8	4	7	5	6	7	8	8	7	3	3	11	3	10	3	10	17	80	4	8	8	16	5	8	3	5	15	7	7	15	0	10	6	7	3	5	12	8	5	3
el	6	8	5	7	10	5	3	11	7	2	6	6	6	5	6	8	5	3	6	2	3	4	8	80	6	3	5	4	5	3	16	3	3	4	4	4	5	1	2	4	2	4	8	6	3
en	7	8	3	4	6	7	5	5	5	4	9	7	9	6	12	7	10	11	8	5	5	6	8	6	80	5	3	1	4	3	5	9	25	15	8	4	6	6	2	9	4	7	3	4	4
f	7	6	5	8	4	6	6	4	4	5	4	3	2	2	5	2	8	4	8	13	11	6	8	3	5	80	5	4	11	10	4	1	8	5	3	9	5	7	15	6	14	7	2	2	6
g	5	8	9	6	4	8	6	10	6	6	9	11	7	8	5	8	6	9	9	10	16	14	16	5	3	5	80	7	10	17	5	5	6	10	9	5	9	11	6	13	9	8	3	5	11
hh	6	9	7	7	4	2	5	8	7	9	6	6	8	7	7	3	4	2	3	3	2	9	5	4	1	4	7	80	6	4	6	10	3	3	11	11	8	9	6	3	7	7	2	4	9
jh	6	8	5	8	10	5	2	10	2	5	3	12	7	7	2	3	9	6	9	20	11	10	8	5	4	11	10	6	80	8	6	5	6	8	9	2	4	18	6	9	13	5	4	4	
k	5	5	4	7	5	8	4	7	3	10	10	6	1	5	9	3	6	6	6	15	9	8	3	3	3	10	17	4	8	80	5	3	3	8	4	20	8	12	7	17	13	7	0	1	10
l	7	7	5	15	13	4	8	11	9	5	8	12	8	8	10	2	10	8	6	8	6	6	5	16	5	4	5	6	6	5	80	10	9	3	6	10	8	5	4	4	5	1	21	5	3
m	8	7	2	6	6	9	6	4	5	8	8	10	7	16	8	5	7	7	11	3	12	12	15	3	9	1	5	10	5	3	10	80	17	14	17	5	6	11	6	2	9	7	11	5	7
n	6	3	3	6	4	11	7	3	6	7	8	8	13	10	8	5	13	20	8	9	13	9	7	3	25	8	6	3	5	3	9	17	80	27	9	7	4	6	8	9	7	8	4	3	5
ng	5	6	9	4	5	12	3	4	5	8	14	6	9	9	9	5	9	8	3	4	13	7	7	4	15	5	10	3	6	8	3	14	27	80	6	10	6	3	0	6	10	10	4	3	2
nx	2	9	8	7	6	7	4	10	6	7	5	10	7	12	5	9	6	10	12	4	6	15	15	4	8	3	9	11	8	4	6	17	9	6	80	3	9	6	6	5	7	10	8	9	6
p	9	7	7	11	7	9	4	5	3	8	7	6	6	5	8	4	11	3	5	11	3	5	0	4	4	9	5	11	9	20	10	5	7	10	3	80	8	9	9	17	17	7	1	0	10
r	9	13	8	8	11	9	21	11	8	21	7	7	15	6	9	8	7	11	10	4	6	7	10	5	6	5	9	8	2	8	8	6	4	6	9	8	80	6	4	6	9	3	10	4	5
s	5	8	7	3	2	2	7	8	5	8	6	9	10	3	7	4	6	3	7	18	9	10	6	1	6	7	11	9	4	12	5	11	6	3	6	9	6	80	13	16	10	5	0	1	48
sh	6	5	5	1	1	6	4	8	5	9	7	10	12	6	5	6	9	10	4	20	13	9	7	2	2	15	6	18	7	4	6	8	0	6	9	4	13	80	9	11	11	5	3	11	
t	3	6	3	4	5	6	10	5	5	7	9	4	7	5	7	1	9	5	10	15	12	8	3	4	9	6	13	3	6	13	4	2	9	6	5	17	6	16	9	80	12	7	2	0	15
th	9	7	7	4	9	5	8	4	6	11	8	7	4	3	6	2	7	1	9	7	14	13	5	2	4	14	9	7	9	13	5	9	7	10	7	17	9	10	11	12	80	16	2	1	11
v	4	3	3	1	5	9	2	5	2	6	5	11	6	5	5	3	7	5	12	6	11	10	12	4	7	7	8	7	13	7	1	7	8	10	10	7	3	5	11	7	16	80	2	3	6
w	6	9	5	11	16	5	9	4	10	3	2	3	3	5	7	3	4	4	4	3	6	6	8	8	3	2	3	2	5	0	21	11	4	4	8	1	10	0	5	2	2	2	80	7	0
y	5	3	3	5	5	2	3	6	7	5	6	6	4	14	0	7	0	5	2	0	3	3	5	6	4	2	5	4	4	1	5	5	3	3	9	0	4	1	3	0	1	3	7	80	0
z	6	6	5	3	2	3	6	10	6	7	10	7	6	4	7	3	5	3	8	22	9	9	3	3	4	6	11	9	4	10	3	7	5	2	6	10	5	48	11	15	11	6	0	0	80

Appendix C

Statistical Significance Tests

The statistical significance test suite from NIST (National Institute of Standards and Technology) is used in ARPA evaluations of automatic speech recognition technologies. It encompasses four tests:

- Matched Pair Sentence Segment (Word Error) Test (MP)

- Signed Paired Comparison (Speaker Word Accuracy) Test (SI)

- Wilcoxon Signed Rank (Speaker Word Accuracy) Test (WI)

- McNemar (Sentence Error) Test (MN).

Here, we apply the test suite to gauge the accuracy differences among four context-independent (CI) SDCHMM systems and four context-dependent (CD) SDCHMM systems of Table 5.2. The four systems in each case have 1, 13, 20, and 39 streams. The eight systems are identified as follows:

- 1stream.g2254.ci — 1-stream CI SDCHMM system with 2,254 full-space Gaussians

- 13stream.g256.ci — 13-stream CI SDCHMM system with 256 subspace Gaussians prototypes per stream

- 20stream.g128.ci — 20-stream CI SDCHMM system with 128 subspace Gaussians prototypes per stream

- 39stream.g32.ci — 39-stream CI SDCHMM system with 32 subspace Gaussians prototypes per stream

- 1stream.g76154.cd — 1-stream CD SDCHMM system with 76,154 full-space Gaussians

- 13stream.g128.cd — 13-stream CD SDCHMM system with 128 subspace Gaussians prototypes per stream

- 20stream.g64.cd — 20-stream CD SDCHMM system with 64 subspace Gaussians prototypes per stream

- 39stream.g16.cd — 39-stream CD SDCHMM system with 16 subspace Gaussians prototypes per stream.

The test results are shown in Table C.1 and Table C.2. A system identifier showing up in a table entry implies it is the better system when compared with the system having the row identifier. We see that results from the three tests other than the MP Test suggest no statistically significant difference in performance among the four systems in each of the two groups. On the other hand, the MP Test finds that the SDCHMM systems are actually better than the original CDHMM system (i.e. 1-stream SDCHMM system). Since each of the four test measures different quantities (word errors, sentence errors and speaker word errors) and employs different assumptions, it is fair to conclude that the new SDCHMM systems are as accurate as the original CDHMM systems.

Table C.1: ATIS: Statistical significance tests on the best CI SDCHMM systems

```
                    Composite Report of All Significance Tests
                     For the Context-Independent SDCHMM Test

                        Test Name                                 Abbrev.
             ----------------------------------------------       -------
             Matched Pair Sentence Segment (Word Error) Test         MP
             Signed Paired Comparison (Speaker Word Accuracy) Test    SI
             Wilcoxon Signed Rank (Speaker Word Accuracy) Test        WI
                     McNemar (Sentence Error) Test                    MN
```

```
|-----------------------------------------------------------------------------------------------------
|                    | 1stream.g2254.ci |  13stream.g256.ci  |  20stream.g128.ci   |  39stream.g32.ci    |
|--------------------+------------------+--------------------+---------------------+---------------------+------------
| 1stream.g2254.ci  |                  | MP 13stream.g256.ci | MP 20stream.g128.ci | MP 39stream.g32.ci |
|                    |                  | SI            same | SI            same | SI            same |
|                    |                  | WI            same | WI            same | WI            same |
|                    |                  | MN            same | MN            same | MN            same |
|--------------------+------------------+--------------------+---------------------+---------------------+------------
| 13stream.g256.ci  |                  |                    | MP 20stream.g128.ci | MP 39stream.g32.ci |
|                    |                  |                    | SI            same | SI            same |
|                    |                  |                    | WI            same | WI            same |
|                    |                  |                    | MN            same | MN            same |
|--------------------+------------------+--------------------+---------------------+---------------------+------------
| 20stream.g128.ci  |                  |                    |                     | MP 39stream.g32.ci |
|                    |                  |                    |                     | SI            same |
|                    |                  |                    |                     | WI            same |
|                    |                  |                    |                     | MN            same |
|--------------------+------------------+--------------------+---------------------+---------------------+------------
| 39stream.g32.ci   |                  |                    |                     |                     |
|                    |                  |                    |                     |                     |
|                    |                  |                    |                     |                     |
|                    |                  |                    |                     |                     |
|-----------------------------------------------------------------------------------------------------
```

Table C.2: ATIS: Statistical significance tests on the best CD SDCHMM systems

```
                  Composite Report of All Significance Tests
                     For the Contest-Dependent SDCHMM Test

                  Test Name                                  Abbrev.
            --------------------------------------------     -------
            Matched Pair Sentence Segment (Word Error) Test      MP
            Signed Paired Comparison (Speaker Word Accuracy) Test SI
            Wilcoxon Signed Rank (Speaker Word Accuracy) Test     WI
                  McNemar (Sentence Error) Test                   MN
```

	1stream.g76154.cd	13stream.g128.cd	20stream.g64.cd	39stream.g16.cd
1stream.g76154.cd		MP 13stream.g128.cd SI same WI same MN same	MP 20stream.g64.cd SI same WI same MN same	MP 39stream.g16.cd SI same WI same MN same
13stream.g128.cd			MP 20stream.g64.cd SI same WI same MN same	MP 39stream.g16.cd SI same WI same MN same
20stream.g64.cd				MP 39stream.g16.cd SI same WI same MN same
39stream.g16.cd				

133

Biographical Note

Brian Kan-Wing MAK was born on the 28th of June, in an interesting year — which, when written in Arabic numerals on a piece of paper, reads the same when the paper is turned 180° — in Hong Kong under the British rule. He received his B.Sc. (Eng.) in Electrical Engineering from the University of Hong Kong in 1983. He did a lot of soul searching during his college years, and was devoted to teaching upon graduation at St. Stephen's College (a high school) in Stanley, Hong Kong. In 1985, he obtained his Certificate of Education from the School of Education, University Hong Kong with distinction.

Under the urge to learn more, he went abroad to further study and received his M.S. in Computer Science from the University of California, Santa Barbara, USA, in 1989. He spent the next three years as a Research Programmer in Speech Technology Laboratory of Panasonic Technologies Inc., Santa Barbara. In the end of 1992, he went to the Chinese University of Hong Kong, where he worked as a Research Assistant on a project of Cantonese Speech Recognition in the Department of Electronic Engineering. It was at that time he finally decided on a future career of speech research. In September 1993, he began his pursuit of a Ph.D. in Computer Science from the Oregon Graduate Institute of Science & Technology, Portland, Oregon, USA. During his Ph.D. study, he had performed perceptual experiments to understand the relative contribution of vowels and consonants in human speech understanding, and research on phone clustering and combining artificial neural networks to improve speech recognition performance. In the Summer of 1996, he went to AT&T Labs on an internship; his mentor was Enrico Bocchieri. The Summer project later turned into his Ph.D. thesis. Since December 1996, he has been a Research Consultant at AT&T Labs while completing his Ph.D. dissertation.

After receiving the Ph.D., he plans to teach and conduct research at the Hong Kong University of Science and Technology.

Brian's main research interest is automatic speech recognition. While he is generally interested in all the various components of a speech recognition system, his current research concentrates on acoustic modeling and speaker adaptation. In addition, he is interested in the closely related area of machine learning.