B.E. PROJECT

ON

SPEAKER RECOGNITION USING

STUDENT-T MIXTURE MODEL

Submitted by

Nilaksh Das (480/IC/10)

Shikhar Kwatra (520/IC/10)

Siddharth Sharma(527/IC/10)

In partial fulfillment of B.E. (Instrumentation and Control Engineering) degree of University of Delhi

Under the Guidance of

Dr. Smriti Srivastava

DIVISION OF INSTRUMENTATION AND CONTROL ENGINEERING

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF DELHI, DELHI

JUNE 2014

DECLARATION

This is to certify that the project entitled, "**SPEAKER IDENTIFICATION USING STUDENT-T MIXTURE MODEL**" by **Nilaksh Das**, **Shikhar Kwatra** and **Siddharth Sharma** is a record of bonafide work carried out by us, in the department of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, University of Delhi, New Delhi, in partial fulfillment of requirements for the award of the degree of Bachelor of Engineering in Instrumentation and Control Engineering, University of Delhi in the academic year **2010-2014**.

The results presented in this thesis have not been submitted to any other university in any form for the award of any other degree.

Nilaksh Das	Shikhar Kwatra	Siddharth Sharma
Roll No. 480/IC/10	Roll No. 520/IC/20	Roll No. 527/IC/10

Instrumentation and Control Engineering Department

Netaji Subhas Institute of Technology (NSIT)

Azad Hind Fauj Marg

Sector-3, Dwarka, New Delhi

PIN - 110078

नेताजी सुभाष प्रौद्योगिकी संस्थान

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

(An Institution of Govt. of NCT of Delhi-Formerly, Delhi Institute of Technology)

Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi - 110 078
Telphone : 25099050; Fax : 25099025, 25099022 Website : http://www.nsit.ac.in

CERTIFICATE

This is to certify that the project entitled, "**SPEAKER IDENTIFICATION USING STUDENT-T MIXTURE MODEL**" by **Nilaksh Das**, **Shikhar Kwatra** and **Siddharth Sharma** is a record of bonafide work carried out by them, in the department of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, University of Delhi, New Delhi, under our supervision and guidance in partial fulfillment of requirements for the award of the degree of Bachelor of Engineering in Instrumentation and Control Engineering, University of Delhi in the academic year **2010-2014**.

The results presented in this thesis have not been submitted to any other university in any form for the award of any other degree.

Prof. Smriti Srivastava
Professor
Instrumentation and Control Engineering Department
Netaji Subhas Institute of Technology (NSIT)
Azad Hind Fauj Marg
Sector-3, Dwarka, New Delhi
PIN - 110078

नेताजी सुभाष प्रौद्योगिकी संस्थान

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

(An Institution of Govt. of NCT of Delhi-Formerly, Delhi Institute of Technology)

Azad Hind Fauj Marg, Sector-3, Dwarka, New Delhi - 110 078
Telphone : 25099050; Fax : 25099025, 25099022 Website : http://www.nsit.ac.in

CERTIFICATE

This is to certify that the report entitled "**SPEAKER IDENTIFICATION USING STUDENT-T MIXTURE MODEL**" by **Nilaksh Das**, **Shikhar Kwatra** and **Siddharth Sharma** is a record of bonafide work carried out by them, in the division of Instrumentation and Control Engineering, Netaji Subhas Institute of Technology, University of Delhi, New Delhi, in partial fulfillment of requirements for the award of the degree of Bachelor of Technology in Instrumentation and Control Engineering, University of Delhi in the academic year **2010-2014**.

Prof. A.P. Mittal
Head of the Department
Department of Instrumentation and Control Engineering
Netaji Subhas Institute of Technology (NSIT)
Azad Hind Fauj Marg
Sector-3, Dwarka, New Delhi
PIN – 110078

ACKNOWLEDGEMENT

We take this unique opportunity to express our heartfelt thanks and gratitude to our respected guide, **Dr. Smriti Srivastava, Professor, Department of Instrumentation and Control Engineering, NSIT** who kindly consented to be our guide for this project along with **Dr. Saurabh Bhardwaj.**

We thank them for their precious time devoted to us, for their expert guidance, kind attitude and the resources arranged for us. It is only because of them that we have been able to successfully complete this project.

We also owe our thanks to all the faculty members for their constant support and encouragement.

Nilaksh Das	Shikhar Kwatra	Siddharth Sharma
Roll No. 480/IC/10	Roll No. 520/IC/20	Roll No. 527/IC/10

Instrumentation and Control Engineering Department

Netaji Subhas Institute of Technology (NSIT)

Azad Hind Fauj Marg

Sector-3, Dwarka, New Delhi

PIN – 110078

ABSTRACT

Speaker recognition is the computational task of validating a person’s identity based on their voice. Speaker recognition can be classified into text dependent and the text independent methods. The features of speech signal that are being used (or have been used) for speaker recognition are presented in this report. The two phases of a speaker recognition system are the enrollment phase where speech samples from the different speakers are turned into models and the verification phase where a sample of speech is tested to determine if it matches a proposed speaker. In a text-independent system, there are no constraints in the words or phrases used during verification. Numerous approaches have been studied to make text-independent speaker recognition systems accurate with very short speech samples and robust against both channel variability (differences due to the medium used to record the speech) and speaker dependent variability (such as health or mood of the speaker).

The commonly used techniques of pattern matching for the decision making process in speaker recognition have been discussed in this report. A text-independent speaker recognition system using Student-T mixture models and factor analysis techniques have been implemented in Matlab and will be tested against the NIST SRE databases for validation.

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	***Definition***
μ	Mean(Expectation value) of data
A	Transition Matrix
B	Emission Matrix
$\pi(i)$	Initialization Probability Matrix
$\lambda(A,B,\pi)$	Model of a system
$N(x\|\mu,\sigma^2)$	Normal Distribution
x	Single Variable
σ^2	Variance
Σ	Covariance Matrix
Δ	Mahalanobis Distance
γ_k	Responsibility of Kth component of x
ν	Degree of freedom

LIST OF EQUATIONS

TABLE OF CONTENTS

CHAPTER ONE: **INTRODUCTION TO HUMAN SPEAKER RECOGNITION**

1.1 Project Background

Humans have the innate ability to recognize familiar voices withi Automatic speaker recognition is the use of a machine to recognize a person from a spoken phrase. These systems can operate in two modes: to identify a particular person or to verify a person's claimed identity. How do we teach a machine to do the same? Research in speaker recognition/verification, the computational task of validating a person’s identity based on their voice, began in 1960 with a model based on the analysis of x-rays of individuals making specific phonemic sounds. With the advancements in technology over the past 50 years, robust and highly accurate systems have been developed with applications in automatic password reset capabilities, forensics and home healthcare verification.

There are two phases in a speaker recognition system: an enrolment phase where speech samples from different speakers are turned into models and the verification phase where a sample of speech is tested to determine if it matches a proposed speaker. It is assumed that each speech sample pertains to one speaker.

A robust system would need to account for differences in the speech signals between the enrolment phase and the verification phase that are due to the channels used to record the speech (landline, mobile phone, handset recorder) and inconsistencies within a speaker (health, mood, effects of aging) which are referred to as channel variability and speaker dependent variability respectively. In text-dependent systems, the words or phrases used for verification are known beforehand and are fixed.

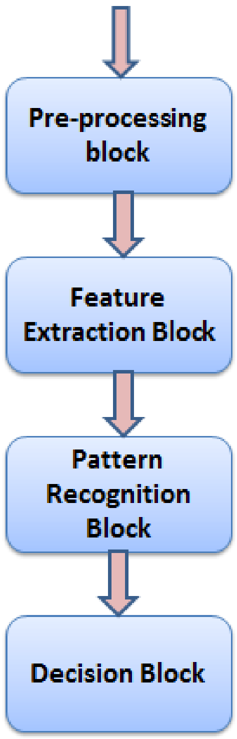

Figure 1.1 Generalized System Description

In a text-independent system, there are no constraints on the words or phrases used during verification. This project will focus on text-independent speaker verification systems.

Speaker recognition systems contain two main modules:

- *feature extraction*
- *feature matching*

Feature extraction:

Extract a small amount of data from the voice signal that can be used to represent each speaker

Feature matching:

Procedure to identify the unknown speaker by comparing extracted features from his/her voice input with the ones from a set of known speakers.

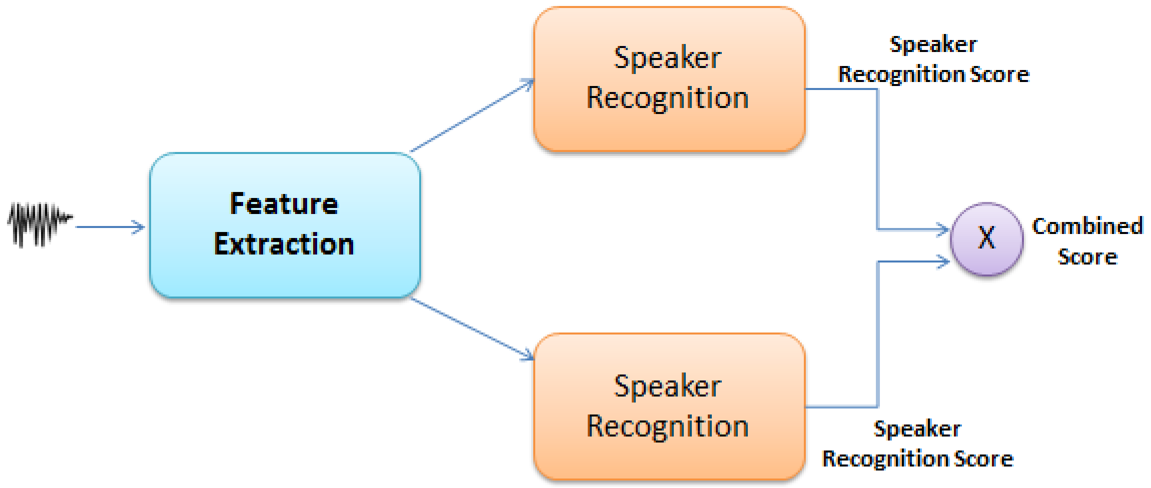

Figure 1.2 Speaker recognition System

A variety of different features can be extracted from the speech samples, or utterances. Low level features relate to physiological aspects of a speaker such as the size of the vocal folds or length of vocal tract, prosodic and spectro-temporal features correspond to pitch, energy or rhythm of the speech, and high level features are behavioral characteristics such as accents or pronunciation. When extracting features, voice activity detectors (VADs) can be used to remove segments in an utterance where there is no speech. VADs can be energy based or based on periodicity. Many advanced systems account for multiple features and fusion is used to find the best overall match.

For text-independent speaker verification the most popular modeling approaches are vector quantization (VQ), Gaussian mixture models (GMMs) and support vector machines (SVM). VQ is a technique that divides the features into clusters using a method such as K-means. GMM is an expansion of the VQ model, allowing each feature to have a nonzero probability of originating for each cluster. A universal background model (UBM) representing an average speaker is often used in a GMM-based model. Adaptations of the UBM are used to characterize each of the individual speakers making the models robust even when the full phonemic space is not covered by the training data. SVM take labeled training data and seeks to find an optimized decision boundary between two classes which can be used to discriminate the different speakers.

Various techniques have been researched to assist in compensating for channel variability and speaker dependent variability, including speaker model synthesis (SMS) and feature mapping (FM). Most approaches require the speaker models to be organized into a high- and fixed-dimensional single vector called a supervector so that utterances with varying numbers of features can be represented in a general and compatible form. Popular methods that focus on compensating SVM supervectors include generalized-linear discriminant sequence (GLDS) kernel and maximum likelihood linear regression (MLLR) . Factor analysis (FA) is a common generative modeling technique that is used on supervectors from GMMs to account for variability by learning low-dimensional subspaces. FA methods used in speaker verification include joint factor analysis (JFA) which model channel variability and speaker dependent variability separately, and total variability which model channel variability and speaker dependent variability in the same space. Normalization methods such as nuisance attribute projection (NAP), within-class covariance normalization (WCCN) and linear discriminant analysis (LDA) are also used for intersession variability compensation.

CHAPTER TWO: **EXTRACTION OF COMPUTATIONAL FEATURES FROM SPEECH USING MFCC**

2.1 Feature Extraction

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

In this project, a simple text-independent speaker verification system will be implemented using mel- frequency cepstral coefficients (MFCCs) as the features used to create UBM-adapted Student-T Mixture models. The mean components in the models will be concatenated into supervectors. LDA methods will be applied to the i-vectors corresponding to the total variability space to maximize inter-speaker variability and minimize speaker-dependent variability. Discrete cosine scoring (DCS) will be used for verifying if a test utterance matches a proposed speaker.

2.1.1 Feature Extraction using MFCC

Low-level features called mel-frequency cepstral coefficients (MFCCs) are extracted from the speech samples and used for creating the speaker models.

2.2 Mel Frequency Ceptrum Coefficients

MFCCs help us in identifying the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff that carries information like background noise, emotion, etc.

The sounds generated by a human being are filtered by the shape of the vocal tract including the tongue, teeth, etc. This shape determines what sound comes out. This shape can give us a accurate representation of the phoneme produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

MFCCs are widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. Before MFCCs, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) and were the main feature type for automatic speech recognition (ASR).

2.2.1 Steps at a Glance

The following steps are followed to calculate the MFCCs:

1. Frame the signal into short frames.
2. For each frame calculate the periodogram estimate of the power spectrum.
3. Apply the mel filterbank to the power spectra, sum the energy in each filter.
4. Take the logarithm of all filterbank energies.
5. Take the DCT of the log filterbank energies.
6. Keep the DCT coefficients, 2-13, discard the rest.

There are a few more things commonly done, sometimes the frame energy is appended to each feature vector. Delta and Delta-Delta features are usually also appended. Liftering is also commonly applied to the final features.

2.2.2 Theory behind the Steps

An audio signal is constantly changing, so to simplify things we assume that on short time scales the audio signal doesn't change much (when we say it doesn't change, we mean statistically i.e. statistically stationary, obviously the samples are constantly changing on even short time scales). This is why we frame the signal into 20-40ms frames. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

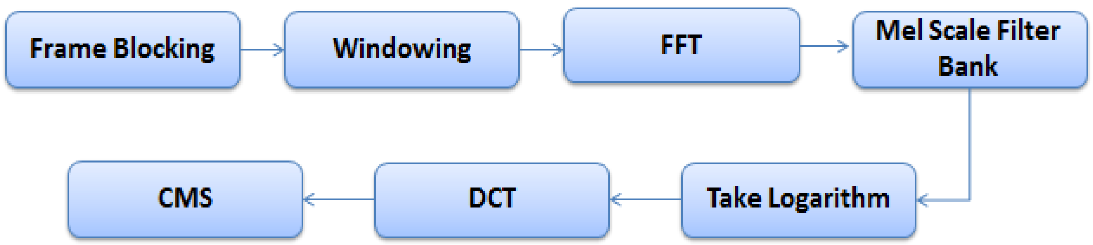

Figure 2.1 Steps involved in MFCC

The next step is to calculate the power spectrum of each frame. This is motivated by the human cochlea (an organ in the ear) which vibrates at different spots depending on the frequency of the incoming sounds. Depending on the location in the cochlea that vibrates (which wobbles small hairs), different nerves fire informing the brain that certain frequencies are present. Our periodogram estimate performs a similar job for us, identifying which frequencies are present in the frame.

The periodogram spectral estimate still contains a lot of information not required for Automatic Speaker Recognition. In particular the cochlea can not discern the difference between two closely spaced frequencies. This effect becomes more pronounced as the frequencies increase. For this reason we take clumps of periodogram bins and sum them up to get an idea of how much energy exists in various frequency regions. This is performed by our Mel filterbank: the first filter is very narrow and gives an indication of how much energy exists near 0 Hertz. As the frequencies get higher our filters get wider as we become less concerned about variations. We are only interested in roughly how much energy occurs at each spot.

The Mel scale tells us exactly how to space our filterbanks and how wide to make them.

Once we have the filterbank energies, we take the logarithm of them. This is also motivated by human hearing: we don't hear loudness on a linear scale. Generally to double the percieved volume of a sound we need to put 8 times as much energy into it. This means that large variations in energy may not sound all that different if the sound is loud to begin with. This compression operation makes our features match more closely what humans actually hear. Why the logarithm and not a cube root? The logarithm allows us to use cepstral mean subtraction, which is a channel normalisation technique.

The final step is to compute the DCT of the log filterbank energies. There are 2 main reasons this is performed. Because our filterbanks are all overlapping, the filterbank energies are quite correlated with each other. The DCT decorrelates the energies which means diagonal covariance matrices can be used to model the features in e.g. a HMM classifier. But notice that only 12 of the 26 DCT coefficients are kept. This is because the higher DCT coefficients represent fast changes in the filterbank energies and it turns out that these fast changes actually degrade ASR performance, so we get a small improvement by dropping them.

MFCC values are not very robust in the presence of additive noise, and so it is common to normalise their values in speech recognition systems to lessen the influence of noise. Some researchers propose modifications to the basic MFCC algorithm to improve robustness, such as by raising the log-mel-amplitudes to a suitable power (around 2 or 3) before taking the DCT, which reduces the influence of low-energy components.

2.2.3 The Mel Scale

The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear.

The formula for converting from frequency to Mel scale is:

$$M(f) = 1125\ \ln(1 + {}^{f}/_{700}) \qquad \text{(Eq. 1.1)}$$

To go from Mel back to frequency scale:

$$M^{-1}(m) = 700(\exp\left({}^{m}/_{1125}\right) - 1) \qquad \text{(Eq. 1.2)}$$

2.2.4 Frame blocking

- In this step the continuous speech signal is blocked into frames of N samples, with adjacent frames being separated by M (M < N)
- The first frame consists of the first N samples
- The second frame begins M samples after the first frame, and overlaps it by N - M samples

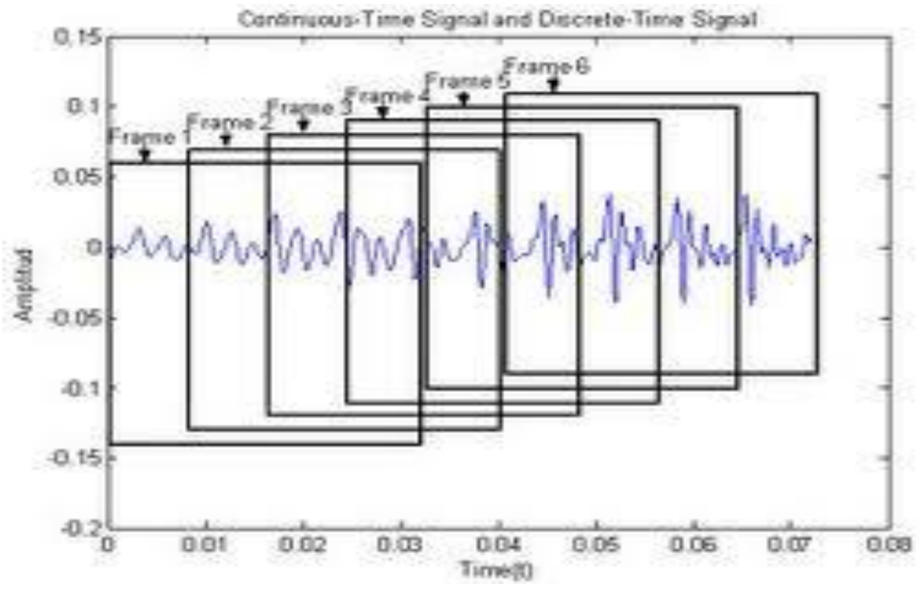

Figure 2.2 Continuous and Discrete Time Signal

- Process continues until all the speech is accounted for within one or more frames
- Taken values for N and M are N = 256 and M = 100
- The result after this step is referred to as spectrum or periodogram.

2.2.5 Mel-frequency Wrapping

For each tone with an actual frequency, f, a subjective pitch is measured on a scale called the 'mel' scale. Mel-frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on the mel-scale. That filter bank has a triangular bandpass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval

The number of mel spectrum coefficients, K, is typically chosen as 20. Filter bank is applied in the frequency domain.

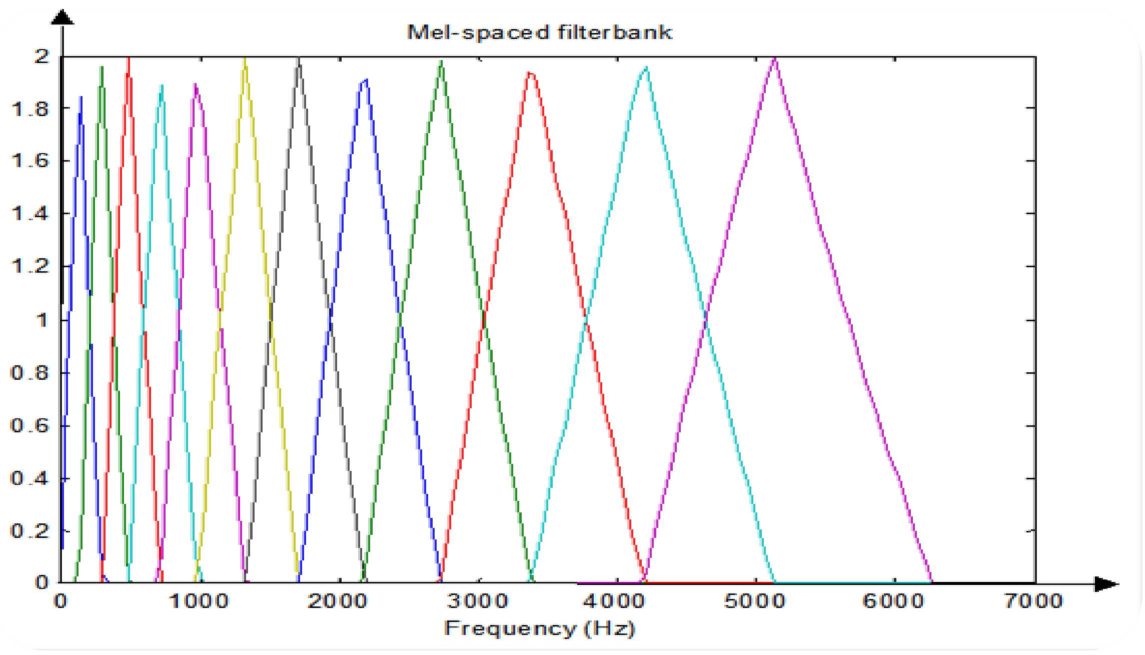

Figure 2.3 Mel-Spaced Filter bank

As mentioned above, psychophysical studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale. A useful way of thinking about this mel-wrapping filter bank is to view each filter as a histogram bin (where bins have overlap) in the frequency domain.

2.3 Applications

Hence, MFCCs are commonly used as features in speech recognition systems, such as the systems which can automatically recognize numbers spoken into a telephone. They are also common in speaker recognition, which is the task of recognizing people from their voices.

MFCCs are also increasingly finding uses in music information retrieval applications such as genre classification, audio similarity measures, etc.

CHAPTER THREE: **CURRENT MATHEMATICAL MODELING TECHNIQUES OF HUMAN SPEAKERS**

3.1 Various Modeling Techniques

The problem of speaker recognition belongs to a much broader topic in scientific and engineering so called pattern classification. The goal of pattern classification is to classify objects of interest into a number of categories or classes. The objects of interest are generically called patterns and in our case are sequences of acoustic vectors that are extracted from an input speech. The classes here refer to individual speakers . Pattern classification plays as a crucial part in speaker modeling component chain. The result of pattern classification will strongly affect the speaker recognition engine to decide whether to accept or reject a speaker.

Many research efforts have been done in speaker recognition pattern classification. There are Dynamic Time Warping (DTW) , Vector Quantization (VQ) , Hidden Markov Models (HMM) , Gaussian mixture model (GMM) and so forth. Most of this works are based on generative model. A generative model is a model for randomly generating observed data, typically given some hidden parameters. Because of the randomly generating observed data functions, they are not able to provide a machine that can directly optimize discrimination.

3.2 Hidden Markov Model (HMM)

An HMM is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. In other words, a hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. A HMM can be considered the simplest dynamic Bayesian network.

The formal definition for HMM is as follows:

$$\lambda = (A, B, \pi)$$

S is our state alphabet set, and V is the observation alphabet set:

$$S = (s_1, s_2, \ldots\ldots, s_N)$$

$$V = (v_1, v_2, \ldots\ldots, v_M)$$

We define Q to be a fixed state sequence of length T, and corresponding observations O:

$$Q = q_1, q_2, \ldots\ldots, q_T$$

$$O = o_1, o_2, \ldots\ldots, o_T$$

A is a transition array, storing the probability of state j following state i . Note the state transition probabilities are independent of time:

$$A = [a_{ij}] , a_{ij} = P(q_t = s_j | q_{t-1} = s_i).$$

B is the observation array, storing the probability of observation k being produced from the state j, independent of t:

$$B = [b_i(k)] , b_i(k) = P(x_t = v_k|q_t = s_i).$$

π is the initial probability array:

$$\pi = [\pi_i] , \pi_i = P(q_1 = s_i)$$

Two assumptions are made by the model. The first, called the Markov assumption, states that the current state is dependent only on the previous state, this represents the memory of the model:

$$P(q_t|q_1^{t-1}) = P(q_t|q^{t-1})$$

The independence assumption states that the output observation at time t is dependent only on the current state, it is independent of previous observations and states:

$$P(o_t|o_1^{t-1}, q_1^t) = P(o_t|q_t)$$

3.2.1 Learning of HMM

Given a set of examples from a process, we would like to be able to estimate the model parameters λ = (A, B, π) that best describe that process. There are two standard approaches to this task, dependent on the form of the examples, which will be referred to here as supervised and unsupervised training. If the training examples contain both the inputs and outputs of a process, we can perform supervised training by equating inputs to observations, and outputs to states, but if only the inputs are provided in the training data then we must used unsupervised training to guess a model that may have produced those observations. The easiest solution for creating a model λ is to have a large corpus of training examples, each annotated with the correct classification. The classic example for this approach is PoS tagging. We define two sets:

- $t_1 \ldots\ldots t_N$ is the set of tags, which we equate to the HMM state set $s_1 \ldots\ldots s_N$
- $w_1 \ldots\ldots w_M$ is the set of words, which we equate to the HMM observation set $v_1 \ldots\ldots v_M$

so with this model we frame part-of-speech tagging as decoding the most probable hidden state sequence of PoS tags given an observation sequence of words. To determine the model parameters λ, we can use maximum likelihood estimates(MLE) from a corpus containing sentences tagged with their correct PoS tags. For the transition matrix we use:

$$a_{ij} = P(t_i|t_j) = \frac{Count(t_i, t_j)}{Count(t_i)}$$

where Count(t_i, t_j) is the number of times t_j followed t_i in the training data. For the observation matrix:

$$b_j = P(w_k|t_j) = \frac{Count(w_k, t_j)}{Count(t_j)}$$

where Count(w_k, t_j) is the number of times w_k was tagged t_j in the training data. And lastly the initial probability distribution:

$$\pi_i = P(q_1 = t_i) = \frac{Count(q_1 = t_i)}{Count(t_j)}$$

In practice when estimating a HMM from counts it is normally necessary to apply smoothing in order to avoid zero counts and improve the performance of the model on data not appearing in the training set.

3.3 Gaussian Mixture Model

The Gaussian, also known as the normal distribution, is a widely used model for the distribution of continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \qquad (Eq. 1.3)$$

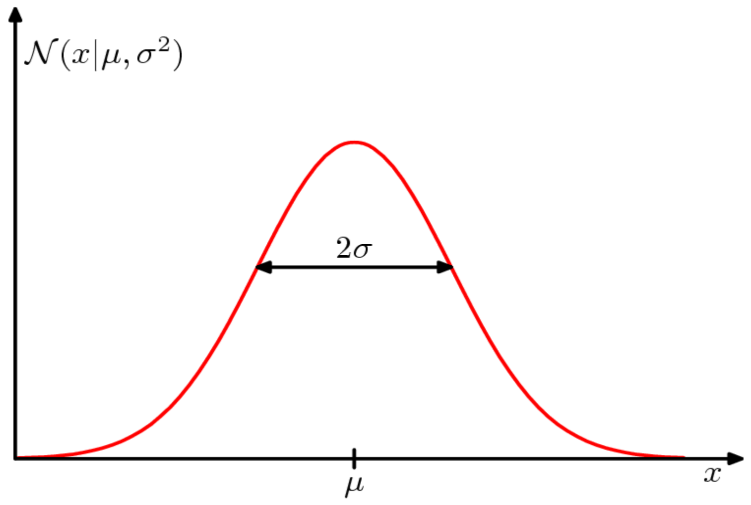

Figure 3.1 Univariate Gaussian Distribution

where μ is the mean and σ^2 is the variance. For a D-dimensional vector x, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \qquad \text{(Eq. 1.4)}$$

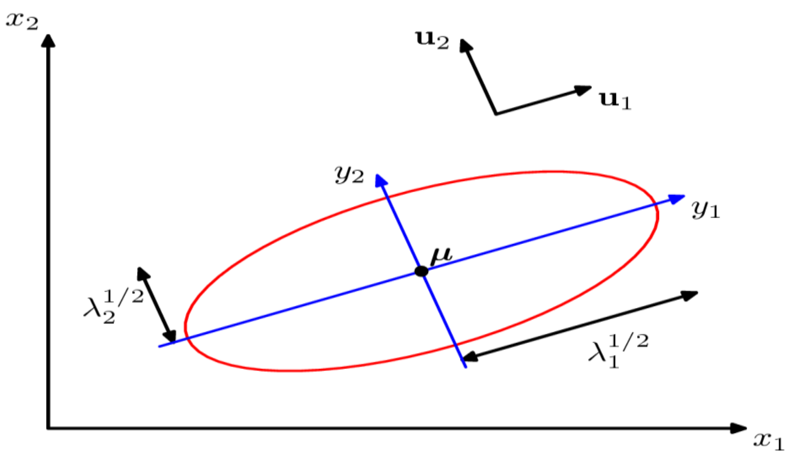

Figure 3.2 Multivariate Gaussian Distribution

where μ is a D-dimensional mean vector, Σ is a D × D covariance matrix, and |Σ| denotes the determinant of Σ. The Gaussian distribution arises in many different contexts and can be motivated from a variety of different perspectives. For example, we have already seen that for a single real variable, the distribution that maximizes the entropy is the Gaussian. .This property applies also to the multivariate Gaussian.

Let's consider the geometrical form of Gaussian Distribution. The functional dependence of the Gaussian on x is through the quadratic form:

$$\Delta^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

which appears in the exponent. The quantity Δ is called the *Mahalanobis distance* from μ to x and reduces to the Euclidean distance when Σ is the identity matrix. The Gaussian distribution will be constant on surfaces in x-space for which this quadratic form is constant.

In short, we assume that the dataset *X* has been generated by a *parametric* distribution *p(X)*.Estimation of the parameters of *p* is known as *density estimation*.

The typical parameters of a GMM are as follows:

- ***Mean*** (μ): average value of $p(X)$, also called expectation.
- ***Variance*** (σ): provides a measure of variability in $p(X)$ around the mean.
- ***Covariance***: measures how much two variables vary together.
- ***Covariance matrix***: collection of covariances between all dimensions.

Diagonal of the covariance matrix contains the variances of each attribute

We obtain following Gaussians for different values of covariance matrix:

- ***Single Covariance Matrix***

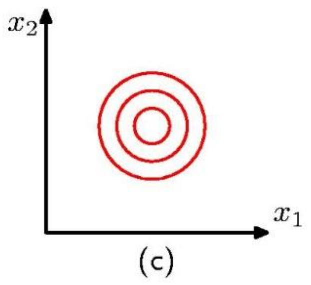

Figure3.3 Single Covariance Matrix

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

- ***Diagonal Covariance Matrix***

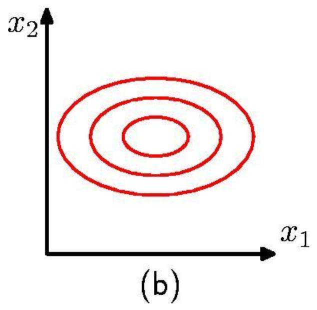

Figure 3.4 Diagonal Covariance Matrix

$$\Sigma = \begin{pmatrix} {\sigma_1}^2 & 0 \\ 0 & {\sigma_2}^2 \end{pmatrix}$$

- ***Full Covariance Matrix***

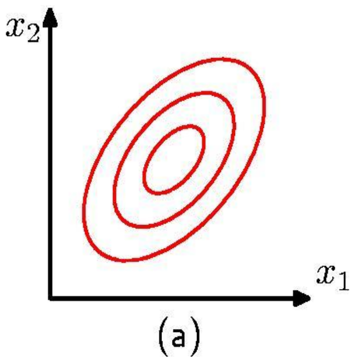

Figure 3.5 Full Covariance Matrix

$$\Sigma = \begin{pmatrix} {\sigma_{11}}^2 & {\sigma_{12}}^2 \\ {\sigma_{21}}^2 & {\sigma_{22}}^2 \end{pmatrix}$$

The complete log likelihood of GMM is given by:

$$\ln p(X) = \ln \prod_{n=1}^{N} Normal(X_n|\mu,\Sigma)$$

Quite often, the real world data is not suitably represented by one Gaussian, since the dataset is not unimodal. In this case we require mixture of Gaussians to represent our dataset accurately. The following figure shows this case:

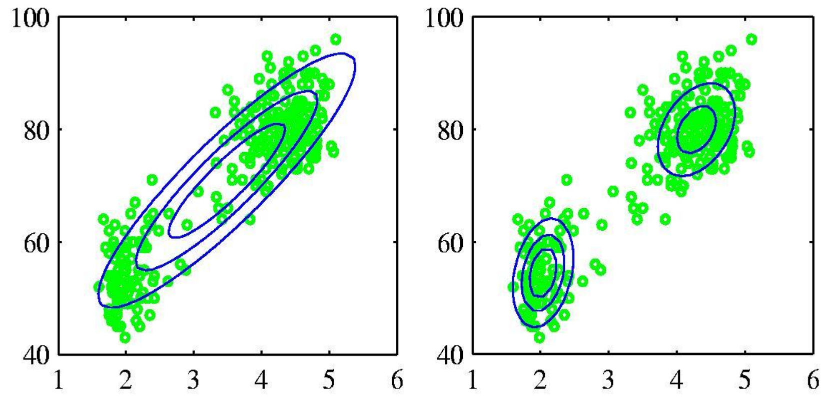

Figure 3.6 Univariate and Multivariate Gaussian Clustering

Mixture of Gaussians is represented by the following equation:

$$p(X) = \sum_{k=1}^{M} \pi_k Normal(x|\mu_k, \Sigma_k) \qquad \text{(Eq. 1.5)}$$

The following figure shows a mixture of Gaussians:

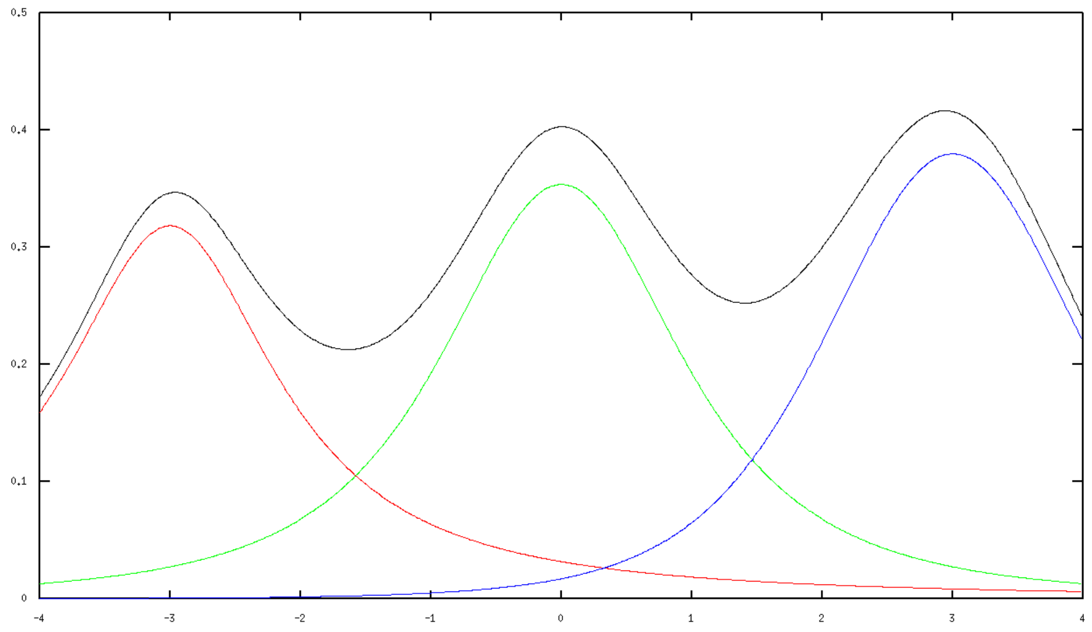

Figure 3.7 Mixture of Gaussians

3.3.1 Expectation Maximization for GMM

The goal of the EM algorithm is to maximize the log likelihood of the whole data.

$$\ln p(X|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln\left\{\sum_{k=1}^{M} \pi_k Normal(X_n|\mu_k,\Sigma_k)\right\} \qquad \text{(Eq. 1.6)}$$

The EM algorithm has two steps, E-step or Expectation step and M-step or Maximization step.

In E-step we calculate how probably observation vector x is from component k. In clustering, responsibilities take values 0 and 1, and thus, it defines the hard partitioning. We can express the marginal density p(x) as:

$$p(x) = \sum_{k=1}^{M} p(k)p(X|k) \qquad \text{(Eq. 1.7)}$$

From this, we can find the responsibility of the kth component of ***x*** using Bayesian theorem:

$$\gamma_k(x) = p(k|x)$$

$$= \frac{p(x)p(x|k)}{\sum_l \pi_l Normal(x|\mu_l,\Sigma_l)}$$

$$= \frac{\pi_k Normal(x|\mu_k,\Sigma_k)}{\sum_l \pi_l Normal(x|\mu_l,\Sigma_l)} \qquad \text{(Eq. 1.8)}$$

During the M-stept, the exact update equations used to calculate the values of mean, covariance and mixing coefficients are as follows:

New mean estimates:

$$\mu_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma_n(X_n)X_n \qquad N_k = \sum_{n=1}^{N}\gamma_k(X_n)$$

Covariance estimates:

$$\Sigma_k = \frac{1}{N_k}\sum_{n=1}^{N}\gamma_k(X_n)(X-\mu)(X-\mu)^T$$

Mixing coefficient estimates:

$$\pi_k = \frac{N_k}{K}$$

After calculating the new updated parameters, we calculate the log likelihood of the new parameters.

3.4 Vector Quantization

Vector Quantization is a process of mapping vectors from a large vector space to a finite number of spaces in that space. Each space is represented by its centroid point, as in k-means and some other clustering algorithms. It is used because of the following two reasons:

- Ease of implementation: The main advantage of VQ in pattern recognition is its low computational burden when compared with other techniques.
- Using VQ provides high accuracy in pattern recognition.

The density matching property of vector quantization is powerful, especially for identifying the density of large and high-dimensioned data. Since data points are represented by the index of their closest centroid, commonly occurring data have low error, and rare data high error. This is why VQ is suitable for lossy data compression. It can also be used for lossy data correction and density estimation.

In VQ, each region is called a cluster and can be represented by its center called a codeword. The collection of all codewords is called a codebook.

3.4.1 VQ Codebook Formation

We will explain VQ codebook formation through the example shown in the following figure:

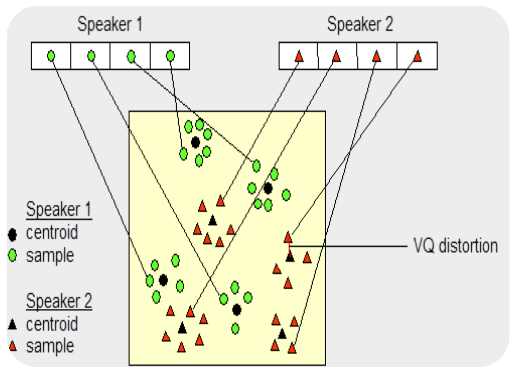

Figure 3.8 VQ clustering Mechanism

The figure shows two speakers and two dimensions of the acoustic space. The circles represent the acoustic vectors from the speaker 1 and the triangles represent acoustic vectors from speaker 2.

During the training phase, clustering algorithm is used to get a speaker-specific VQ codebook for each and every speaker. Thus we get result codewords (centroids) – black circles and black triangles for speaker 1 and speaker 2. Distance from a vector to the closest codeword of a codebook is called VQ-distortion. Input utterance of an unknown voice is "vector-quantized" using each trained codebook and the total VQ distortion is computed. The speaker corresponding to the VQ codebook with smallest total distortion is identified as the speaker of the input utterance. This is how Vector Quantization is used in speaker recognition.

The Algorithmic State Machine representing the above process is shown in the following figure:

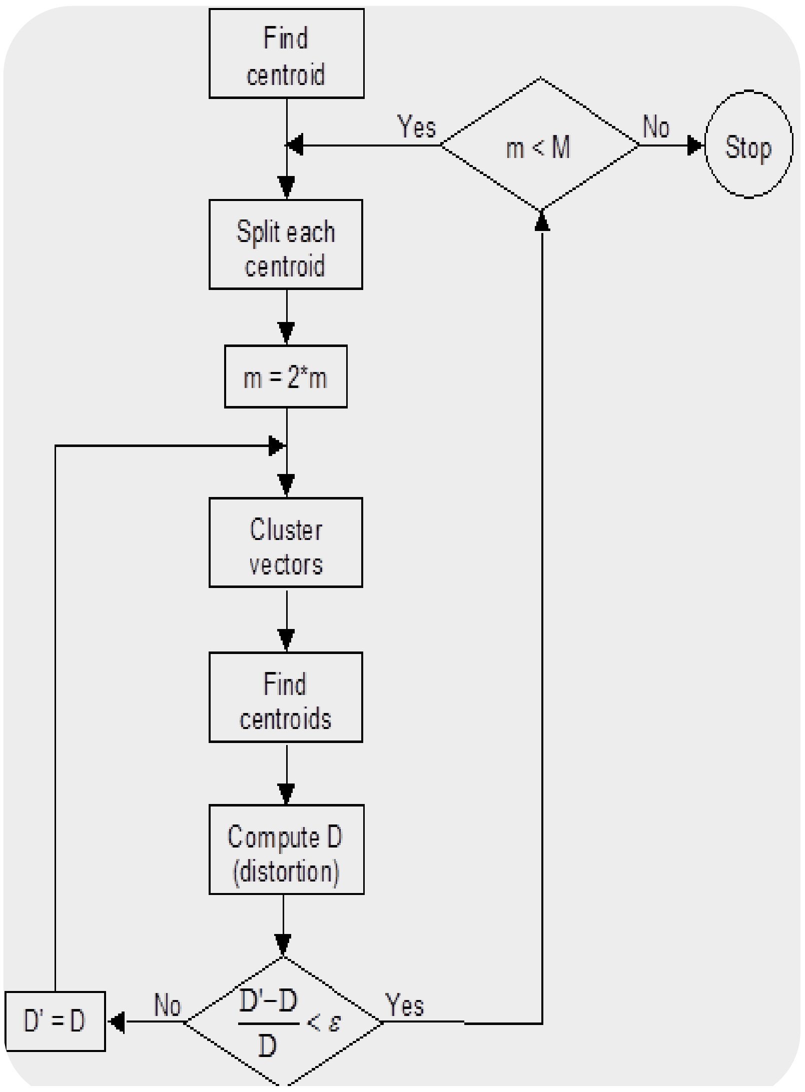

Figure 3.9 VQ ASM chart

Thus we discussed the other popular pattern recognition techniques that are commonly used for speaker recognition. In the next chapter we will discuss in detail about Student’s T Mixture Model technique that we have used for speaker recognition in this project.

CHAPTER FOUR: **AN INTRODUCTION TO STUDENT'S T MIXTURE MODELS**

4.1 Student's- T Mixture Model

Different forms of modelling techniques include Hidden Markov Model (HMM), Gaussian Mixture Model (GMM), Vector quantization etc. which have been discussed in detail in the previous chapter.

Student-T mixture was a new approach derived later as a generalised Gaussian Model with variable degrees of freedom and was proposed to be more robust with outliers as compared to GMM.

Student-T Mixture model represents each speaker by a finite mixture of multivariate Gaussians based on the d-dimensional feature vector.

Various parameters of Student-T mixture models include:

- Mean (μ): average value of p(X), also called expectation.
- Covariance (Σ): measures how much two variables vary together.
- Degree of freedom (ν): controls the length of the tails.
- Covariance matrix: collection of covariances between all dimensions.

- Diagonal of the covariance matrix contains the variances of each attribute

If we have a univariate Gaussian N (x|μ, τ−1) together with a Gamma prior Gam (τ|a, b) and we integrate out the precision, we obtain the marginal distribution of x in the form:

$$\Pr(x) = Stud_x[\mu, \Sigma, \nu]$$

$$= \frac{\Gamma\left[\frac{\nu+D}{2}\right]}{\nu\pi^{D/2}|\Sigma|^{1/2}\Gamma\left[\frac{\nu}{2}\right]}\left(1 + \frac{(x-\mu)^T\Sigma^{-1}(x-\mu)}{\nu}\right)^{-\frac{\nu+D}{2}} \qquad \text{(Eq. 1.9)}$$

which is known as Student’s t-distribution. The parameter λ is sometimes called the precision of the t-distribution, even though it is not in general equal to the inverse of the variance. The parameter ν is called the degrees of freedom, and its effect is illustrated in Figure 4.1. For the particular case of ν = 1, the t-distribution reduces to the Cauchy distribution, while in the limit ν → ∞ the t-distribution St(x|μ, λ, ν) becomes a Gaussian N(x|μ, λ−1) with mean μ and precision λ.

A universal background model or speaker independent model is first created using speech samples from a large number, T, of speakers. The parameters of the UBM are found using an Expectation Maximization Algorithm which iteratively refines the random initialization of Student-T mixture parameters to monotonically increase the maximum likelihood based on the given feature vectors from the set of training data.

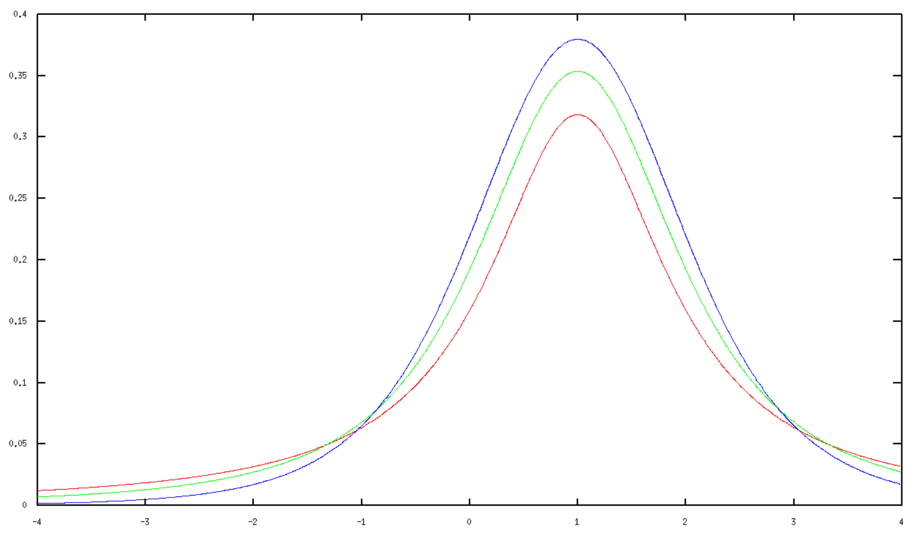

Figure 4.1 Student's T distribution with varying DOF

4.2 Expectation Maximization

The Expectation-Maximization (EM) iterative algorithm is a broadly applicable statistical technique for maximizing complex likelihoods and handling the incomplete data problem.

The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization(M) step, which computes parameters maximizing the expected log-likelihood found on the E step.

The approach of modeling of training data using Expectation Maximization algorithm is shown in the next chapter.

The Student’s t-distribution is a heavy tailed approximation to the Gaussian. It is therefore, natural to consider the mean and covariance of the SMM components to approximate the parameters of a GMM on the same data as it was described in the previous section. If the statistics of the images follow a Gaussian model, the degrees of freedom are relatively large and the SMM tends to be a GMM with the same parameters. If the images contain outliers, the parameters are weak and the mean and covariance of the data are appropriately weighted in order not to take into account the outliers.

CHAPTER FIVE: **AN APPROACH TO MODEL HUMAN SPEAKERS WITH STUDENT'S T MIXTURES**

5.1 Feature Extraction Approach

As discussed above, we use MFCC in order to extract the features of the speaker to further the modelling of training set.

Prior to performing the feature extraction using MFCC, we use a pre-emphasis filter technique in order to improve the signal to noise ratio.

5.1.1 Pre-Emphasis

Pre-emphasis refers to a system process designed to increase (within a frequency band) the magnitude of some (usually higher) frequencies with respect to the magnitude of other (usually lower) frequencies in order to improve the overall signal-to-noise ratio by minimizing the adverse effects of such phenomena as attenuation distortion or saturation of recording media in subsequent parts of the system.

Hence, it is required to boost the relative amplitudes of the modulating voltage for higher audio frequencies from 2 to approximately 15 KHz.

Figure given above was a result of application of Pre-emphasis filter on our data so as to degrade the noise level to the utmost minimalistic value.

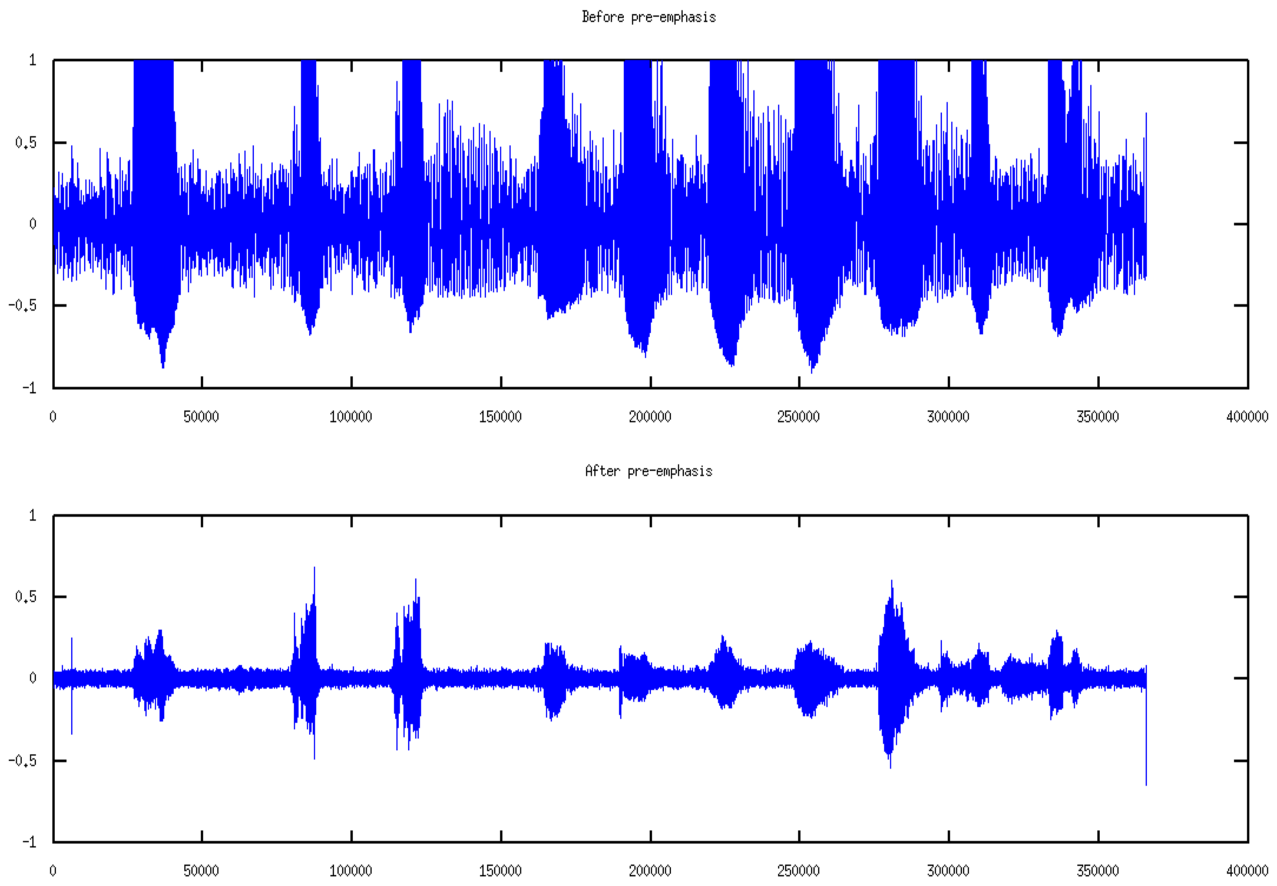

Figure 5.1 Pre-emphasis Filter to improve S/N ratio

5.1.2 Implementation Steps For MFCC

We start with a speech signal, we'll assume sampled at 16kHz.

1. Frame the signal into 20-40 ms frames. 25ms is standard. This means the frame length for a 16kHz signal is 0.025*16000 = 400 samples. Frame step is usually something like 10ms (160 samples), which allows some overlap to the frames. The first 400 sample frame starts at sample 0, the next 400 sample frame starts at sample 160 etc. until the end of the speech file is reached. If the speech file does not divide into an even number of frames, pad it with zeros so that it does.

The next steps are applied to every single frame, one set of 12 MFCC coefficients is extracted for each frame. A short aside on notation: we call our time domain signal $s(n)$. Once it is framed we have $s_i(n)$where n ranges over 1-400 (if our frames are 400 samples) and i ranges over the number of frames. When we calculate the complex DFT, we get $S_i(k)$, where the i denotes the frame number corresponding to the time-domain frame. $P_i(k)$, is then the power spectrum of frame i.

2. To take the Discrete Fourier Transform of the frame, perform the following:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \quad 1 \leq k \leq K \qquad \text{(Eq. 1.10)}$$

where $h(n)$ is an N sample long analysis window (e.g. hamming window), and K is the length of the DFT. The periodogram-based power spectral estimate for the speech frame $s_i(n)$ is given by:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \qquad \text{(Eq. 1.11)}$$

This is called the Periodogram estimate of the power spectrum. We take the absolute value of the complex fourier transform, and square the result. We would generally perform a 512 point FFT and keep only the first 257 coefficents as is shown in the figure below.

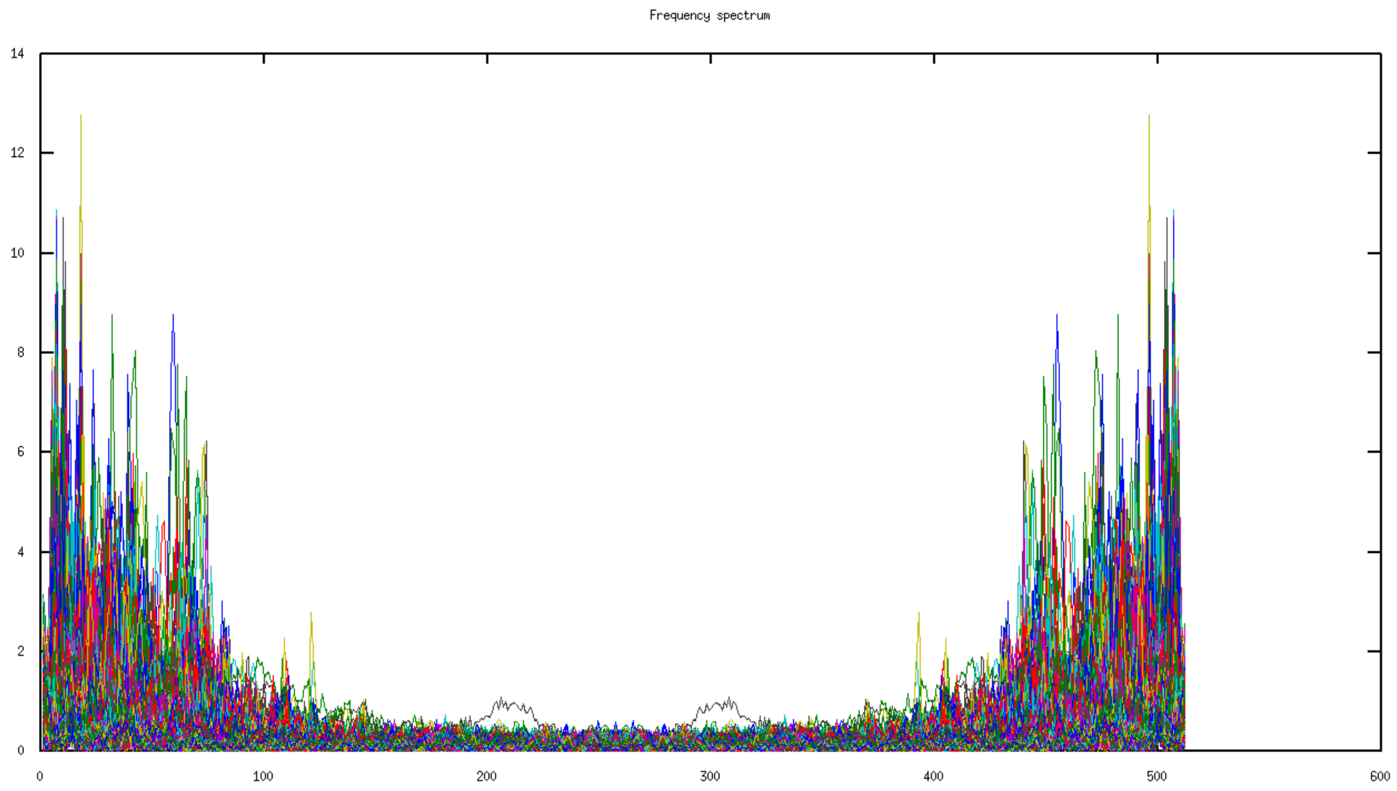

Figure 5.2 512 point Frequency Spectrum of frames

3. Compute the Mel-spaced filterbank. This is a set of 20-40 (26 is standard) triangular filters that we apply to the periodogram power spectral estimate from step 2. Our filterbank comes in the form of 26 vectors of length 257 (assuming the FFT settings from step 2). Each vector is mostly zeros, but is non-zero for a certain section of the spectrum. To calculate filterbank energies we multiply each filterbank with the power spectrum, then add up the coefficents. Once this is performed we are left with 26

numbers that give us an indication of how much energy was in each filterbank. For a detailed explanation of how to calculate the filterbanks see below.

4. Take the log of each of the 26 energies from step 3. This leaves us with 26 log filterbank energies.

5. Take the Discrete Cosine Transform (DCT) of the 26 log filterbank energies to give 26 cepstral coefficents. For ASR, only the lower 12-13 of the 26 coefficients are kept.

The resulting features (12 numbers for each frame) are called Mel Frequency Cepstral Coefficients.

5.1.3 Liftering

Before and after liftering. a filter that operates on a cepstrum might be called a *lifter*. A low pass lifter is similar to a low pass filter in the frequency domain. It can be implemented by multiplying by a window in the quefrency domain and when converted back to the frequency domain, resulting in a smoother signal.

From the above performed liftering technique, we take 2nd to 13th index as the features of the speaker.

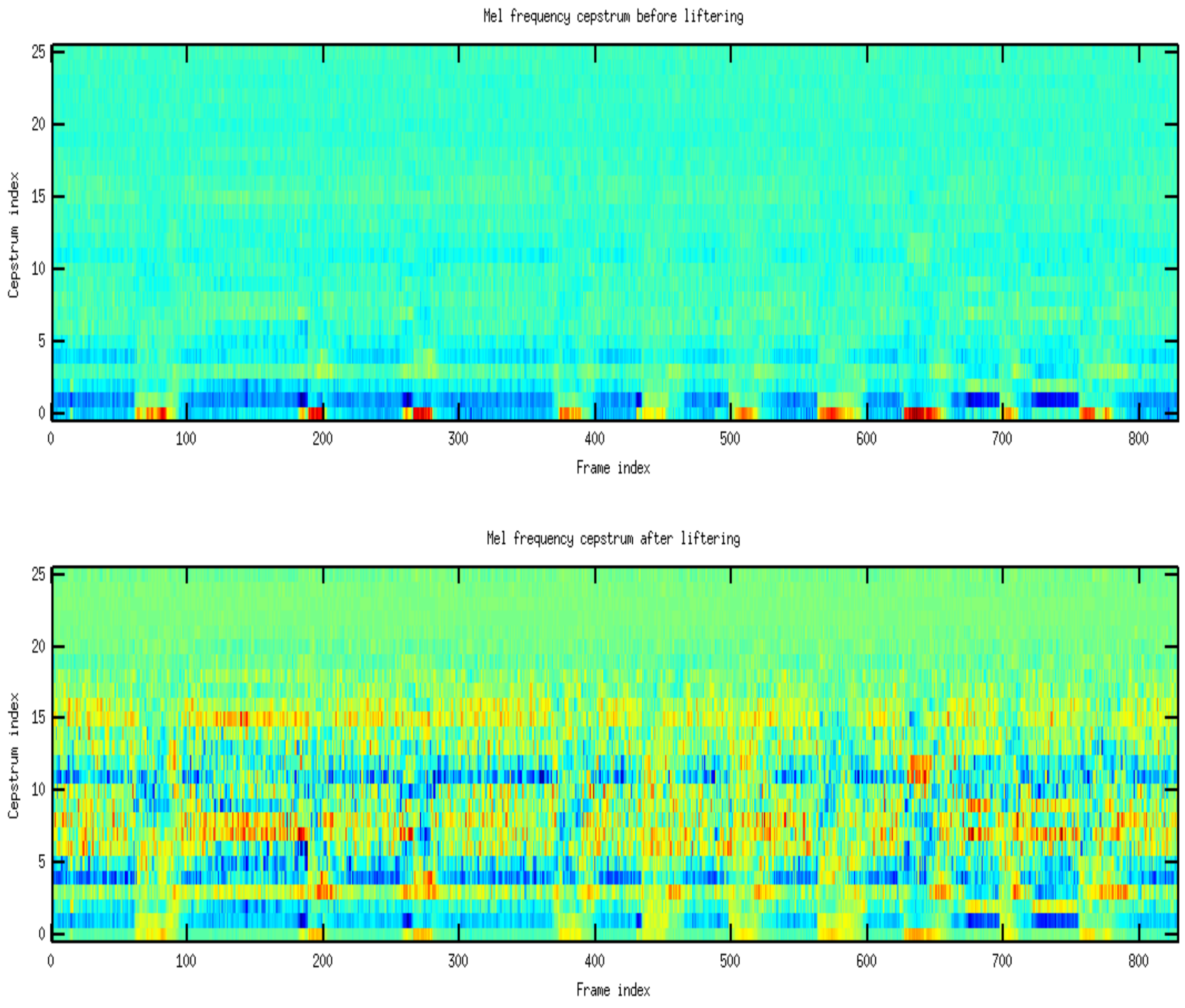

Figure 5.3 Frequency Cepstrum Before and After Liftering

5.2 Feature modeling

As explained in the previous topics, Modeling of the features obtained via MFCC using various techniques, for instance, HMM, GMM, VQ etc.

However, we have performed the analysis and modeling of the training set using Student's T mixture model.

5.2.1 Student's T-Mixture Modeling- EM Approach

The approach towards Student's T-mixture Modeling using EM Approach is discussed as follows:

5.2.1.1 ML estimation of mixtures of Student's t-distributions

A d-dimensional random variable X that follows a multivariate t-distribution with mean μ, positive definite, symmetric and real d x d covariance matrix Σ, $\nu \,\epsilon$ [0, ∞) degrees of freedom has a density expressed by:

$$p(x;\mu,\Sigma,\nu) = \frac{\Gamma\left(\frac{\nu+d}{2}\right)|\Sigma|^{-\frac{1}{2}}}{(\pi)^{\frac{d}{2}}\,\Gamma\left(\frac{\nu}{2}\right)\left[1+\nu^{-1}\delta(x,\mu;\Sigma)\right]^{\frac{\nu+d}{2}}} \qquad \text{(Eq. 1.12)}$$

Where $\delta(x,\mu,\Sigma) = (x-\mu)^T\Sigma^{-1}(x-\mu)$is the Mahalanobis squared distance and Γ is the Gamma function.

It can be shown that the Student's t distribution is equivalent to a Gaussian distribution with a stochastic covariance matrix. In other words, given a weight u following a Gamma distribution parameterized by

$$u \sim \Gamma(\nu/2, \nu/2)$$

The variable X has the multivariate normal distribution with mean μ and covariance, Σ/u.

$$X|\mu,\Sigma,\nu,u \sim N(\mu,\Sigma/u)$$

It can be shown that for $\nu \to \infty$the Student's t-distribution tends to a Gaussian distribution with covariance Σ. Also, if ν> 1; μ is the mean of X and if ν > 2; $\nu(\nu-2)^{-1}\Sigma$ is the covariance matrix of X. Therefore, the family of t-distributions provides a heavy-tailed alternative to the normal family with mean μ and covariance matrix that is equal to a scalar multiple of Σ, if ν > 2. A K-component mixture of t-distributions is given by:

$$\Phi(x,\psi) = \sum_{i=1}^{K} \pi_i p(x; \mu_i, \Sigma_i, \nu_i),$$

Where $x = (x_1, \dots\dots.., x_N)$ denotes the observed-data vector and

$$\psi = (\pi_1, \dots\dots\dots, \pi_K, \mu_1 \dots\dots, \mu_K, \textstyle\sum_1 \dots\dots, \sum_K, \nu_1 \dots\dots, \nu_K)^T$$

are the parameters of the components of the mixture. A Student's t-distribution mixture model (SMM) may also be trained using the EM algorithm. Consider now the complete data vector.

$$x_c = (x_1, \dots\dots, x_N, Z_1, \dots\dots, Z_N, \mu_1, \dots\dots, \mu_N)$$

where Z_1 ; ...; Z_N are the component-label vectors and z is either one or zero, according to whether the observation x is generated or not by the ith component. In the light of the definition of the t distribution, it is convenient to view that the observed data augmented by the z , j ¼ 1; ...; N are still incomplete because the component covariance matrices depend on the degrees of freedom. This is the reason that the complete-data vector also includes the additional missing data u1j.

5.2.1.2 E-Step

The E-step on the (p +1)th iteration of the EM algorithm requires the calculation of the posterior probability that the datum x_j belongs to the ith component of the mixture:

$$z_{ij}^{t+1} = \frac{\pi_i^t p(x_j;\, \mu_i^t, \Sigma_i^t, \nu_i^t)}{\sum_{m=1}^{K} \pi_m^t p(x_j; \mu_m^t, \Sigma_m^t, \nu_m^t)} \qquad \text{(Eqs. 1.13)}$$

as well as the expectation of the weights for each observation:

$$u_{ij}^{t+1} = \frac{\nu_i^t + d}{\nu_i^t + \delta(x_j, \mu_i^t; \Sigma_i^t)}$$

5.2.1.3 M-Step

Maximizing the log-likelihood of the complete data provides the update equations of the respective mixture model parameters:

$$\pi_i^{t+1} = \frac{1}{N} \sum_{j=1}^{N} z_{ij}^t \qquad \text{(Eqs. 1.14)}$$

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{N} z_{ij}^t \mu_{ij}^t x_j}{\sum_{j=1}^{N} z_{ij}^t \mu_{ij}^t}$$

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{N} z_{ij}^t \mu_{ij}^t x_j}{\sum_{j=1}^{N} z_{ij}^t \mu_{ij}^t}$$

The degrees of freedom for the ith component, at time step t + 1, are computed as the solution to the equation:

$$\log \frac{\nu_i^{t+1}}{2} - \psi\left(\frac{\nu_i^{t+1}}{2}\right) + 1 - \log\left(\frac{\nu_i^t + d}{2}\right) + \frac{\sum_{j=1}^{N} z_{ij}^t \left(\log u_{ij}^t - u_{ij}^t\right)}{\sum_{j=1}^{N} z_{ij}^t} + \psi\left(\frac{\nu_i^t + d}{2}\right) = 0$$

Where ψ is the digamma function.

At the end of the algorithm, the data are assigned to the component with maximum responsibility using a maximum a posteriori (MAP) principle.

CHAPTER SIX: **CONCLUSION,**

FUTURE SCOPE & REFERENCES

7.1 Conclusion

There has been a considerable amount of development in the field of speech and speaker recognition. The techniques for speaker recognition are yet to be successfully used in practical systems as the recognition rate is drastically reduced due to many reasons such as the distortion in the channel and the recording conditions and speaker-generated variability. Therefore it is important to explore stable features that remain insensitive to variation of speakers voice over time and are robust against variation in voice quality due to colds or disguises. The problem of distortion in the channels and background noise also requires being resolved with better techniques.

7.2 Future Work

The future scope of this project would be to compare speaker recognition performance of Student T- mixture model with Gaussian Mixture Model on TIMIT, VoxForge and NIST Speaker Data.

REFERENCES

1. Rabiner, L., "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE , vol.77, no.2, pp.257,286, Feb 1989

2. Bhardwaj, Saurabh, et al. "GFM-Based Methods for Speaker Identification." Cybernetics, IEEE Transactions on 43.3 (2013): 1047-1058.

3. Gerogiannis, D., Nikou, C., & Likas, A. (2009). The mixtures of Student's t-distributions as a robust framework for rigid registration. Image and Vision Computing, 27(9), 1285-1294.

4. Wong, C. S., W. S. Chan, and P. L. Kam. "A Student t-mixture autoregressive model with applications to heavy-tailed financial data." Biometrika 96.3 (2009): 751-760.

5. Lin, Tsung I., Jack C. Lee, and Wan J. Hsieh. "Robust mixture modeling using the skew t distribution." Statistics and Computing 17.2 (2007): 81-92.

6. Hasan, Md Rashidul, Mustafa Jamil, and Md Golam Rabbani Md Saifur Rahman. "Speaker identification using mel frequency cepstral coefficients." variations 1 (2004): 4.

7. Seddik, H.; Rahmouni, A.; Sayadi, M., "Text independent speaker recognition using the Mel frequency cepstral coefficients and a neural network classifier," Control, Communications and Signal Processing, 2004. First International Symposium on , vol., no., pp.631,634, 2004

8. Bilmes, Jeff A. "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models." International Computer Science Institute 4.510 (1998): 126.

9. Bishop, Christopher M. Pattern recognition and machine learning. Vol. 1. New York: springer, 2006.

10. Forsyth, David A., and Jean Ponce. Computer vision: a modern approach. Prentice Hall Professional Technical Reference, 2002.

11. Gish, Herbert, and Michael Schmidt. "Text-independent speaker identification." Signal Processing Magazine, IEEE 11.4 (1994): 18-32.

APPENDIX A

SOURCE CODE

A.1 MFCC Implementation

```
function [Cepstra] = mymfcc(signal, fs)

    %% INLINE FUNCTIONS

    freq2mel  = @(f)(1125 * log(1 + (f / 700))); % Convert frequency to Mel scale

    mel2freq  = @(m)(700 * (exp(m / 1125) - 1)); % Convert Mel scale to frequency

    window_fn = @(N)(0.54 - 0.46 * cos(2 * pi * [0 : N - 1].'/(N - 1))); % Hamming window

    ceplifter = @(N, L)(1 + 0.5 * L * sin(pi * [0 : N - 1] / L)); % Liftering function



    %% TUNING PARAMETERS

    Tw = 25; % Signal frame length (ms)

    Ts = 10; % Signal frame step (ms)

    R  = [300 8000]; % Frequency range (Hz)

    P  = 26; % Number of filters

    nfft = 2^9; % Length of Discrete FFT (number of frequency bins)

    preemph = 0.97;
```

```
%% USEFUL VARIABLES

Nw = round(1E-3 * Tw * fs); % Number of samples in frame duration

Ns = round(1E-3 * Ts * fs); % Number of samples for frame overlap

f_min = 0; % Filter coefficients start at this frequency (Hz)

f_max = fs / 2; % Filter coefficients end at this frequency (Hz)



%% PROCESSING

% Pre-emphasis filtering

% figure(1); subplot(211); plot(signal); title("Before pre-emphasis");

signal = filter([1 -preemph], 1, signal);

% figure(1); subplot(212); plot(signal); title("After pre-emphasis");



% Add padding to signal

L = length(signal);

M = ceil(((L - Nw) / Ns) + 1); % Number of frames

N = Nw + (Ns * (M - 1)); % Number of samples after padding

pad = N - L; % Number of samples to be padded

signal = [signal; zeros(pad, 1)];
```

```
% Convert signal vector to matrix of frames (each frame as a column)

indf    = Ns * [0 : (M - 1)];

inds    = [1 : Nw].';

indexes = indf(ones(Nw, 1), :) + inds(:, ones(1, M));

frames  = signal(indexes);



% Apply analysis window to the implicit rectangular window frame

window = window_fn(Nw);

frames = diag(window) * frames;



dft = abs(fft(frames, nfft, 1)); % Discrete FFT

% figure(2); plot(dft); title("Frequency spectrum");



% Compute the triangular filter bank

mel_low  = freq2mel(R(1));

mel_high = freq2mel(R(2));

Q = (nfft / 2) + 1; % Number of unique FFT values
```

```
    c = mel2freq(mel_low + ([0 : P + 1] * (mel_high - mel_low) / (P + 1))); % Frequency
cutoffs

    f = linspace(0, (fs / 2), Q);

    filterbank = zeros(P, Q);

    for (m = 1 : P)

        k = (f >= c(m)) & (f <= c(m + 1)); % left-to-centre

        filterbank(m, k) = (f(k) - c(m)) / (c(m + 1) - c(m));

        k = (f >= c(m + 1)) & (f <= c(m + 2)); % centre-to-right

        filterbank(m, k) = (c(m + 2) - f(k)) / (c(m + 2) - c(m + 1));

    end

    % figure(2); plot(filterbank(:, [1 : 100]).'); title("Mel Filterbank");



    % Apply filter bank to unique part of frequency spectrum

    fbe = filterbank * dft(1 : Q, :);



    Cepstra = dct(log(fbe));



    % figure(3); subplot(211); plot(Cepstra); title("Cepstral coefficients before liftering");

    % figure(4, 'Position', [30 100 800 200], 'PaperPositionMode', 'auto', ...
    %       'color', 'w', 'PaperOrientation', 'landscape', 'Visible', 'on' );

    % subplot(211);

    % imagesc([1 : size(Cepstra, 2)], [0 : P - 1], Cepstra);

    % axis('xy');

    % xlabel('Frame index');

    % ylabel('Cepstrum index');

    % title('Mel frequency cepstrum before liftering');



    lifter = ceplifter(P, 22);

    Cepstra = diag(lifter) * Cepstra;



    % figure(3); subplot(212); plot(Cepstra); title("Cepstral coefficients after liftering");

    % figure(4, 'Position', [30 100 800 200], 'PaperPositionMode', 'auto', ...

    %       'color', 'w', 'PaperOrientation', 'landscape', 'Visible', 'on' );

    % subplot(212);

    % imagesc([1 : size(Cepstra, 2)], [0 : P - 1], Cepstra);

    % axis('xy');

    % xlabel('Frame index');
```

```
    % ylabel('Cepstrum index');

    % title('Mel frequency cepstrum after liftering');



    Cepstra = Cepstra';

end
```

A.2 Mahalanobis distance computation

```
function [retval] = mahalanobis(x, Mu, S)

        D = size(x, 2);

        K = size(Mu, 1);

        retval = zeros(K, 1);

        for m = 1:K

                retval(m) = (x - Mu(m, :)) * inv(S(:, :, m)) * (x - Mu(m, :))';

        end
```

A.3 Student’s-T distribution PDF

```
function [pdf] = tpdf(x, Mu, S, nu)

        D = size(x, 2);

        K = size(Mu, 1);



        pdf = zeros(K, 1);



        for m = 1:K

                pdf(m) = (gamma((nu(m) + D) / 2) * (det(S(:, :, m)) ^ (-1 / 2))) ...

                        / (((pi * nu(m)) ^ (D / 2)) * gamma(nu(m) / 2) ...

                        * ((1 + (mahalanobis(x, Mu(m, :), S(:, :, m)) / nu(m))) ^
((nu(m) + D) / 2)));

        end
```

A.4 Student's-T Mixture Model

```
function model = tmixture(X, K)
   % debug_on_warning (true);


   %% USEFUL VARIABLES
   N = size(X, 1); % Number of samples
   D = size(X, 2); % Number of states


   %% PARAMETERS
   tol     = 1e-10;
   maxiter = 300;


   %% INITIALIZE
   [idx, Mu] = kmeans(X, K);


   S   = zeros(D, D, K);
   nu  = 10(ones(K, 1));
   Z   = double(bsxfun(@(a, b)(a == b), idx'(ones(K, 1), :), [1:K]').'); % (N * K)
Matrix
   U   = gampdf(Z, (mean(nu) / 2), inv(mean(nu) / 2)); % (N * K) Matrix
   pie = (sum(Z, 1) / N).'; % (K * 1) Vector


   for m = 1:K
      S(:, :, m) = (nu(m) / (nu(m) - 2)) .* cov(X(idx == m, :));
```

```
end


llh = tmixturellh(X, Mu, S, nu, Z, U, pie);


%% PROCESSING
t = 0;
converged = false;


while ~converged && (t < maxiter)
    tic;
    t++;


    fprintf('Doing iteration %d/%d...\n', t, maxiter);


    % if (t == 2)
    %     figure;
    %     hold on;
    %     plot(X(:, 1), X(:, 2), 'ro');


    %     for m = 1:K
    %         plot_gaussian_ellipsoid(Mu(m, :), S(:, :, m));
    %     end
    %     hold off;
    % end
    Mu_old  = Mu;
    S_old   = S;
    nu_old  = nu;
    Z_old   = Z;
    U_old   = U;
    pie_old = pie;
    llh_old = llh;

    % E step
    for q = 1:N
        tmpval  = abs((pie_old .* tpdf(X(q, :), Mu_old, S_old, nu_old)).');
        Z(q, :) = tmpval / sum(tmpval);

        U(q, :) = ((nu_old .+ D) .* ((nu_old + mahalanobis(X(q, :), Mu_old, S_old)) .^
(-1))).';
    end

    % M step
    pie = (sum(Z_old, 1) / N).';

    for p = 1:K
        tmpval     = (Z_old(:, p) .* U_old(:, p));
        Mu(p, :)   = sum(bsxfun(@(a, b)(a .* b), tmpval, X), 1) / sum(tmpval, 1);

        S(:, :, p) = bsxfun(@(a, b)(a .* b), tmpval, ...
            bsxfun(@minus, X, Mu(p, :))).' * bsxfun(@minus, X, Mu(p, :)) / sum(Z(:,
p), 1);
    end

    func = @(y)(log(y / 2) - digamma(y / 2) + 1 - log((nu_old + D) / 2) +
digamma((nu_old + D) / 2) ...
        + (sum((Z_old .* (log(U_old) - U_old)), 1) ./ sum(Z_old, 1)).');
    nu = abs(fsolve(func, nu_old));

    llh = tmixturellh(X, Mu, S, nu, Z, U, pie);
    dif = llh - llh_old;
    fprintf('Difference is %d\n', dif);
    converged = abs(dif) < tol;

    toc
    fflush(stdout);
end

% figure;
% hold on;
% plot(X(:, 1), X(:, 2), 'ro');

% for m = 1:K
%     plot_gaussian_ellipsoid(Mu(m, :), S(:, :, m));
% end
% hold off;


model.K  = K;

model.Mu = Mu;

model.S  = S;

model.nu = nu;

model.pi = pie;

end
```

A.4 Student’s T-mixturellh model

```
function llh = tmixturellh(X, Mu, S, nu, Z, U, pie)

    %% USEFUL VARIABLES

    N = size(X, 1); % Number of samples

    D = size(X, 2); % Number of states

    K = size(Mu, 1); % Number of hidden components


    %% PROCESSING

    M = zeros(N, K);

    detm = zeros(K, 1);


    for p = 1:N, M(p, :) = mahalanobis(X(p, :), Mu, S); end


    for q = 1:K, detm(q) = det(S(:, :, q)); end


    llh1 = sum(sum(bsxfun(@(a, b)(a .* b), Z, log(pie).')));


    llh2 = sum(sum(Z .* bsxfun(@plus, (bsxfun(@(a, b)(a .* b), ...

        (log(U) - U), (nu / 2).') - log(U)), ...

        (((nu / 2) .* log(nu / 2)) - log(gamma(nu / 2))).')));


    llh3 = sum(sum(Z .* ((D * log(2 * pi)) + ...

        bsxfun(@plus, detm', (U .* M))) * (-1 / 2)));
    llh = llh1 + llh2 + llh3;
end
```

A.6 T probability of mixtures

```
function p = tprobability(X, model)

        %% USEFUL VARIABLES

        N = size(X, 1);

        D = size(X, 2);

        K = model.K;



        %% INITIALIZE

        p = 1;



        %% PROCESSING

        for g = 1:N

                ptemp = 0;



                for h = 1:K
                ptemp += model.pi(h) * tpdf(X(g, :), model.Mu(h), model.S(:,
:, h), model.nu(h));
            end

            p *= ptemp
        end

end
```

Brief Information about Authors

Nilaksh Das:

Roll Number: 480/IC/10

Email: nilakshdas@gmail.com

Shikhar Kwatra

Roll Number: 520/IC/10

Email: kwatra.contact@gmail.com

Siddharth Sharma

Roll Number: 527/IC/10

Email: siddharthsharma52@gmail.com