ACCENT CLASSIFICATION: LEARNING A DISTANCE METRIC OVER PHONETIC STRINGS

by

Swetha Machanavajhala

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computing

School of Computing

The University of Utah

December 2013

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Swetha Machanavajhala**

has been approved by the following supervisory committee members:

Suresh Venkatasubramanian	, Chair	**06/10/2013** Date Approved
Jeffrey M. Phillips	, Member	**06/07/2013** Date Approved
Ellen Riloff	, Member	**06/07/2013** Date Approved

and by **Alan Davis** , Chair of

the Department of **School of Computing**

and by David B. Kieda, Dean of The Graduate School.

ABSTRACT

Presently, speech recognition is gaining worldwide popularity in applications like Google Voice, speech-to-text reporter (speech-to-text transcription, video captioning, real-time transcriptions), hands-free computing, and video games. Research has been done for several years and many speech recognizers have been built. However, most of the speech recognizers fail to recognize the speech accurately. Consider the well-known application of Google Voice, which aids in users search of the web using voice. Though Google Voice does a good job in transcribing the spoken words, it does not accurately recognize the words spoken with different accents. With the fact that several accents are evolving around the world, it is essential to train the speech recognizer to recognize accented speech. Accent classification is defined as the problem of classifying the accents in a given language. This thesis explores various methods to identify the accents. We introduce a new concept of clustering windows of a speech signal and learn a distance metric using specific distance measure over phonetic strings to classify the accents. A language structure is incorporated to learn this distance metric. We also show how kernel approximation algorithms help in learning a distance metric.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Prof. Suresh Venkatasubramanian, for encouraging me to explore this new field of Accent Classification. His expertise, patience, understanding, and constant encouragement has immensely contributed to my graduate experience. I would also like to express my thanks to my committee, Prof. Jeff Phillips and Prof. Ellen Riloff, for their assistance and support to my thesis.

I would like to thank Dr. Piyush Rai for his excellent teaching in Machine Learning that has formed the foundation for starting this thesis, and for his non-academic support.

Many thanks to my friends, Radhika Gupta, Samira Daruki, Chinmayee, Dr. Avishek Saha, Namrata Dey, Parasaran Raman, and John Moeller, who were there in needful times and with whom I have shared fun times.

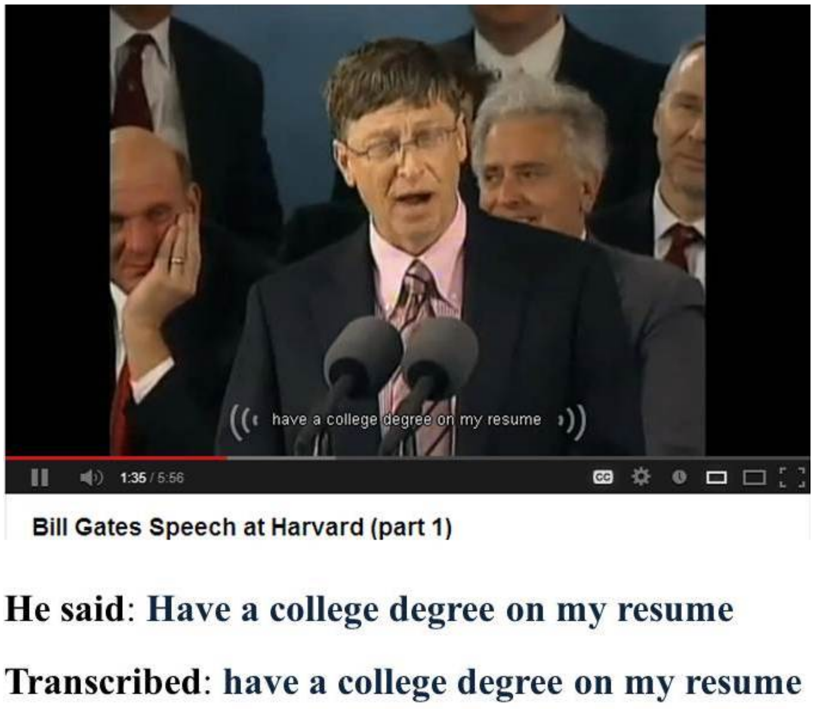
Finally, I thank my family for their continuous support throughout my masters. They have always encouraged me in all walks of life.

CHAPTER 1

INTRODUCTION

Presently, speech recognition is gaining popularity [1] and is being used in a lot of applications like Google Voice, speech-to-text reporter (speech-to-text transcription, video captioning, real-time transcriptions), hands-free computing, and video games. Automatic speech recognition is a very difficult problem [2] and research has been done for a few decades [3]. Applications like Google Voice, automatic captions in YouTube videos, Siri, and S-Voice softwares in mobile phones have been developed to serve the purpose of recognizing the speech and thereby transcribing them into text. Though these applications do a good job in recognizing the speech, we found that automatic captions in YouTube videos fail to recognize the words spoken with different accents. Figure 1.1 shows an example where the speech is being accurately transcribed while Figure 1.2 shows how inaccurate the transcription could be when there is a variation in the accent. The University of Birmingham did a study [4] on the speech recognition technology used by call centers and discovered that speakers with strong accents were frequently misunderstood when dealing with call centers and concluded that accents were the main problem behind the performance of voice recognition. Huang et al. [5] points out that the accuracy of speech recognition fluctuates depending on the speaker variability, especially when the speaker has a strong accent. It has also been discovered that Siri, Apple's personal assistant, is unable to recognize words spoken with different accents [6, 7].

With the above drawbacks and with the fact that several accents are evolving around the world [8], it is essential to train the speech recognizer to be a universal one such that it performs accurately around the globe. This is where the problem of recognizing accents comes into picture. Accents are one of the most important features that distinguishes speakers in the way they pronounce the words differently. In order for a speech recognizer to identify the accented spoken words, the first step is to classify the accents such that a speech recognizer trained for that particular accent can perform well in transcribing the accented spoken words. This type of speech recognition is referred to as Accented Speech



Bill Gates Speech at Harvard (part 1)

He said: **Have a college degree on my resume**

Transcribed: **have a college degree on my resume**

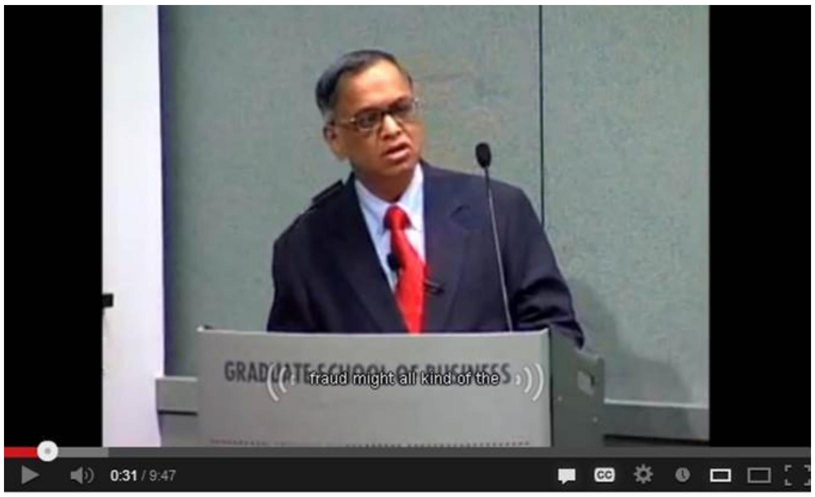
Figure 1.1: Example 1: Accurate transcription of speech using automatic captions in YouTube

Recognition (ASR).

The first step in Accented Speech Recognition is to classify the accents using the features extracted from the speech. Accent classification [9] is defined as the problem of classifying the accents in a given language. The motivation of this problem is that, if accent classification is implemented successfully, the accent classifier component can be fed into a speech recognition engine. This can improve the accuracy of detecting spoken words and thereby improve the quality of transcription in YouTube videos. This entire software, if fed into the online videos and real-time captioning systems, can enable the automation of captions in videos and deliver real-time captions in any surrounding. As a result, this could avoid the need of costly manual transcriptions.

The first step in classifying accents is to extract meaningful features so that the classifier can accurately identify the accents. Techniques like Perceptual Linear Prediction (PLP) [10], Linear Predictive Coding (LPC)[11], and Mel Frequency Cepstral Coefficients (MFCC)[12] were used to extract feature vectors from the speech.

Next, classifiers like LibSVM [13] and Gaussian Mixture Models (GMM) [14] were run on the extracted feature vectors to detect the accent of the speech. This is a simple architecture of accent classification and the above methods were used in previous studies [15, 16, 9, 17].



N. R. Narayan Murthy: Do As You Say to Gain Credibility

He said: **Throughout my talk**

Transcribed: **fraud might all kind of the**

Figure 1.2: Example 2: Inaccurate transcription of speech using automatic captions in YouTube

The architecture is shown in Figure 1.3.

This thesis explores new methods of classifying accents and is described as three phases in the accent classification architecture. Figures 1.4, 1.5, and 1.6 show the architecture consisting of three phases as proposed in this thesis and below is a brief description of each phase.

The first phase involves reducing the number of windows so as to reduce the amount of data to deal with. This step is done after grouping the samples into windows during the feature extraction process. At this point, each speaker contains 2046 windows and each window consists of a set of 39 features. Since 2046 windows per speaker is large and this makes the classifiers run very slowly, Watanaprakornkul et al. [16] chose 10 windows randomly to reduce the amount of data. Since choosing the windows randomly might not be a feasible option, we proposed to reduce the windows using K-Means as a clustering step. We also experimented sampling the windows using column and random sampling.

In the second phase, after reducing the number of windows, we incorporated a language structure on every accent. Using this information, we learned a distance metric that pulls apart different accents by a specific distance measure. This distance measure was found by determining the edit distance between phonetic strings of accents.

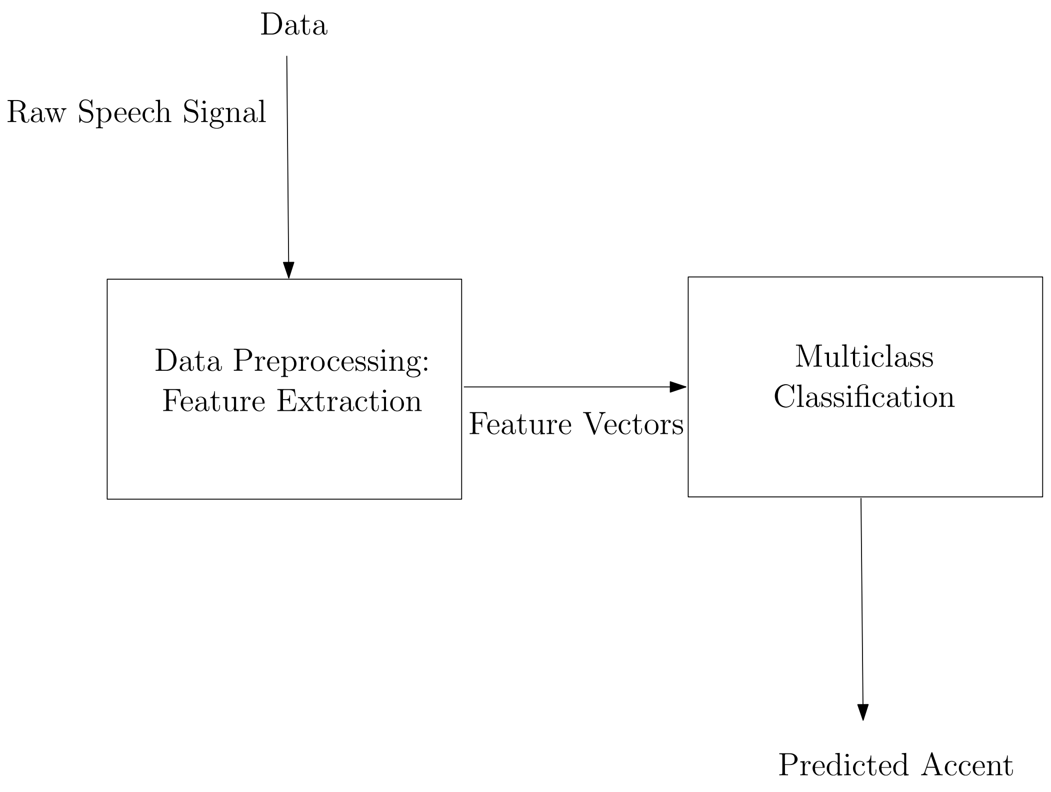

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Feature Vectors
Multiclass Classification
Predicted Accent

Figure 1.3: Accent classification: Existing architecture

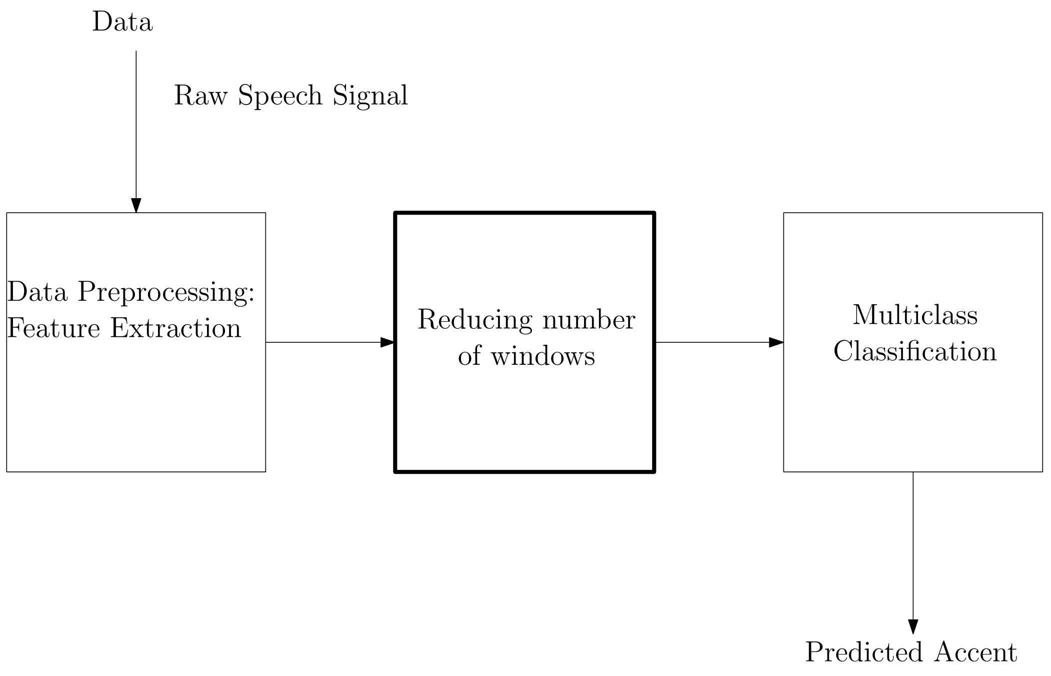

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Reducing number of windows
Multiclass Classification
Predicted Accent

Figure 1.4: Phase 1: Reducing number of windows

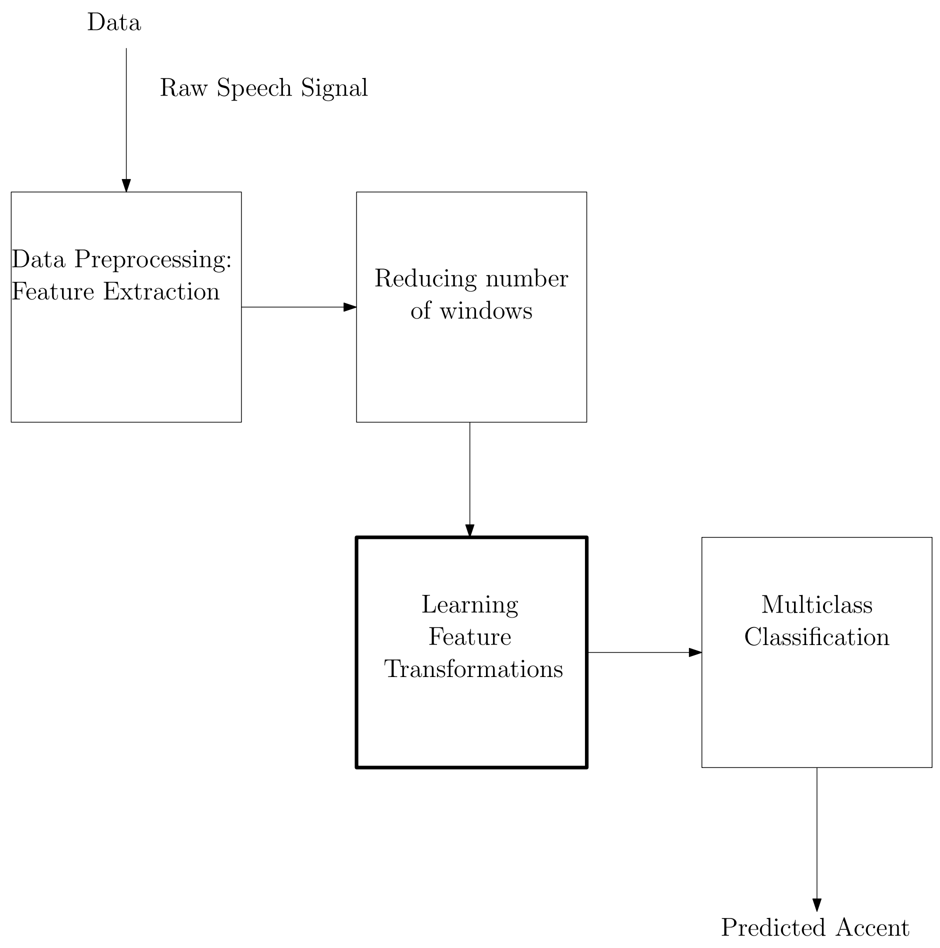

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Reducing number of windows
Learning Feature Transformations
Multiclass Classification
Predicted Accent

Figure 1.5: Phase 2: Learning feature transformations - better representation of classifying accents

Finally, we also looked at how to deal with nonlinear class boundaries prior to learning a distance metric and classification of accents. In the case of nonlinear data, we used kernel approximation algorithms like Kernel Principal Component Analysis (KPCA) [18] and the Rahimi Recht method [19].

1.1 Thesis Outline

This thesis is organized as follows:

Chapter 2 gives a detailed literature survey on the techniques that have been researched previously from the perspective of Linguistics as well as Computer Science. Chapter 3 gives an overview on the feature extraction process using Perceptual Linear Prediction (PLP) and explains the procedure. Chapter 4 describes the various classification techniques that were experimented. Chapter 5 explains the background and existing methods that have been researched in the area of Distance Metric Learning. Chapter 6 gives a detailed

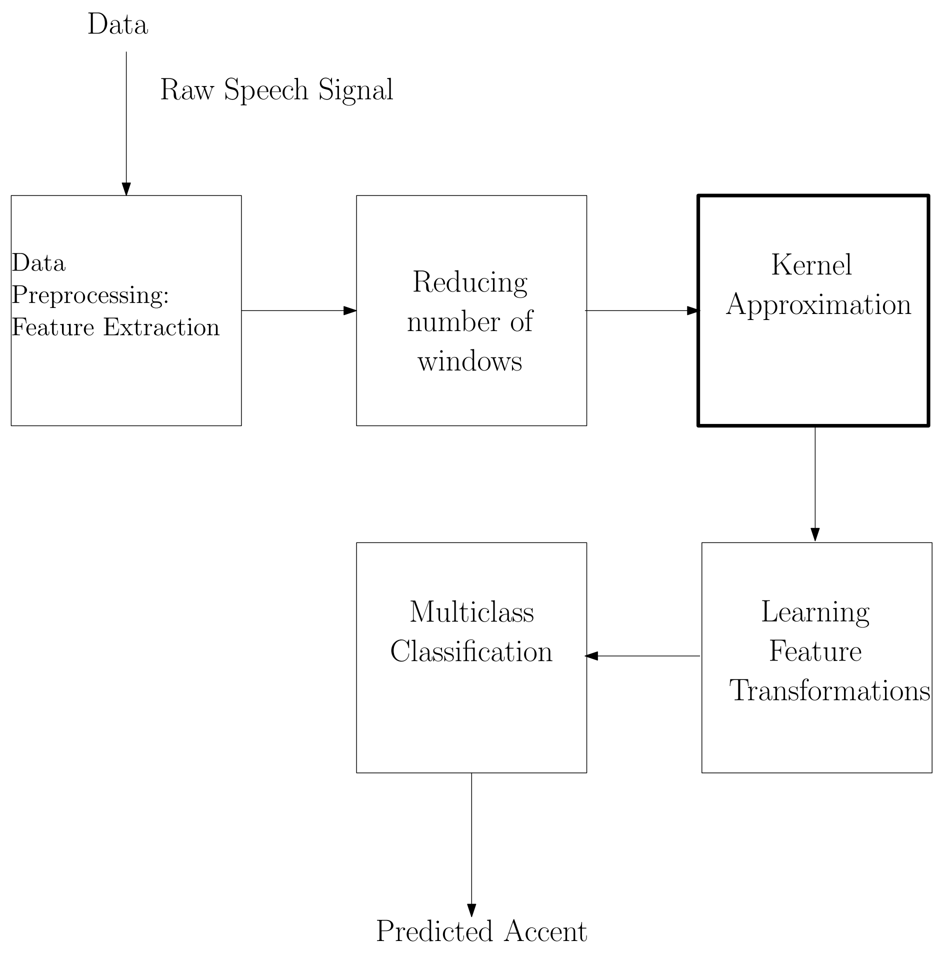

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Reducing number of windows
Kernel Approximation
Learning Feature Transformations
Multiclass Classification
Predicted Accent

Figure 1.6: Phase 3: Kernel approximation - to deal with nonlinear data

account on the three phases of the proposed methods along with experimental results. Chapter 7 concludes by giving an overview of the performance of the various techniques experimented with in this thesis. Chapter 8 talks about the possibilities of improving the accent classification system and the methods that can be experimented with in future.

CHAPTER 2

LITERATURE SURVEY

Accented Speech Recognition (ASR) is gaining more importance today since accent is one of the factors that affects the speaker variation. Due to this fact, the presence of accents in speech could be one of the reasons for degradation of performance in speech recognizers. Furthermore, since a lot of accents have evolved around the world, extensive research is being done on this problem so as to make the speech recognizers universal and compatible with all accents. There has been a lot of study on accent detection in the areas of linguistics as well as computer science. This chapter gives a detailed description of the research studies in both linguistics and computer science areas.

2.1 Linguistics Perspective

Ikeno and Hansen [20] proposed a method to distinguish the accents by identifying the speech characteristics across a variety of native and nonnative English accents. They conducted experiments on 3 types of listeners: US, British native English, and nonnative English listeners. As a result, they concluded that listeners who were familiar with accents could detect the accents more accurately than the listeners who were not familiar with the accent. In addition to just detecting the accents, the listeners were a given a task to transcribe the speech samples. This was the road to achieving a benchmark for an automatic Accented Speech Recognition system.

Adank et al. [21] recorded the haemodynamic responses (response to stimuli such as exercise, emotional stress, or variation in accent in this study) of the participants in an MR scanner. They listened to two sentences that were presented in quick succession. Two hundred and fifty-six sentences were taken from the Speech Reception Threshold corpus (SRT) where 128 sentences were recorded in standard Dutch and the remaining in the novel accent that was merely different from the standard accent. It was observed that there was a relative increase in the areas of temporal, anterior, posterior, and frontal lobes in the brain in the case of an accent variation.

Luca Ragnoni [22] claimed that the previous studies [23, 24, 25, 26] did not quantify the role of prosodic aspects such as intonation patterns, pitch range, rhythm, and speech rate in foreign accent detection. The dataset was partitioned into 4 sets. One set consisted of natural speech samples while the other 3 sets contained monontonized-only, monotonized and PURR (Prosody Unveiling through Restricted Representation)-filtered, and PURR-filtered sentences. The listeners were presented randomly with the speech samples that were repeated thrice. The listeners scores were well defined for the sets: PURR-filtered and monotonized PURR-filtered sentences. This confirmed that prosody was an essential cue to detect the foreign accent.

2.2 Computer Science Perspective

2.2.1 Classification on Foreign Accented English

Chouiter et al. [9] did an empirical study on the 23 accents and claimed a detection rate of 32.7% using GMM baseline after cumulative application of Heteroscedastic Linear Discriminant Analysis (HLDA), Maximum Mutual Information (MMI), and Gaussian Tokenizer (GT).

Neidert, Chen, and Lee [15] experimented over 2 accents, German and Mandarin, and extended the classification experiment to 12 accents. The resulting accuracy for 2 accents was 57.12% using Support Vector Machine (SVM) and Linear Predictive Coding (LPC) as the feature extraction method. On 12 accents, they obtained an accuracy of 13.32%

Watanaprakornkul, Eksombatchai, and Chien [16] classified 3 foreign accents: Cantonese, Hindi, and Russian, using SVM and Gaussian Mixture Models (GMM). The features were extracted using Mel Frequency Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP) approaches. Using SVM, they obtained an accuracy of 41.18% and 37.5% accuracy using GMM. Out of 2000 windows, of each speech utterance that was obtained after feature extraction, 10 windows were randomly picked for the experiments.

Tang et al. [27] classified accents using Hidden Markov Models (HMM), Directed Acyclic Graph SVM (DAGSVM), and Support Vector Machines (SVM). They found that DAGSVM performed similarly to that of HMM but better than SVM, whereas SVM was effective in classifying different accents.

Novich et al. [28] did a study on accent classification using neural networks where they extracted the format features from the vowels of the speech utterances.

2.2.2 Classification Based on Vowel Distances

Huckvale [29] came up with a metric, AccDist, for comparing the similarities among accents and classified 14 British accents with an accuracy of about 80%. The similarities were computed by considering the distance among stressed vowels like *after* , *father* , *cat*. The correlation of these distances across speakers was computed in the AccDist metric.

Ferragne et al. [30] explored another similarity methodology by measuring the distance between vowels in the 14 British accents. The distance between vowels were measured among the Mel-cepstrum features of the speech. Hierarchical clustering and Multi-Dimensional Scaling (MDS) were applied to make the acoustic vowel distances explicit. The acoustic distance between accents was estimated by correlating the individual vowel distance matrices.

2.2.3 Classification Based on Distance Metric Learning

Ullah and Karray [31] and [32] employed a distance metric learning approach based on Mahanalobis distance. Their goal was to maximize the distance between dissimilar pairs and the similarities between the accents were not considered since they mainly focused on distinguishing the accents. They classified 2 accents, North midland and Western, and came up with 74.12% accuracy.

2.2.4 Other Notable Approaches

Zheng et al. [33] built an optimized MAP/MLLR combination and developed new approaches in detecting the degree of accent in Accented Speech Recognition. Their approaches were phoneme-based automatic accent detection, formant-augmented acoustic features for accented speech recognition, and accent-based model selection. They focused on only one form of Shanghai-Accent and obtained a character error rate of 39% to 49% in classifying it into more standard and more accented speech.

Vergyri et al. [34] examined the accent variation in English broadcast news data. They aimed at building multiple models of smaller accent variances. An acoustic level and Maximum-A-Posteriori (MAP) to refine the acoustic models were used to handle the accents at the acoustic level. An accent-independent model was first trained on the corpus and it was found that the performance increased greatly but decreased on some subsets of data. To improve the performance on these subsets, an accent-dependent model was used where the accents were classified using 2 GMMs for each accent (one for male and one for female) since the gender was one of the main factors influencing the variability in accents.

Lin et al. [35] proposed phoneme-less hierarchical algorithm for classifying accents. In each utterance of the speech signals, the gender was detected and modeled as female and male accents. On top of these models, the accents were classified using GMM with the features extracted using MFCC. They were able to distinguish American and British accent with an accuracy of 83%.

Biadsy et al. [36] introduced a new approach of recognizing dialects using a Phone-GMM-supervector-based SVM Kernel. A kernel was designed such that it could compute the similarities of phones between pairs of utterances. This kernel function was obtained by recognizing the phones of each utterance using a phone recognizer and extracting the GMM supervector for each phone. This resulted in a set of vectors needed to design the kernel function. They were able to achieve an Equal Error Rate (EER) of 4.9% overall. Furthermore, the performance was improved when compared to existing approaches like Phone Recognition followed by Language Modeling (PRLM), GMM Universal Background Model (GMM-UBM) with feature space Maximum Likelihood Linear Regression (fMLLR).

Humphries et al. [37] focused on modeling accent-specific pronunciation variations. First, accurate phone level transcriptions of accented data were obtained with an assumption that accent variation is more evident in vowels than consonants. From these aligned transcriptions, a list of context-dependent phone substitutions, deletions, and insertions are obtained such that it represented the pronunciation of target accent speakers which was different from the seed accent speakers. These context-dependent variations were then clustered in a binary decision tree which built a new pronunciation dictionary for recognizing the accents. Two accents in England region, London, and South East were considered. When a sufficient number of pronunciation variants were used, speech recognition word error rate (WER) decreased by 20%.

2.3 Summary

In this chapter, we did a literature survey of the methods dealing with accent classification in linguistic and computer science fields. We saw that research studies in the lingustic field (Section 2.1) has provided essential clues on identifying the important features to train a computer in classifying the accents. Section 2.2.1 shows that a lot of statistical classifiers have been trained in order to get accurate classification of accents. Among them, it was found that SVM was efficient in identifying different accents. As in a previous study [16], speech signal windows were randomly sampled before the accents were classified. We feel that this random sampling of windows might be an infeasible option and we proposed ways of reducing the number of windows as explained in Chapter 6. We also saw a new approach in classifying accents by learning a distance metric in Section 2.2.3 and this could lead us to find a better way of classifying accents. In Chapter 6, we propose a new method in learning a distance metric based on phonetic strings.

CHAPTER 3

DATA

3.1 Foreign Accented English (FAE)

This speech corpus contains utterances of American English spoken by non-native speakers of 23 accents. The different accents are: Arabic, Brazilian Portuguese, Cantonese, Czech, Farsi, French, German, Hindi, Hungarian, Indonesian, Italian, Japanese, Korean, Mandarin, Malay, Polish, Portuguese, Russian, Swedish, Spanish, Swahili, Tamil, and Vietnam. The text fragments were different for each speaker where they gave an introduction about themselves. There are 4925 utterances of telephone quality and were rated as strongly accented based on three independent judgements on each utterance. Since these utterances are of telephone quality, the sampling rate was found to be 8KHz and the samples are stored as 8-bit WAV files. For experimentation, we considered voice samples that had duration of 20 seconds [38]

3.2 George Mason University (GMU) Speech Accent Archive

We also experimented using another dataset that consisted of accented speakers of over 200 accents. We considered only 22 accents that were similar to the FAE dataset. The speakers were made to read an elicitation paragraph:

> ”Please call Stella. Ask her to bring these things with her from the store: Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station.”

The phonetic trancription of each speaker was done by 2 to 4 English-speaking judges who were phonetically educated. The voice samples had a sampling rate of 16 KHz and were stored as 16 bit MOV files. We used a converter to convert the MOV files to WAV format [39].

3.3 Data Preprocessing - Feature Extraction

An automatic speech recognition system performs several tasks like speaker recognition, speech recognition and more recently accent classification. All these tasks perform pattern recognition for classification or recognition on speech utterances in the form of training and test sets. We know that a speech utterance consists of several thousands of samples; hence, it would be essential to provide a stable representation of the signal for the classifier or machine to perform well in the classification stage. For this reason, the first step in speech recognition is the front-end or data processing also known as feature extraction.

There are many popular feature extraction techniques in signal processing: Mel-frequency cepstral coefficients (MFCCs), Linear Predictive Coding (LPC), Perceptual Linear Prediction (PLP). Out of these techniques, we implemented Perceptual Linear Prediction (PLP) technique [40] as this extracts the phonetic features of the utterances. Furthermore, PLP is more robust to noisy data compared to MFCC and LPC. [10]

This section is organized as follows: First we illustrate the block diagram of PLP technique, followed by an explaination of each phase [10] in the feature extraction process.

3.3.1 PLP Block Diagram

The procedure for extracting features using PLP is shown in Figure 3.1

3.3.2 Spectral Analysis

Spectral analysis is the key step in describing the characteristics of a signal. It breaks down a complex signal to simpler parts. This analysis produces a power spectrum which has the ability of describing the frequencies which contains the signal's power. In this step, the speech segment is weighted by a Hamming window $W(n)$ as follows:

$$W(n) = 0.54 + 0.46cos\left[\frac{2\pi n}{(N-1)}\right]$$

where, N is length of the window; for example, if a signal has a sampling frequency of 10KHz, then the length of the window is 256. The window length is fixed and n is any number within the range of 0 to 255, more formally in the range of 0 to N-1.

Fast Fourier Transform (FFT) is used to compute a Discrete Fourier Transform (DFT) which tranforms the windowed segment to the frequency domain. The real and imaginary parts of the short-term speech spectrum are squared and added to get the power spectrum $P(\omega)$

$$P(\omega) = Re[S(\omega)]^2 + Im[S(\omega)]^2$$

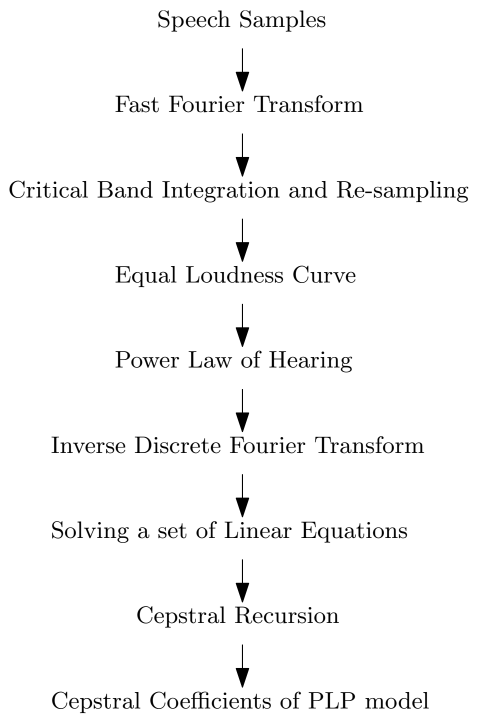

Speech Samples
Fast Fourier Transform
Critical Band Integration and Re-sampling
Equal Loudness Curve
Power Law of Hearing
Inverse Discrete Fourier Transform
Solving a set of Linear Equations
Cepstral Recursion
Cepstral Coefficients of PLP model

Figure 3.1: PLP Block Diagram

3.3.3 Critical-band Spectral Resolution

The power spectrum $P(\omega)$ obtained in the previous step is converted to the Bark frequency Ω by using the approximation:

$$\Omega(\omega) = 6ln\frac{\omega}{1200\pi} + \left[\left(\frac{\omega}{1200\pi}\right)^2 + 1\right]^{0.5}$$

where ω here is the angular frequency in rad/s. The bark-scaled spectra is then convolved with the power spectrum of the critical-band filter. The critical-band curve is given by:

$$\psi(\Omega) = \begin{cases} 0, & \Omega < -1.3, \\ 10^{2.5(\Omega+0.5)}, & -1.3 < \Omega < -0.5, \\ 1, & -0.5 < \Omega < 0.5, \\ 10^{-1.0(\Omega-0.5)}, & 0.5 < \Omega < 2.5, \\ 0, & \Omega > 2.5 \end{cases}$$

This smoothed Bark scale spectrum is then down-sampled in approximately 1 Bark interval so that the entrire analysis band is covered by an integral number of spectral bands.

3.3.4 Equal-loudness Preemphasis

Preemphasis is done based on loudness by simulating the equal-loudness curve:

$$\Xi[\Omega(\omega)] = E(\omega)\Theta[\Omega(\omega)]$$

where $\Theta[\Omega(\omega)]$ is the sampled Bark scale spectrum. Furthermore, the function $\mathrm{E}(\omega)$ computes the ranges of frequencies that are not well defined to the human ear and approximates it. This approximation is given as follows:

$$E(\omega) = \frac{[(\omega^2 + 56.8 * 10^6)\omega^4]}{[(\omega^2 + 6.3 * 10^6)^2 * (\omega^2 + 0.38 * 10^9)]}$$

This explains that the first frequency sample (Bark) and the last frequency sample (Nyquist frequency) are not sensitive to the human ear and are made equal to their nearest neighbours. Hence, at the end of this stage, $\Xi[\Omega(\omega)]$ begins and ends with frequency samples that are at the same value.

3.3.5 Intensity-loudness Power Law

This step computes the perceived loudness $\psi(\Omega)$ by approximately taking the cube root of the intensity $\Xi(\Omega)$

$$\psi(\Omega) = \Xi(\Omega)^{\frac{1}{3}}$$

The perceived loudness is a reasonable approximation of hearing the speech and the loudness is neither too loud or lower than the sensitive levels of hearing.

3.3.6 Autoregressive Modeling

This is the final stage of the PLP technique. Here the perceived loudness $\psi(\Omega)$ is approximated using the auto-correlation method of the all-pole spectral modeling. Inverse DFT is run on the above result since a few autocorrelation values are needed. Next, a Linear Predictive Coding (LPC) model is fit in and the autocorrelation coefficients are transformed to cepstral coefficients.

The cepstral coefficients provide an estimate of short-term energy as a function of frequency. It describes the changes and speed of change of the feature vector coefficients in time.

3.3.7 Order of PLP Model

Hermansky [10] says that higher the order of the PLP model, the higher the chances are that the spectrum of the all-pole PLP model reaches the auditory spectrum. Hence, a pole order of 8 and above is suitable for experimentaion. We used a pole order of 13 in our experimentations.

3.3.8 Description of Each Feature Vector

As a result of processing the speech samples using a PLP model of order 13, we obtained the 1st 13 features where the 1st feature defines the energy of the speech sample and the remaining 12 features are the basic PLP parameters. Since, in general, a speech recognizer uses the format of 39 feature vectors, we extended the 13 features to 39 features. This extension was done by taking the first- and second-order derivative using the delta function. Therefore, the next 13 features correspond to the first-order derivative or the velocity of speech sample. Furthermore, the last 13 features denote the second-order derivative or the acceleration of the speech sample.

To sum up, each speech sample of a particular accent, after feature extraction, contains 39 features and 2046 windows embedded as number of examples. For an accent group containing 100 samples, 204600 examples and 39 features are obtained.

CHAPTER 4

CLASSIFICATION

After extracting meaningful features from the speech utterances, the next step is to classify the accents. Classification can be either supervised or unsupervised. Supervised classification involves learning from the labeled training data whereas unsupervised classification refers to the problem of identifying a structure in the unlabeled data.

Previously, in the area of accent classification, several classification techniques like Hidden Markov Models (HMM), Support Vector Machines (SVM), Gaussian Mixture Models (GMM), Artificial Neural Networks (ANN), k-Nearest Neighbors (KNN), decision trees, and Naive Bayes algorithms have been used. Apart from these statistical classifiers, researchers have also experimented with classifying accents by modeling acoustic features using a combination of Maximum Likelihood Linear Regression (MLLR), and Maximum-A-Posteriori (MAP).

We chose to experiment with supervised classification using SVM, GMM, and DAGSVM, since they were found to improve the performance of classification using phonetic parametric features. This section provides an overview of Binary and Multiclass classification followed by an explanation of each of the classification techniques that were used in the experiments.

4.1 Supervised Binary and Multiclass Classification

Binary classification is a problem of classifying items in data into two classes. This comparison is made by a hyperplane and checking onto which side of the hyperplane the unseen test data falls.

Multiclass classification is an extension to binary classification, with the difference being $k > 2$ choices must be made where k is the number of classes. There are two popular approaches to multiclass classification: One Versus All (OVA) and All Versus All (AVA).

4.1.1 One Versus All Approach (OVA):

Given $k-$classifiers, C$_1$, C$_2$, C$_3$, . . . , C$_k$, each classifier C$_i$ sees the entire training data and gets all the examples with labels belong to class i as positive. The rest of the examples are seen by this particular classifier as negative. During the testing phase, if a data point is predicted to belong to class i, then the classifier C$_i$ gets a vote. If it is predicted that this data point does not belong to class i, then all the other classifiers excluding C$_i$ gets a vote.

4.1.2 All Versus All Approach (AVA):

This approach is also called an all-pairs approach as it considers every pair of classes in the classification stage. Consider a classifier C$_{ij}$ that classifies data into two classes i and j. This classifier gets all the examples that belong to class i as positive and all the examples belonging to class j as negative. This classifier is trained on all pairs of classes and in the testing phase, for a given test point, C$_{ij}$ determines whether the test point belongs to class i or class j. If a positive prediction is made, then class i gets a vote else class j gets a vote. In this process, $\binom{k}{2}$ classifiers are trained and run in the testing phase. Among these classifiers, the class that has the majority of votes denotes the class to which the test point belongs.

4.2 Support Vector Machine - SVM

Support Vector Machine is one of the most popular supervised classification algorithms. It refers to the problem of classifying data into classes by using the hyperplane - maximum margin principle.

Given training data $X = \{x_1, x_2, \ldots, x_n\}$ and labels $Y = \{y_1, y_2, \ldots, y_n\}$ where Y ∈ {-1 , 1}, and we have a hyperplane linear classifier that is defined by two parameters, weight w and bias b, as shown in Figure 4.1

The main goal of the SVM is to learn w and b that maximizes the margin between the two classes. The prediction rule is given by:

$$y = sign(w^\top x + b)$$

An assumption is made on the hyperplane such that: For $y_i = +1$,

$$w^\top x_i + b \geq 1$$

and for y_i = -1,

$$w^\top x_i + b \leq -1$$

The above assumptions can be visualized in Figure 4.2.

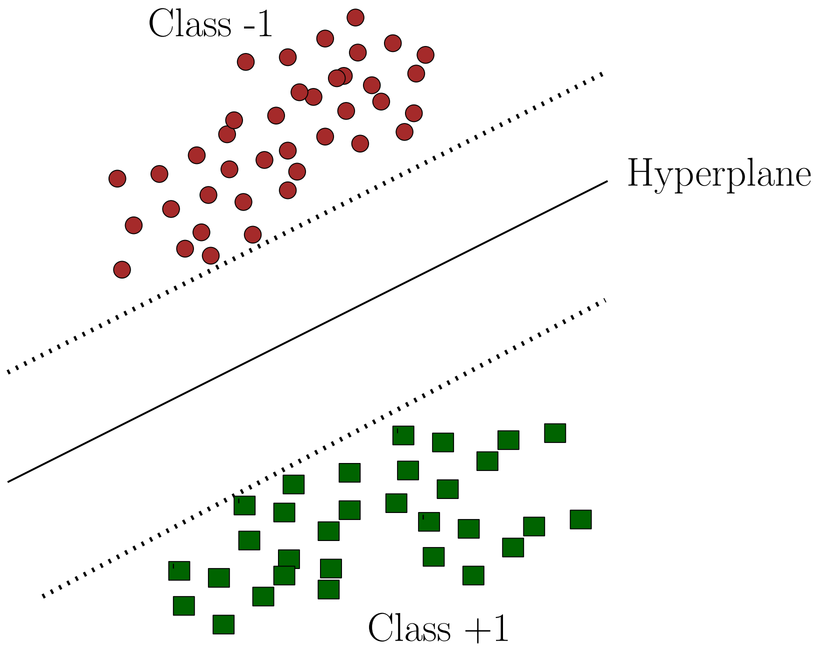


Figure 4.1: Support Vector Machine: Maximum margin principle

The assumptions can be rewritten as:

$$y_i(w^\top x_i + b) \geq 1 \; for \; all \; 1 \leq i \leq n$$

$$\implies \min_i |w^\top x_i + b| = 1$$

We want to maximize this margin by some quantity by minimizing the w parameter. The optimization problem of $\min ||w||$ in (w,b) can be solved by finding the hyperplane's margin γ:

$$\gamma = \frac{\min_i |w^\top x_i + b|}{||w||}$$

$$\implies \gamma = \frac{1}{||w||}$$

Therefore, SVM classifies the data to classes by finding the maximum margin which is nothing but minimizing the norm (w).

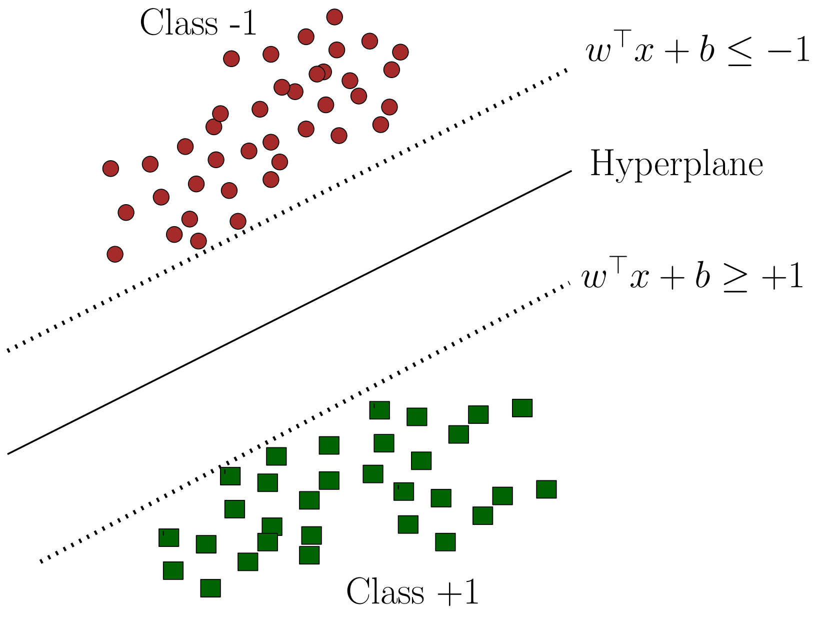


Figure 4.2: Assumptions on hyperplane

In our experiments, we used LibSVM [13] which is an integrated software for performing supervised classification using SVM. It also helps in multiclass classification.

4.2.1 Learning with Imbalanced Data

Imbalanced data occurs when the number of data points in classes differ from each other by a great amount. It can lead to misclassification as the class with a larger number of data points tends to be the preferred class. This results to an inaccurate classification. The speech samples that we received were greatly imbalanced among accents. This problem was solved by using the weights option $-w$ in LibSVM. This follows the concept of the SVM soft-margin optimization problem [41] which includes minimizing the penalties associated with misclassifications along with maximizing the margin.

The SVM soft-margin optimization problem is defined as:

$$\min_{i} \frac{1}{2}||w||^2 + C\sum \epsilon_i$$

such that

$$y_i(w^\top x_i + b) \geq 1 - \epsilon_i$$

where ϵ is a slack variable and if ϵ is found in the range of 0 to 1, then the data point is on the correct side of the hyperplane and lies near the margin. Otherwise, if ϵ is greater than 1, then the data point is misclassified. C is the regularization parameter that is used as a misclassification cost. In the case of imbalanced classes, Ben-Hur and Weston [42] explained that the C parameter could be expressed as below:

$$C\sum_{i=1}^{n}\epsilon_i - > C_+\sum_{i\in I_+}\epsilon_i + C_-\sum_{i\in I_-}\epsilon_i$$

where C_+ is the soft-margin constant for the examples that belong to the positive class and I_+ denotes the set of examples in the positive class. Similary, C_- is the soft-margin constant for the examples in the negative class and I_- denotes the set of examples in the negative class.

Ben-Hur and Weston [42] assumed that the number of misclassified examples in each class is proportional to the number of examples that each class contains. Based on this assumption, the value of C_+ and C_- is found by taking the ratio of the number of examples in each class as follows:

$$\frac{C_+}{C_-} = \frac{n_-}{n_+}$$

where n_+ and n_- are the number of examples in positive and negative classes, respectively.

4.3 Gaussian Mixture Models - GMM

This is one of the approaches that has been widely used in accent classification and speech recognition.

A Gaussian Mixture Model is a weighted sum of k-Gaussians denoted by the below equation:

$$p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(x|\mu_i, \Sigma_i)$$

where w_i is the probability of the i^{th} Gaussian and in general, the weights of all Gaussians sum up to 1.

$$\sum_{i=1}^{k} w_i = 1$$

and X = $\{x_1, x_2, \ldots, x_n\}$ is a D-dimensional data vector drawn from a possible Gaussian distribution. $\mathcal{N}(x|\mu_i, \Sigma_i)$ for $1 \le i \le k$ are the Gaussian densities of each component i.

Each component density is a Gaussian function of the form:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{\Sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\Sigma^2}$$

where μ is the mean and Σ is the covariance of the Gaussian.

The goal of a GMM is to estimate the parameters μ and Σ that fit the data. This estimation is done by training the GMM with the Expectation-Maximization algorithm. The EM algorithm is initialized with the mean and covariance of the K-Means algorithm.

In our experiments, we used the supervised form of learning. Given labeled data, each accent is modeled as a Gaussian distribution with parameters of mean (μ) and variance (σ^2). The GMM was created using the diagonal version where the diagonal matrix for each component was stored as rows of a matrix. For each accent, a Gaussian Mixture Model (GMM) is trained. The probability of a test example belonging to each accent is calculated and the GMM that has the highest probability for the test example was found. This can be visualized in Figures 4.3 and 4.4.

4.4 Directed Acyclic Graph SVM - DAGSVM

We also experimented using DAGSVM for the purpose of comparing with LibSVM. The training phase of a DAGSVM is similar to one-against-one SVM [43].

The testing phase involves a pairwise decision between accents by constructing a tree. Starting at the root, given a test point, a pairwise SVM decision is made and either class is rejected. Depending on the result, it either moves to the left or right of the tree and continues until it reaches one of the leaves that indicates the predicted class. The testing process is shown in Figure 4.5

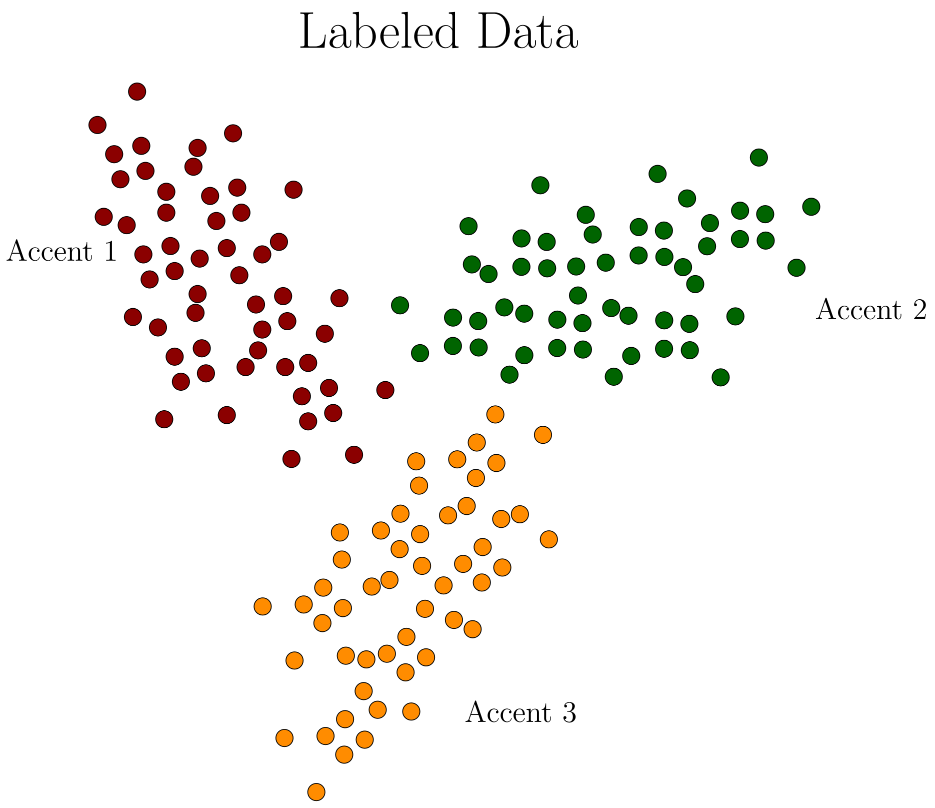


Figure 4.3: Data representation

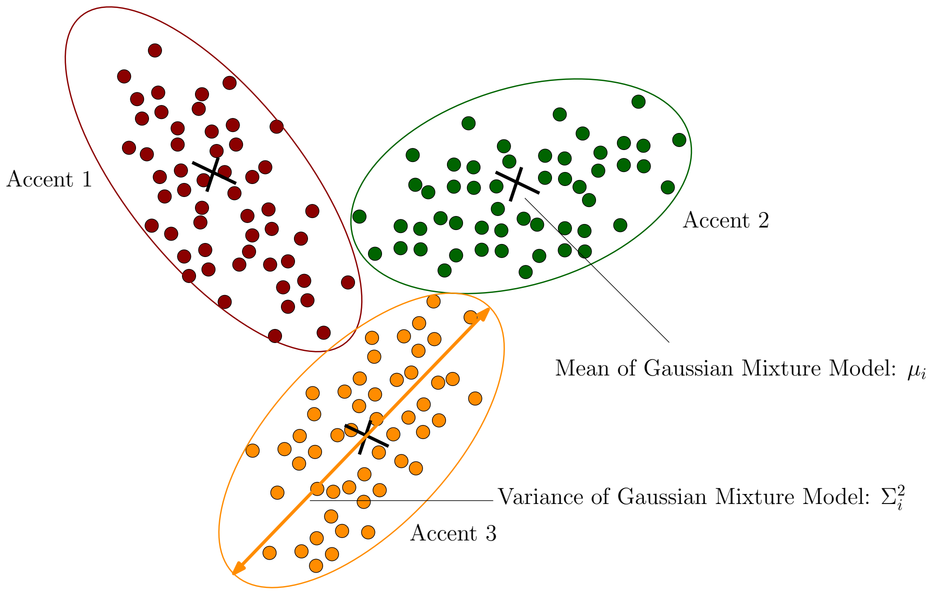

Accent 1
Accent 2
Mean of Gaussian Mixture Model: μ_i
Variance of Gaussian Mixture Model: Σ_i^2
Accent 3

Figure 4.4: Gaussian modeling of data

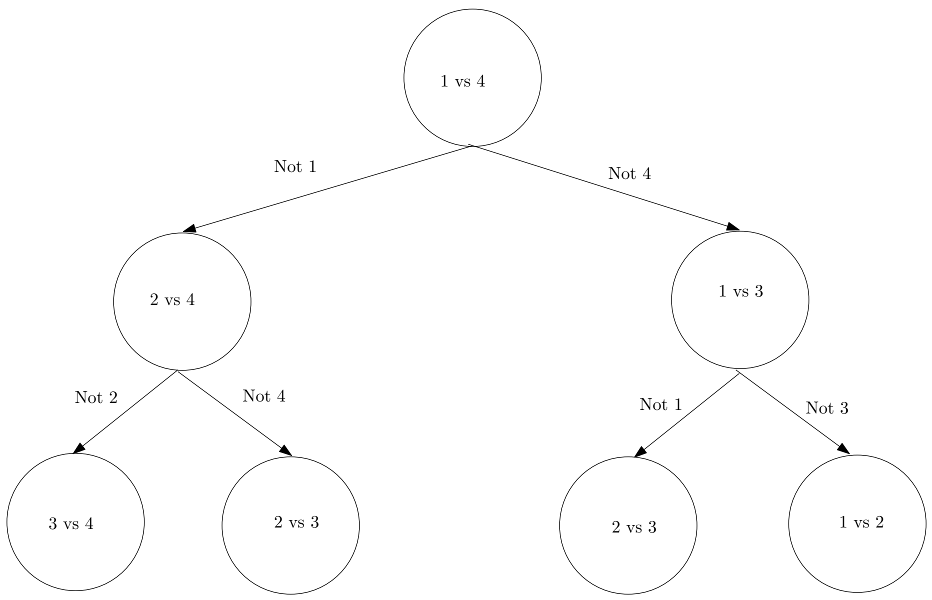


Figure 4.5: Directed Acyclic Graph SVM

CHAPTER 5

DISTANCE METRIC LEARNING

This chapter briefly describes the well-known distance metric learning algorithms, Distance Metric Learning, with application to side information [44], Pareto Discriminant Analysis [45], and Learning Distance Metric from Relative Comparisons [46], and explains the drawbacks and introduces the proposed method that could possibly improve the classification. First we start by describing the Linear Discriminant Analysis which was used as a baseline in our experiments in the concept of applying Distance Metric Learning prior to classifying the accents.

5.1 Linear Discriminant Analysis

We experimented with the multiclass version of LDA which relates to the problem of finding a subspace that contains the class variability. An assumption is made that each class C_i has the same covariance Σ and a mean μ_i. The sample covariance of the class means μ was used to define the between class variability as:

$$\Sigma_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^\top$$

and the classes were separated by:

$$S = \frac{\vec{w}^\top \Sigma_b \vec{w}}{\vec{w}^\top \Sigma \vec{w}}$$

5.2 Distance Metric Learning, with Application to Side Information

Xing et al. [44] proposed a systematic methods for learning a distance metric using examples of similar data points. Given a data set $\{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^n$ and a set that contains pairs of *similar* data points:

$$S : (x_i, x_j) \in S \text{if } x_i \text{and } x_j \text{ are similar}$$

They considered learning a mahanalobis distance metric of the form:

$$d(x, y) = d_A(x, y) = ||x - y||_A = \sqrt{(x-y)^\top A(x-y)}$$

where A was required to be positive semidefinite, $A \geq 0$. The optimization problem to solve for A was defined as:

$$\min_A \sum_{x_i, x_j \in S} ||x_i - x_j||_A^2$$
$$\text{such that} \sum_{x_i, x_j \in D} ||x_i - x_j||_A \geq 1, A \geq 0$$

where D denotes the set of dissimilar data points. The optimization problem states that given a set of similar points, minimize the distance between these similar points such that they are close to each other while keeping the constraint that the data points belonging to different classes need to be far from each other. The positive semidefinite A is solved which denotes the metric that can be learned for transforming the data points to a new feature space.

Xing et al. [44] experimented with learning *diagonal* $A = diag(A_{11}, A_{22}, \dots, A_{nn})$ by solving the Newtons Raphson method:

$$\begin{aligned} g(A) &= g(A_{11}, A_{22}, \dots, A_{nn}) \\ &= \sum_{x_i, x_j \in S} ||x_i - x_j||_A^2 - \log\left(\sum_{x_i, x_j \in D} ||x_i - x_j||_A\right) \end{aligned}$$

given that $A \geq 0$.

They showed that it is efficient to optimize g using the Newtons Raphson method and that minimizing g is equivalent to solving the optimization problem up to a multiplication of A by a positive constant.

They also experimented learning a $full\ matrix\ A$ by find the gradient descent and projecting the data iteratively such that the similar points are closer to each other. They formulated the following optimization problem:

$$\max_A g(A) = \sum_{x_i, x_j \in D} ||x_i, x_j||_A$$
$$\text{such that } f(A) = \sum_{x_i, x_j \in S} ||x_i - x_j||_A^2 \leq 1$$
$$A \geq 0$$

This optimization problem states that the distance between dissimilar data points should be maximized such that they are far apart while performing iterative projections such that the distance between similar data points is minimized.

5.3 Pareto Discriminant Analysis

Abou Moustafa et al. [45] proposed Pareto Discriminant Analysis to solve the problems faced by LDA (Linear Discriminant Analysis). They came up with an idea to separate the data points in different classes by an equal distance; i.e., the distances between all classes are equal.

Kullback-Leibler Divergence (KLD) was used as a measure of separating two Gaussians; based on this, an objective function was formulated to maximize the distance between two classes. Multimodal Oriented Discriminant Analysis (MODA) [47] formulation based on symmetric KLD was used to define the appropriate objective function.

They defined the KLD for two classes C_i and C_j as considering the difference in mean and covariance:

$$KL(i||j) = tr(\Sigma_i \Sigma_j^{-1} + \Sigma_i^{-1}\Sigma_j - 2I) + (\mu_i - \mu_j)^\top (\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j)$$

A linear transformation B is learned using MODA such that the KLD is maximized between the two classes:

$$\text{Class } C_i \text{ with distribution } \mathcal{N}(\mu_i, \Sigma_i) \text{ is transformed to } \mathcal{N}(B^\top \mu_i, B^\top \Sigma_i B)$$
$$\text{Class } C_j \text{ with distribution } \mathcal{N}(\mu_j, \Sigma_j) \text{ is transformed to } \mathcal{N}(B^\top \mu_j, B^\top \Sigma_j B)$$

They showed that Pareto Discriminant analysis outperformed the current methods like LDA.

5.4 Learning Distance Metric from Relative Comparisons

Schultz and Joachims [46] proposed a method for learning a distance metric that uses the relative information among classes such as: example x_i is closer to example x_j than to example x_k where $x_i, x_j, x_k \in X_{train}$ which is a set of training examples. A parametrized distance metric is learned by employing kernels which estimates the metric A. If a kernel is defined as $k(x, y) = \phi(x)\phi(y)$, and Φ is the feature space where the training example is projected using the function $\phi(x)i$, then the following is the distance metric in the feature space.

$$\begin{aligned} d_{\Phi,W}(\phi(x),\phi(y)) &= \sqrt{((\phi(x)-\phi(y))^\top \Phi)W(\Phi^\top(\phi(x)-\phi(y)))} \\ &= \sqrt{\sum_{i=1}^{n} W_{ii}(K(x,x_i)-K(y,x_i))^2} \end{aligned}$$

The idea behind using kernels is to separate the nonlinear data in the feature space before applying the distance metric learning algorithm.

5.5 Conclusion

Distance metric learning, with side information [44], looks at just maximizing the distance between data points in different classes and minimizing the data points in the same class. It does not look at separating the class as a whole and the maximum, minimum distance is not given in specific terms.

Pareto Discriminant analysis does not look at minimizing the data points in a class. Instead it concentrates at only maximizing the distance between classes. There are cases where the data points in a class might not be as close as possible, so the assumption, that data points in the same class are close to each other, cannot be made. In this case, if the constraint on minimizing the distance between data points is not considered, then the classifier might not perform well.

Learning the distance metric from relative comparisons [46] does not explain clearly how a new test point is transformed to the feature space after applying the kernel function.

As a result, we looked at the missing components of the above existing methods and proposed a method which considers a refined way of learning a distance metric, taking into account minimizing the distance between examples within the same class while maximizing the distance between classes by a specified distance. We also looked at applying distance metric learning to kernels where we approximate the kernels. The proposed method is explained in detail in Chapter 6.

CHAPTER 6

RESEARCH CONTRIBUTIONS

This chapter is organized as follows. First, an overview of the existing system is presented. Next, we explain the proposed system consisting of 3 phases followed by experiments. Finally, a comparison study between the existing and proposed system is explained.

6.1 Existing System

6.1.1 System Design

Most of the accent classification systems are composed of two components wherein first features are extracted from the raw input speech signal using the feature extraction techniques, as explained in Chapter 2. These features are then classified using several classifiers, as described in Chapter 3. See Figure 6.1

6.1.2 Drawbacks

The data after feature extraction, in our case using Perceptual Linear Prediction (PLP), consist of 2046 windows and 39 features for a single speech utterance. The concept of windows is explained in the next section. These 2046 windows are embedded into the examples; thus, a single speaker is comprised of 2046 examples and 39 features, in terms of machine learning.

For an accent containing 200 speakers, the resulting data would contain 200 * 2046 examples and 39 features. This does not seem like large data but performing multiclass classification can be time- and memory-consuming. Hence, Watanaprakornkul et al. [16] experimented by *randomly* choosing 10 windows for each speaker. We feel that this is not an optimal solution as windows describe each sample of a speech utterance and it is important that the description is in a flow in accordance to the speech sample.

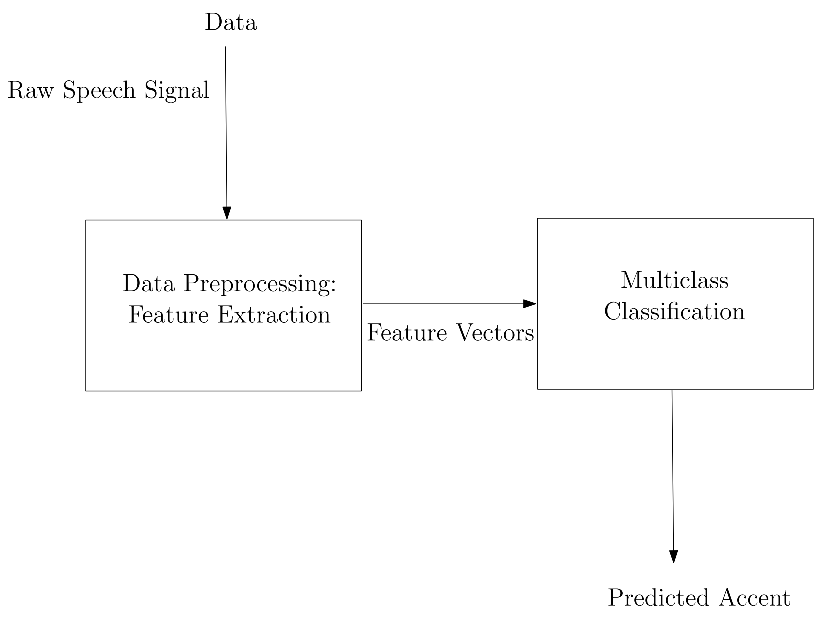

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Feature Vectors
Multiclass Classification
Predicted Accent

Figure 6.1: Existing accent classification system

6.2 Proposed System: Phase 1

6.2.1 System Design

The first phase of the proposed system design is shown in Figure 6.2. After the features are extracted from the raw speech input signal, algorithms are applied to reduce the number of windows. With the reduced amount of speech data, multiclass classification methods are run to predict the accent.

6.2.2 Concept of Windows

Initially, a raw speech input signal, i.e., the audio of each speaker that we can hear directly, is made up of a number of samples. In our case, we estimated this number of samples, for each input signal of 20ms, to be 163890 samples. Now we want to find the frequency contained in each sample for the purpose of extracting features. Since 163890 samples is really large, it is infeasible to calculate the frequency contained in each sample.

Hence, in the feature extraction stage, a bunch of these samples are taken and the fourier transform is calculated for each group rather than each sample. This makes the computation of the transform easier. This group constitutes a *frame* or *Window.*

6.2.3 Reducing Number of Windows

In our experiments, after applying Perceptual Linear Prediction feature extraction on the data, we obtained 2046 windows for each speaker. The drawbacks of classifying accents with each speaker containing 2046 windows were explained in Subsection 6.2.2. To reduce

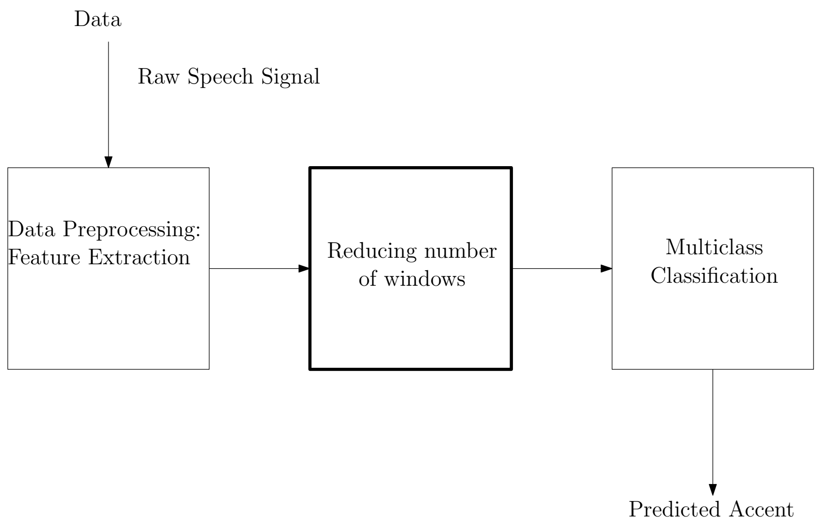

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Reducing number of windows
Multiclass Classification
Predicted Accent

Figure 6.2: Proposed accent classification system - Phase 1

the amount of data that we are dealing with, we proposed two methods: K-Means and column sampling.

6.2.3.1 K-Means

K-Means is one of the popular clustering algorithms. It takes as input the data $X = \{x_1, x_2, \ldots, x_n\}$ where $x_i \in \{\mathbb{R}^D\}$ and k which defines the number of groups into which the data should be partitioned.

Initially, we want to partition the data to k groups, so we define the k-centers as $\mu_1, \mu_2, \ldots, \mu_k$. These k-centers are randomly intialized in $\mathbb{R}^D$.

The below steps are repeated until the algorithm converges, i.e., until the cluster centers remain the same through all the iterations.

Each example x_i is assigned to its closest cluster center. Let C_k be the set of examples closest to the the cluster center μ_k, then

$$C_k = argmin||x_i - \mu_k||^2$$

The new cluster centers are then computed by taking the mean of the set C_k

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

In the feature extraction stage, after extracting the cepstral coefficients and the 2046 windows, K-Means is run to obtain 10 windows for each speaker in a particular accent.

6.2.3.2 Column Sampling

Column sampling refers to the problem of selecting k columns that represents a subspace of the input data. Let $C = \{c_1, c_2, \ldots, c_{2046}\}$ be the set of columns for each speaker, and in this case, the columns are the windows which we want to sample.

For each column c_i for $1 \leq i \leq 2046$, the squared norm *snorm* is calculated as:

$$snorm = norm(c_i)^2$$

Finally, k columns proportional to the value snorm, for each speaker after extracting the cepstral coefficients, are selected.

6.2.4 Experiment Results - Classification Using LibSVM

The results after applying algorithms to reduce the number of windows and classifying with 13, 39, and 54 features are shown in Table 6.1. We observed that K-Means as a clustering method to reduce number of windows gave better results than column and random sampling. Also we carried out experiments using Perceptual Linear Prediction (PLP) as a feature extraction method and compared the accuracy with the previous studies which used Linear Predictive Coding (LPC) as the feature extraction method. Hermansky [10] in his study of analyzing speech using Perceptual Linear Predictive (PLP) claimed that PLP could be useful in speaker-independent ASR since it can represent the linguistic information, that is present in speech, better than LPC. We chose to compare the accuracy obtained using PLP and LPC to see which method performs better as a feature extraction technique.

6.2.5 Experiment Results - Classification Using Gaussian Mixture Models (GMM)

The results after applying algorithms to reduce the number of windows and classifying with GMM using 13, 39, and 54 features are shown in Table 6.2. We observed that the previous study [16] used 35 Gaussians as the best number of Gaussians for each accent. We experimented with 40 Gaussians to see if we can achieve a better accuracy. We also compared the overall accuracy obtained using GMM with the accuracy obtained using LibSVM. The results show that using GMM, we get a lower accuracy of classifying accents when compared with LibSVM. The reason we chose 10 windows is just for comparing the performance of classifiers with the previous study [16] where 10 windows were randomly sampled.

Table 6.1: Classification of accents using LibSVM. 2-accents (Mandarin and German), 3-accents (Cantonese, Hindi, and Russian), 12-accents (Portuguese, Cantonese, Farsi, French, German, Hindi, Hungarian, Italian, Japanese, Mandarin, Russian, and Swedish)

Accents	**Reduction to get 10 windows**	**13 features**	**39 features**	**52 features**	**Accuracy as in previous studies**
2-accents	Kmeans	52.20%	60.61%	60.32%	57.12% (using LPC) [15]
	Column Sampling	53.21%	52.31%	51.40%	
	Random	52.91%	51.40%	51.10%	
3-accents	Kmeans	45.28%	45.35%	46.69%	41.18% [16]
	Column Sampling	45.28%	45.28%	37.38%	
	Random	45.28%	44.92%	40.34%	
12-accents	Kmeans	13.11%	15.13%	16.03%	13.32% [15]
	Column Sampling	13.11%	13.36%	9.66%	
	Random	13.11%	13.19%	10.76%	

Table 6.2: Classification of accents using GMM. 2-accents (Mandarin and German), 3-accents (Cantonese, Hindi, and Russian), 12-accents (Portuguese, Cantonese, Farsi, French, German, Hindi, Hungarian, Italian, Japanese, Mandarin, Russian, and Swedish)

Accents	**Reduction to get 10 windows**	**13 features**	**39 features**	**52 features**	**Accuracy as in previous studies**
2-accents	Kmeans	54.51%	48.70%	51.30%	
	Column Sampling	53.31%	51.20%	52.81%	
	Random	53.11%	50.00%	52.30%	
3-accents	Kmeans	42.88%	43.79%	39.77%	37.5% for 35 Gaussians [16]
	Column Sampling	35.90%	45.28%	38.65%	
	Random	36.39%	29.48%	27.64%	
12-accents	Kmeans	11.36%	12.38%	11.46%	
	Column Sampling	9.38%	10.38%	9.72%	
	Random	9.93%	8.72%	9.13%	

6.2.6 Experiment Results - Comparison between Directed Acyclic Graph SVM (DAGSVM) and LibSVM with FAE and GMU Data

Table 6.3 shows the result of classifying accents using DAGSVM and LibSVM. Here we chose K-Means as a technique to reduce the number of windows and we experimented with 39 features. We also compared the accuracy of classification on FAE and GMU datasets. The results show that LibSVM performed with a better accuracy than DAGSVM in the case of 3-accents and 12-accents, whereas DAGSVM performed better while classifying 2-accents.

6.3 Proposed System: Phase 2

Apart from rating which classifier gives the best performance, we also observed the behavior among the accents based on their accuracy. We classified certain accents using LibSVM. Table 6.4 shows the accuracy between pairwise accents. We observed that maybe the accents that are close to each other had a lesser accuracy than the accents that are far apart. For example, we know that a speaker with Hindi accent would sound the same as a speaker with a Tamil accent. These two accents are known to be closer to each other and we can see from the results that it has the lowest accuracy whereas the speech spoken by an Arabic accented speaker varies greatly with the speaker of Portuguese accent. According to the results, these two accents are detected with the highest accuracy. This introduces the idea of learning new feature transformations applying the principle of distance metric learning that will be explained in the next Subsection, 6.3.2.

6.3.1 System Design

Figure 6.3 shows the system design that involves the second phase. Based on our observations explained earlier, we wish to pull apart different accents while keeping the data points within the same accent closer to each other. This could lead to a better classification and is described as learning new feature transformations. Once the features are extracted and the number of windows are reduced, new feature transformations could be learned by applying the Distance Metric Learning (DML) concept. On the new feature transformations, multiclass classification methods are run to predict the accent.

Table 6.3: Comparison between DAGSVM and LibSVM with FAE and GMU accents. 2-accents (Arabic and Brazilian Portuguese), 3-accents (Cantonese, Hindi, and Russian), 12-accents (Portuguese, Cantonese, Farsi, French, German, Hindi, Hungarian, Italian, Japanese, Mandarin, Russian, and Swedish)

Accents	**Data**	**LibSVM Accuracy**	**DAGSVM Accuracy**
2-accents	FAE	70.58%	79.98%
	GMU	60.83%	60.10%
3-accents	FAE	44.15%	43.86%
	GMU	47.18%	52.11%
12-accents	FAE	14.79%	12.21%
	GMU	17.09%	9.42%

6.3.2 Distance Measure on Phonetic Transcriptions

The first step in learning new feature transformations was to choose a good distance measure that would be able to distinguish between accents. Previous studies have shown that based on geographical locations, accents could be separated. However, we wanted to define a specific measure that would be able to tell us that accent1 is far from or near to accent2 by a specific distance.

Charlotte Gooskens [48] has proven phonetic distance to be a reliable source for distinguishing between dialects and/or languages. This can be extended in the area of differentiating accents.

With the above reference, we proposed a method to find distance between accents by estimating the edit-distance between phonetic strings of pairwise accents. This edit distance is defined as a distance measure by which the accents need to be separated. This problem has not been researched previously.

The phonetic strings for each accent were taken from the GMU speech accent archive.

Table 6.4: Classified using LibSVM on FAE dataset, with KMeans as clustering technique

Accents	**LibSVM**
Arabic and Brazilian Portuguese	70.58%
Japanese and Hindi	69.7%
Hindi and Tamil	56.13%

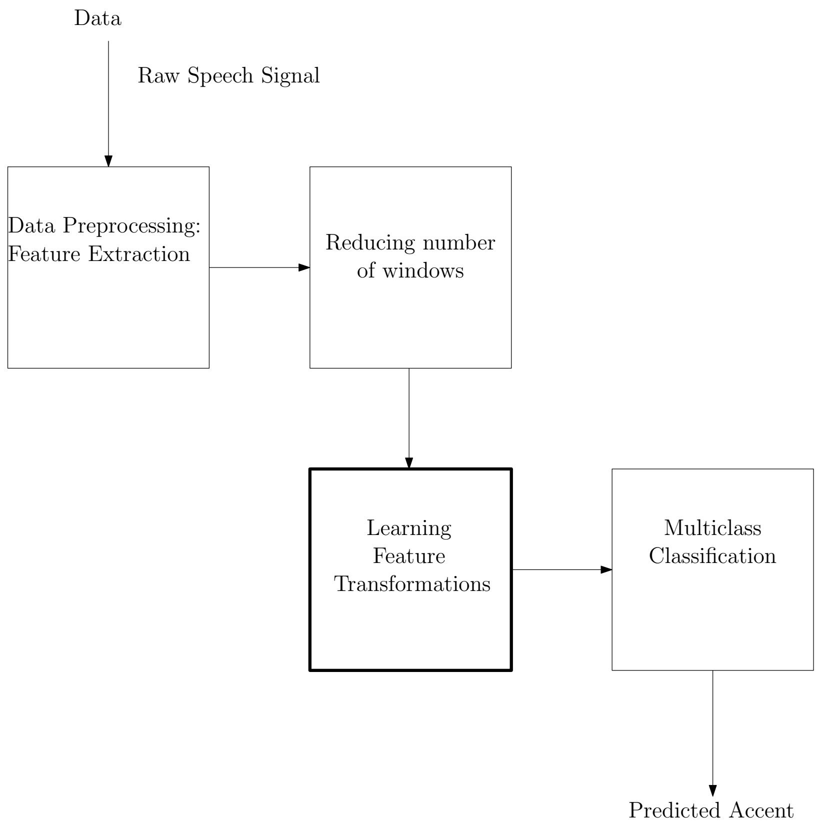

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Reducing number of windows
Learning Feature Transformations
Multiclass Classification
Predicted Accent

Figure 6.3: Phase 2 of the proposed accent classification system

The phonetic strings are depicted in Figures 6.4 and 6.5.

The GMU speech accent archive [39] provides phonetic transcription of speech spoken by males and females. For a pair of foreign accents, the edit distance between their respective phonetic transcriptions is found. Since the phoneme varies slightly between male and female accents, we have considered the distance between a pair of accents to be the mean of edit distance between the male and female phonetic transcriptions of those two accents.

The edit distance d_{mn} for every pair of accents results in a dissimilarity matrix. Let a_m and a_n denote two classes of accents. We wish to learn a distance metric such that distance between data points belonging to accent a_i where $i \in \{m, n\}$ is minimized and the data points of different accents a_m and a_n are separated from each other based on the distance d_{mn} between a_m and a_n. This is illustrated in Figure 6.6.

6.3.3 Concept of Distance Metric Learning

Given a set of pairwise constraints, distance metric learning aims to find a distance matrix, A, such that the distance between dissimilar pairs is greater and distance between similar pairs is lesser. There are several distance metric learning algorithms, as explained in Chapter 5, and among them, we implemented Linear Discriminant Analysis (LDA) as a baseline to see if it performed better than LibSVM.

6.3.4 Experiment Results - Comparison between LibSVM and LDA

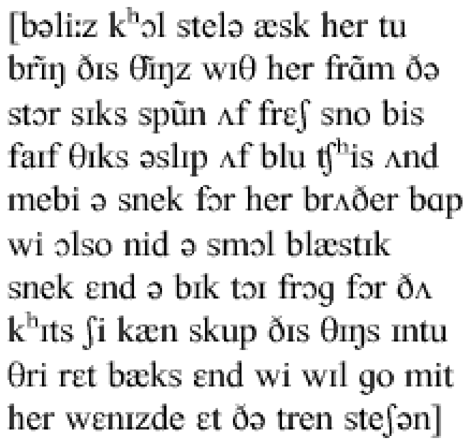
Table 6.5 shows the results on classifying accents using LibSVM and LDA. Since LDA has the same principle of maximizing the distance between data points of different classes while minimizing data points belonging to the same class, we compared LDA with LibSVM to see if LDA performs better. As per the results, LDA fails to get a higher classification accuracy and this resulted in finding a transformation matrix that transforms the points as per the constraints. This is explained in detail in Subsection 6.3.5.

6.3.5 Applying Distance Metric Learning

Given: Edit distance d$_{mn}$ based on phonetic strings between classes (accents). A labeled data matrix, X of a examples and b features.

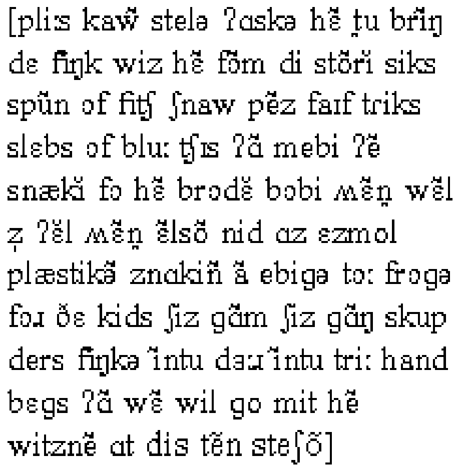
To find: transformation matrix A that transforms the data matrix X such that the interclass distance is equal to the distance in the distance matrix, d$_{mn}$, and the intraclass distance is minimized such that it is equal to ϵ where ϵ lies between 0 and 1.

Formulation:



[bəliːz kʰɔl stelə æsk her tu
brĩŋ ðɪs θĩŋz wɪθ her frãm ðə
stɔr sɪks spũn ʌf frɛʃ sno bis
faɪf θɪks əslɪp ʌf blu tʃʰis ʌnd
mebi ə snek fɔr her brʌðer bɑp
wi ɔlso nid ə smɔl blæstɪk
snek ɛnd ə bɪk tɔɪ frɔg fɔr ðʌ
kʰɪts ʃi kæn skup ðɪs θɪŋs ɪntu
θri rɛt bæks ɛnd wi wɪl go mit
her wɛnɪzde ɛt ðə tren steʃən]

Figure 6.4: Phonetic transcription of Arabic accent



[pliːs kaw̃ stela ʔaskə hẽ t̪u brĩŋ
dɛ fĩŋk wiz hẽ fõm di stõrĩ siks
spũn of fitʃ ʃnaw pẽz faɪf triks
slɛbs of bluː tʃɪs ʔã mebi ʔẽ
snækĩ fo hẽ brodẽ bobi ʍẽn̪ wẽl
z̩ ʔẽl ʍẽn̪ ẽlsõ nid az ɛzmol
plæstikã znakiñ ã ebigə toː frogə
foɹ ðɛ kids ʃiz gãm ʃiz gãŋ skup
ders fĩŋkə ˈintu dəɹˈintu triː hand
bɛgs ʔã wẽ wil go mit hẽ
witznẽ at dis tẽn steʃõ]

Figure 6.5: Phonetic transcription of Portuguese accent

Let X = $\{x_1, x_2, .., x_n\}$ be a collection of data points where n is the number of examples and each $x_i \in R^D$ is a vector of D features.

Let the set of equivalence constraints be denoted by:

S = $\{(x_i, x_j) \mid$ x_i and x_j belong to the same class$\}$

D = $\{(c_m, c_n) \mid$ c_m and c_n are the centroids belonging to different classes$\}$

We wish to learn a matrix 'A' such that the below conditions are satisfied

$$\min_{A \epsilon R^{DxD}} \sum_{(x_i, x_j \epsilon S)} ||x_i - x_j||_A^2$$

$$\text{such that } ||c_m - c_n||_A^2 = d_{mn}$$

$$\text{and } A \geq 0$$

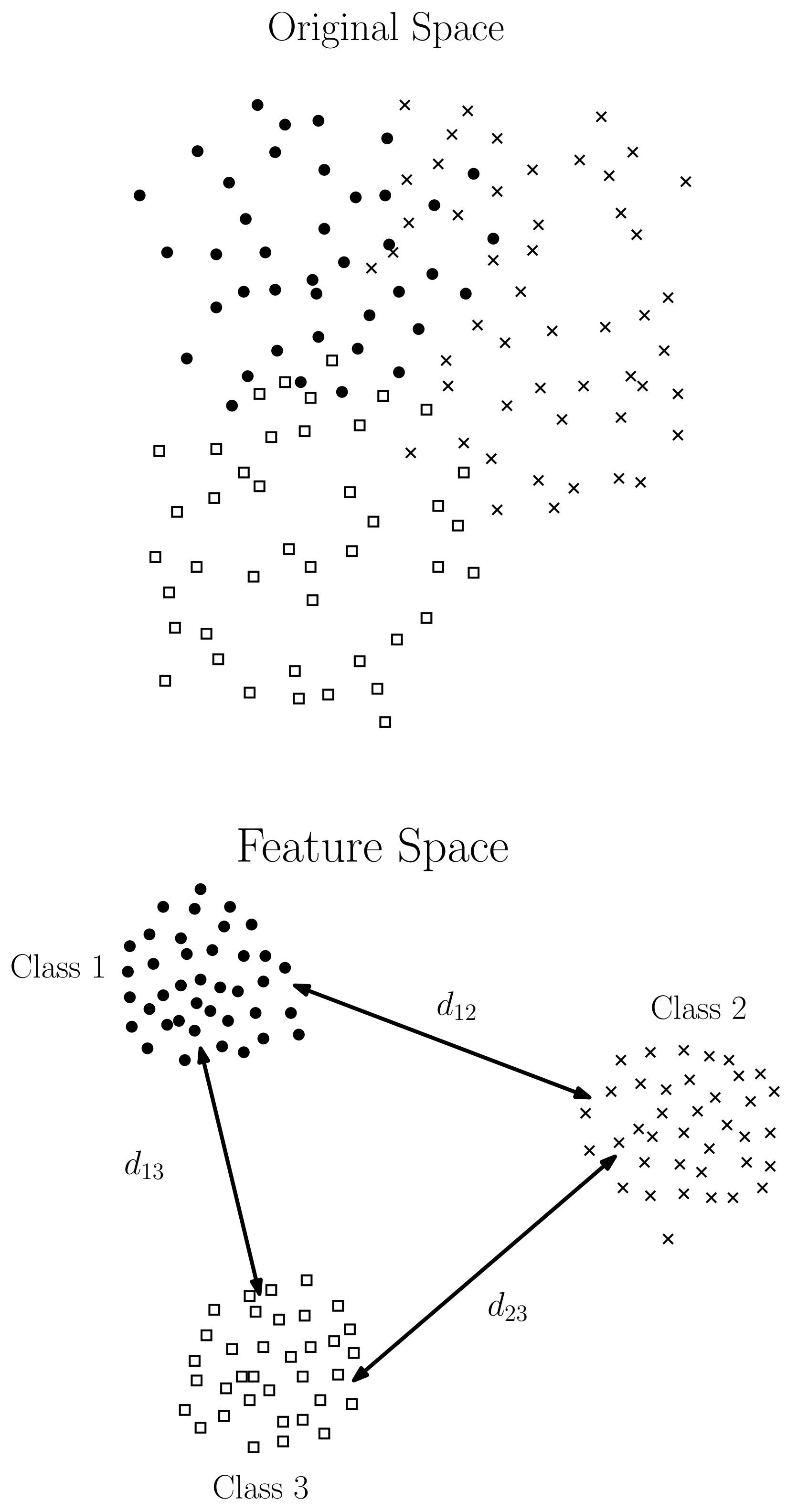


Figure 6.6: Illustration of the proposed problem

Table 6.5: Comparison between LibSVM and LDA

Accents	**Data**	**LibSVM Accuracy**	**LDA Accuracy**
Arabic and Brazilian Portuguese	FAE	70.58%	65.88%
Cantonese, Hindi and Russian	FAE	45.35%	40.41%
12 way (Portuguese, Cantonese, Farsi, French, German, Hindi, Hungarian, Italian, Japanese, Mandarin, Russian, Swedish)	FAE	14.79%	11.92%

A is a positive semidefinite matrix and the above formulation is a semidefinite programming problem.

Let X denote the data points in original space, and A is the positive semidefinite matrix that transforms the data points in original space to a new feature space based on the above formulation. Let B denote the data points in the new feature space.

In the training phase, we can depict the above in the following equation:

$$X * A = B$$

Since A is positive semidefinite, $A = U^\top U$, $||x_i - x_j||_A^2$ can be written as:

$$\begin{aligned}
||x_i - x_j||_A^2 &= (x_i - x_j)A(x_i - x_j)^\top \\
&= (x_i - x_j)U^\top U(x_i - x_j)^\top \\
&= (U^\top x_i - U^\top x_j)(U^\top x_i - U^\top x_j)^\top \\
&= (b_i - b_j)(b_i - b_j)^\top \\
&= ||b_i - b_j||^2
\end{aligned}$$

where $b_i = U^\top x_i$

This implies that a new test point can be classified using the above transformation. Hence, let X' be a new test point, then $X' * U = B'$, denotes the transformation of the new test point where B' is the feature space for the new test point.

The proposed distance metric learning problem was experimented with on the FAE dataset and binary classification was run using LibSVM. Table 6.6 in Subsection 6.3.6 compares the resulting accuracy of detecting accents using LibSVM and the proposed method. It can be inferred that in 2 cases, Arabic and Portuguese accents were classified by a large margin of 8% compared to LibSVM. Hindi and Tamil accents were classified with an accuracy of 66.25% which is 10% more than the accuracy obtained using LibSVM. However, there was no change in the accuracy while classifying Japanese and Hindi and there was a drop of 4% in classifying Mandarin and German accents. We are not sure what led to this decrease in accuracy.

6.3.6 Experiment Results - Classification Using Proposed Distance Metric Learning

We experimented classifying accents based on learning a distance metric over phonetic strings and the results are shown in Table 6.6. We chose to experiment on a FAE dataset using K-Means to reduce the number of windows. We compared the classification performance using the proposed distance metric learning and LibSVM. We found that for some accents, learning a distance metric over phonetic strings gives better classification results.

6.4 Proposed System: Phase 3

We chose to approximate the kernels in feature space so that a distance metric can be learned on the data based on equivalent and inequivalent criteria. As described in Chapter 5, equivalent criteria refers to a set of points within the same class and inequivalent criteria refers to a set of points belonging to different classes.

Xing et al. [44] proposed a method for learning a relative distance metric using kernels. The main idea behind using kernels is that data could be nonlinearly separable and in such cases, applying distance metric learning could cause an overlap of classes while trying to separate them. Kernels help in pulling apart data points towards their classes, thereby making it easier to separate the classes.

However, [44] does not clearly explain how a new test point is transformed to the feature space after applying a kernel function and learning a distance metric. Hence, we came up with another idea to include kernel approximation methods prior to learning a distance metric. The reason is, if a kernel function is applied to the training data $X = \{x_1, x_2, \ldots, x_n\}$

Table 6.6: Comparison between LibSVM and learning distance metric over phonetic strings

Accents	**Data**	**LibSVM Accuracy**	**Distance metric over phonetic strings Accuracy**
Arabic and Brazilian Portuguese	FAE	70.58%	78.18%
Japanese, and Hindi	FAE	69.7%	69.36%
Hindi and Tamil	FAE	56.13%	66.25%
Mandarin and German	FAE	60.61%	56.77%

where $X \in \mathbb{R}^D$, we get a resulting kernelized data of the form $KX \in \mathbb{R}^n$. In other words, the number of features of the training kernelized data are the same as the number of training examples. Now consider testing data $TX = \{tx_1, tx_2, \ldots, tx_m\}$ where $tx \in \mathbb{R}^D$; here the resulting kernelized data would be of the form $KTX \in \mathbb{R}^m$. A distance metric on the training data is learned and is of the form $L \in \mathbb{R}^n$; in short L is a square matix of dimensions nXn. In order to transform the test kernelized data, one would have to use the distance metric L to perform the matrix multiplication between the test kernelized data and L. Since the dimensions vary as explained above, it becomes impossible to transform the test kernelized data to a new feature space using the distance metric.

Therefore, we proposed to use kernel approximation methods that take kernelized data and approximates the features such that the training and test data contain the same number of features, $KX \in \mathbb{R}^k$ and $KTX \in \mathbb{R}^k$. Now, L, the distance metric learned from the training data, would be a square matrix of dimensions kXk. This would be feasible to transform the test kernelized data to a new feature space using the learned distance metric.

We considered experimenting with two existing methods, Kernel Principal Component Analysis (KPCA) and the Rahimi Recht method, which are explained in this section. This section is organized as follows: First, the system design is presented followed by an explanation of the two kernel approximation methods and experimental results obtained by classifying accents using KPCA and the Rahimi Recht method.

6.4.1 Kernel System Design

Figure 6.7 shows the third phase of the system design. Once the features are extracted and the number of windows are reduced, kernel approximation algorithms are applied to deal with nonlinear data. On the resulting data which are transformed to the kernel space, we wish to learn a distance metric over phonetic strings before classifying the accents.

6.4.2 Kernel Principal Component Analysis - KPCA

Kernel PCA [18] is similar to PCA except that the dataset $(N \times D)$ is transformed nonlinearly into a $M-$dimensional space where $M >> D$. KPCA takes each example x and applies nonlinear transformation using C which is given by the following formula:

$$C = \frac{1}{N}\sum_{n=1}^{N} \phi(x_n)\phi(x_n)^{\top}$$

where ϕ is the lifting map. The examples x live in a $D-$dimensional space and ϕ takes x from $D-$dimensional space to a higher $M-$dimensional space with the above mapping. Now we have mapping $\phi(x)$ for each point and based on this, covariance matrix is computed in the new higher dimensional space and then followed by eigen decomposition of this matrix. This matrix is not similar to the covariance matrix in PCA and is called the kernel matrix.

The eigenvector decomposition of the kernel matrix is given by:

$$\frac{1}{N}\sum_{n=1}^{N} \phi x_n \phi x_n{}^{\top} v_i = \lambda_i v_i$$

From the above equation, it can be noticed that $\phi(x_n)^{\top} v_i$ is a scalar dot product where v_i is a linear combination of $\phi(x_n)$. This implies that each of the quantities in the kernel vector is a scalar quantity; hence, v_i can be written as a scalar number of times of $\phi(x_n)$.

$$v_i = \sum_{n=1}^{N} a_{in}\phi(x_n)$$

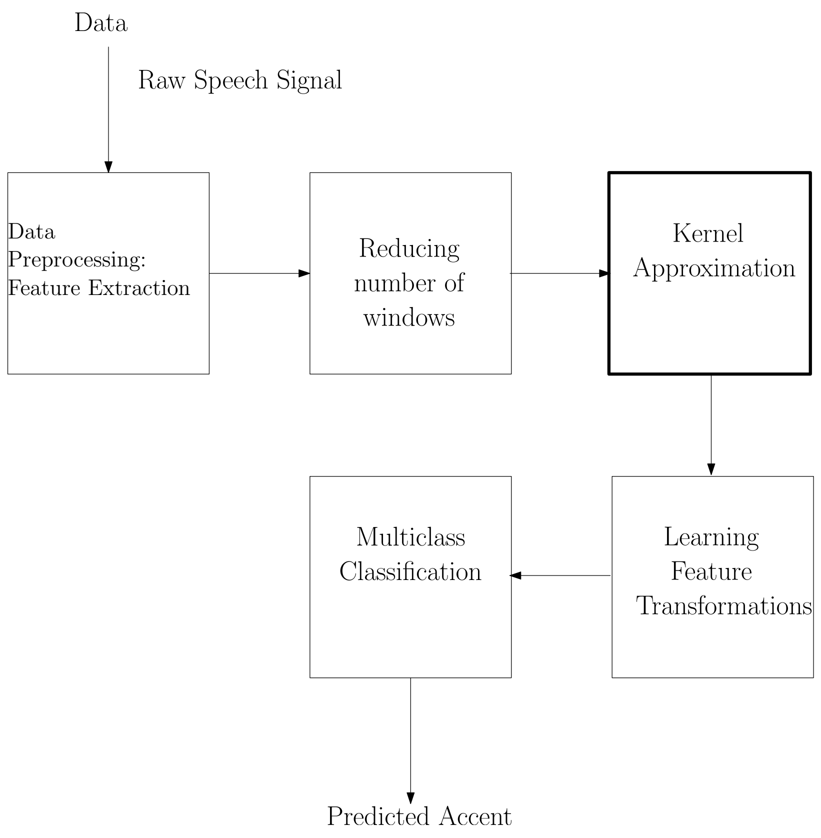

Data
Raw Speech Signal
Data Preprocessing: Feature Extraction
Reducing number of windows
Kernel Approximation
Learning Feature Transformations
Multiclass Classification
Predicted Accent

Figure 6.7: Accent classification system design using kernel approximation prior learning distance metric

where a_{in} is some coefficient which when combined with $\phi(x_n)$ gives a linear combination. Thus, plugging the above equations in 'C', we get the following eigenvector equation:

$$\frac{1}{N}\sum_{n=1}^{N} k(x_l, x_n) \sum_{m=1}^{N} a_{im} k(x_n, x_m) = \lambda_i \sum_{n=1}^{N} a_{in} k(x_l, x_n)$$

where k(x$_m, x_n$) is the NxN kernel matrix.

Using this, top K eigenvectors are retrieved and the projection here is the projection of the initial kernel matrix along with the eigenvectors of the centered kernel. Thus, KPCA can be used to nonlinearize a linear dimensionality reduction method and is used when the projections are assumed to be nonlinear.

6.4.3 Experiment Results - Classification Using Kernel Principal Component Analysis (KPCA)

We experimented classifying accents using KPCA as the kernel approximation method. Table 6.7 shows the comparison of classification accuracies among LibSVM, learning a distance metric over phonetic strings, and KPCA. We wanted to see if KPCA performs

Table 6.7: Comparison between LibSVM, Proposed Distance Metric Learning method, KPCA

Accents	**Data**	**LibSVM Accuracy**	**Proposed distance metric Accuracy**	**KPCA Accuracy**
Arabic and Brazilian Portuguese	FAE	70.58%	78.18%	59.81%
Japanese, and Hindi	FAE	69.7%	69.36%	68.42%
Hindi and Tamil	FAE	56.13%	66.25%	53.29%
Mandarin and German	FAE	60.61%	56.77%	49.60%

better as a kernel approximation method prior to learning a distance metric over phonetic strings and classification. As per the results, KPCA fails to perform better than LibSVM and the proposed distance metric.

6.4.4 Rahimi-Recht Method

Rahimi and Recht [19] proposed a feature mapping using Bochner's theorem and random sampling to approximate the Radial Basis Function (RBF) kernel. Basically, the input data are mapped to a randomized low dimensional feature space and then fast linear methods are applied to approximate the kernel.

According to Bochner's theorem [49]: A continuous kernel $k(x, y) = k(x - y)$ on $\mathbb{R}^d$ is positive definite if and only if $k(\delta)$ is the fourier transform of a non-negative borel measure.

$$k(x - y) = \int_{\mathbb{R}^d} p(\omega)\zeta_\omega(x)$$

They claimed that if the kernel was scaled properly, then the above could be rewritten by defining $\zeta_\omega(x)$ as $e^{j\omega' x}$

$$k(x - y) = \int_{\mathbb{R}^d} p(\omega)e^{j\omega' x} = E_\omega[z_\omega(x)z_\omega(y)]$$

For the purpose of converging the above integral, they replaced the complex exponentials with cosines so as to get a real-valued mapping. The condition to satisify this mapping:

$$E_\omega[z_\omega(x)z_\omega(y)] = k(x, y)$$

where $z_\omega(x)$ was set to $\sqrt{2}cos(\omega' x + b)$. ω as drawn from the distribution $p(\omega)$ and b was drawn uniformly from the interval $[0, 2\pi]$

The resulting inner product was computed by taking the average of D-randomly chosen z_ω and this is the lower variance approximation to the expectation.

$$z(x)' z(y) = \frac{1}{D}\sum_{j=1}^{D} z_{\omega j}(x)z_{\omega j}(y)$$

6.4.5 Experiment Results - Classification Using Rahimi-Recht Method

Table 6.8 shows the results of classifying accents using the Rahimi-Recht method as a kernel approximation method. For experiments, the *bandwidth* was chosen to be 1 and ρ which defines the dimension to approximate the kernel was kept as 200. We experimented with different values of ρ and we did not want the number of features in the low-dimensional space to be too less or more. Among the different values of ρ, projecting the data to low-dimensions with ρ as 200, we got a higher accuracy. Hence, we settled with ρ as 200.

Table 6.8: Comparison between LibSVM, Proposed Distance Metric Learning method, Rahimi-Recht method

Accents	**Data**	**LibSVM Accuracy**	**Proposed distance metric Accuracy**	**Rahimi Recht Method Accuracy**
Arabic and Brazilian Portuguese	FAE	70.58%	78.18%	77.56%
Japanese, and Hindi	FAE	69.7%	69.36%	68.42%
Hindi and Tamil	FAE	56.13%	66.25%	63.88%
Mandarin and German	FAE	60.61%	56.77%	51.02%

CHAPTER 7

PERFORMANCE AND CONCLUSION

This chapter gives a summary of the performance in each phase of the proposed method.

7.1 Phase 1

After performing feature extraction on the raw input speech signal using PLP, each speaker is characterized by 2046 windows and 39 features. Classification on such data could take a long time and consumes more memory; hence, we decided to reduce the number of windows. We ran algorithms like K-Means, Column Sampling, and Random Sampling and compared their performances as shown in Table 6.1 in Chapter 6.2.4. From this table, it was inferred that we obtained higher accuracy with K-Means.

We also compared the performance using various features like 13, 39, and 52 features. On the speech data that has 39 features, the classifiers detected the accents more accurately compared to the data that had 13 or 52 features. Similar experiments were run using GMM and DAGSVM classifiers. The results are shown in Table 6.2 of Section 6.2.5 and in Table 6.3 of Section 6.2.6. In the case of comparing the performance between LibSVM and DAGSVM, we also compared the performance on different datasets *FAE* and *GMU*.

7.1.1 Conclusion

As per the results, we conclude that K-Means as a clustering step gives better classification accuracy when compared with column and random sampling. Among the classifiers, LibSVM performed better than Gaussian Mixture Models (GMM) and Directed Acyclic Graph SVM (DAGSVM) and classifiers. Since speech recognition systems in industry uses 39 features, we chose to have 39 features and also for the fact that the performance was better.

7.2 Phase 2

Based on the observations in Table 6.4 of Section 6.3, the accents that were far apart had higher accuracy than the ones closer to each other. As a baseline, we experimented classifying using Linear Discriminant Analysis (LDA). As shown in the results in Table 6.5 of Section 6.3.4, the accuracy with LDA as classifier was much lower than the accuracy with LibSVM as classifier.

With this observation, we proposed to learn a distance metric that would bring the data points in the same class closer and separate the classes by a specific distance. This specific distance was the edit distance computed over phonetic strings of the accents. We compared the performance of classification between LibSVM and the proposed method. As per the results in Table 6.6 of Section 6.3.6, the proposed distance metric learning method outperformed the LibSVM on most of the pairwise accent classification. In some cases, the accuracy of the proposed method was found to be slightly decreased and we think it might be due to the speech data that might not be strongly accented.

7.2.1 Conclusion

Based on results, we conclude that learning a distance metric over phonetic strings gives better classification results than existing LibSVM. We considered only binary classification in this case. The advantage of the proposed method is that it requires less computation in considering the distance between different classes rather than the distance between every data point in different classes. Incorporating the phonetic distance helps to learn a structure over the classes. It also gives a more refined way of separating different classes by a specified distance rather than defining the distance as maximum or minimum.

7.3 Phase 3

Xing et al. [44] proposed a method to learn a distance metric using kernels but did not clearly explain how to transform a new test point to the feature space. Therefore, we experimented learning a distance metric using kernels that involved approximating kernels in order to make it feasible to transfer a new test point to the feature space. We used Kernel PCA (KPCA) and Rahimi-Recht methods for approximating kernels. Results obtained using KPCA are shown in Table 6.7 of Section 6.4.3 and the Rahimi-Recht method in Table 6.8 of Section 6.4.5. The accuracy seems to be lower than the proposed distance metric learning method.

7.3.1 Conclusion

Though both the kernel approximation methods did not yield further benefits when compared to LibSVM, the Rahimi-Recht method gave an higher accuracy than KPCA.

With these results, we conclude that incorporating supervision (class information) in approximating kernels might be an improvement and this could be a possible future work.

CHAPTER 8

FUTURE WORK

Though we have shown that learning a distance metric over phonetic strings can achieve higher accuracy, we focused on just binary classification. More research needs to be done to extend this for multiclass classification and also look into simultaneously minimizing the distance between data points within a class while separating the different classes by a specified distance.

In the kernel approximation stage, supervision (class information) needs to be considered to get a better classification accuracy since we are dealing with supervised data. Also, while learning the kernel transformations since the training and test data are transformed separately, it needs to be ensured that the transformation is independent of data.

Improvements in creating the accented FAE data should be made like including more accented data of microphone type with a higher sample rate. Therefore, it would be easier for the classifier to perform better.

REFERENCES

[1] P. Li and H. Tang, “Design of a low-power coprocessor for mid-size vocabulary speech recognition systems,” *IEEE Trans. on Circuits and Systems*, vol. 58-I, no. 5, pp. 961–970, 2011.

[2] S. Novotney and C. Callison-Burch, “Cheap, fast and good enough: automatic speech recognition with non-expert transcription,” in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. HLT ’10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 207–215. [Online]. Available: http://dl.acm.org/citation.cfm?id=1857999.1858023

[3] C. Fook, M. Hariharan, S. Yaacob, and A. Adom, “A review: Malay speech recognition and audio visual speech recognition,” in *Biomedical Engineering (ICoBE), 2012 International Conference on.* IEEE, 2012, pp. 479–484.

[4] Regional accents a problem for call centres as voice recognition technology fails to understand them. [Online]. Available: http://www.dailymail.co.uk/news/article-562830/Regional-accents-problem-centres-voice-recognition-technology-fails-understand-them.html

[5] C. Huang, E. Chang, and T. Chen, “Accent issues in large vocabulary continuous speech recognition (lvcsr),” *Microsoft Research China, Technical Report MSR-TR-2001-69*, 2001.

[6] Watch: Siri has trouble recognizing scottish accents. [Online]. Available: http://techland.time.com/2011/10/27/watch-siri-has-trouble-recognizing-scottish-accents/

[7] Apple’s siri vs. japanese-accented english. [Online]. Available: http://boingboing.net/2012/11/20/apples-siri-vs-japanese-acc.html

[8] J. Loviglio, “Researchers track evolution of philly’s odd accent,” www.philly.com/philly/news/science/20130426_ap_researcherstrackevolutionofphillysoddaccent.html.

[9] G. F. Choueiter, G. Zweig, and P. Nguyen, “An empirical study of automatic accent classification,” in *ICASSP’08*, 2008, pp. 4265–4268.

[10] H. Hermansky, “Perceptual linear predictive (plp) analysis of speech.” *Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752, 1990. [Online]. Available: http://lectures.idiap.ch/winter2005-2006/ic-48/courseslide/PLP.pdf

[11] D. O’Shaughnessy, “Linear predictive coding,” *Potentials, IEEE*, vol. 7, no. 1, pp. 29–32, 1988.

[12] S. B. Davis and P. Mermelstein, "Readings in speech recognition," A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, pp. 65–74. [Online]. Available: http://dl.acm.org/citation.cfm?id=108235.108239

[13] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[14] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1, pp. 19–41, 2000.

[15] J. Neidert, P. Chen, and J. Lee, "Foreign accent classification," 2011, cS 229.

[16] P. Watanaprakornkul, C. Eksombatchai, and P. Chien, "Accent classification," 2011, cS 229.

[17] S. Ullah, F. Karray, A. Abghari, and S. Podder, "Soft computing-based approach for natural language call routing systems," in *IEEE International Symposium on Signal Processing and its Applications*, 2007.

[18] B. Schlkopf, A. J. Smola, and K. R. Müller, "Kernel principal component analysis," *Advances in kernel methods: support vector learning*, pp. 327–352, 1999.

[19] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *NIPS*, 2007.

[20] A. Ikeno and J. H. L. Hansen, "Perceptual recognition cues in native english accent variation: – listener accent, perceived accent, and comprehension –," in *icassp*, vol. 1, 2006, pp. 401–404.

[21] P. Adank, M. Noordzij, and P. Hagoort, "The role of planum temporale in processing accent variation in spoken language comprehension." *Hum Brain Mapp*, 2011.

[22] L. Ragnoni, "The impact of prosody in foreign accent detection. a perception study of italian accent in english," in *Methodological Perspectives on Second Language Prosody*, 2012.

[23] P. B. de Mareil and B. Vieru-Dimulescu, "The contribution of prosody to the perception of foreign accent," 2006.

[24] K. v. Leyden and V. J. van Heuven, "On the prosody of orkney and shetland dialects," *Phonetica*, vol. 63, no. 2-3, pp. 149–74, 2006.

[25] I. Mennen, "Phonological and phonetic influences in non-native intonation," in *Trouvain*, J. and U. Gut, Eds. Non-native Prosody: Phonetic Descriptions and Teaching Practice (Nicht-muttersprachliche Prosodie: phonetische Beschreibungen und didaktische Praxis, 2007, pp. 53–76.

[26] M. J. Munro, "Nonsegmental factors in foreign accent," *Studies in Second Language Acquisition*, vol. 17, pp. 17–34, 2 1995.

[27] H. Tang and A. A. Ghorbani, "Accent classification using support vector machine and hidden markov model."

[28] P. R. Scott Novich and A. Trevino, "Accent classification using neural networks," Rice University.

[29] M. Huckvale, "Accdist: a metric for comparing speakers' accents," in *INTERSPEECH'04*, 2004, pp. –1–1.

[30] E. Ferragne and F. Pellegrino, "Vowel systems and accent similarity in the british isles: exploiting multidimensional acoustic distances in phonetics," *Journal of Phonetics*, vol. 38, no. 4, pp. 526–539, 2010.

[31] S. Ullah and F. Karray, "An evolutionary approach for accent classification in ivr systems," in *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference*, 2008.

[32] S. Ullah, "A soft computing based approach for multi-accent classification in ivr systems," Ph.D. dissertation, University of Waterloo, 2009.

[33] Y. Zheng, R. Sproat, L. Gu, I. Shafran, H. Zhou, Y. Su, D. Jurafsky, R. Starr, and S. youn Yoon, "Accent detection and speech recognition for shanghai-accented mandarin," in *Proc. Interspeech*, 2005.

[34] D. Vergyri, L. Lamel, and J.-L. Gauvain, "Automatic speech recognition of multiple accented english data," in *INTERSPEECH*, 2010, pp. 1652–1655.

[35] X. Lin and S. Simske, "Phoneme-less hierarchical accent classification," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2. IEEE, 2004, pp. 1801–1804.

[36] F. Biadsy, J. Hirschberg, and M. Collins, "Dialect recognition using a phone-gmm-supervector-based svm kernel," in *INTERSPEECH*, 2010, pp. 753–756.

[37] J. Humphries, P. Woodland, and D. Pearce, "Using accent-specific pronunciation modelling for robust speech recognition," 1996.

[38] T. Lander, "Cslu: Foreign accented english release 1.2, linguistic data consortium, philadelphia," 2007.

[39] S. Weinberger, "Speech accent archive," 2012. [Online]. Available: http://accent.gmu.edu

[40] H. Hermansky and N. Morgan, "Rasta processing of speech," *IEEE Transactions on Speech and Audio Processing*, pp. 578–589, 1994.

[41] R. Batuwita and V. Palade, "Class imbalance learning methods for support vector machines," 2012.

[42] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," in *Data mining techniques for the life sciences.* Springer, 2010, pp. 223–239.

[43] M. Sabzekar, M. GhasemiGol, and M. Naghibzadeh, "Improved dag svm: a new method for multi-class svm classification," in *Int. Conf. Artificial Intelligence ICAI 2009*, 2009.

[44] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in Neural Information Processing Systems 15.* MIT Press, 2002, pp. 505–512.

[45] K. T. Abou-Moustafa, F. D. la Torre, and F. P. Ferrie, "Pareto discriminant analysis," in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010.* IEEE, 2010, pp. 3602–3609.

[46] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *NIPS*, 2003.

[47] F. De la Torre and T. Kanade, "Multimodal oriented discriminant analysis," in *Proceedings of the 22nd international conference on Machine learning.* ACM, 2005, pp. 177–184.

[48] C. Gooskens, "The contribution of linguistic factors to the intelligibility of closely related languages," *Journal of Multilingual and Multicultural Development*, vol. 28, no. 6, pp. 445–467, 2002.

[49] W. Rudin, *Fourier Analysis on Groups.* John Wiley & sons, 1962.