INVESTIGATION OF SUBSPACE GAUSSIAN MIXTURE MODEL FOR AUTOMATIC SPEECH RECOGNITION

A Project Report

submitted by

ATTA SWETHA

in partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY

Department of Electrical Engineering

Indian Institute of Technology Madras, India.

JUNE, 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **"Investigation of Subspace Gaussian Mixture Modelling for Automatic Speech Recognition"**, submitted by **Atta Swetha (EE12M111)**, to the Indian Institute of Technology, Madras for the award of the degree **Master of Technology**, is a bona fide record of the research work done by him under my supervision. The contents of the thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Umesh S
Research Guide
Professor
Department of Electrical Engineering
IIT Madras, 600 036

Place: Chennai
Date: 13th June 2014

ACKNOWLEDGEMENTS

First, I would like to thank my project adviser, Dr. Umesh S. He has continuously encouraged to challenge myself and explore the domain to its depths and intricacies. He has provided me with a research atmosphere that allowed me delve into pressing questions in the field. He has been a constant source of inspiration and support for me. Thank you.

I would like to thank my lab mates Sekhar, Bhargav, Neethu, Basil, Angel and others, for their valuable inputs to help me through with ideas. I would like to thank them for making the entire project a wonderful learning experience. I would like to thank all my friends for their continuous support through my four years at this institute.

I would like to thank all my professors and teachers for their continuous mentoring throughout the course of my studies. I would like to thank the Department of Electrical Engineering for providing me a unique learning experience that enables to compete with the best in the world. I would like to thank IIT Madras for providing me exciting opportunities and making my whole Postgraduate education a part of my life that is worth remembering forever.

I would like to thank my parents for their love and encouragement throughout the course of my studies and my entire life. I have no words to express my gratitude to my parents for making me what I am today.

ABSTRACT

KEYWORDS: GMM; SGMM;

In this thesis, an acoustic modelling technique, Subspace Gaussian Mixture Modelling, for Speech Recognition Introduced by Daniel Povey has been Investigated on Mandi databases of Indian languages for Tamil and Hindi . Various simulations have been performed by varying different parameters involved in SGMM and results have been obtained. Comparison of performance is done for various parameters and also with CDHMM and LDA+MLLT. The exact procedure to be followed and the various optimized parameters have been explained in detail. The significance of each parameter is also explained. Also, the results for various tied states and UBMs have been given in detail.

CONTENTS

LIST OF TABLES

ABBREVIATIONS

ASR Automatic Speech Recognition

CAT Cluster Adaptive Training

CDHMM Continuous Density Hidden Markov Model

CMVN Cepstral Mean and Variance Normalization

EM Expectation Maximization

GMM Gaussian Mixture Model

HMM Hidden Markov Model

MFCC Mel-frequency Cepstral Coefficients

MLLR Maximum Likelihood Linear Regression

p.d.f. Probability Density Function

TIMIT Texas Instruments MIT

SGMM Subspace Gaussian Mixture Model

WER Word Error Rate

UBM Universal Background Model

LDA Linear Discriminant Analysis

MLLT Maximum Likelihood Linear Transform

Chapter 1

INTRODUCTION

Automatic Speech Recognition (ASR) is a prominent field that aims at conversion of spontaneous speech into machine understandable text. It is a difficult problem because of the different kinds of variability in speech due to changes in speaker and environment. Statistical parametric models like HMM are generally used to model the production of speech sounds. The performance of the speech recognition systems entirely depends on the how good the modeling is and how well the parameters of the model can be estimated using the available training data. There is a lot of focus on using compact modeling techniques that can be easily trained with limited resources. This is of particular interest in the context of Indian languages, many of which have considerably less data resources than English and other European languages.

In conventional CDHMM systems that are typically used in speech recognition applications, the p.d.f. of each HMM state is a Gaussian Mixture Model (GMM). A lot of parameters(means, variances and weights) are required to define these GMMs, thus demanding a large amount of training data. A relatively new acoustic modeling technique, known as SGMM, was introduced in Povey (2009), which takes advantage of the high correlation between the state's distributions to generate the GMM parameters indirectly using only a small number of state specific parameters. The state GMM parameters are constrained to lie in a low dimensional subspace of the total parameter space. The parameters that are used to define this subspace are shared among all the states and thus can be estimated robustly using limited amount of data and even out-of-domain data. This has been verified through several multilingual experiments(Burget et al. (2010), Mohan et al. (2012)).

We introduce a very different UBM based approach that has fewer parameters, and it is shown that it can be discriminatively trained and still provide a performance improvement under ML training similar to our previous UBM based approach. We are introducing here a subspace approach, in which a vector of low dimension (e.g.50) controls all the mean and weight parameters of the speech-state specific mixture model. We also generalize to have a mixture of substates in each state, i.e. each state's distribution is controlled by a number of these 50-dimensional vectors each with its own mixture weight.

Chapter 2

SPEECH RECOGNITION

2.1 Introduction

Information in the real world is communicated in the form of signals. Most of these signals (like speech signals) are generated continuously in time and are analog in nature (can take continuous values). But in all practical applications, we can extract only a finite number of samples of the signal and they need to be quantized to take only a finite number of values. The statistics of signals such as speech vary over time and hence are non-stationary. But they can be assumed to be stationary over a short observation window (25ms) and fall into a category of pseudo-stationary signals. This allows us to model the signals with efficient parametric models.

The models used to characterize signals can be broadly classified into deterministic and statistical models. Signals like speech can be modelled as the outcome of a random process and the parameters of this process can be estimated accurately. For temporal pattern recognition applications like speech recognition, stochastic models known as Hidden Markov Models (HMM) are widely used. It is called "hidden" because the underlying states are not observed; but only the output of the states is observed. The output is conventionally modelled to be generated from a Gaussian Mixture Model (GMM). This is referred to as the HMM-GMM system.

Section 2.2 gives a brief introduction to the speech recognition problem and the HMM based speech recognition system. Section (2.3) describes the conventional HMM-GMM system. The subsequent section reviews more complex approaches to modelling and adapting the GMM-based systems. Section 2.4 describes the Subspace Gaussian Mixture Model (SGMM)based system

2.2 HMM-based Speech Recognition

The pseudo-stationary property of speech signals allows the speech signal to be divided into25ms observation windows. The statistical properties of the signal can be assumed to be constant over this window. The data in this window is converted into discrete parameter vectors. This process of conversion of continuous speech signal into a sequence of discrete vectors is known as Feature Extraction. These vectors are also known as feature vectors or observation vectors. One of the most widely used features is the Mel Frequency Cepstral Coefficients(MFCC). The objective of the speech recognition system is to convert this sequence of observations into a sequence of symbols (or words) that can be “understood” by a machine. The observation sequence can be modeled as to be generated by a sequence of states as defined by a HMM. A typical acoustic modeling uses a 3-state left-to-right HMM topology to model the features generated by a single phonetic unit. A first order Hidden Markov process is assumed meaning that the transition into a particular state depends only on the previous state and that the observation depends only on the current state. The following characterizes the HMM:

- N, the number of states in the model. The set of states in the model is defined by S $=\{S_1, S_2, .., S_N\}$ .The state at the observation window or frame t is denoted as q_t. For the model of a basic phonetic unit such as a phoneme, we typically use N = 3.
- A, the state transition probability distribution. A = $\{a_{ij}\}$ where

  $$a_{ij} = P\left[q_t = S_j \middle| q_{t-1} = S_i\right] 1 \leq i, j \leq N.$$

  For speech systems, we use a left-to-right topology, which implies that aij = 0 for j < i.

- π, the initial state distribution. $\pi = \{\pi_i\}$ where

  $$\pi_i = P[q_0 = S_i], 1 \leq i \leq N.$$

  The model of a basic phonetic unit such as phoneme has $\pi_i = 0$ for $i \neq 1$

- The observation probability distribution in state j. In the case of a discrete HMM with output vectors $v_1, v_2, \dots, v_k$ the probability of observing v_k in the state j is given by

  $$b_j(k) = P\left[v_k \; at \; t \middle| q_t = S_j\right], 1 \leq j \leq N, 1 \leq k \leq M.$$

  The observation vector, x(t), can assume to be generated from a continuous distribution. The probability density function (p.d.f) can be modelled as a mixture of Gaussians or a GMM:

  $$b_j(x(t)) = P\left[x(t) \middle| q_t = S_j, \mu_{ji}, \Sigma_{ji}\right] = \sum_{i=1}^{I} w_{ji} N(x(t); \mu_{ji}, \Sigma_{ji}), 1 \leq j \leq N.$$

  where I is the number of Gaussians in the GMM; μ_{ji}, Σ_{ji} are the means and the covariance matrix of the Gaussian component i of state j; and w_{ji} is the Gaussian prior or the Gaussian weight with the constraint $\sum_{i=1}^{I} w_{ji} = 1$.

  The parameters of the HMM can be put together as a parameter set λ.

2.2.1 Linguistic units

The basic linguistic unit that we model is the phoneme (also referred to as monophones or just phones). There are around 40 phones in English language. Using only these gives a very simplistic model. For large vocabulary recognition, we need to look at the left and the right context of the phone; i.e. we need to model the co-articulation in vocal tract by considering the phones uttered before and after the phone in consideration. Such a model is called a triphone model. There are as many as 40^3triphones possible, but many of them are not used or are not observed in the training data. The GMMs used to model the triphones have many parameters to be estimated. We require a large amount of data to get a good estimate of the parameters. So, we “tie” similar triphones using a decision-tree based top-down clustering approach. The decision tree based clustering has been described in detail in Young et al. (1994). At the end of such a clustering process, we get a few thousand triphone models

.

2.3 HMM-GMM system

HMM-GMM system, also known as CDHMM system, is the conventionally used system for speech recognition. It models each context-dependent phone (usually the triphone) with a generative model based on a left-to-right three state HMM topology. The total number of context-dependent phonetic states after tree-based clustering is of the order of a few thousands. Each state is denoted by the index j with $, 1 \leq j \leq J.$. The observation vector is assumed to be generated within each HMM state j from a GMM:

$$P(x|j) = \sum_{i=1}^{M_j} w_{ji} N(x; \mu_{ji}, \Sigma_{ji}),$$

where x is the observation vector, w_{ji},μ_{ji}, Σ_{ji} are the prior, mean and covariance matrixof the i^{th} Gaussian component and M_j is the number of Gaussians in the j^{th} state.

2.4 Subspace Gaussian Mixture Model (SGMM)

SGMM is similar to the GMM-based system, but the model parameters for each state are specified by a single state vector v_j. Thus $\mu_i$lies in a state- independent subspace defined by the columns of M_i. The covariance is shared across all states, so that we have a state-independent Σ_i. The basic model can be expressed as:

$$P(x|j) = \sum_{i=1}^{I} w_{ji} N(x; \mu_{ji}, \Sigma_{ji}),$$

$$w_{ji} = \frac{\exp\,(w_i^T v_j)}{\sum_{i'=1}^{I} \exp\,(w_{i'}^T v_j)}$$

$$\mu_{ji} = M_i v_j$$

where $v_j \epsilon \mathbb{R}^s$is the state projection vector, x is the feature vector, $M_i$and $w_i$define thesubspaces in which the means and the unnormalized log weights respectively lie and $\Sigma_i$is the shared covariance. j is the index of the context-dependent state ($1 \leq j \leq J$.) with J in the order of a few thousands. i is the Gaussian index in the GMM of I mixtures (usually 200 < I < 2000). $v_j$is the only state specific parameter. $M_i, w_i, \Sigma_i$are “shared” parameters.The basic strategy of the SGMM is to reduce the number of state specific parameters andincrease the number of shared (global) parameters. The intuition is that the means of the tied state models span a smaller subspace of the entire acoustic space. This allows us to reduce the number of state specific parameters. Also, since the global parameters do not depend on a specific phone, there is a lot of data available to train the parameters. It is possible to train these parameters using out-of-domain data even from other languages as shown in Povey et al.(2011a).

2.4.1 Training procedure

The training of the SGMM system begins with the traditional HMM-GMM system. First, a large GMM consisting of all the gaussians in the HMM-GMM system is built. This is typically in the order of tens of thousands. The gaussians are repeatedly merged to get a desired number of gaussians with diagonal covariances. The actual procedure of doing this can be found in Povey et al. (2011a). These gaussians are trained with around 8 iterations of EM algorithm for full covariance re-estimation. The resulting model is called a Universal Background Model(UBM). The UBM can be viewed as a compact model representing all kinds of speech from all speakers. The UBM need not necessarily be built from a specific HMM-GMM system; any generic UBM can be used. This UBM is used to initialize the SGMM model. This is done in such a way that the initial p.d.f. of all states is equal to the UBM. The HMM-GMM system provides the Viterbi alignments for the initial SGMM parameter re-estimation iterations. Once the SGMM parameters are estimated by EM algorithm to a sufficient extent, the SGMM training can be continued with self-alignment (alignments from the SGMM itself).

Chapter 3

SUBSPACE GAUSSIAN MIXTURE MODELS

This model is a large shared GMM whose parameters vary in a subspace of relatively low dimension (e.g. 50), thus each state is described by a vector of low dimension which controls the GMM's means and mixture weights in a manner determined by globally shared parameters. In addition we generalize to having each speech state be a mixture of substates, each with a different vector. This technique was introduced by Daniel Povey (2009) in his paper "Subspace Gaussian Mixture Models for Speech Recognition".

What we are introducing here is a subspace approach, in which a vector of low dimension (e.g.50) controls all the mean and weight parameters of the speech-state specific mixture model. We also generalize to have a mixture of substates in each state, i.e. each state's distribution is controlled by a number of these 50-dimensional vectors each with its own mixture weight.

3.1 Basic model

In this section we describe the Subspace Mixture Model. First we describe the basic model without substates. We use the index $1 \leq i \leq I$ to represent the Gaussians in the UBM (e.g. I = 750 Gaussians), and the index $, 1 \leq j \leq J$to represent the clustered phoneticstates (e.g. J = 8000 for a typical large vocabulary system). Let the feature dimension be $, 1 \leq d \leq D$, e.g. D = 40, and let the subspace dimension be $, 1 \leq s \leq S$, e.g. S = 50. The subspace dimension can take any value; it represents the number of different directions in which we allow the phonetic states to differ from each other.

For each state j, the probability model $P(x|j)$is:

$$P(x|j) = \sum_{i=1}^{I} w_{ji} N(x; \mu_{ji}, \Sigma_{ji}),$$

$$w_{ji} = \frac{\exp\,(w_i^T v_j)}{\sum_{i'=1}^{I} \exp\,(w_{i'}^T v_j)}$$

$$\mu_{ji} = M_i v_j$$

Thus, each state has a shared number of mixtures (e.g., I = 750).The means vary linearly with the state-specific vector v_j (we denoteby v_j the same vector, extended with a 1, to handle constant offsets).The log weights prior to normalization also vary linearly with v_j .The parameters of the system are the mean-projection matrices$M_i$the weight-projection vectors w_i, the variances Σ_i, and the statespecific vectors v_j . To give the reader a feel for the number of parameters involved, for the values of I, J,D and S mentioned above the total number of parameters would be, in reverse order of size: mean-projections,IDS $= 750 \times 40 \times (50 + 1) = 1.53 \times 10^6$;variances,$\frac{1}{2} ID(D + 1) = \frac{750 \times 40 \times 41}{2} = 0.615 \times 10^6$; state-specific vectors,JD $= 0.4 \times 10^6$, weight-projections, IS = $750 \times (50+1) = 38.25 \times 10^3$. Thus the total number of parameters is 2.58×10^6, andmost of the parameters are shared, not state-specific. For reference, atypical mixture-of-Gaussians system might have 100000 Gaussiansin total, each with a 40-dimensional mean and variance, which givesus 8×10^6parameters total, more than twice this subspace GMM system. Note that the quantity of state-specific parameters in the subspace GMM system is less than one tenth of that in the normal GMM system. For this reason, we extend the model to include mixtures of substates.

3.2 Subspace mixture model with substates

The subspace mixture model with substates is the same as in Equations 1 to 3 except each state is now like a mixture of states; each state j has substates $1 \leq m \leq M_j$with associated vectors $v_{jm}$and mixture weights $c_{jm}$with$\sum_{m=1}^{M_j} c_{jm} = 1$ we can write out themodel as:

$$P(x|j) = \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} N(x; \mu_{jmi}, \Sigma_i)$$

$$\mu_{jmi} = M_i v_{jm}$$

$$w_{jmi} = \frac{\exp\,(w_i^T v_{jm})}{\sum_{i'=1}^{I} \exp\,(w_{i'}^T v_{jm})}$$

It is useful to think about the substates as corresponding to Gaussians in a mixture of Gaussians, and in fact as we describe later, we use a similar mixing up procedure to increase the number of states. This model is in effect a mixture of mixtures of Gaussians, with the total number of Gaussians in each state being equal to I J_m. Clearly this large size could lead to efficiency problems. In fact, computing each mean would involve a matrix multiply taking time O(SD), and since the variances $\Sigma_i$are not diagonal the actual likelihood computation would be O(D^2). In the next section we show that despite this, likelihoods given this model can be computed in a time similar to a normal diagonal mixture of Gaussians.

3.3 SUBSPACEMODEL TRAINING

The subspace model training proceeds as follows. Firstly we initialize the UBM, which is a mixture of full-covariance Gaussians that models all speech data regardless of speech state or speaker. Next, we do a first pass of accumulation and update, using a previous system to align speech states to frames. In this first pass of accumulation and update, we are essentially estimating the basic subspace mixture model without substates or speaker vectors. In later passes over the data, we accumulate different kinds of statistics and the update equations have a different form.

3.4 UBM initialization

The method we use for initialization of the UBM parameters $\overline{\mu_\iota}$ and$\overline{\Sigma_\iota}$i may not be optimal as we have not experimented with this. Wetake an already-trained conventional diagonal Gaussian system andcluster the Gaussians into I clusters (e.g. 750). This is done byconsidering all the Gaussians as one large mixture model (using as weights the weights within each state, divided by the total number of states), and then computing the mixture of I Gaussians that maximizes the auxiliary function likelihood. The algorithm we use to compute this is like a form of k-means except with pruning to avoid excessive compute (this involves a notion of neighbouring clusters), starting from a random assignment to clusters. The variances are thus initialized to diagonal. From that point we do 3 iterations of EM over a subset (e.g. 1/10) of the training data, updating the means and (full) variances but leaving the mixture weights uniform to encourage even distribution of data.

3.5 First pass of training: accumulation

The first pass of training involves getting mean statistics for each state j and UBM index i, and using this to initialize the parameters with a single vector per state. By storing statistics in a different form for the first iteration of update than for later iterations, we can avoid making unnecessary passes over the data. However, to store the mean statistics requires a lot of memory and storage: e.g. for our example system using floats, it would take 4IJD =4 × 750 × 8000 × 40 bytes of memory, or 0.96 GB. To reduce this, we avoid storing statistics with very small counts, as we describe below. Our state posteriors $\gamma_j(t)$ are zero-one posteriors based onViterbi alignments obtained using a baseline (mixture-of-Gaussians) system. On each frame we also compute UBM Gaussian posteriors $\gamma_i(t)$ (with pruning to the top 5 as described above). We thencompute initial posteriors:

$$\gamma_{ji}(t) = \gamma_j(t)\gamma_i(t)$$

The statistics we accumulate are count statistics (sums of the posteriors)and state-specific mean statistics, and also a scatter for each UBM Gaussian index which we will use to compute within-class covariances Σ_i. There is a slight complication in that we want to avoid accumulating mean statistics where the count is very small. Therefore we define the "pruned" count $\widetilde{\gamma_{ji}}(t)$ to be zero if the sum of$\gamma_{ji}(t)$ up to the current point in the current parallel job is less than a threshold τ (we have used τ values from 0.1 to 2 depending onsystem size). The statistics we accumulate are named $\widetilde{m_{ji}}$ for thefirst order statistics and $\widetilde{S_i}$ for the scatter to emphasize that they are accumulated using the pruned counts. So we have:

$$\gamma_{ji} = \sum_{t=1}^{T} \gamma_{ji}(t)$$

$$\widetilde{\gamma_{ji}} = \sum_{t=1}^{T} \widetilde{\gamma_{ji}}(t)$$

$$\widetilde{m_{ji}} = \sum_{t=1}^{T} \widetilde{\gamma_{ji}}(t)x(t)$$

$$\widetilde{S_i} = \sum_{t=1}^{T} \sum_{j=1}^{J} \widetilde{\gamma_{ji}}(t)x(t)x(t)^T$$

3.5.1 First pass of training: update

The first pass of update is an iterative one in which we first initialize the vectors to random values (e.g. Gaussian noise), initialize the projections to zero and the variances to the UBM variances, then iteratively optimize in turn each of the four types of parameters: the weight-projection vectors w_i,the mean-projection matrices M_i, the variances $\Sigma_i$and the state-specific vectors v_j (at this point we have no substates). This is done for about ten iterations.

3.5.2 Weight-projection vector update

The update of the weight-projection vectors $w_i$is based on maximizingthe auxiliary function:

$$Q(\dots) = \sum_i \sum_j \gamma_{ji} log w_{ji} = \sum_{i,j} \gamma_{ji} (w_i^T v_j - log \sum_{i'=1}^{I} exp w_{i'}^T v_j)$$

We can use the inequality$1 - \left(\frac{x}{\bar{x}}\right) \leq -log\left(\frac{x}{\bar{x}}\right)$ (which is anequality at$x = \bar{x}$), to maximize, where $\overline{w_i}$ is the pre-update value of w_i.

To maximize the above we use a second order approximation to the exponential function, but then in certain cases we take a heuristic over estimate of the negated second gradient, for safety; this leads to the max(·) function below (without this heuristic we would just have its first term). The update procedure is as follows. First we compute all the un-normalized log weights, let us call them x_{ji}=$w_i^T v_j$, and the normalizers $x_j = log \sum_i exp x_{ji}$ these are used tocompute the weights $w_{ji} = \exp(x_{ji} - x_j)$ during the computation. We also compute the total counts per state$\gamma_{ji} = \sum_{i=1}^{I} \gamma_{ji}$. Then for each UBM Gaussian index i we compute the first order term g_i andnegated second order term Hi in a quadratic approximation to theauxiliary function in$w_i - \overline{w_i}$, i.e. around the current point. After updating eachw_i, we update the affected x_{ji} and the x_j beforeupdating the next i so we can continue with up to date values ofw_{ji}. The value of the auxiliary function should be checked as wecannot prove that this procedure will converge, although we havenever observed it not converging.

3.5.3 Mean-projection matrix update

The update for the mean-projection matrices M_i (which have sizeD × S + 1) is as follows. For a particular i, we first make a coordinatechange so that the variance $\Sigma_i$is unit. We use the transform $= \Sigma_i{}^{-0.5}$, and project to get $M_i{}' = TM_i$ in the new co-ordinates. Then the computation is as follows: for each of its D rows m'_{id} wewill compute a linear term g_{id} of the auxiliary function as a functionof the change in that row, and a negated quadratic term H_i which isshared for all d.

$$g_{id} = \sum_j \left(T_i \widetilde{m_{ji}} - \widetilde{\gamma_{ji}} M'{}_i v_j\right)_d v_j \; H_i = \sum_j \widetilde{\gamma_{ji}} v_j^T v_j \; m'{}_{id} = m'{}_{id} + \mathrm{H}_i{}^{-1} g_d$$

3.5.4 Vector updates

The update for the state-specific vectors v_j involves incorporatinga quadratic auxiliary function for the means, and our previouslydescribed quadratic approximation to the auxiliary function for theweights. Again we accumulate a linear term g_j and a negatedquadratic term H_j which describe how the auxiliary function varies with a *change* in v_j . In the expressions below, the top line in eachexpression refers to the weights and the bottom line to the means.

We use the notation $x^{-}$to mean the vector x without its last element;for matrices the notation M^{-} means removing the last rowand column.

$$g_j = \sum_{i=1}^{I} (\gamma_{ji} - \gamma_j w_{ji}) w_i^{-} + \sum_{i=1}^{I} (M_i^T \Sigma_i{}^{-1} \, (\widetilde{m}_{ji} - \widetilde{\gamma_{ji}} M_i v_j))^{-} v_j = v_j + \mathrm{H}_j{}^{-1} g_j$$

The matrices only dependent on i in the last equation should be precomputed.

3.6 Later iterations of training: accumulation

The method of accumulation differs in later iterations of training, versus the first iteration. We store statistics in a more memory efficient way, without pruning. This enables a more exact optimization, and also allows us to have more mixtures without increasing the size of the statistics too much. The size of the statistics are dominated by the need to store data counts for each i, j and m. For these later iterations we assume that we already have a “substate” model; we initialize this by having a single substate per state as estimated above, and using unit weight. The state posteriors are, as before, zero-one posteriors based on Viterbi alignment using a previous system.

3.6.1 Discretized posteriors

The within-state posteriors $\gamma_{jmi}(t)$ are computed by evaluating thelikelihoods. However, we also randomlydiscretize the posteriors into steps of typically $= 0.05$. This reducescompute time by getting rid of most very small posteriors, and also allows us to compress the posteriors in memory and on disk in a variable length coding scheme in which counts γ_{jmi} typically takeonly one byte to store. The discretized posteriors $\widetilde{\gamma_{jmi}}(t)$ consist ofthe part of $\gamma_{jmi}(t)$ that can be expressed in whole increments of δ,plus with probability equal to the remaining part divided by δ, oneextra increment of δ. The random element of the discretization processis necessary to preserve expectations. All statistics are storedusing the discretized posteriors.

3.6.2 Statistics

The weight statistics are straightforward:

$$\gamma_{ji} = \sum_{t=1}^{T} \widetilde{\gamma_{ji}}(t)$$

The statistics we store in order to update the vectors v_{jm} are the firstorder term in the quadratic auxiliary function written in terms of thev_{jm} directly (i.e. not in terms of offsets from the current value).Again,x^{-} is x without its last dimension. So we have:

$$x_{jm} = \sum_{t=1}^{T} \sum_{i=1}^{I} \widetilde{\gamma_{ji}}(t) \left(M_i^T \Sigma_i{}^{-1} x(t) \right)$$

3.7 Later iterations of training: update

The update for later iterations of training is somewhat harder to justify than the update for the first iteration. The reason is that there are updates which we do at the same time (for the variance, the vectors and the mean projections) which cannot easily be proved to converge unless they are done on separate iterations. However, we are confident that these parameter types are sufficiently orthogonal that this is not a problem, and in practice we find that our approach converges. Note that when any the updates below refer to other types of parameters (e.g. if the update for $M_i$refers to v_{jm}), this means thepre-update versions of those parameters. This is important becausethe stored statistics are a function of the other parameters, and using the newly updated versions can lead to inconsistency.

3.7.1 Weight-projection vector update

The update for the weight projection vectors is the same as that described in Section 3.5.2, except that we have to replace any sums over j with sums over both j and m. We do the update for up to4 iterations given the stored statistics, or until the auxiliary function improvement per frame is small (e.g. less than 0.0001).

3.7.2 Mean-projection matrix update

The update for the mean-projection matrix is similar to that given in Section 3.5.3 except we formulate the quadratic auxiliary function in terms of the transformed matrix row m'_{id} rather than the offset from its current value. Again we use the data transform $T_i = \Sigma_i{}^{-0.5}$to make the variances unit, so $M'_i = T_i M_i$

3.7.3 Vector update

In the vector update as follows, we split the second gradient H_j intotwo parts that relate to the weights and the means respectively, anduse the second one H_j in our computation of the gradient to convertfrom a formulation in terms of the vector v_{jm}, to the change inthe vector. We make use of the summed counts

$$\gamma_{jm} = \sum_{i=1}^{I} \gamma_{jmi}$$

3.7.4 Variance update

The variance update is trivial:

$$\widehat{\Sigma_\iota} = \frac{S_i}{\sum_{j,m} \gamma_{jmi}}$$

The auxiliary function improvement can be computed as described in Section 3.5.4.

3.7.5 Substate weight

We now have a new parameter to estimate: the weight of substates. This is given by:

$$c_{jm} = \frac{\sum_i \gamma_{jmi}}{\sum_{j,m} \gamma_{jmi}}$$

3.7.6 Mixing up

Here we describe how we increase the number of substates. The initial model has one substate per state. We have a target total number of mixtures per state, e.g. M = 50,000 and we allocate mixture components to states based on a power rule with a default exponent of 0.2. Thus, if a state has total count $\gamma_j \sum_{m,i} \gamma_{jmi}$, thetarget number of mixture components T_j is the closest integer to M $\frac{{\gamma_j}^{0.2}}{\sum_j {\gamma_j}^{0.2}}$. We do mixing up on a subset of iterations (currently{2,4,6,8,10,12}). On each iteration and for each state j, the number of mixture components to split shall be the difference between the target T_j and the current number of mixture components M_j ; butno more than the current M_j . If it is less than that, we split thosewith the largest counts. In addition, we enforce a minimum count for mixtures to be split, which is 200 by default. For each substate vector v_{jm} that is selected to be split, we compute the negated second gradient H_{jm} as used in section 3.7.3, and then compute the scale S =$\left(\frac{H_{jm}}{\gamma_{jm}}\right)^{-0.5}$, which provides a scale to the vector (think of S like a standard deviation). We then compute a random vector whose elements are drawn from zero-mean Gaussian distribution with variance 0.1, and our perturbed vectors shall be v_{jm}± Sr. Weassign half of the old mixture weight to each of the two new mixture components. Mixing up is done after all other phases of update are complete (i.e., starting from the already updated vectors).

3.7.7 Updating the UBM

The UBM parameters $\overline{\mu_\iota}$ and $\overline{\Sigma_\iota}$ which are used for pruning arealso updated in our training setup. This is done by accumulating zeroth, first and second order statistics for each i and doing the normal Gaussian update. The posteriors used are the sum over substate j,m of the posteriors $\gamma_{jmi}(t)$. Because of the discrete nature of thepruning operation it is not easy to say very much theoretically abouthow these parameters should be trained, in fact it might seem safer to leave them fixed. Experiments have failed to show any difference between training and not training these parameters.

Chapter 4

EXPERIMENTS AND RESULTS

4.4.1 Experimental Setup

The performance of the SGMM model is tested on Hindi and Tamil languages of Mandi database along with TIMIT database.

Mandi database is used in Automatic Speech Recognition-based application to help farmers stay updated with the latest commodity price .It is an interactive speech recognition engine that has been developed by a consortium of seven institutions (IIT-M, IIT-K, IIT-B, IIT-G, IIIT-Hyd, TIFR & CDAC-Kol) and is coordinated by IIT-Madras.

The Hindi database consists of 1hr, 3hr, 5hr, and 22hrs of training data along with 5974 utterances of test data. Similarly, TAMIL database also has 1hr, 3hr, 5hr and 22hrs of training data for training along with 3564 utterances for testing. TIMIT has a total of 3,396 utterances for training and 192 utterances for testing. 13-dimensional MFCC were used as features for parameterizing the speech waveforms. The delta and acceleration of these features were augmented to get 39-dimensional features. Cepstral Mean and Variance Normalization (CMVN) were done to increase the noise-robustness of features. The Kaldi toolkit (Povey et al. (2011b)) was used for training and testing the acoustic models. Standard C++ programs in the Kaldi toolkit were used to build the baseline HMM-GMM system and also LDA+MLLT to initialize the SGMM acoustic models. The SGMM system is implemented using the standard programs in the toolkit. Various libraries in the toolkit were used for the standard computations in the algorithms

4.2 Parameters

The LDA+MLLT system used for TIMIT task has a total of 1040 tied states and 22047 Gaussians. The dictionary had a set of 38 phones. The silence was modelled as a context independent phone with a 8 state HMM, while all other phones were context-dependent with 3 state HMMs. This was used to initialize the SGMM model. Since the feature vector used was of 39 dimension, full-MLLR matrices of dimension 39 x 40 was used for the cluster transforms. The UBM was initialized by a bottomupclustering approach by merging the Gaussians from the LDA+MLLT system till I mixtures were obtained. I was varied from in a range to obtain best result.

The baseline LDA+MLLT system used for Tamil and Hindi task had different number of tied states and Gaussians for different hours of data which is mentioned in the next table. Dictionary with 39 phones was used for TAMIL and 41 phones for Hindi. The modeling of the phones was similar to that in TIMIT task.

4.3 Experiments and Discussion

Tables 5.5 and 5.7 show the results of experiments evaluating SGMM models on the, Tamil and Hindi tasks respectively. The details of the experiments, along with the motivation and the conclusions are described in the subsequent sections.

4.3.1 Baseline System

At first basic CDHMM system is built and then LDA+MLLT is done on top of it which is used to initialize SGMM . All the other experiments are compared with this baseline system in terms of Word Error Rate (WER).

4.3.2 Increasing the number of tied states

The number of tied states is increased by choosing the tied states by going further down the context-dependency decision tree. And there are serious limitations to increasing the number of tied states, as we may not have enough data to estimate some tied state parameters. There is not much improvement possible on this front, but optimizing the number of tied states gave a better model for initializing the system.

4.4 Tables:

4.4.1 TAMIL:

Baseline system for various tied States for Tamil

Results for tri1 (CDHMM) and tri2 (LDA+MLLT) are tabulated for varying tied states and Gaussians for 1hr, 3hr, 5hr and 22hr of Tamil training data.

1hr data

(# pdfs, # Gaussians)	CDHMM(%WER)	LDA+MLLT(%WER)
174, 802	43.78	39.86
174,1202	43.19	40.67
213,1005	41.51	39.83
213,1504	**42.00**	**37.93**
249,1202	42.10	41.07
249,1808	42.13	42.30
260,1402	42.62	39.29
260,2107	43.95	39.44

260,1606	41.07	38.57
260,2405	43.26	39.17
260,1804	43.16	39.24
260,2710	44.25	41.34
260,2007	42.69	38.97
260,3010	44.64	40.03
260,2207	42.60	39.12
260,3312	43.53	41.54
260,2408	42.77	41.86
260,3612	45.04	42.77
260,2605	43.46	41.51
260,3915	43.83	41.21

Table 5.1 TAMIL 1hr Baseline Results

3hr

(#pdfs, Gaussians)	CDHMM(%WER)	LDA+MLLT(%WER)
179,803	33.46	30.80
179,1207	32.31	28.65
220,1004	32.92	28.73
220,1502	31.17	28.78
267,1204	32.72	28.48
267,1803	**31.61**	**27.34**
306,1408	32.21	28.16
306,2107	32.31	29.47
338,1603	32.21	27.74
338,2409	32.58	27.89
372,1803	32.26	27.71
372,2705	32.11	29.47
405,2010	31.74	29.54
405,3011	32.72	28.48
443,2206	32.08	28.21
443,3311	31.71	28.36
459,2405	32.01	29.91
459,3610	32.75	30.40
459,2608	32.21	29.42
459,3911	33.07	29.89
459,2807	32.63	29.94
459,4215	33.19	30.50
459,3009	32.31	30.38
459,4512	33.49	30.85

Table 5.2 TAMIL 3hr Baseline Results

5hr

(#pdfs, Gaussians)	CDHMM(%WER)	LDA+MLLT(%WER)
410,2008	29.81	25.35
410,3012	28.83	24.16
442,2208	29.94	26.78
442,3310	28.92	23.96
475,2408	29.39	25.69
475,3608	29.15	23.69
515,2609	28.21	24.31
515, 3914	28.11	23.91
554,2808	28.92	25.30
554,4214	27.00	24.04
583,3012	28.60	24.83
583,4518	27.89	23.45
624,3210	27.84	23.52
624,4817	27.49	24.04
650,3408	28.11	24.58
650,5112	27.00	23.35
668,3612	28.01	24.19
668,5414	**25.91**	**23.17**
668,3813	28.26	23.49
668,5716	27.22	23.27
668,4010	27.42	24.16
668,6020	27.54	24.90

Table 5.3 TAMIL 5hr Baseline Results

22hr

(#pdfs, Gaussians)	CDHMM(%WER)	LDA+MLLT(%WER)
887,16145	22.43	19.84
887,18451	22.01	19.79
921,16846	21.45	19.45
921,19248	22.31	19.74
952,17556	21.50	19.87
952,20060	21.32	19.69
993,18246	21.69	19.47
993,20850	21.96	19.60
1030,18948	22.14	20.04
1030,21652	22.16	19.37
1071,22458	21.92	19.45

1114,23262	**22.11**	**19.37**
1139,24052	22.24	20.06
1172,21767	22.93	19.40
1172,24861	22.36	20.09
1203,22465	21.89	19.45
1203,25662	22.43	19.89
1245,23154	22.09	20.16
1245,26481	22.38	19.97

Table 5.4 TAMIL 22hr Baseline Results

4.4.2 SGMM Results for Varying parameters for Tamil database

In order to obtain optimal results we have kept the variable –use-no-substate= false and –use-same-tree = false. We can observe that, by varying number of states beyond a range the obtained HMM states reaches peak value and performance will be unchanged.

If we copy the tree or do not use varying substates the performance degrades. Speaker dimension is kept as 0. We can do SGMM on top of SAT also but in case of Indian Mandi database there is no speaker information. So we have done on top of tri2.The Dimension of Phone vector space is 41 and dimension of feature vectors is 40.

Hours of data	**Parameters**		**%WER**
	UBM	**# HMM states**	
1hr	64	324	34.85
3hr	128	516	26.28
5hr	64	778	23.10
22hr	400	1555	19.10

Table 5.5 TAMIL 1hr, 3hr, 5hr, 22hr SGMM Results

4.4.3 Hindi

Hindi Baseline Results

The baseline system used for Initializing SGMM for Hindi database is tabulated below showing their %WER and the total formed tied states &gaussians. The System is optimized for various tied state value and Gaussians but only the best result is being shown in the tables.

Hours of data	**# pdfs, #Gaussians**	**CDHMM (%WER)**	**LDA+MLLT (%WER)**
1hr	305,1405	16.18	14.12
3hr	454,2206	11.59	10.77
5hr	571,4216	9.06	8.53
22hr	1061,10834	5.76	5.68

Table 5.6 Hindi 1hr, 3hr, 5hr, 22hr Baseline Results

Hindi SGMM results

Here the Dimension of phone vector space is 41, Dimension of speaker vector space is 0 and the feature vectors dimension is 40.To obtain Optimal solution the UBM mixtures and HMM states are varied over a wide range and the best results are only tabulated.

Hours of data	**Parameters**		**%WER**
	UBM	**# HMM states**	
1hr	90	338	13.25
3hr	200	470	10.73
5hr	256	577	7.87
22hr	300	1090	5.09

Table 5.7 Hindi 1hr, 3hr, 5hr, 22hr SGMM Results

4.4.4 TIMIT

Results are shown for Baseline System and SGMM

Name of Expt	UBM Mixtures	Tied States formed	% WER
CDHMM	-	1049	28.38
LDA+MLLT	-	1040	25.45
SGMM	400	2526	**20.60**

Table 5.8 TIMIT Baseline and SGMM Results

Observations

- Results have varied when there is a change in the processor (Cluster).In order to maintain consistency we have performed all experiments on libra IITM cluster. The experiments on this cluster were twice faster and gave optimal Results
- Change in the number of jobs is varying the obtained results. So we maintained a constant number(nj) all over
- Various experiments were performed to optimize SGMM for Hindi and Tamil during which it has been observed that the result is optimal if we do not copy the same tree as in baseline system and gave it a larger degree of freedom. We should even keep varying the total number of substates. So make sure the variable
- Varying of tied states was done to optimize the solution during which it was observed that beyond a certain range of parameter variation the solution remained unchanged and reaching a peak HMM state value
- For huge amount of training data more UBM Gaussians and more Tied states are required.

Chapter 5

CONCLUSIONS

We have Investigated a new type of statistical model, the Subspace Gaussian Mixture Model (SGMM) on Indian languages Hindi and Tamil of Mandi Database, and demonstrated that it can give substantially better results than a conventionally structured model, particularly without adaptation. We have shown the importance of various features of the model, such as modeling the weights; using full-covariance Gaussians; and using sub-states.

Particularly for low resource of training data the percentage improvement for Hindi and Tamil was high. The total number of parameters obtained in SGMM model are also less compared to conventional CDHMM system. But the time taken for experiment is more compared to conventional methods.

Appendix A

Things To Be Noted While Performing The Experiments:

- All the experiments were performed under IITM Libra Cluster because change in processor is giving different results.
- Experiments should be done by using 20 cores as change in splits gives different performance.
- Also, all the features should be sorted out because unsorted features are giving different results.
- All the scripts used should be latest standard KALDI scripts because they have slight modifications compared to old scripts and hence performance might be different.
- Optimization should be done for both LDA+MLLT and CDHMM but not just alone CDHMM because the input to SGMM is LDA+MLLT in our current experiments.
- The value of the TIED States formed in SGMM will be reaching a saturation limit at certain point beyond which they will not be any increase in TIED states and performance.
- Also, one should be careful at giving the number of TIED states to CDHMM and LDA+MLLT as giving too many to lesser amount of data will hinder performance.
- The best result in SGMM might be obtained for any value of UBM mixtures and tied states which is unknown, so can be determined only by experiments.

BIBLIOGRAPHY

1. **Burget, L., P. Schwarz, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, D. Povey**, et al., Multilingual acoustic modeling for speech recognition based on subspace gaussian mixture models. In Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. IEEE, 2010.
2. **Gales, M. J.** (1998). Maximum likelihood linear transformations for hmm-based speech recognition. Computer speech and language, 12(2).
3. **Gales, M. J.** (2000). Cluster adaptive training of hidden markov models. Speech and Audio Processing, IEEE Transactions on, 8(4), 417–428.
4. **Leggetter, C. and P. Woodland** (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. Computer speech and language, 9(2), 171.
5. **Mohan, A., S. Umesh, and R. Rose**, Subspace based for indian languages. In Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on. IEEE, 2012.
6. **Povey, D.** (2009). A tutorial-style introduction to subspace gaussian mixture models for speech recognition. Technical Report MSR-TR-2009-111, Microsoft Research.
7. **Povey, D., L. Burget, M. Agarwal, P. Akyazi, K. Feng, A. Ghoshal, O. Glembek, N. K. Goel, M. Karafiát, A. Rastrow**, et al., Subspace gaussian mixture models for speech recognition. In Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on. IEEE, 2010.

8. **Povey, D., L. Burget, M. Agarwal, P. Akyazi, F. Kai, A. Ghoshal, O. Glembek, N. Goel, M. Karafiát, A. Rastrow, R. C. Rose, P. Schwarz, and S. Thomas** (2011a). The subspace gaussian mixture model - a structured model for speech recognition. Computer Speech & Language, 25(2), 404 – 439. ISSN 0885-2308.
9. **Povey, D., A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz**, et al., The kaldi speech recognition toolkit. In IEEE 2011 Workshop on Automatic Speech Recognition and Understanding. 2011b.
10. **Rabiner, L. R.** (1989). A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257–286.
11. **Srinivas, B., N. M. Joy, R. R. Bilgi, and S. Umesh**, Subspace modeling technique using monophones for speech recognition. In Communications (NCC), 2013 National Conference on. 2013.
12. **Young, S. J., J. Odell, and P. Woodland**, Tree-based state tying for high accuracy acoustic modelling. In Proceedings of the workshop on Human Language Technology. Association for Computational Linguistics, 1994.