ADAPTATION OF HYBRID DEEP NEURAL NETWORK-HIDDEN MARKOV MODEL SPEECH RECOGNITION SYSTEM USING A SUB-SPACE APPROACH

A Dissertation
Presented to
The Academic Faculty

By

Muhammad Rizwan

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2017

ADAPTATION OF HYBRID DEEP NEURAL NETWORK-HIDDEN MARKOV MODEL SPEECH RECOGNITION SYSTEM USING A SUB-SPACE APPROACH

Approved by:

Dr. David V. Anderson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark A. Clements
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark A. Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Omer T. Inan
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Fang (Cherry) Liu
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Wayne Daley
Georgia Tech Research Institute

Date Approved: June 30, 2017

Dedicated to my parents & teachers

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, David V. Anderson for his guidance and support. Especially his mentorship style that any graduate student requires namely the freedom to explore new ideas, friendly style of advising and encouragement. I am indebted and grateful to him for allowing me to explore and work on other projects besides my Ph.D. thesis area of research. These projects have provided me with learning opportunities and have broadened my horizon of research. I feel privileged and lucky to have the opportunity to work with him during my Ph.D. In addition to research skills, I also learned from him how to be a good human-being.

I am thankful to Dr. Mark Clements, Dr. Mark Davenport, Dr. Wayne Daley, Dr. Omer Inan, and Dr. Fang (Cherry) Liu for their time and consideration as members of my thesis committee. Also, I owe gratitude to Dr. James McClellan for teaching me how to link ideas from different research papers and build connections in mind, Dr. Mary Weitnauer for introducing me to an exciting area of machine learning, and Dr. John Barry for being teaching mentor and providing me with an opportunity to teach lectures.

My wife has been a significant support throughout the journey of Ph.D. I appreciate her for continuous support and unconditional love. I would like to thank my parents for their encouragement and constant support. I would not have achieved this without their prayers. Special thanks to my sister for advice and well wishes.

Finally, big thanks to all of my friends at the Georgia Tech for making my time enjoyable and enlightening. Especially the members of the Efficient Signal Processing Lab - Dr. Nashlie, Dr. Kaitlin, Femi, Nathan, Brad, Brandon, Courtney, Daniel, and Jeff. Thank you for making this entire Ph.D. journey exciting.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The performance of automatic speech recognition (ASR) system can be enhanced by adaptation of the ASR for a particular speaker or a group of speakers. In ASR, training and testing data often do not follow the same statistics; they are often mismatched, which leads to a gap in performance. The difference between training and testing statistics can be minimized by speaker adaptation techniques, which require adaptation data from a target speaker to optimize system performance. In the past, ASR systems were based on Gaussian mixture model-hidden Markov models (GMM-HMM). A resurgence of neural networks has resulted in the popularity of hybrid deep neural network-hidden Markov models (DNN-HMM) for speech recognition. The adaptation techniques developed for GMM-HMM systems cannot be directly applied to DNN-HMM systems because GMMs are generative models and DNNs are discriminative models. Also, DNN-HMM systems contain large numbers of parameters and require a huge amount of data from target speaker to adapt ASR. In many cases, only a limited amount of adaptation data is available for the target speaker. This thesis proposes multiple methods for the adaptation of speech recognition system by using a limited amount of data (a few words). The first method uses multiple words for accent classification in order identify variability in speaking style. Next adaptive phoneme classification is proposed based on target speaker similarity with speakers in the training data. Finally, we present adaptation of ASR by augmenting the speech features with speaker-specific information learned using sparse coding.

CHAPTER 1
INTRODUCTION

Communication through speech is the most natural and convenient way that we all use every day for interacting with one another. There are other ways through which people can communicate with one another. Communication through speech is the most efficient and versatile way as it allows a fast flow of information. In the last decade, we have seen a pervasive spread of electronic devices that have revolutionized and have become an integral part of our daily life. These electronic devices have improved the way we get information from all around the world and have a profound impact on our everyday activities. Speech recognition can help people to interact with these devices seamlessly and can revolutionize the landscape of human-machines interaction. Automatic speech recognition (ASR) is a thriving and promising topic that can provide more opportunities for getting more benefits from these electronic devices.

The goal of a speech recognition system is to convert an audio waveform (speech signal) to words accurately, independent of a speaker and environmental variations by using a computer interface. In other words, ASR is a system that takes a speech signal as an input and gives words as an output corresponding to the given input speech signal. The conversion of speech signal into words is a challenging task as the speech signal intrinsically exhibits many variations: physiological, environmental, linguistic, etc.

1.1 Recent Work

Speech recognition has been an active research area for four decades. Speech signals contain temporal structure, and the role of ASR system is to convert variable length speech utterances into variable length sequences of words. Hidden Markov models (HMMs) provide a statistical framework for acoustic modeling of speech signal [1]. HMMs map sequences

of observations (acoustic frames) to sequences of labels (phonemes). For a given acoustic observation, HMMs provide the probability distribution over all possible sequences of labels. In the past, Gaussian mixture models (GMMs) were used for estimating the probability distributions of speech utterances associated with the states of HMMs. Acoustic models based on GMMs-HMMs were trained using the maximum likelihood algorithm. In the 2000s the maximum likelihood algorithm was replaced by the sequence discriminative algorithm. Sequence discriminative algorithms, such as minimum classification error and minimum phone error, further improved the performance accuracy of ASR. Recently GMMs have been replaced by discriminative hierarchical models such as deep neural networks (DNNs) and have significantly improved the accuracy of ASR system. A major driving force for these discriminative hierarchical models is the availability of a large amount of data and computational resources [2].

1.2 Current Challenges

Although in recent years there has been a significant improvement in ASR system accuracy, people still mostly interact with various devices through a keyboard or touchscreen. The primary reason is that typing is more convenient than dictation considering the low accuracy of ASR. Also, the recent improvement in performance is limited to certain conditions [3, 4]. To have a seamless interaction with the devices and widespread use of ASR, one needs to improve the accuracy of ASR under all conditions. The most prominent of these variations are speaker mismatch, channel mismatch, and noise [5].

1.3 Contributions

This thesis focuses on variation due to speaker mismatch. The speaker mismatch variations result in performance degradation of ASR system when a new speaker data does not match with the data used for training ASR system. To improve accuracy further, the biggest challenge is the adaptation of ASR system to different dialects, accents, speaking

styles, etc. There is a plethora of work done for the adaptation of GMM-HMM speech recognition systems. The resurgence of neural networks resulted in hybrid deep neural network-hidden Markov model (DNN-HMM) speech recognition systems. The techniques developed for speaker adaptation of GMM-HMM speech recognition systems cannot be directly applied to DNN-HMM speech recognition systems because GMMs are generative models and DNNs are discriminative models. There is no clear structure in the model parameters of DNN. Training DNN-HMM speech recognition systems require large amounts of data and are computationally expensive. Also, adaptation techniques proposed recently for the DNN-HMM speech recognition systems require a significant amount of data (10's - 100's of sentences) for the speaker adaptation.

The objective of this thesis is to develop a framework for the adaptation of ASR systems by using a limited amount of data (a few words). To achieve this objective, we break down the speaker adaptation problem of ASR in the following specific aims.

1.3.1 Accent classification

Variability in speech due to accents results in performance degradation of ASR systems. The performance degradation of ASR system can be overcome by identifying the accent of the speaker and using accent information for the adaptation of ASR. An algorithm is proposed that uses multiple words for accent classification. The algorithm uses a novel architecture to classify North American accents into seven groups [6].

1.3.2 Adaptive phoneme classification

Speech recognition systems decode words for a given speech signal by splitting the speech signal into small fragments known as frames. The overall performance of a speech recognition system is dependent on the frame phoneme classification accuracy of these frames. A speaker similarity score algorithm is proposed that uses k-nearest neighbor (k-NN) on the deep neural network (DNN) features to find the similarity of a given test speaker with

speakers in an instance space (training data). Based on the speaker similarity score information, the algorithm does adaptive phoneme classification [7, 8, 9].

1.3.3 Robust phoneme classification

The computation time of the speaker similarity score algorithm is improved by reducing the dimension of the feature vectors using neighborhood component analysis that learns low dimensional linear embeddings from the training data. Also, an adaptive data condensation scheme for instance space (training data) reduction is proposed based on the speaker ranking. Dimensionality reduction using neighborhood component analysis and adaptive data condensation using speaker ranking provide a significant reduction in the computational time of the speaker similarity score algorithm at the cost of a slight increase in phoneme frame error rates [10].

1.3.4 Speaker adaptation of ASR

The performance of ASR system is improved by augmenting the speech features with speaker features. The speaker features contain speaker-specific information. Universal background sparse coding and a multi-layer bootstrap networks are used to extract speaker features using a few words. The extracted speaker features are augmented with speech features for the speaker adaptation of ASR system.

1.4 Outline of thesis

The rest of the thesis is organized as follows. Chapter 2 begins with an overview of ASR and speaker adaptation. The chapter also provides a summary of speaker adaptation methods for GMM-HMM based speech recognition. Finally, a comprehensive review of speaker adaptation methods for DNN-HMM based speech recognition system is discussed.

Chapter 3 discusses the weighted accent classification algorithm. The weighted accent classification algorithm is based on extreme learning machines is compared with support

vector machines. The performance of the accent classification algorithm is compared by using different words and varying the number of words.

Chapter 4 describes a novel algorithm for adaptive phoneme classification named as the speaker similarity score algorithm. The speaker similarity score algorithm learns a speaker similarity score based on a small amount of adaptation data from each target speaker using the deep neural network-based acoustic features. A comparison regarding frame-wise phoneme classification is made between adaptive phoneme classification with the baseline deep neural network.

Chapter 5 focuses on methods to improve the computational time of the speaker similarity score algorithm. The first method reduces the dimension of the feature space by using neighborhood component analysis and the second method reduces the number of samples in the instance space by doing adaptive data condensation. Comparison of these methods is made with principal components analysis and linear discriminant analysis.

Chapter 6 extends the speaker similarity score algorithm for the adaptation of ASR system. A feature augmentation approach for speaker adaptation based on sparse coding and bootstrap network is also presented. A summary of the contributions made in this thesis, along with future directions, is provided in Chapter 7.

CHAPTER 2

ADAPTATION OF AUTOMATIC SPEECH RECOGNITION

In the past, ASR systems have been dominated by GMM-HMM for acoustic modeling. Recently hybrid DNN-HMM have shown significant improvements over GMM-HMM. This chapter provides an overview of speaker adaptation methods for both GMM-HMM and DNN-HMM speech recognition systems. This chapter is organized as follows: first, a brief overview of ASR is provided; then a summary of methods developed for GMM-HMM based speech recognition systems is presented; finally, a comprehensive review of speaker adaptation methods for hybrid DNN-HMM speech recognition systems is discussed.

2.1 Automatic speech recognition

The goal of the speech recognition system is to find the most probable word sequence corresponding to a given speech signal. Mathematically, this is represented as [11]:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W}|\mathbf{X}) = \arg\max_{\mathbf{W}} \frac{P(\mathbf{W})P(\mathbf{X}|\mathbf{W})}{P(\mathbf{X})} \tag{2.1}$$

where $\mathbf{X}$ represents speech signal with observation sequence $\mathbf{X} = [X_1, X_2, ..., X_n]$. ASR has to map the speech signal of variable length into a sequence of words which are also of variable length $\mathbf{W} = [W_1, W_2, ..., W_m]$. The above equation is rewritten below for a given fixed observation $\mathbf{X}$ as:

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{X}|\mathbf{W}) \tag{2.2}$$

where $P(\mathbf{W})$ represents the language model, and $P(\mathbf{X}|\mathbf{W})$ accounts for the acoustic model. Automatic speech recognition systems use large amounts of training data to learn the acoustic and language models to accurately predict words for a given sequence of observations, $\mathbf{X}$, of the speech signal. The underlying architecture of an ASR system is shown

in Fig. 2.1. ASR comprises mainly four components: feature extractor, acoustic model, language model, and decoder.

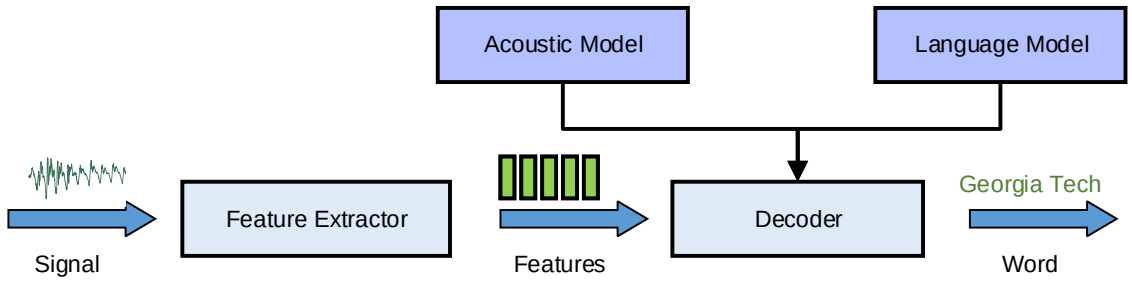

Figure 2.1: Automatic speech recognition block diagram

The feature extractor takes a speech signal as an input and converts it into features that are invariant to changes in the speaker, environment, and are suitable for building an accurate acoustic model. Feature extraction plays a vital role in the overall performance of an ASR. Many methods have been proposed over the years for feature extraction. These include Mel-frequency Cepstral coefficients (MFCCs), linear predictive coding (LPC), perceptual linear predictive coefficients (PLP), and relative spectral transform (RASTA). The speech signals are quasi-stationary and have stationary characteristics over a short period (10-100 msec) [12]. The speech signal is divided into frames so that the signal within each frame remains stationary and the corresponding features can be represented by a fixed length feature vector.

The acoustic model represents a statistical relationship between the features extracted from the speech signal and the linguistic unit. The most commonly used method to learn the acoustic model is the HMM. In HMMs, the acoustic input which comprises frames is represented by a set of states. A GMM is used to model the relationship between the states in the HMMs and given speech observations. Recently, DNNs replaced GMMs to estimate the posteriori probabilities of each state for a given observation [13]. Language models specify the likelihood of a word sequence by constraining the search in terms of limiting the number of possible words that need to be considered at a given point. The language

model helps in faster searches at the decoding stage and improves the overall accuracy. The most widely used language model for ASR is N-gram. The decoder uses the acoustic and language models to search for the best sequence of words by maximizing the score computed by the acoustic and language models.

2.2 Speaker Adaptation

The accuracy of speaker independent and speaker dependent ASR systems significantly differs, which leads to a gap in performance. The performance of an ASR system can be enhanced by adapting the ASR for a particular speaker or group of speakers [14]. In ASR, as training and testing data do not follow the same distribution, they are often mismatched. The difference between training and testing statistics/parameters can be minimized by speaker adaptation techniques, which require adaptation data from a target speaker to optimize system performance. In most cases, only a limited amount of adaptation data are available for the target speaker.

Speaker adaptation has different modes [15]. In the supervised adaptation, word-level transcriptions of speaker utterances are known. In unsupervised adaptation, word-level transcriptions are not available, and the ASR estimates them [16]. In static adaptation (also called batch mode), all adaptation data is given to the ASR before the adaptation process. In dynamic adaptation (also known as online adaptation), data is incrementally given to ASR.

2.3 Speaker Adaptation for GMM-HMM ASR

In the past, speech recognition systems were dominated by GMM-HMM systems and a plethora of work has been done for the adaptation of GMMs-HMMs based speech recognition systems. Several survey papers are written on GMM-HMM adaptation techniques [17, 18, 19, 20]. Speaker adaptation techniques for GMM-HMM systems can be broadly classified into maximum a posteriori (MAP), transformation, and speaker space-based methods

[17]. Fig. 2.2 provides a summary of GMMs-HMMs adaptation methods.

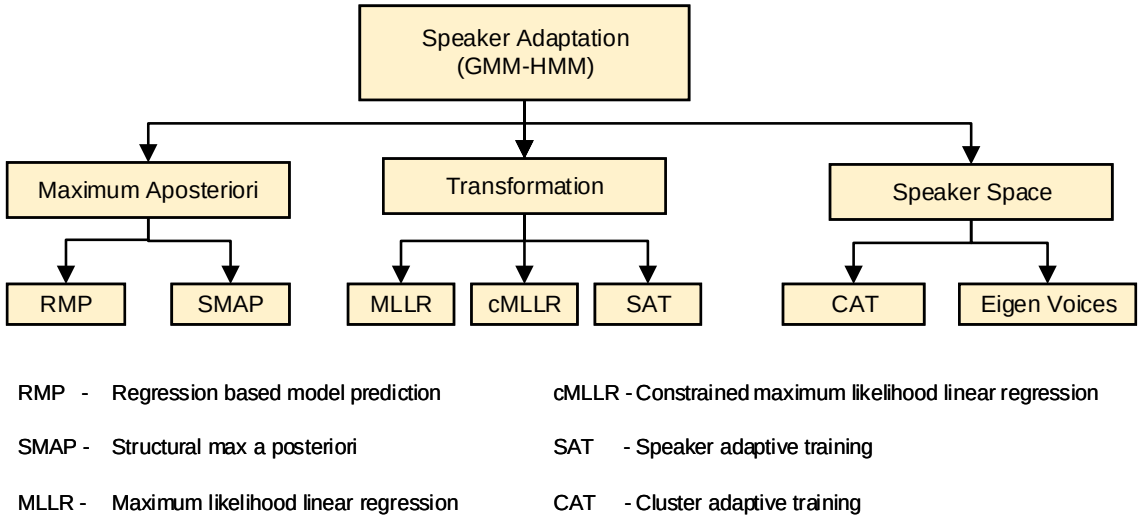

Figure 2.2: GMM-HMM based adaptation methods for ASR

In MAP-based methods, the parameters of the model are re-estimated by using adaptation data from the test speaker [21]. MAP-based methods require a significant amount of data from the test speaker to re-estimate HMM parameters. Given a large amount of data from the test speaker, the MAP estimate converges to the maximum likelihood (ML) estimate. When there is limited adaptation data from the test speaker, regression-based model prediction, and structural maximum a posteriori are used [22, 23].

The most widely used techniques for transformation-based methods are the maximum likelihood linear regression (MLLR), constrained maximum likelihood linear regression (cMLLR), and speaker adaptive training (SAT) [24, 25]. These techniques estimate a transformation of the model parameters. Speaker space-based methods estimate HMMs for a group of speakers. A few utterances are used from the test speaker to identify his/her group. Promising techniques for speaker space-based methods are cluster adaptive training (CAT) and Eigen-voices [26, 27]. For details regarding these methods, readers are referred to a review paper by P. C. Woodland [17].

2.4 Speaker Adaptation for DNN-HMM ASR

2.4.1 Feature space adaptation

The simplest approach for a feature space adaptation is a linear input network [28]. The linear input network uses linear mapping to transform input feature vectors as shown in Fig. 2.3. For a new test speaker, a linear input network is initialized with an identity matrix so that the initial point is based on a speaker independent model. The linear input network is trained using the back-propagation algorithm, and weights of the speaker independent neural network are kept frozen during training of the linear input network.

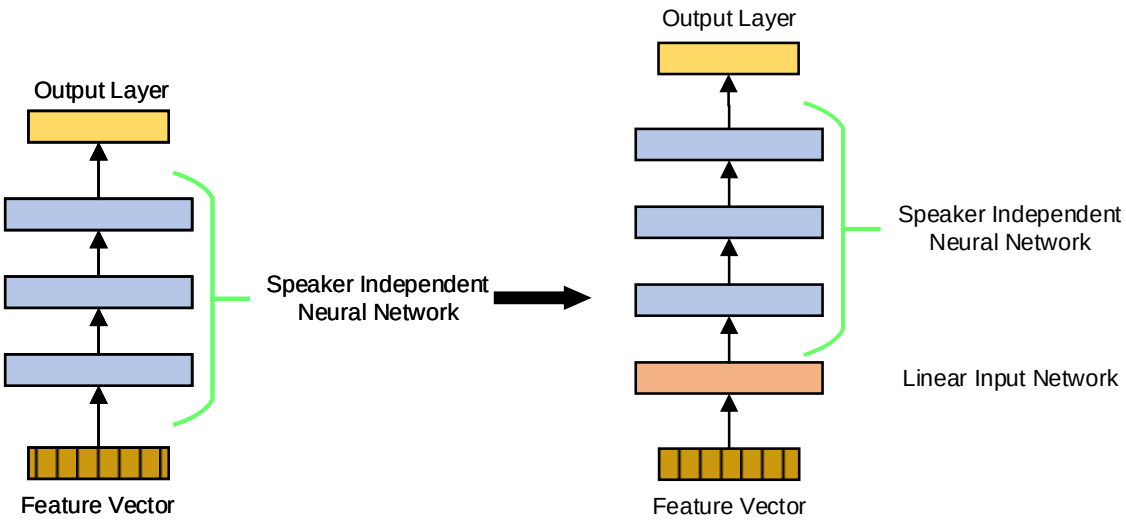

Figure 2.3: Linear input network

J. Neto et al. proposed a restrained speaker independent scheme for speaker adaptation [28]. In the restrained speaker independent scheme, weights of the original speaker independent neural network were adapted based on the test speaker adaptation data. The challenge is that there are large numbers of free parameters with a small amount of adaptation data, and training must be stopped before the system over-fits the adaptation data. Cross-validation determines the stopping criterion for network training on an independent set of data.

V. Abrash et al. used a transformation network as a pre-processor to the original speaker independent neural network [29]. The transformation neural network learns speaker dependent characteristics with a small number of parameters by applying a linear transformation to the incoming speech features. The architecture of the transformation neural network depends on the amount of adaptation data from the test speaker. The transformation neural network can be jointly trained with the speaker independent neural network. The parameters of the combined system (speaker independent and transformation neural network) are learned using a small learning rate. The transformation neural network provides quick adaptation with a small amount of adaptation data from the test speaker. F. Seide et al. applied heteroscedastic linear discriminant analysis, vocal tract length normalization, and feature space maximum likelihood linear regression to DNN-HMM speech recognition systems [30]. The authors found that DNN features are better than features obtained by heteroscedastic linear discriminant analysis and vocal tract length normalization.

2.4.2 Model space adaptation

J. Neto et al. proposed an architecture for model space adaptation in which a new neural network is placed in parallel with the speaker independent neural network [28]. The parallel neural network as shown in Fig. 2.4 has the same input layer and the same output layer as the speaker independent neural network. The original speaker independent network is kept frozen, and the new parallel neural network is trained using adaptation data from the test speaker. The intuition for the parallel neural network is that it compensates for the difference between the speaker independent neural network and the new speaker through weights. The parameters of the original speaker independent neural network are kept frozen, and weights of the parallel neural network are learned to provide speaker dependent information.

The neural network hidden layers activations learn better and refine features that are useful for phoneme classification as we move from input layer to the output layer of the

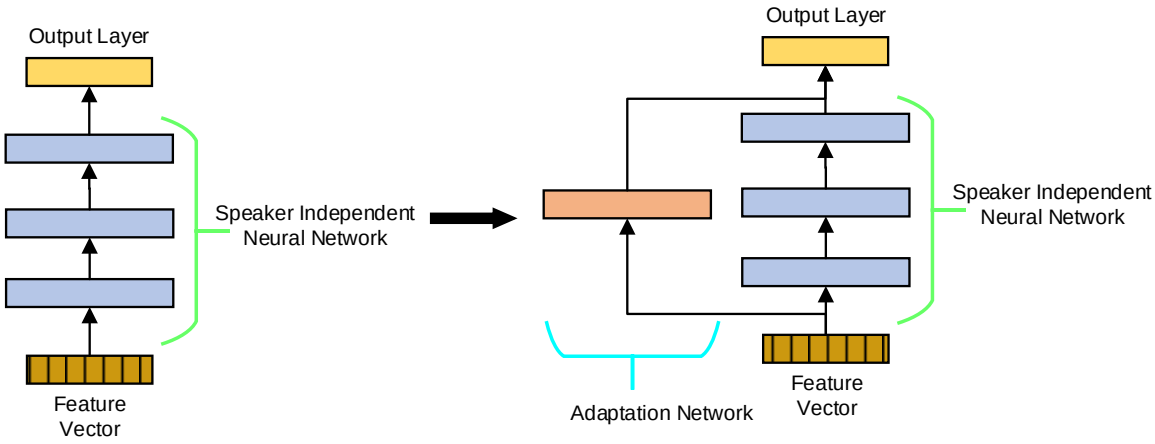

Figure 2.4: Parallel hidden network

neural network. The weights between the last hidden layer and the output layer provide a linear discrimination of the phoneme classes. R. Gemello et al. proposed a linear hidden network in which the linear transformation network is placed between the last hidden layer and the output layer [31]. The linear hidden network learns weight using adaptation data from the test speaker and provides better separation of phoneme classes. Fig. 2.5 shows the architecture of the linear hidden network. The linear hidden network is learned using the same procedure as discussed earlier for the linear input network. Conservative training is used to overcome the missing data classes in the adaptation data by replacing missing data class outputs with the outputs computed from the original speaker independent neural network.

K. Yao et al. further extended the idea of the transformation neural network before the output layer by applying an affine transformation to the parameters of softmax layer [32]. Only the bias vector is modified due to the scarcity of adaptation data from the test speaker. S. M. Siniscalchi et al. applied the idea of model space adaptation to large vocabulary continuous speech recognition [33]. A Hermitian activation function is used and the shape of the Hermitian activation function was modified based on speaker characteristics. The same degree of polynomial was used for the Hermite as an activation function for all the

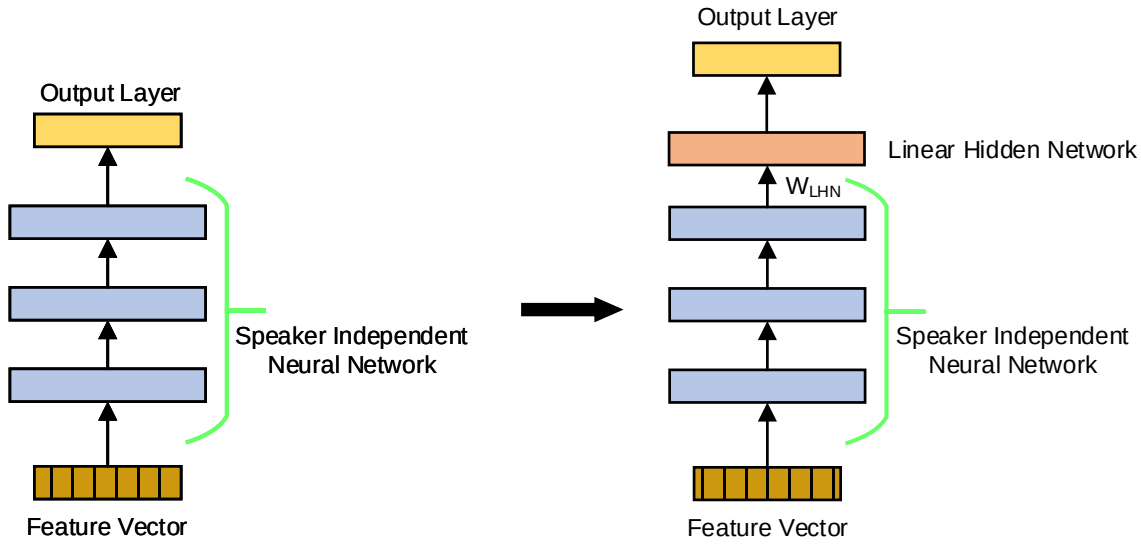

Figure 2.5: Linear hidden network

hidden layers. Adaptation of the non-linear activation function also overcomes the problem of test speaker adaptation data scarcity. Based on their experimental study, the authors did not find any effect of doing an adaptation of bias and slope of the sigmoid activation function.

D. Yu et al. proposed a conservative adaptation method that constrained the output probabilities of senones by adding Kullback-Leibler divergence regularization to the neural network objective function [34]. The Kullback-Leibler divergence forces the senone distributions to have probability values that are close to those estimated by the original speaker independent neural network.

S. Xue et al. applied singular value decomposition to the weight matrices of the speaker independent neural network [35]. The weight matrices of each layer are decomposed using singular value decomposition as shown in Fig. 2.6. The decomposition of the weight matrices is regularized such that the maximum singular value in S is one. During the adaptation, only the singular values of matrix S are updated for each target speaker. The non-diagonal zero elements of S, matrices U and V are kept frozen. This approach overcomes the overfitting problem as only limited parameters (singular values of matrices S) are updated by

using the adaptation data of the test speaker.

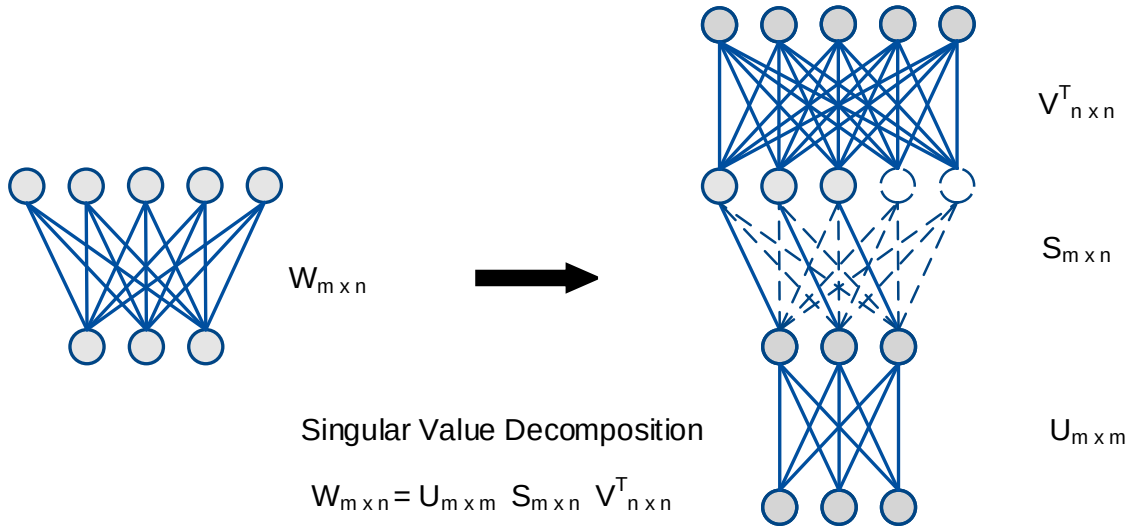

Figure 2.6: Singular value decomposition of weight matrix

Y. Zhao et al. modified the bias and slope of the sigmoid activation function for speaker adaptation [36]. The total number of adaptation parameters is twice the number of hidden units in the speaker independent neural network. This approach requires a small amount of adaptation data from the test speaker and has a low footprint, which makes it appealing for deployment in large-scale speech recognition systems.

R. Price et al. proposed speaker adaptation technique that overcomes the issue of classes with no representation at all or smaller representation in the adaptation data [37]. In their approach, they appended an additional output layer, termed the hierarchy output layer, that maps the original phonetic classes at the output layer to a smaller set of phonetic classes. During adaptation, weights of the hierarchy output layer are kept fixed, and weights of all other layers are adjusted using back-propagation.

Z. Huang et al. applied multi-task learning for speaker adaptation [38]. Senone classification is used as the primary task and monophone as the secondary task as shown in Fig. 2.7. Multi-task learning improves the generalization capability of the neural network

by enhancing the acoustic space for unseen senone scenarios. Multi-task learning also results in better discrimination capabilites of the neural network specifically when the adaptation data is small and most of the senones classes are not available in the adaptation data.

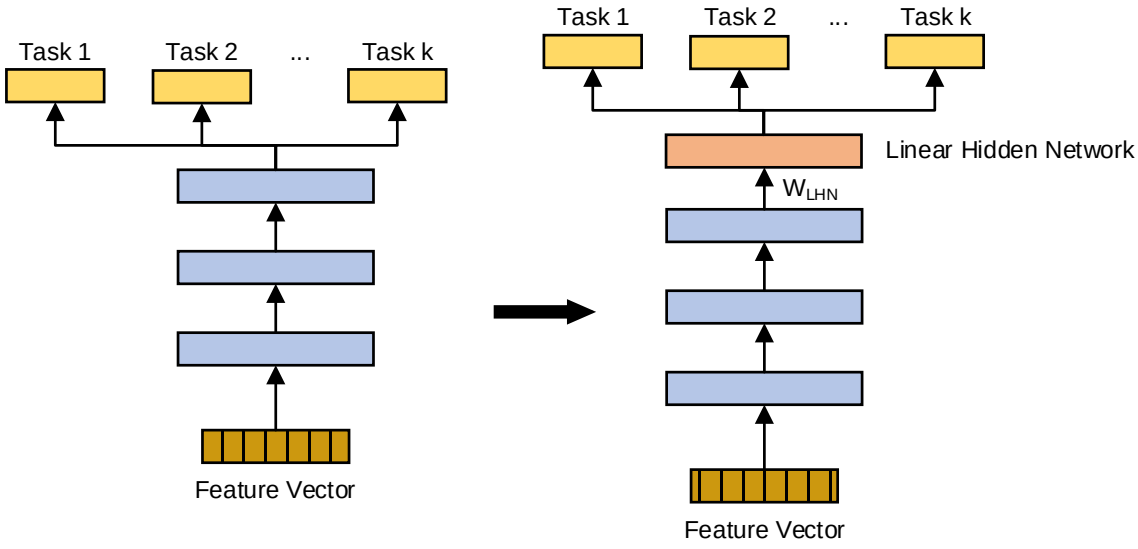

Figure 2.7: Multi-task learning for speaker adaptation

MAP framework is applied to the parameters of the top layer for speaker adaptation [39]. The prior information required by MAP is learned from the training data. Activation functions of the hidden layer are parameterised with a linear output scaling factor for speaker adaptation [40]. The parameters for the activation function are incrementally learned at each layer. The speaker dependent model comprises an adaptive linear factor associated with each activation function.

2.4.3 Sub-space data augmentation

The performance of ASR can also be improved by augmentation of speaker-specific information to the speech features. Speaker-specific information is added both during the training of ASR and at the testing stage. Various methods have been proposed to learn speaker-specific information that can be augmented with the speech features in the hybrid

DNN-HMM speech recognition system.

I-vectors can provide speaker information by using a very small amount of data from the speaker. The DNN-HMM system is adapted by giving speaker-specific information (i-vectors) along with acoustic feature vectors as input. Y. Miao et al. used i-vectors along with speech features for the speaker adaptation [41]. The speaker independent neural network is first trained using the training data. In the next step, i-vectors for each speaker in the training data are extracted and given as input along with speech feature vectors to learn the weights of adaptation neural network. During this step, the weights of the speaker independent neural are kept fixed. In the last step, the parameters of the adaptation network are kept fixed and the parameters of the speaker independent neural network are updated with the features obtained from adaptation network and speech feature vectors as shown in Fig. 2.8.

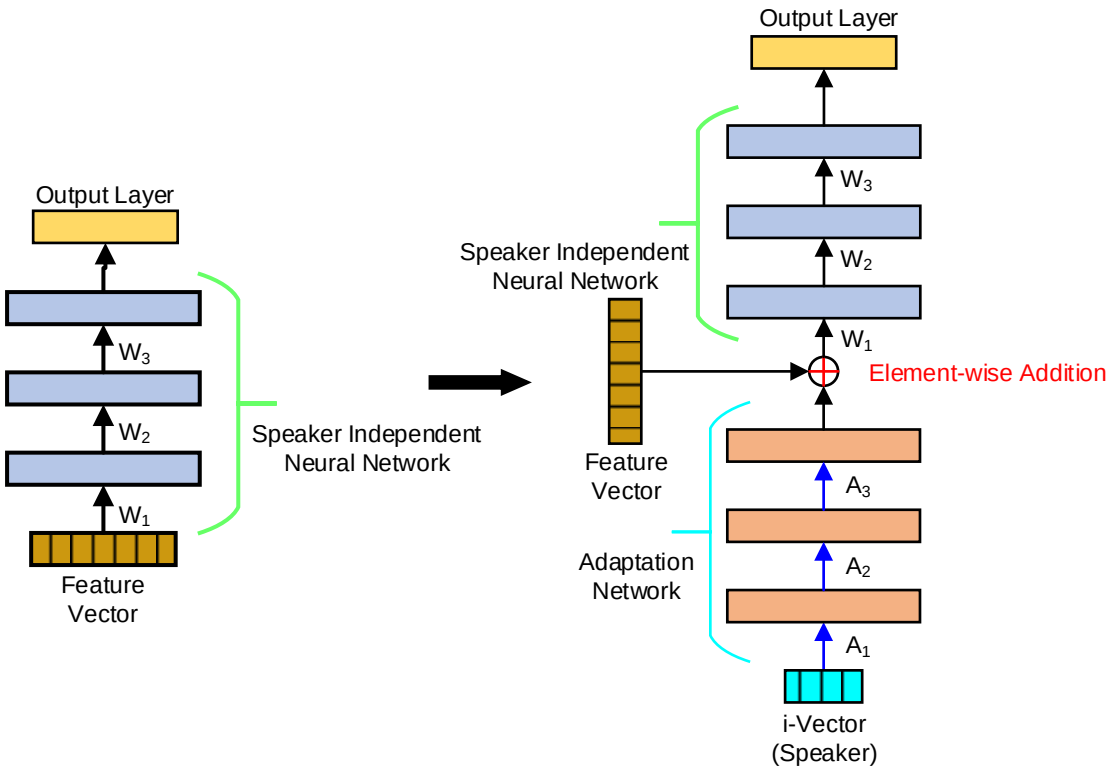

Figure 2.8: I-vector based speaker adaptation

Abdel et al. proposed three methods for speaker adaptation based on speaker codes [42, 43, 44]. The speaker code is a discriminative condition code associated with each speaker and represents speaker-specific information in a compact form. In first method, the adaptation neural network is added prior to the speaker independent neural network as shown in Fig. 2.9. Speaker codes are connected to the adaptation neural network. In the second method, speaker codes are directly used to adapt the speaker independent neural network by directly connecting speaker codes with all the layers of the speaker independent neural network. In third method, a joint learning procedure is used to learn speaker independent neural network weights, speaker codes, and connection weights from scratch.

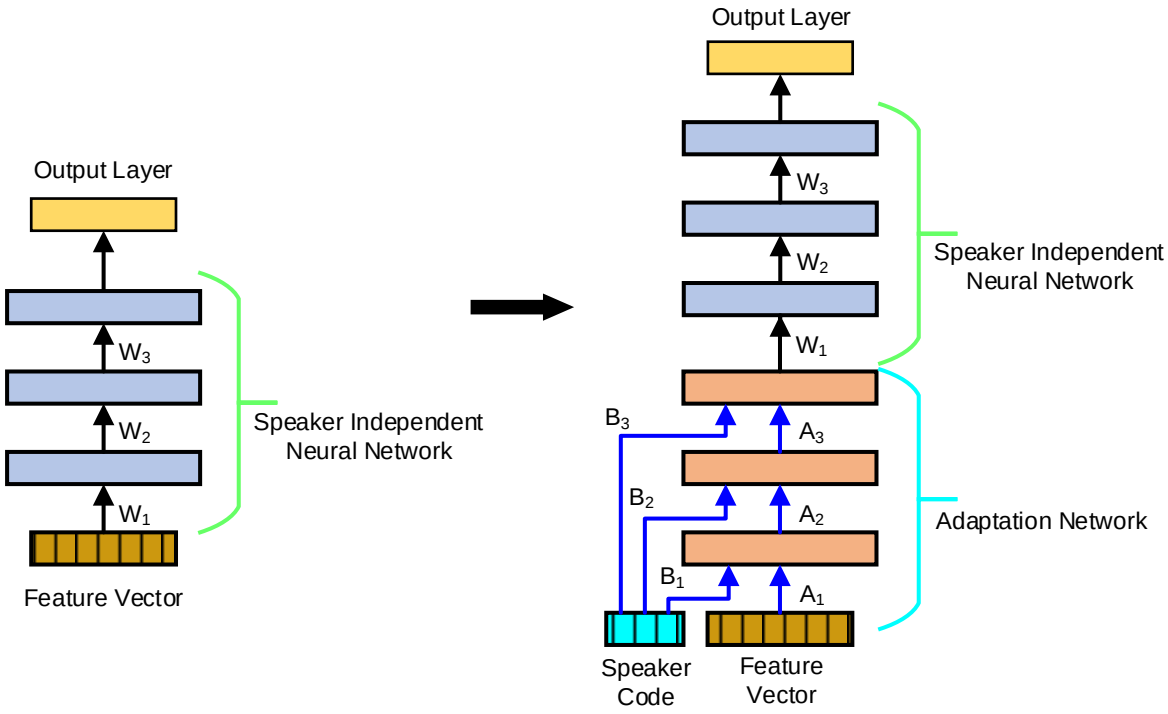

Figure 2.9: Speaker codes based speaker adaptation

CHAPTER 3

ACCENT CLASSIFICATION USING MULTIPLE WORDS

Speech recognition systems exhibit performance degradation due to variability in speech caused by the accents or dialects of speakers. The performance degradation can be overcome by correctly identifying the accent or dialect of the speaker and using accent or dialect information to adapt speech recognition systems. In this chapter, novel accent classification algorithm based on extreme learning machines (ELMs) is presented. ELMs are attractive for the accent classification task as they can be quickly trained and also provide a better generalization capability for small amounts of training data [45, 46]. Accent classification algorithm performance based on ELMs is compared with support vector machines (SVMs) as classifiers. The rest of the chapter is organized as follows: Section 3.1 summarizes related work in accent/dialect classification, Section 3.2 and 3.3 presents the theory of ELMs and SVMs, Section 3.4 provides theoretical comparison between ELMs and SVMs, Section 3.5 discusses accent classification algorithm, Section 3.6 describes dataset and feature extraction, and Section 3.7 presents results.

3.1 Related work

Speech signals intrinsically exhibit many variations, even in the absence of background noise. The three most prominent types of variations are due to acoustic effects, accent, and dialect. Acoustic variations are primarily related to inherited physical characteristics of size and shape of a vocal tract. Two different people saying the same sentence results in different spectrograms. The variations due to accent result from the relative prominence of a particular syllable or a word in pronunciation determined by the regional or social background of the speaker [47]. Different accents affect a change in the order and number of phonemes used to construct each word of an utterance, *i.e.* phoneme deletion, insertion and

substitution with respect to some reference accent. Dialect is defined as a regional variety of a language distinguished by pronunciation, grammar or vocabulary. Every individual develops a characteristic speaking style at an early age that is highly dependent on his or her language environment as well as the region where the language is spoken [48, 49].

The performance of a speech recognizer can be further improved by adapting a system based on accent/dialect. S. Goronzy achieved a 37% relative reduction in word error rate by adapting a speech recognizer based on accent [50]. There has been little past research in the area of accent classification. In particular, most of the previous work in the field involves accent classification among non-native English speakers. Accent variation among native American speakers is more challenging and has not enjoyed the same amount of attention in speech community research.

G. Choueiter et al. extended language identification techniques to a large-scale accent classification task [51]. The authors performed several experiments using heteroscedastic linear discriminant analysis and maximum mutual information on the Foreign Accented English dataset. The dataset is composed of utterances spoken by native speakers of 23 languages. They found that acoustic-only methods are useful for accent classification in contrast to typical language identification systems. P. Angkititrakul et al. used a phoneme-based model to design a text independent automatic accent classification scheme [52]. They performed experiments capturing the spectral evolution information as potential accent sensitive cues. They generated subspace representations using principal component analysis and linear discriminant analysis. The authors compared a spectral trajectory model framework with a traditional HMM framework using an accent sensitive word corpus. System evaluation was performed using a corpus that represents five English speaker groups, which consisted of native American English and English speakers having Mandarin Chinese, French, Thai, and Turkish accents for both male and female speakers. J. Guarasa used GMMs and Bayes' classifiers for German versus Spanish accent classification [53]. C. Clopper et al. did an extensive study of vowel variation in different regions of North

America by measuring duration of the first and second formant frequencies [54]. J. Hansen et al. did an extensive analysis and modeling of speech accents on NATO N-4, TIMIT and the WSJ corpus [55]. The authors analyzed prosodic structure (formants, syllable rate, and sentence duration), phoneme acoustic space and did word-level based modeling on large vocabulary data. The authors found that using the most discriminating vowels from each group improves the accent detection rate.

3.2 Extreme learning machines

Extreme learning machine (ELM) is a robust learning algorithm for single layer feed-forward neural networks (SLFNs) [56]. Currently, SLFNs mostly use gradient based methods for training neural networks. Gradient-based methods often get trapped in local minima solutions and, as a result, give suboptimal solutions. Genetic and evolutionary algorithms have been used to overcome local minima problems, but they are computationally expensive [57].

In ELMs, input weights of the hidden layer neurons are randomly generated, and output weights of the hidden layer neurons are learned analytically [56, 58]. By determining weights analytically, there is a high-performance speedup for training neural networks as compared to learning methods such as back-propagation [59]. Theoretically, it has been shown that by using ELMs universal approximation can be achieved [60, 61]. ELMs can also be used for training multilayer perceptrons by using hierarchical frameworks [62].

Various other architectures for ELMs have been proposed. In incremental-ELM, hidden nodes are added incrementally, and output weights are determined analytically [63]. In online sequential-ELM, training data is fed to the network in chunks [64]. Local receptive fields-ELM uses local structures and combinatorial nodes for incorporating translational invariance in the network [65]. ELMs can be used for both regression and multiclass classification problems directly [66].

ELMs transform the input data to the hidden layer by via randomly initialized weighted

connections. A single hidden layer network with M hidden nodes is shown in Fig. 3.1.

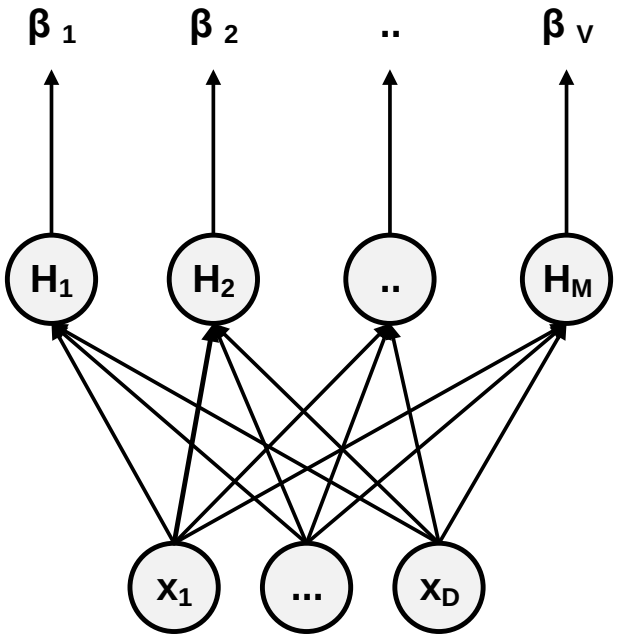

Figure 3.1: Extreme learning machines

The output function of the single layer network with M hidden neurons can be written as [45]:

$$f(\mathbf{x}) = \sum_{i=1}^{M} \boldsymbol{\beta}_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{3.1}$$

where

$$h_i(\mathbf{x}) = \sigma(\mathbf{w_i x} + b_i) \tag{3.2}$$

and σ is a non-linear activation function given by:

$$\sigma(\mathbf{w}_i\mathbf{x} + b_i) = \frac{1}{1 + e^{-(\mathbf{w}_i\mathbf{x} + b_i)}} \tag{3.3}$$

$\boldsymbol{\beta}$ is the vector of weights between M neurons in the hidden layer and the output layer:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_M \end{bmatrix} \tag{3.4}$$

The goal of ELM is to minimize the training error as well as the norm of the output weights. It does not require any adjustments to the input weights of neurons in the hidden layer [45, 65, 67, 68].

$$\text{minimize} \quad \|\boldsymbol{\beta}\|_p^{\sigma_1} + C \|\mathbf{H}\boldsymbol{\beta} - \boldsymbol{T}\|_q^{\sigma_2} \tag{3.5}$$

where $\sigma_1 > 0$, $\sigma_2 > 0$, and $p, q = 0, 1, 2, ..., \infty$, and $\mathbf{H}$ is the output matrix at the hidden layer given by:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \dots & h_M(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \dots & h_M(\mathbf{x}_N) \end{bmatrix}. \tag{3.6}$$

and $\mathbf{T}$ is the training data target values:

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_N \end{bmatrix} \tag{3.7}$$

Minimizing the training error and norm of weight vector results in good generalization capability of the network [63, 69]. The optimal output weights are then computed as:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T} \tag{3.8}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$ [70]. Various methods can be used to calculate the inverse of the matrix $\mathbf{H}$ such as orthogonal projection methods, iterative methods, and singular value decomposition [66, 71]. A closed-form solution for calculating $\boldsymbol{\beta}$ is given by [65]:

$$\beta = \begin{cases} \mathbf{H}^T(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{T}, & \text{if } N \leq M \\ (\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}, & \text{if } N > M \end{cases} \tag{3.9}$$

where C is a regularization parameter, $\mathbf{I}$ is the identity matrix, and $\mathbf{H}$ and $\mathbf{T}$ are as previously defined in Eqs. 3.6 and 3.7 respectively. ELM classifier expression can be written as:

$$\mathbf{f}(\mathbf{x}) = \begin{cases} \mathbf{h}(\mathbf{x})\mathbf{H}^T(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{T}, & \text{if } N \leq M \\ \mathbf{h}(\mathbf{x})(\frac{\mathbf{I}}{C} + \mathbf{H}^{\mathbf{T}}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T}, & \text{if } N > M \end{cases} \tag{3.10}$$

where $\mathbf{h}(\mathbf{x})$ is the hidden layer output vector corresponding to the input samples $\mathbf{x}$ and $\boldsymbol{\beta}$ are the output weight vector between a hidden layer of M nodes and the output node. In fact, $\mathbf{h}(\mathbf{x})$ is a feature mapping from input space of D-dimensions to random feature space (or ELM space) of M-dimensions.

The training data consists of N distinct input-output pairs of words and their corresponding accent type given by:

$$\text{Training Data} = \{(\mathbf{x}_1, \mathbf{t}_1), (\mathbf{x}_2, \mathbf{t}_2), ..., (\mathbf{x}_N, \mathbf{t}_N)\} \tag{3.11}$$

where each ($\mathbf{x}_i$,$\mathbf{t}_i$) respectively represent an input word data and its corresponding accent label. Specifically, $\mathbf{x}_i \in \mathbf{R}^D$ is the vector of extracted speech signal features for a complete "word", and $\mathbf{t}_i \in \mathbf{R}^V$ is the corresponding accent type. In this case, ELMs are trained to distinguish between two accent types.

3.3 Support vector machines

Support vector machines (SVMs) are based on the intuition of placing a hyperplane in such a way that it separates data classes with a large margin. An SVM is thus a maximum margin classifier. The margin in an SVM is the distance between the hyperplane and data point closest to it [72, 73, 74, 75]. When the data to be classified is not linearly separable (usually the case), a kernel function may be used to map the data from a given input space to a high dimensional space known as a kernel space. Using the kernel space may result in a better separability of data [76, 77]. For a given training set comprising N data points with class labels $\mathbf{t}_i \in \{1, -1\}$, the goal of SVM classifier is to separate data classes by finding an optimal hyperplane in the kernel space by solving a minimization problem with the inequality constraint given by Eq. 3.12:

$$\begin{aligned} &\underset{\mathbf{w},\zeta_i}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\zeta_i \\ &\text{subject to} \quad \zeta_i \geq 0, t_i\left[\mathbf{w}^T\phi(\mathbf{x}_i) + b\right] \geq 1 - \zeta_i \end{aligned} \tag{3.12}$$

where $i = 1, 2, ..., N$, $\phi(\cdot)$ is a mapping function, $\mathbf{w}$ is the normal to the optimal decision hyperplane, b is the bias term, C is the regularization parameter which determines the generalization capability of SVM (*i.e.* trade-off between margin and misclassification errors; the higher the value of C, the stricter the constraint and the lower the likelihood of over-fitting [78]), and ζ_i is the slack variable.

The above equation (Eq. 3.12) is in non-convex form and, therefore, difficult to solve. The above optimization problem is transformed into its dual form with an equality con-

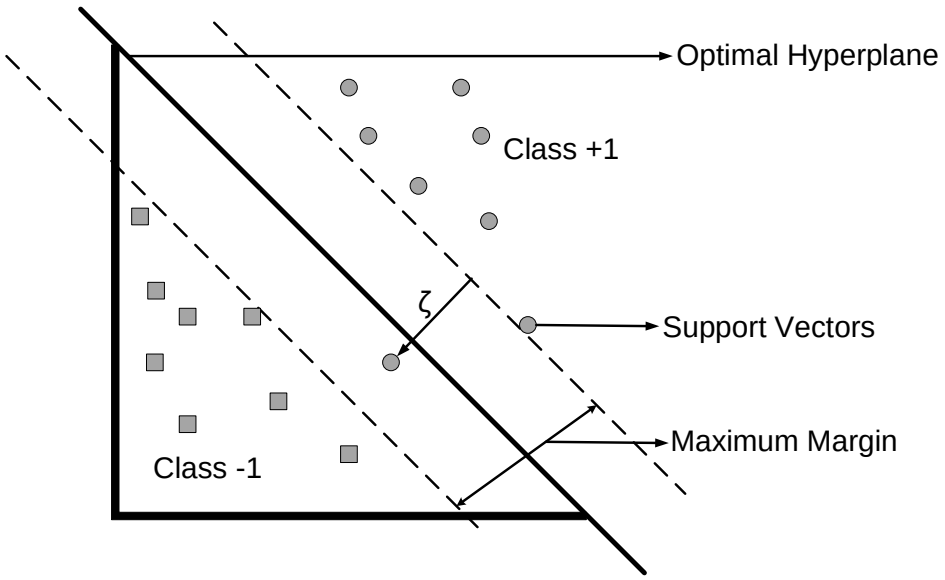

Figure 3.2: Support vector machine

straint by using the Lagrange multiplier. Lagrange function is given by:

$$\begin{aligned} L(\mathbf{w}, b, \zeta_i; \alpha_i, \lambda_i) &= \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N}\zeta_i \\ -\sum_{i=1}^{N}\alpha_i(\mathbf{t}_i[\mathbf{w}^T\phi(\mathbf{x}_i) + b] - 1 + \zeta_i) &- \sum_{i=1}^{N}\lambda_i\zeta_i \end{aligned} \tag{3.13}$$

where $\alpha_i \geq 0$ and $\lambda_i \geq 0$.

The solution can be obtained by solving the Lagrange function and calculating the partial derivative of the Lagrange function with respect to $\mathbf{w}$, b, and ζ_i [79].

$$\max_{\alpha_i,\lambda_i}\min_{\mathbf{w},b,\zeta_i} L(\mathbf{w}, b, \zeta_i; \alpha_i, \lambda_i) \tag{3.14}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0, \frac{\partial L}{\partial b} = 0, \frac{\partial L}{\partial \zeta_i} = 0 \tag{3.15}$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \mathbf{t}_i \phi(\mathbf{x}_i) \tag{3.16}$$

$$\sum_{i=1}^{N} \alpha_i \mathbf{t}_i = 0 \tag{3.17}$$

$$0 \leq \alpha_i \leq C \tag{3.18}$$

By using the above constraints, SVMs dual optimization problem can be written as:

$$\begin{aligned} &\underset{\alpha}{\text{maximize}} \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \mathbf{t}^{(i)} \mathbf{t}^{(j)} \alpha_i \alpha_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle \\ &\text{subject to} \quad \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i \mathbf{t}^{(i)} = 0, i = 1, ..., m \end{aligned} \tag{3.19}$$

SVMs decision function in a kernel space is given by:

$$\mathbf{f}(\mathbf{x}) = sgn \left(\sum_{i=1}^{M} \alpha_i \, \mathbf{t}_i \, \kappa \Big(\mathbf{x}_i, \mathbf{x} \Big) + b \right) \tag{3.20}$$

where $\kappa(.)$ is a kernel function. Several kernel functions with their hyperparameters are summarized in Table 3.1 below.

Table 3.1: SVM - Kernel functions

Kernel	**Function**	**Parameters**
Linear	$\mathbf{x_i}^T \mathbf{x_j}$	-
Polynomial	$(\gamma \mathbf{x_i}^T \mathbf{x_j} + r)^d$	γ, r, d
Radial basis function	$exp(-\gamma \|\|\mathbf{x_i} - \mathbf{x_j}\|\|^2)$	γ
Sigmoid	$\tanh(\gamma \mathbf{x_i}^T \mathbf{x_j} + r)$	γ, r

When using SVMs, three choices must be made: kernel type, corresponding kernel parameters, and regularization parameter [80]. SVMs computational complexity depends

on the number of samples in the training data and is independent of kernel space dimension.

3.4 Comparison between ELMs and SVMs

Both ELMs and SVMs converge to a single global optimum solution. ELMs optimize sum of squared errors, while SVMs construct a hyperplane that maximizes the separation between the data classes [79, 81, 82].

3.4.1 Decision surface

ELMs and SVMs have the same dual optimization objective functions. In ELMs, optimal solutions are learned from the entire cube $[0, C]^N$, while in SVMs optimal α_i is learned from one hyperplane $\sum_{i=1}^{N} \alpha_i \mathbf{t}_i = 0$ within the cube $[0, C]^N$ as shown in Figs. 3.3 and 3.4. This results in SVM solution being sub-optimal as compared to ELM.

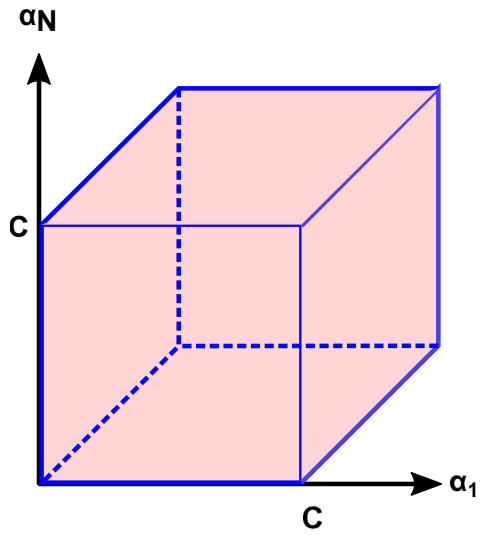

Figure 3.3: Extreme learning machine decision surface

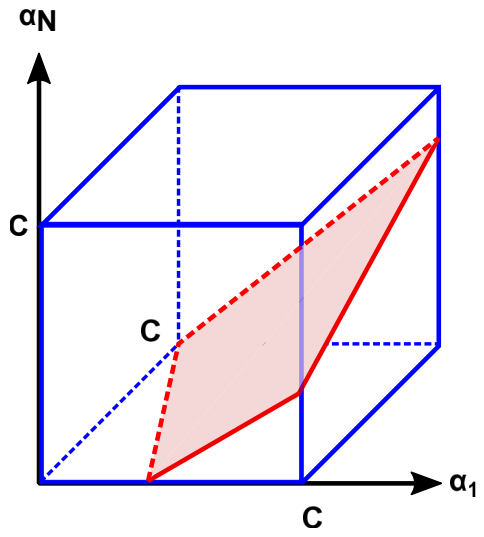

Figure 3.4: Support vector machine decision surface

3.4.2 Loss function

The training of classifier depends on the loss functions. Loss function has a significant impact on the training time of the classifier, as well as on the computational cost for the classification of new data [81].

ELM uses a quadratic loss function and minimizes the sum of square errors between the class labels and the network output. It not only penalizes wrong answers but also penalizes correct answers which are far from the decision boundary. The quadratic loss function is smooth and the resulting Karush-Kuhn-Tucker (KKT) system has a closed form solution [83]. This makes the training of ELM easy. Decision boundary of the ELM classifier is determined by using all samples present in the training data [84].

SVM constructs hyperplane as a loss function. The loss function is not smooth which results in an iterative solution for KKT dual system. It not only penalizes answers which are incorrect, but also that are correct but lie close to decision boundary [85]. Decision boundary is decided by only those samples from the training data for which Lagrange

multiplier is non-zero (*i.e.* support vectors).

3.4.3 Feature transformation

ELM uses random feature transformation and classifier can be trained by using primal or dual formulations [81]. SVM transforms data in kernel space by using kernel functions. SVMs are always trained in the dual space [72].

3.4.4 Hyperparameters

ELM requires selection of regularization parameter C and a number of neurons in hidden layer as hyper-parameters. The number of neurons in the hidden layer determine the dimensionality of the feature space. SVM requires hyperparameters depending on the kernel function, in addition to regularization parameter C. In short, SVM requires more hyperparameters as compared with ELM [81].

3.4.5 Training and testing time

ELM training time can be estimated as it uses a closed form solution for calculating weights. Let N be the number of training samples, D be the dimensionality of input data, and M be the number of neurons in the hidden layer of ELM. To calculate weight matrix given by Eq. 3.9, we first need to calculate H. Calculating H matrix requires $O(NDM)$ operations. The weight matrix β requires $O(NM^2 + M^3)$ operations [66, 81]. Training and testing time of ELM is given by Eqs. 3.21 and 3.22 for the case when $N \gg M$ and $N \gg D$.

$$ELM\ Training\ Time = O(NM^2) \tag{3.21}$$

$$ELM\ Testing\ Time = O(MD) \tag{3.22}$$

SVM training time estimation is difficult because of its iterative training procedure [81]. SVM training is related to the number of support vectors [86, 87]. For S number of support vectors, the testing time of SVM classifier is given by Eq. 3.23:

$$SVM\ \ Testing\ \ Time = O(SD) \tag{3.23}$$

Table 3.2: Comparison of ELMs and SVMs

Characteristics	**ELMs**	**SVMs**
Optimization	Sum of squared errors	Maximum margin classifier
Loss function	Smooth	Not smooth
Feature transformation	Random features	Kernel functions - Linear - Polynomial - Radial basis function - Sigmoid
Hyperparameters	Regularization parameter "C" Number of neurons in hidden layer Activation functions	Regularization parameter "C" Kernel function parameters - Linear (γ) - Polynomial (γ, d, r) - Radial basis function (γ) - Sigmoid (γ, r)
Computational complexity	Dependent on dimension of feature space	Independent of feature space dimension
Training time	Less	More
Multi-class classification	Directly	Indirectly

3.5 Weighted accent classification algorithm

Weighted accent classification algorithm uses either ELMs or SVMs for accent classification. The algorithm involves three stages. In the first stage, multiple ELMs (or SVMs) are trained using word samples from two accent classes at a time. This pair-wise classification helps to find right decision boundaries [6]. Hyperparameters of ELMs (or SVMs) are learned by cross-validation. In the second stage, the output of multiple ELMs (or SVMs) are combined to obtain a classification score. Finally, the classification score is encoded, and output accent class decision is made based on the highest encoded score. Fig. 3.5 shows the overall block diagram.

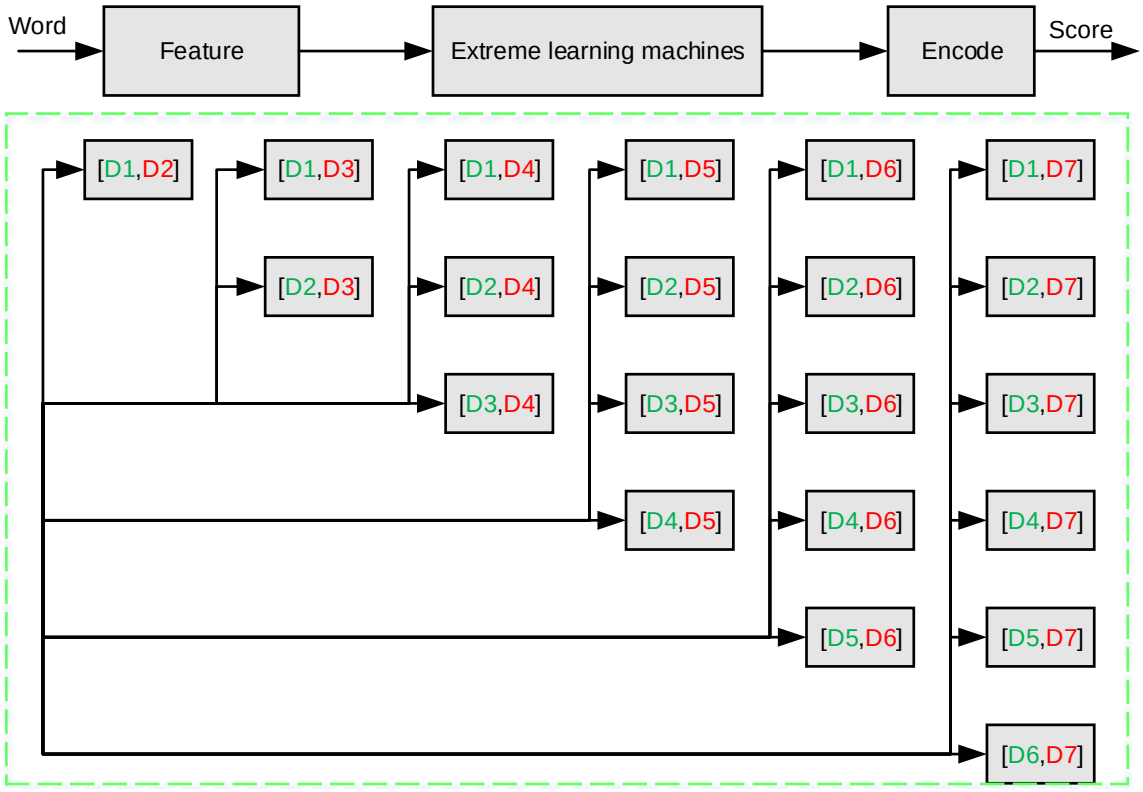

Figure 3.5: Weighted accent classification algorithm - Block diagram

3.5.1 Classifier training

Each ELM (or SVM) is individually trained and optimized for a single pair-wise decision. For example, an ELM (or SVM) [D1, D2] is trained using word samples from speakers belonging to accents D1 (New England) and D2 (Northern). Similarly, an ELM (or SVM) [D1, D3] is trained using word samples from speakers belonging to dialects D1 (New England) and D3 (North Midland), and so on. For ELMs, the number of hidden layer neurons was varied during training. For SVMs, kernel parameters were learned using grid search approach. The best hyperparameters were selected based on cross-validation.

Although ELMs (or SVMs) are capable of complex decision boundaries, in practice, developing a system that can reliably distinguish the seven accent classes is demanding, especially because there are many similarities between accents. The method is shown in Fig. 3.5 utilizes only pair-wise classification to make it easier to find good decision boundaries. This will result in 21 ELMs as shown in Fig. 3.5.

Let Z_{A1},Z_{A2},...,Z_{Aj} be the training word samples from accent "D_A" and let "j" be the total number of speakers in that particular accent group. Similarly, Z_{B1},Z_{B2},...,Z_{Bk} are the training word samples from accent "D_B," and "k" is the total number of speakers in accent group "D_B." Thus we have $D_A \in \{D1, D2, D3, \ldots, D7\}$, $D_B \in \{D1, D2, D3, \ldots, D7\}$, and $D_A \neq D_B$. Fig. 3.6 and Fig. 3.7 shows how ELMs or SVMs are trained in a pairwise manner for the weighted accent classification algorithm.

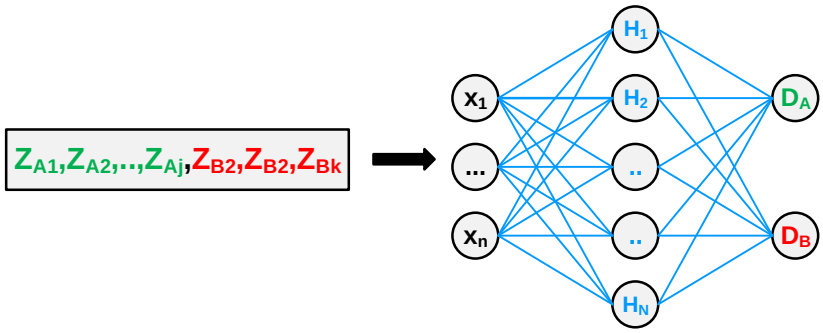

Figure 3.6: Extreme learning machine - Training

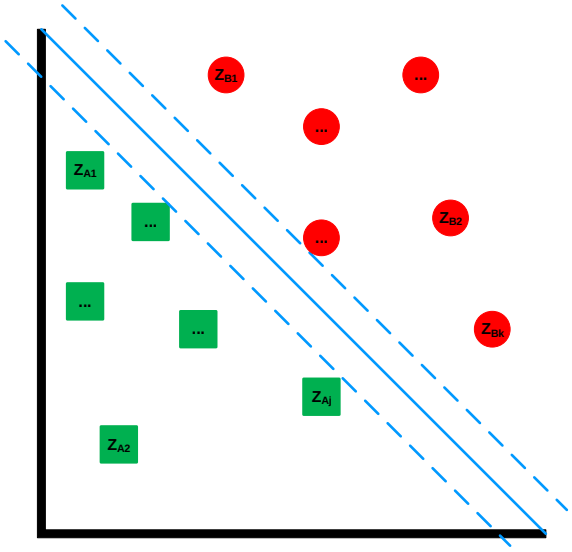

Figure 3.7: Support vector machine - Training

3.5.2 Classification score

Each ELM (or SVM) can receive all the samples from a typical word at once. All the word samples from a particular speaker are given as input to all these 21 ELMs (or SVMs) at once as shown in Fig. 3.8. Each of these ELMs (or SVMs) trained in a pair-wise manner will classify the particular speaker accent. For example, if the true class of the input word was D2, most of the pair-wise classifiers that were trained on D2 (*i.e.*, (D1, D2), (D2, D3), ...(D2, D7)) will correctly identify the class as D2. Those not trained on D2 (*i.e.*, (D1, D3), (D1, D4), ...(D6, D7)) will have effectively random outputs, choosing among the other classes with approximately equal probability as shown in Fig. 3.8. Thus, the class D2 will win the vote, and the class will be correctly identified. No hard decision is made on a single word, so the results are combined over the entire utterance using a weighting scheme as described below in Section 3.5.3.

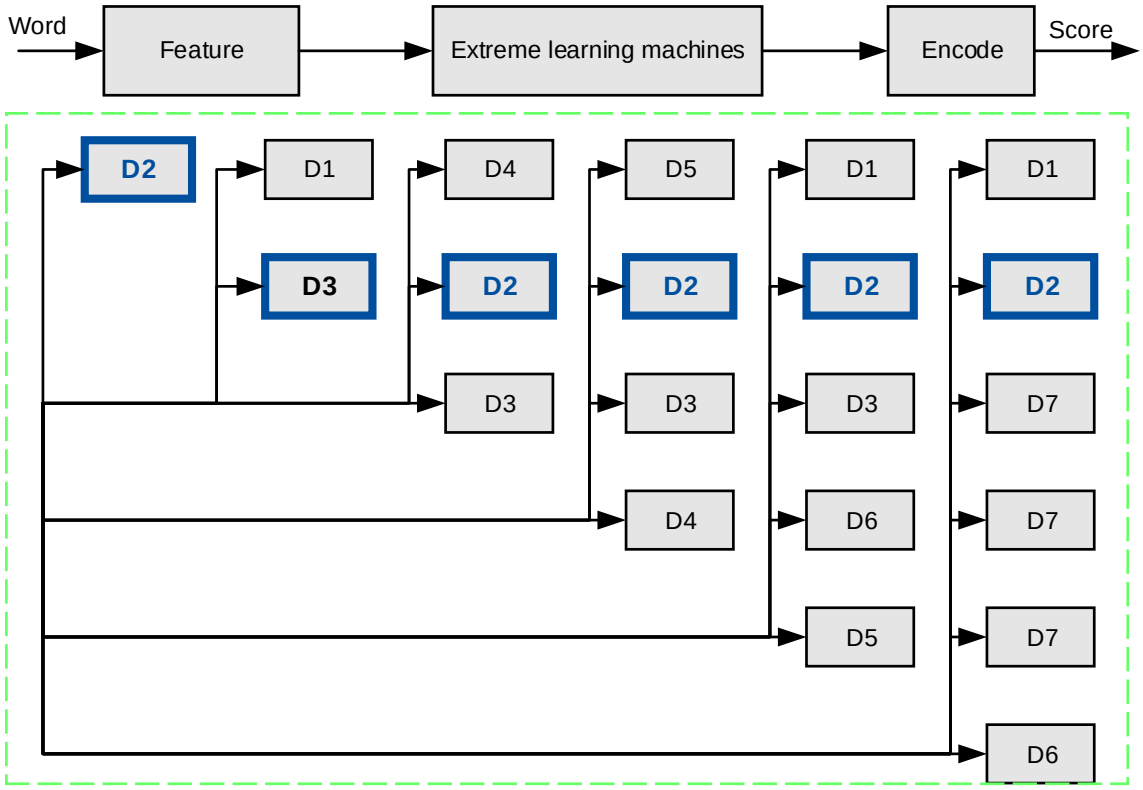

Figure 3.8: Weighted accent classification - Architecture

3.5.3 Accent decision

Classification results from multiple words are combined using a weighting scheme that improves overall performance. The output classes from each of the 21 ELMs (or SVMs) are tallied, and a score is given to each class according to the number of times that class was selected. The maximum count that any class can have is 6, and the count⇒score mapping is given in Fig. 3.9. The highest total score determines the overall dialect class.

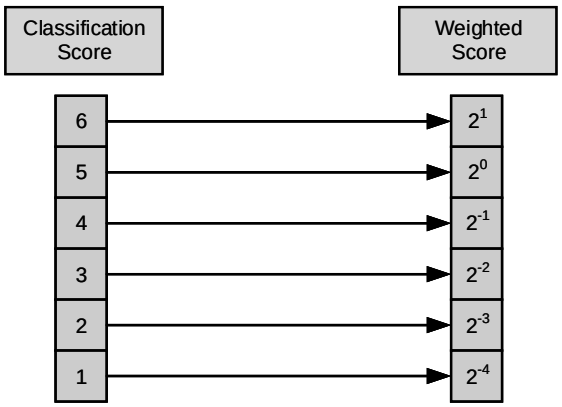

Figure 3.9: Weighted score

3.6 Experiment

3.6.1 Dataset

The dataset used in the experiment is TIMIT, a speech dataset developed by Texas Instruments (TI) and Massachusetts Institute of Technology (MIT) and considered as one of the standard datasets in speech research [88, 89]. The TIMIT dataset contains utterances from 630 speakers representing eight different dialect regions of the United States. The dialect regions are New England (D1), Northern (D2), North-Midland (D3), South-Midland (D4), Southern (D5), New York City (D6), Western (D7), and Army Brat. In TIMIT dataset, they used the term dialect for specifying these regions. To be consistent with the dataset we use the word dialect here. These utterances are read, so there are no word and grammar variations. The only variation in the acoustic waveform is the accent variations. For each utterance, the text, the signal sampled at 16 kHz, and hand-labeled segmentation at the word and phonetic level are provided. In our experiment, we used the first seven accent regions as the Army Brat accent group comprises speakers who moved around often during their childhood. For each speaker, we have ten utterances consisting of two accent sentences (SA) which are the same for each speaker, five phonetically compact sentences (SX) and three phonetically diverse sentences (SI). In our proposed method we are using

words from sentence "SA" as these words are available for each speaker.

3.6.2 Feature extraction

The TIMIT dataset is provided with word label information. Using word-label information, we extracted speech samples of words from the TIMIT dataset. These speech samples were normalized between -1 and 1. We extracted 12 Mel-frequency cepstral coefficients (MFCCs) [90] and normalized energy parameter using Auditory Toolbox [91]. We used a Hamming window and triangular filter bank for the MFCCs [92]. To incorporate temporal dependencies we used Δ and $\Delta - \Delta$'s coefficients. Delta (Δ) coefficients are computed by the regression Eqs. 3.24 and 3.25:

$$\Delta_i = \frac{\sum_{n=1}^{M} n(c_{i+n} - c_{i-n})}{2\sum_{n=1}^{M} n^2} \tag{3.24}$$

$$(\Delta - \Delta)_i = \frac{\sum_{n=1}^{M} n(\Delta_{i+n} - \Delta_{i-n})}{2\sum_{n=1}^{M} n^2} \tag{3.25}$$

For each word sample we have 39 dimension feature vectors consisting of 13 static cepstral feature, 13 Δ cepstral features, and 13 $\Delta - \Delta$'s cepstral features. The Δ's improve the accent classification accuracy by adding temporal dependencies.

3.6.3 ELMs and SVMs hyperparameters

During ELM training, the number of neurons in hidden layer was varied from 100 to 1000 with an increment of 100 and sigmoid is used as a non-linear activation function. The number of neurons in the hidden layer was learned using trial and error procedure based on cross-validation. For SVM training, a grid search method was used to find optimal SVM model parameters [93]. SVMs in the weighted accent classification algorithm was trained using LIBSVM library [94]. We used linear, polynomial, radial basis function, and sigmoid kernels with $d = \{1, 2, ..., 15\}$, $\gamma = \{2^{-15}, 2^{-14}, ..., 2^{5}\}$, and $C = \{2^{-3}, 2^{-2}, ..., 2^{15}\}$.

3.7 Results

The accuracy of the weighted accent classification algorithm is compared by using ELMs and SVMs as classifiers. Also, the performance comparison of different words and the improvement resulting from using multiple words from a particular speaker is evaluated. Finally, the relative computational time between ELMs and SVMs as classifiers in weighted accent classification is compared.

3.7.1 Comparison of different words

Eleven different words: “dark,” “like,” “oily,” “suit,” “that,” “wash,” “year,” “your,” “carry,” “water,” and “greasy” were used to classify speaker into one of the seven different accents. The words with three or more letters were selected so that they can capture variability in terms of accents and are available for all speakers in the TIMIT dataset. The weighted accent classification algorithm (Section 3.5) was tested using ELMs and SVMs as classifiers with only one word at a time. The performance of weighted accent classification algorithm was compared with multi-class classification. Figs. 3.10 and 3.10 show the comparison of weighted accent classification algorithm with multi-class classification using ELMs and SVMs as a classifier.

Weighted accent classification algorithm gives better results as compared with multi-class classification. The proposed weighted accent classification algorithm with ELM-based classifier performed best with the word “like” while the SVM-based classifier performed best with the word “carry”. In this experiment, only one word at a time is used for a particular speaker.

3.7.2 Classification accuracy and number of words

In this experiment, the improvement in accent classification accuracy obtained by using multiple words from a given speaker is compared. The number of words from one to

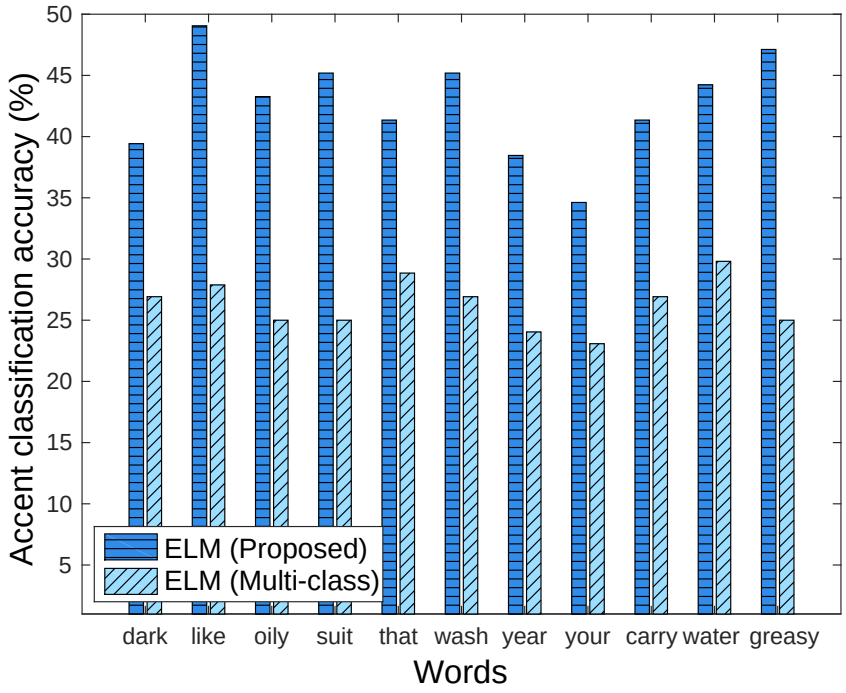

Figure 3.10: Comparison of classification accuracy with different words using ELM as a classifier

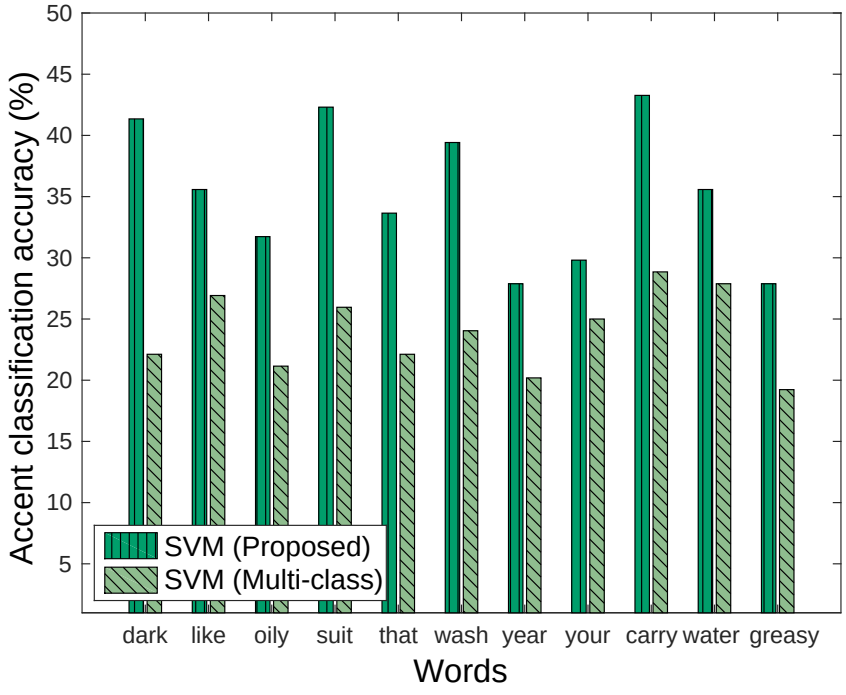

Figure 3.11: Comparison of classification accuracy with different words using SVM as a classifier

five are varied for a particular speaker. Fig. 3.12 shows the comparison of weighted accent classification algorithm with multi-class classification by using ELMs and SVMs as classifiers for multiple words.

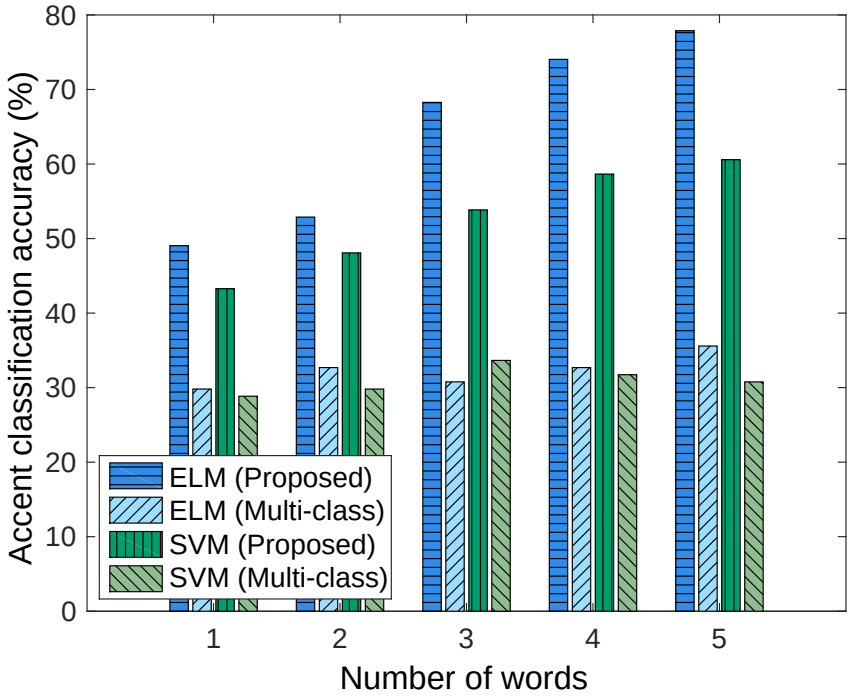

Figure 3.12: Comparison of classification accuracy with number of words

We used the top five words in terms of their performance as presented in Figs. 3.10 and 3.11. For weighted accent classification using ELMs, the following words are used: "like," "greasy," "suit," "wash," and "water." For multi-class classification using ELMs following words are used: "water," "that," "like," "wash," and "dark." Similarly, for weighted accent classification using SVMs the following words are used: "carry," "suit," "dark," "wash," and "water." For multi-class classification using SVMs the following words are used: "carry," "water," "like," "suit," and "your."

As the number of words from one to five are increased for a particular speaker, the weighted accent classification algorithm using ELMs and SVMs results in an improvement of accuracy from 49.04% to 77.88% and from 43.27% to 60.58% respectively. In the case of multi-class classification using ELMs and SVMs, there is a very slight improvement in

accent classification accuracy (multi-class ELMs 29.81% to 35.58% and multi-class SVMs 28.85% to 30.77%).

3.7.3 Classification accuracy of each accent

Fig. 3.13 shows the accent classification accuracy as a function of true accent. In this experiment, five words from each speaker are used. As shown in Fig. 3.13, accent D6 (New York City) shows the worst performance for both ELM and SVM classifiers. This is because speakers from accent region D6 (New York City) intermixed with speakers from accent region D1 (New England) and D2 (Northern).

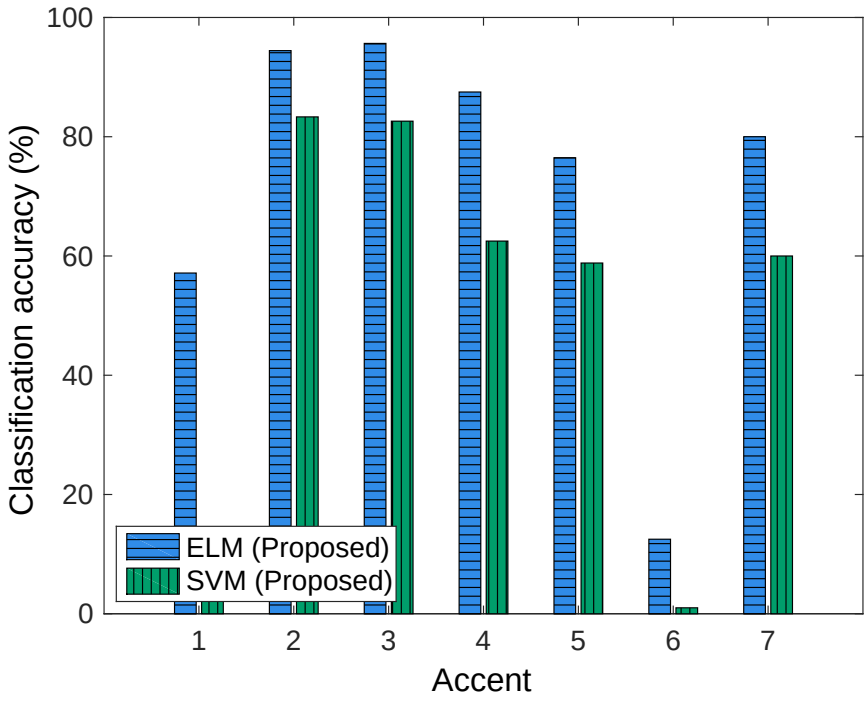

Figure 3.13: Classification accuracy per different accents

3.7.4 Comparison of ELMs and SVMs training and testing time

Using ELMs as classifiers for the accent classification algorithm gives a better accent classification accuracy relative to SVMs. Fig. 3.14 shows the relative comparison of training and testing time using ELMs and SVMs as classifiers for the proposed weighted accent

classification algorithm. ELMs take less time to train and operate by more than a factor of 2 relative to SVMs.

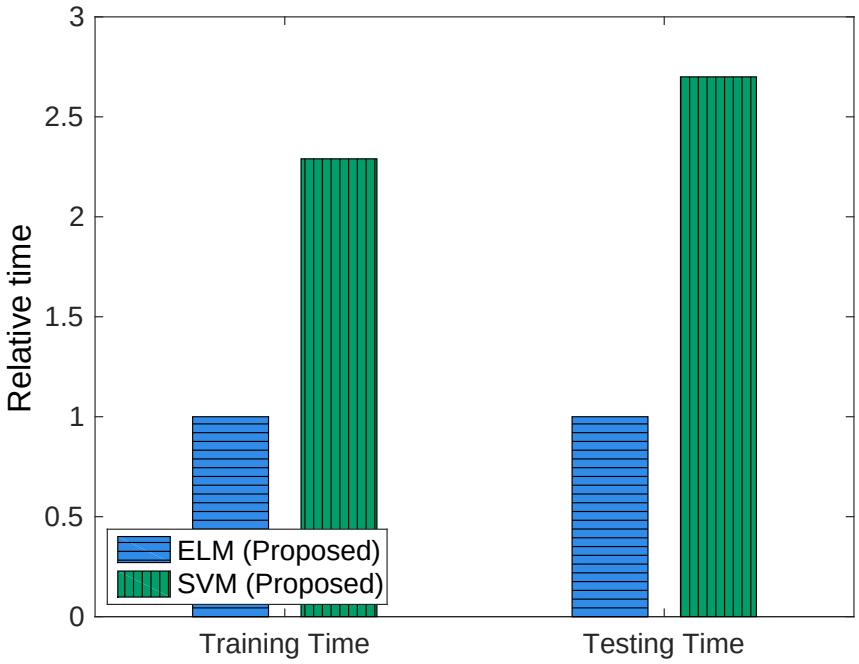

Figure 3.14: Comparison of ELMs and SVMs training and testing time

3.7.5 Comparison of accent classification results

As discussed, accent classification is a challenging problem, and it becomes more challenging on read sentences because word selection and sentence structure are not part of the message as in spontaneous speech. Different researchers have tried various approaches on different datasets, and most of the work is done in classifying accents of non-native speakers. In the empirical study of 23-way classification on a Foreign Accented English dataset [51], an average accuracy of 32.7% was obtained. J. Guarasain used acoustic methods to classify a German and Spanish group in an Foreign Accented English dataset [53]. By using GMMs and the Bayesian classifier, detection rates of 73% and 58.9% respectively were obtained. In the text-independent automatic accent classification using phoneme-based models, average classification accuracies of 64.90% at the phone level and 75.18% at the

word level for pairwise classification were obtained [52]. For a pool of four accents, the average classification accuracy rate was 37.57% at the phone level and 46.72% at the word level. In another study on the TIMIT dataset that used the most discriminating vowels, a detection rate of 42.52% was obtained [55].

To summarize, this chapter describes a novel accent classification algorithm based on ELMs. The algorithm uses five words from a speaker to differentiate between different accents and is comprised of three stages. In the first stage, a given word from a test speaker is presented as the input to 21 ELMs which are each trained to distinguish between two accents. In the second stage, the outputs of multiple ELMs are combined to obtain a classification score for that word. Finally, the classification score is encoded and optionally combined with the scores from other words and a decision about an accent class is based on the highest total score. Experiments were conducted on seven different accent groups from the TIMIT dataset. The proposed method classifies speakers into seven groups with an accuracy of 77.88% using five words from a given test speaker. Weighted accent classification algorithm is compared with SVMs as classifiers and also with multi-class classification using ELMs or SVMs.

CHAPTER 4
ADAPTIVE PHONEME CLASSIFICATION

Speech recognition is a complex problem because of inherent variability among speakers due to vocal tract length, dialect, accent, speaking rate, etc., and uncertainties such as environmental noise. Speech recognition systems seek to find a sequence of words corresponding to an acoustic waveform by splitting the acoustic waveform into small fragments known as frames. Data driven models are used to correctly identify the phonetic identity of the frames. The phonetic identity, along with language and pronunciation models are used for decoding words from the acoustic waveform.

The overall goal of the speech recognition system is to learn a model from given data (also known as training data) that generalizes well to testing data. These models can be learned either by using all the available training data to build a model before the testing sample is seen or by judiciously selecting a subset of exemplars from the training data to build a local model specifically for every test sample. In machine learning, the former belongs to global learning methods and the latter to local learning methods.

This chapter discusses adaptive phoneme classification method based on speaker similarity. The proposed method attempts to overcome the problems inherent in HMMs by utilizing the feature learning capabilities of DNNs. The proposed speaker similarity score algorithm requires a small amount of adaptation data from the target speaker to learn a speaker similarity score. Based on the speaker similarity score information, the algorithm adapts and predicts phonemes for the target speaker. Reduction in frame error rate yield an overall improvement in the performance of the speech recognition system.

This chapter is organized as follows: Section 4.1 reviews related work, Section 4.2 discusses DNNs for feature learning, Section 4.3 presents propose speaker similarity score algorithm, Section 4.4 discusses the dataset and experiments, and Section 4.5 presents re-

sults.

4.1 Related work

Currently, speech recognition systems are dominated by a statistical data modeling technique known as hidden Markov models (HMMs). HMMs are based on global learning methods to model the time-varying nature of the speech. In the past, GMMs were used to model the observation probabilities required by HMMs. Recently, DNNs have also become popular to convert speech inputs into observation probabilities. Although HMMs are popular and dominant in current speech recognition systems, they have weaknesses such as conditional independence and piecewise stationary assumptions. They fail to capture speaker-specific properties such as accents, dialects, *etc.* This is because HMMs learn a global model that generalizes well to all the samples present in the training data. In the process of learning a generalized global model, speaker specific properties are aggregated to determine statistical parameters of the model, resulting in a loss of speaker specific information. Also, they fail to generalize well to classes of examples in the training data which have small numbers of samples or observations (with limited training data, the models are incapable of representing the fine detail in the distribution of the data). Exemplar-based approaches do not share these weaknesses. An examplar-based approach can construct a local model using a very small amount of data. An exemplar-based approach keeps all the information from the training data, while methods such as HMMs build statistical approximations from the training data.

Exemplar-based approaches are popular in machine learning and are applied to various signal processing applications such as image processing [95, 96], video processing, and biomedical signal processing [97, 98]. In audio signal processing, exemplar-based approaches were initially applied for content based audio classification [99, 100, 101, 102, 103, 104], music genre classification [105, 106, 107], audio information retrieval, source separation [108], and speaker identification [109, 110, 111].

Various attempts have been made using exemplar-based approaches for speech recognition. The promising techniques based on an exemplar-based approach include k-NN classifiers, sparse representation, support vector machines, and template matching. A k-NN classifier searches the entire training data during classification, and the test sample is classified based on the class(es) of the closest neighbor or neighbors for a given distance measure. k-NN classifiers require minimal training [112, 113, 114, 115]. Sparse representation approaches represent the data as a linear combination of dictionary atoms which are learned from the training data [116, 117, 118]. The underlying structure of the data as embodied in the choice of dictionary atoms can then be exploited to make classification easier. Template-matching compares testing data with reference templates from training data by using dynamic time warping [119, 120, 121]. For extensive details on exemplar-based methods for speech recognition, are referred to an overview by T. Sainath et al. [122].

4.2 Feature learning using deep neural networks

Feature extraction plays a significant role in the accuracy of any classification engine. In the case of speech, extracted features should be chosen to represent information from the speech signal and at the same time eliminate information that is irrelevant for classification (*e.g.*, intensity of the speech signal, background noise). Speech signal comprises a small number of parameters produced by modulating a dynamical system. The underlying structure of speech is low dimensional and lies on a nonlinear manifold [123]. Several methods have been proposed to deal with the task of modeling and to represent speech signals. Some of them are based on physiological research on human hearing while others are human engineered and require significant expertise. Using DNNs for feature learning, one can discover abstractions from low-level features to high-level concepts with little human effort. These algorithms can learn nonlinear mathematical models with multivariate statistics related to each other by intricate statistical relationships [124]. A typical DNN architecture consists of an input layer, hidden layers and an output layer, as shown in Fig.

4.1.

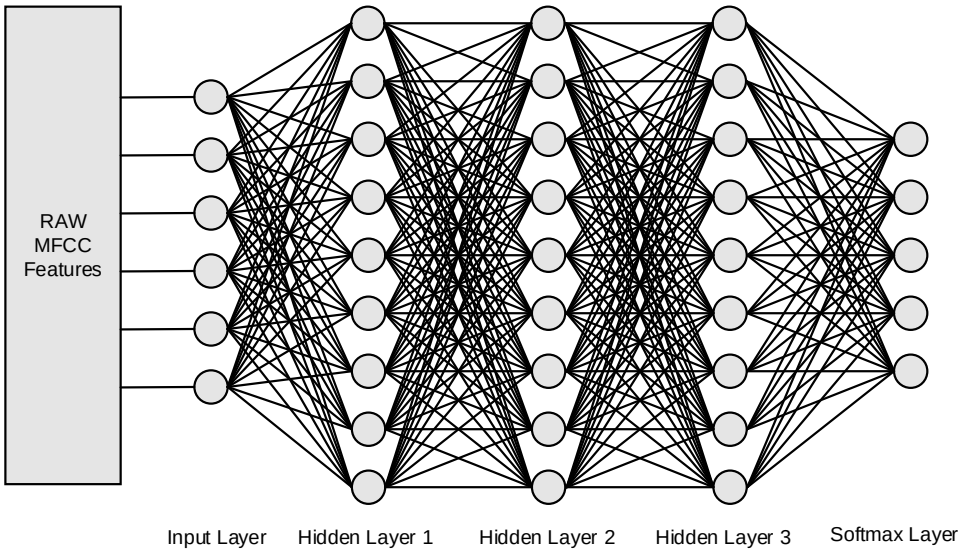

Figure 4.1: Deep neural networks

Mathematically we can write a neural network with H hidden layers as below

$$\mathbf{o}^l = \sigma(\mathbf{z}^l) = \sigma(\mathbf{W}^l\mathbf{o}^{l-1} + \mathbf{b}^l), 0 < l < H \tag{4.1}$$

where σ is an activation function given below.

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{4.2}$$

$\mathbf{o}^0 = \mathbf{x} \in \mathbb{R}^D$ is the input to the network, $\mathbf{W}^l \in \mathbb{R}^{\eta^l \times \eta^{l-1}}$ is the weight matrix, $\mathbf{b}^l \in \mathbb{R}^{\eta^l}$ is the bias vector, η^l is the number of neurons at layer l, and $\mathbf{o}^H = \mathbf{o}^* \in \mathbb{R}^P$ is the output of the neural network. The number of outputs, P, is the same as the number of phoneme classes $\Psi \in \{\psi_1, \psi_2, ..., \psi_P\}$.

Each neuron at the output layer represents one class of phonemes. We use the softmax function (ϕ) at the output to assign probability for each phoneme class as given by Eq. 4.3.

$$o_i^H \;=\; \phi_i(\mathbf{z}^H) \;=\; \frac{e^{z_i^H}}{\sum_{j=1}^{P} e^{z_j^H}} \tag{4.3}$$

The features learned using DNNs tend to eliminate irrelevant variabilities of raw input data and at the same time preserve information that is useful for classification. The network is trained using training data given by Eq. 4.4.

$$\mathbb{S} = \{(\mathbf{x}_i^I, \mathbf{y}_i^I) \mid 1 \leq i \leq N\} \tag{4.4}$$

where N is the total number of samples in our training data. $\mathbf{x}_i^I$ and $\mathbf{y}_i^I$ is the i^{th} input feature and corresponding output vector respectively. Cross entropy is used as cost function given by Eq. 4.5.

$$J(\mathbf{W}, \mathbf{b} : \mathbf{x}^I, \mathbf{y}^I) = -\sum_{i=1}^{P} y_i^I \log o_i^H \tag{4.5}$$

To learn optimal $\mathbf{W}$ and $\mathbf{b}$ parameters we use the backpropagation learning algorithm using all the training samples as given below.

$$\mathbb{J}(\mathbf{W}, \mathbf{b}; \mathbb{S}) \;=\; \frac{1}{N} \sum_{i=1}^{N} J(\mathbf{W}, \mathbf{b} : \mathbf{x}_i^I, \mathbf{y}_i^I) \tag{4.6}$$

Once the deep neural network is trained using backpropagation, the output of the last hidden layer is used as new features. The inputs consist of 13 raw Mel-frequency cepstral coefficients along with the first and second temporal differences (the deltas and delta-deltas). A deep neural network as feature extractor is shown in Fig. 4.2. The data after the last hidden layer activations are used as features for training and testing speaker similarity score algorithm discussed in Section 4.3.

4.3 Adaptive phoneme classification

The speaker similarity score algorithm uses a k-NN approach on adaptation data from the target speaker to learn the speaker similarity score (see Fig. 4.3). The speaker similar-

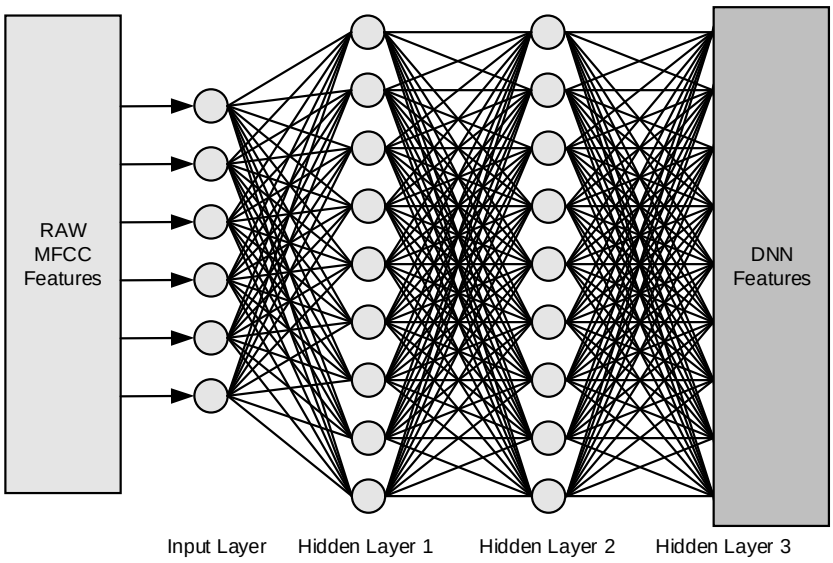

Figure 4.2: Deep neural network as feature extractor

ity score is used to adapt a distance metric of the k-NN classifier for adaptive phoneme classification as shown in Fig. 4.4. Using k-NN for phoneme classification, the algorithm estimates a target function locally and differently for each new speaker instead of estimating the target function over the entire instance space. This has a significant advantage when the target function is very complex but can still be described by a collection of less complex local approximations [125].

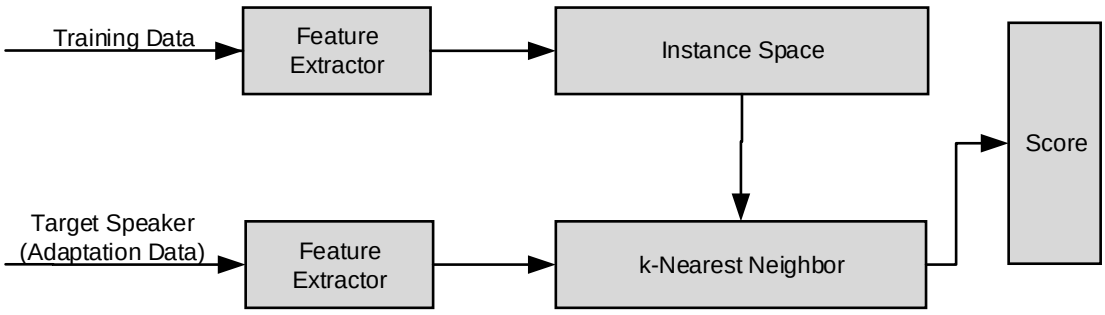

Figure 4.3: Speaker similarity score

The instance space consists of phoneme samples from the training data in the DNN

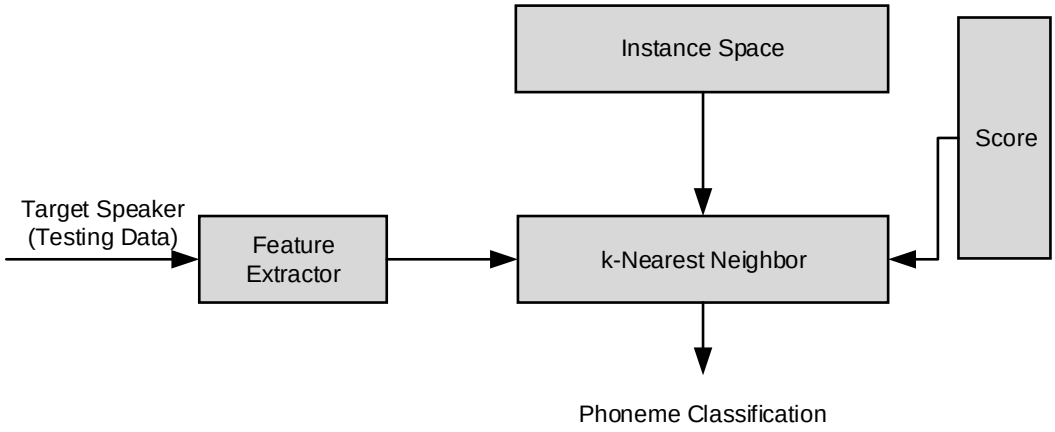

Figure 4.4: Adaptive phoneme classification

feature space. For each phoneme frame instance, we have a corresponding phoneme label and speaker information available as shown in Fig. 4.5. The instance space in total consists of 'N' phoneme frame samples from continuous speech as given by Eq. 4.7.

$$Phoneme\ \ Samples\ =\ \left\{\mathbf{x}_1^I, \mathbf{x}_2^I, \mathbf{x}_3^I, ..., \mathbf{x}_N^I\right\} \tag{4.7}$$

$\mathbf{x}_i$ is a feature vector that lies in 'M' dimensional DNN feature space $\mathbb{R}^M$ as given by Eq. 4.8.

$$\mathbf{x}_i^I\ =\ \left\{x_{i1}^I, x_{i2}^I, x_{i3}^I, ..., x_{iM}^I\right\} \tag{4.8}$$

For each of these phoneme speech samples from the instance space, corresponding phoneme and speaker labels are given by Eqs. 4.9-4.10.

$$Phoneme\ \ Labels\ =\ \left\{\mathbf{y}_1^I, \mathbf{y}_2^I, \mathbf{y}_3^I, ..., \mathbf{y}_N^I\right\} \tag{4.9}$$

$$Speaker\ \ Labels\ =\ \left\{\mathbf{z}_1^I, \mathbf{z}_2^I, \mathbf{z}_3^I, ..., \mathbf{z}_N^I\right\} \tag{4.10}$$

where, $\mathbf{y}_i^I\ \in\ \{\psi_1, \psi_2, \psi_3, ..., \psi_P\}$ and $\mathbf{z}_i^I\ \in\ \{s_1, s_2, s_3, ..., s_L\}$. There are P phoneme

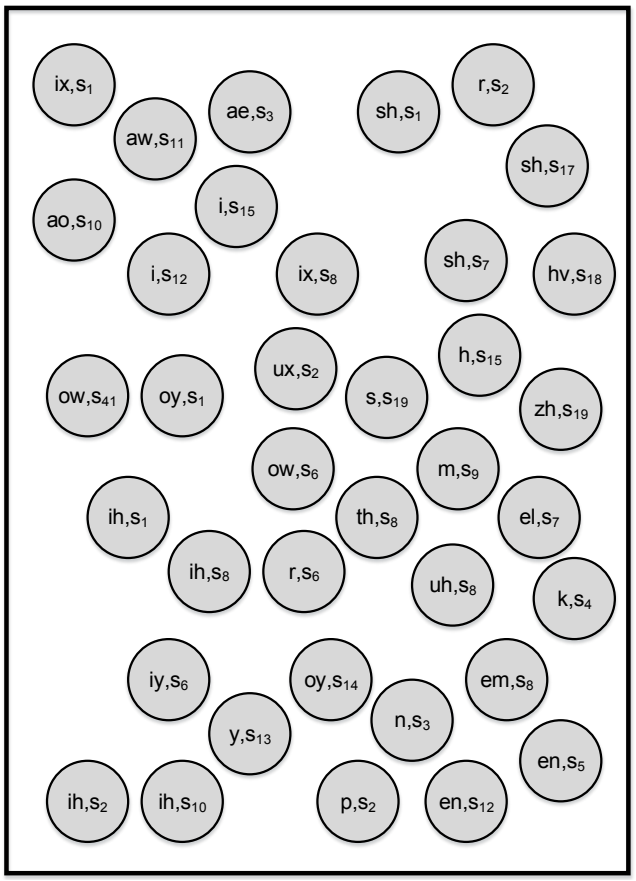

Figure 4.5: Instance Space. Each gray circle represents phoneme samples from training data. For each of these phoneme speech samples we have phoneme label and speaker information available. There are 'L' speakers and 'P' phoneme classes. For simplicity and clarity we have shown our instance space in two dimensions with limited phoneme and speaker labels.

classes and L different speakers in our instance space.

The adaptation data used from the target speaker for learning the speaker similarity score, and evaluated our approach on the testing data from the target speaker for adaptive phoneme classification. In the training phase, target speaker similarity scores are learned by using k-NN. While in the testing phase predictions about the phoneme classes are made by combining target speaker similarity score information with k-NN.

4.3.1 Speaker Similarity Score

When a new target speaker is encountered, we must first learn a speaker similarity score, γ_i, for each of the speakers, s_i, in the instance space relative to the target speaker. This is done using k-NN—a non-parametric, lazy learning algorithm widely used in various types of classification, estimation, and prediction problems due to its simplicity and versatility. Because of its non-parametric nature k-NN does not require building statistical models of the underlying data. While statistical models provide structure that aids in generalization, they may also result in a loss of information because of that generalization. Using a non-parametric approach, different training observations are not generalized but are rather retained and used for doing fine comparison between the input samples with the training observations resulting in better speech recognition.

For the target speaker the adaptation data consists of Q phoneme frame samples in the DNN feature space and corresponding phoneme labels given by Eqs. 4.11 & 4.12

$$Adaptation\ Phoneme\ Samples\ =\ \left\{\mathbf{x}_1^A, \mathbf{x}_2^A, ..., \mathbf{x}_Q^A\right\} \tag{4.11}$$

$$Adaptation\ Phoneme\ Labels\ =\ \left\{\mathbf{y}_1^A, \mathbf{y}_2^A, ..., \mathbf{y}_Q^A\right\} \tag{4.12}$$

where, $\mathbf{y}_i^A \in \{\psi_1, \psi_2, \psi_3, ..., \psi_P\}$. To find the speaker similarity scores, $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_L\}$, we initialize $\gamma_i = 0$ for $i = 1, \ldots, L$. Then we iterate through the Q adaptation phoneme samples and find those speakers among the L different speakers in the instance space who best predict the labels of the adaptation samples using k-NN as follows. For each adaptation phoneme sample, $\mathbf{x}_j^A$, find the k_s most similar phoneme samples in our instance space using the Euclidean distance measure given by Eq. 4.13:

$$\rho(\mathbf{x}_i^I, \mathbf{x}_j^A) = \sqrt[2]{\sum_{k=1}^{M}(x_{ik}^I - x_{jk}^A)^2} \tag{4.13}$$

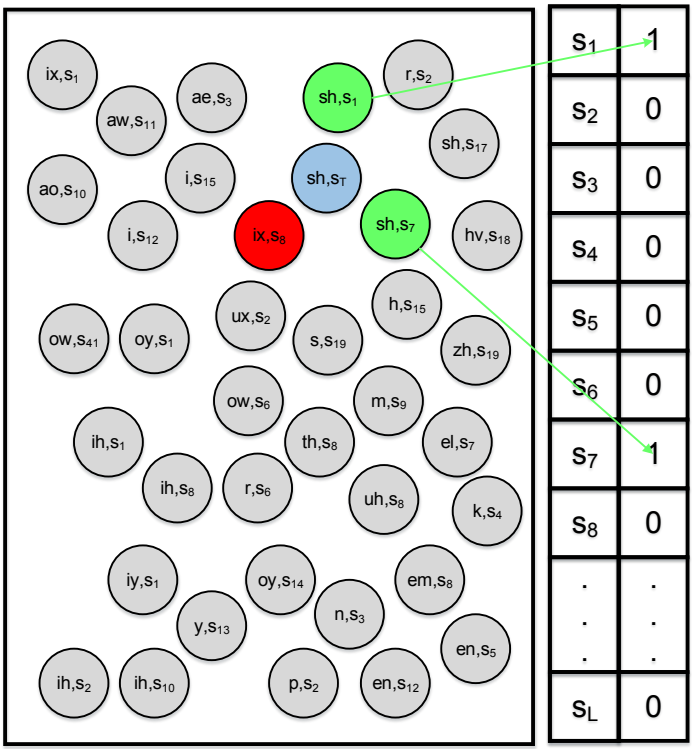

Figure 4.6: Speaker similarity score calculation. The blue circle indicates an adaptation phoneme speech sample from the target speaker. For this adaptation phoneme speech sample from the target speaker we have phoneme label information available (*i.e.* "sh"). The green circle represents k-nearest neighbor speech samples from training data similar to the target speaker adaptation phoneme speech sample ("sh"). The red circle represents the k-nearest neighbor phoneme speech sample that does not match with the target speaker adaptation speech sample ("ix"). For speaker similarity score, we will incrementally increase the score of speakers correspoding to correct phoneme speech samples (*i.e.* "s_1" and "s_7") by "1"

Each of the k_s nearest samples found in the instance space will either have the same label as the adaptation sample, $\mathbf{x}_j^A$, or a different label. For each phoneme match, that is for every i for which $\mathbf{y}_i^I = \mathbf{y}_j^A$ among the k_s nearest neighbors of $\mathbf{x}_j^A$, we increase the score, γ_i, of that particular speaker by one as shown in Fig. 4.6. Selection of "k_s" can be done in a heuristic way or by cross-validation. This process is repeated for all the phoneme samples from our adaptation data of the target speaker to find the total score, γ_i, for each of the speakers in our instance space as shown in Fig. 4.7. The greater the score of a particular

speaker in our instance space, the more similar it is to the given target speaker.

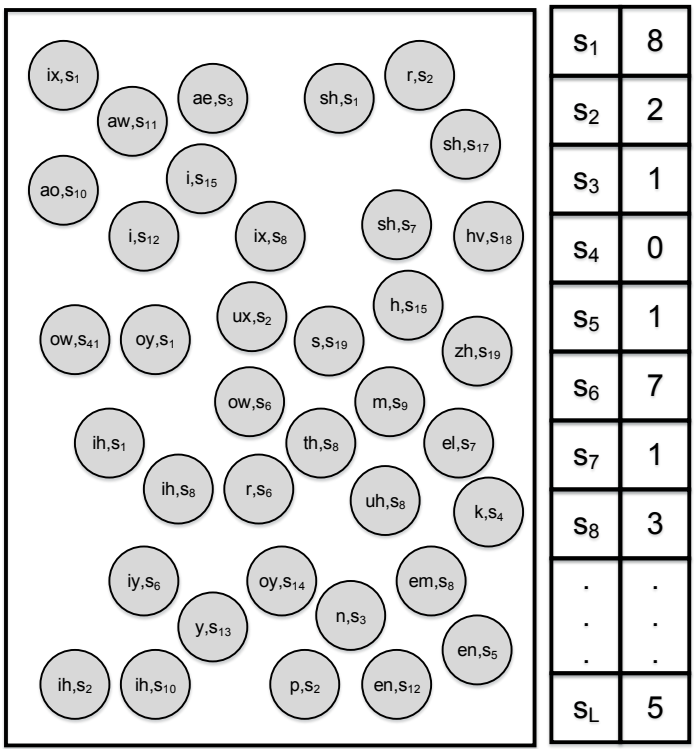

Figure 4.7: Speaker similarity score after going through all adaptation phoneme samples from the target speaker

4.3.2 Adaptive Phoneme Classification

For adaptive phoneme classification the speaker similarity scores γ found for the target speaker are used to modify a k-NN classifier. Given an unknown phoneme from the target speaker testing data, the k-nearest neighboring phonemes in the instance space are found using the same Euclidean distance measure described previously. The classification decision for the unknown phoneme is then made by a vote of these k nearest instance-space samples; however, the votes are weighted according to the corresponding speaker similarity scores. In other words, for each phoneme represented among the k-nearest neighbors, the corresponding speaker similarity scores, γ_i, are added. The classifier then assigns a phoneme label to the given test data based on the highest score as shown in Fig. 4.8.

Algorithm 1: Speaker similarity score

Result: Speaker similarity score of a target speaker
Input : $\mathbf{x}^I$, $\mathbf{y}^I$, $\mathbf{z}^I$, $\mathbf{x}^A$, $\mathbf{y}^A$, k_s
Output: γ

/* L is the number of speakers in $\mathbf{z}^A$ */
/* Q is the number of samples in $\mathbf{x}^A$ */
Initialize $\gamma_i = 0$ for $i = 1, \ldots, L$

for $j = 1$ *to* Q **do**
  $\zeta = \text{knn}(\mathbf{x}^I, \mathbf{x}_j^A, k_s)$ /* ζ is a vector containing the indices of the k_s nearest neighbors in $\mathbf{x}^I$ to $\mathbf{x}_j^A$. */
  for $m = 1$ *to* k_s **do**
    if $\mathbf{y}^I_{\zeta(m)} == \mathbf{y}^A_j$ **then**
      /* Increase the speaker score for index $\zeta(m)$ */
      $\gamma_{\zeta(m)} = \gamma_{\zeta(m)} + 1$
    end
  end
end

Algorithm 2: Adaptive phoneme classification

Result: Predicted phoneme
Input : $\mathbf{x}^I$, $\mathbf{y}^I$, γ, $\mathbf{x}_i^T$, k_{prd}
Output: $\mathbf{p}$

$\zeta = \text{knn}(\mathbf{x}^I, \mathbf{x}_j^T, k_{prd})$ /* ζ is a vector containing the indices of the k_{prd} nearest neighbors in $\mathbf{x}^I$ to $\mathbf{x}_j^T$. */
Initialize $\vartheta_i = 0$ for $i = 1, \ldots, P$
/* The score for each phoneme. */
for $m = 1$ *to length*(Ψ) **do**
  for $i = 1$ *to* k_s **do**
    /* If the i^{th} nearest neighbor is phoneme ψ_m then increment the score (ϑ_m) for ψ_m according to the corresponding speaker score. */
    if $\mathbf{x}^I_{\zeta(i)} == \psi_m$ **then**
      $\vartheta_m = \vartheta_m + \gamma_{\zeta(i)}$
    end
  end
end
$m^* = \arg\max_m(\vartheta_m)$ /* Get the index of the largest score. */
$\mathbf{p} = \psi_{m^*}$ /* Return the predicted phoneme label. */

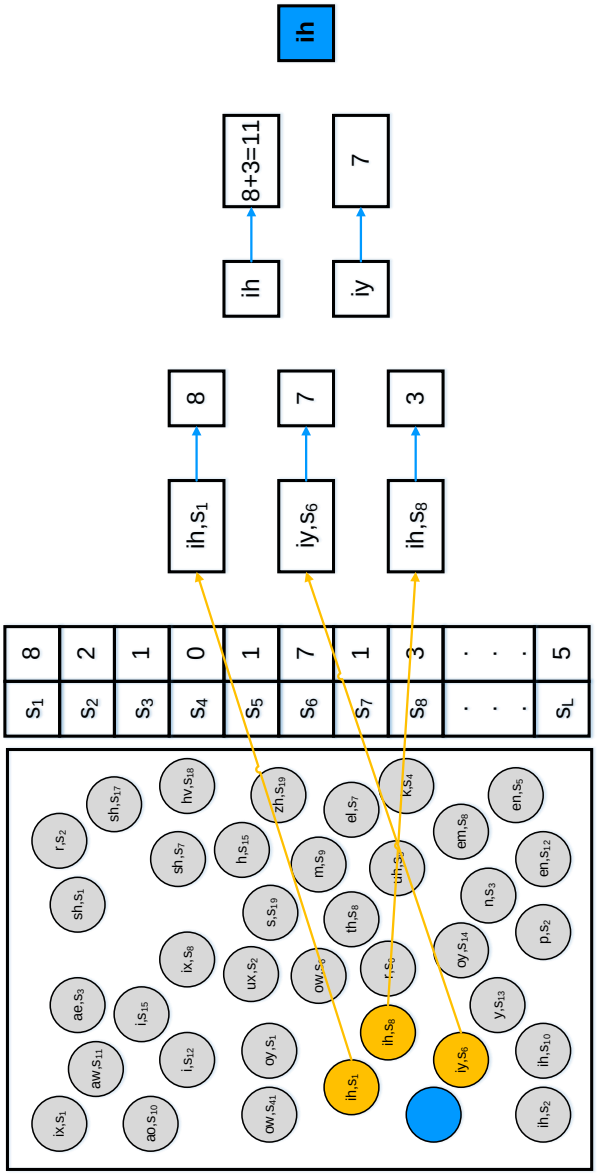

Figure 4.8: Adaptive phoneme classification. Blue circle represents target speaker testing phoneme sample. Orange circle shows k-nearest neighbor phoneme samples from the training data. First, it looks for the speaker similarity score of each of these speakers in orange circle with the target speaker using the speaker similarity score learned in the last step. *e.g.* speaker s_1, s_6, and s_8 speaker similarity score is 8,7, and 3 respectively. Then it looks for the unique phonemes in k-nearest neighbor. *e.g.* here "ih" and "iy". It adds the speaker score for similar phonemes. *e.g.* adding the score of speaker s_1 and s_8. *e.g.* $8 + 3 = 11$ Now, phoneme "ih" has a value of 11 and "iy" has a value of "7". Finally, it makes the decision based on highest score. *e.g.* "ih" has the highest score, so phoneme decision here is "ih"

4.4 Experiment

4.4.1 Dataset

The dataset used in this experiment is the TIMIT, which consists of 630 speakers from eight different dialect regions of the United States [88, 89]. For each speaker, ten prompted utterances (2 SA, 5 SX and 3 SI), which are phonetically rich are available. These utterances are accompanied with transcriptions at the word and phoneme levels. The dataset is divided into training, validation, and testing data. Testing data consists of 24 speakers (3 speakers from each dialect region). [7]. Originally, the phoneme labels in the dataset were categorized into 61 phoneme classes. These 61 phoneme classes are mapped into 39 phoneme classes. This is standard practice for experimentation on TIMIT dataset by merging those with similar sounds and combining "closures" with "stops" [126]. There are $1,477,824$ and $57,919$ frames in the training and testing set respectively. From training set, 10% frames are held out as a validation set for tuning hyper-parameters. In experiments, phoneme classification is done on speech frames.

4.4.2 Feature Learning Using DNN

For feature extraction, a DNN is trained using speakers from training data comprising sentences "SX" and "SI". Seven different architectures of DNN for feature extraction are used. Architecture "A" comprises one layer, architecture "B" comprises two layers and architecture "C" comprises three layers, and so on. We varied the number of neurons in the last hidden layer as shown in Table 4.1, where value of η^H can be either one of these $\{25, 50, 100, 200, 400, 800\}$. The speech signal is divided into hamming windows of 25ms with a frame rate of 10ms. For each window 39 (13 static, 13 delta and 13 delta-delta) Cepstral coefficients were computed. For the evaluating frame, we included three contextual frames on each side. This makes the dimension of input vector 273 (39x7). Each frame was labeled as one of the 39 standard phoneme class. DNNs are trained using backpropagation

Table 4.1: Deep neural network architectures

Name	Layers (H)	Neurons in each hidden layer (η^l)
DNN "A"	$H = 1$	$\eta^i \in \{25, 50, 100, 200, 400, 800\}$, $i = 1$
DNN "B"	$H = 2$	$\eta^i = 1000$,$i = 1$, $\eta^H \in \{25, 50, 100, 200, 400, 800\}$
DNN "C"	$H = 3$	$\eta^i = 1000$,$i = 1, 2, ..., (H-1)$, $\eta^H \in \{25, 50, 100, 200, 400, 800\}$
DNN "D"	$H = 4$	$\eta^i = 1000$,$i = 1, 2, ..., (H-1)$, $\eta^H \in \{25, 50, 100, 200, 400, 800\}$
DNN "E"	$H = 5$	$\eta^i = 1000$,$i = 1, 2, ..., (H-1)$, $\eta^H \in \{25, 50, 100, 200, 400, 800\}$
DNN "F"	$H = 6$	$\eta^i = 1000$,$i = 1, 2, ..., (H-1)$, $\eta^H \in \{25, 50, 100, 200, 400, 800\}$

algorithm with cross entropy error as the objective function (Eq.4.5). The training of the DNN is further optimized using momentum. Once the DNN is trained, the softmax layer is removed and outputs at the last hidden layer activations are used as features as discussed in Section 4.2.

4.4.3 Speaker Similarity Score Algorithm

The instance space for the speaker similarity score algorithm comprises phoneme samples from sentences SI and SX from the training dataset. An input comprising Mel-frequency cepstral coefficients of phonemes from target speaker-adaptation data are fed as input to a DNN feature extractor as shown in Fig. 4.3. For adaptive phoneme classification speaker similarity score calculated for the target speaker from adaptation data is used in above step. Input comprising Mel-frequency cepstral coefficients of target speaker-testing data is fed as input as shown in Fig. 4.4.

4.5 Results

4.5.1 Comparison with baseline system

The speaker similarity score (SSS) algorithm for phoneme frame error rate is compared with a baseline system. The baseline system uses softmax layer at the output of DNN for phoneme classification of frames. Fig. 4.9 shows the comparison of the speaker similarity

score algorithm with the baseline system for DNN architectures "B (2 layers)," "D (4 layers)," and "F (6 layers)." We also varied the number of neurons in the last hidden layer. The speaker similarity score algorithm gives 1.93%, 2.43%, 2.65%, 2.32%, 2.01%, and 1.37% absolute reduction in frame error rate with 25, 50, 100, 200, 400, and 800 neurons respectively in the last hidden layer for DNN architectures as compared with the baseline system. The value of 'k' used for speaker similarity score (k_s) and adaptive phoneme prediction (k_{prd}) in this experiment was 40 and 50 respectively.

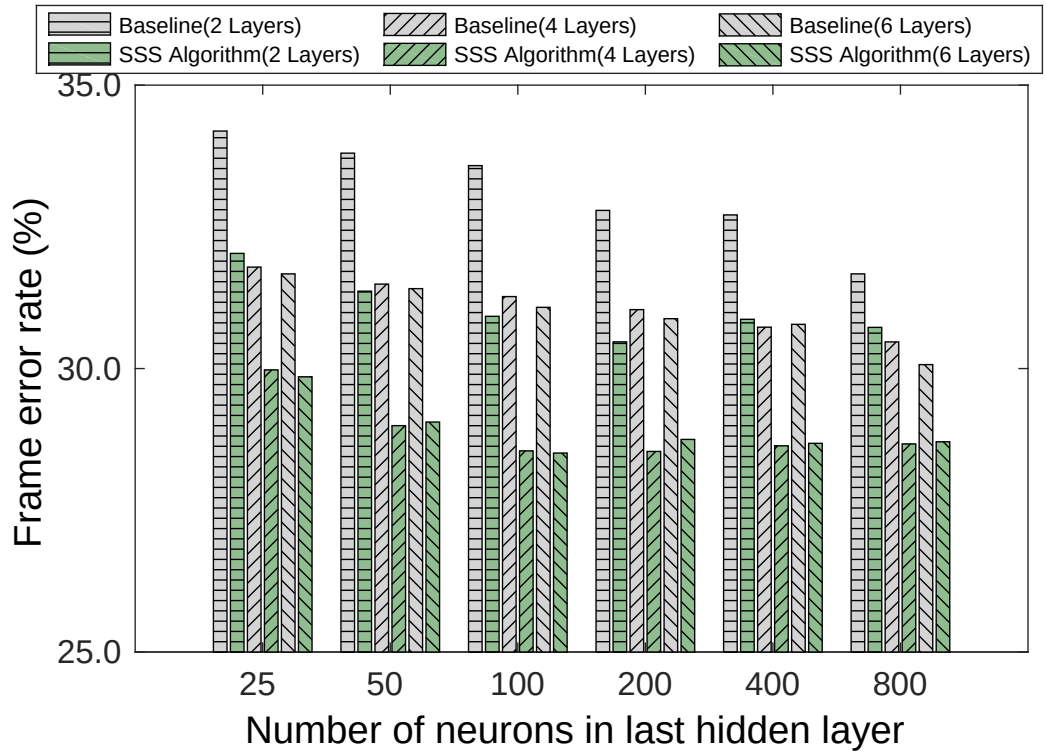

Figure 4.9: Comparison of the speaker similarity score algorithm (SSS) with baseline system

4.5.2 Deep neural network architecture

Fig. 4.10 shows comparison of frame error rate with the variation in number of neurons in the last hidden layer. Since our approach is based on k-NN, and computational cost of

k-NN increases with an increase in the dimension of feature vector. The number of neurons in last hidden layer decides the dimension of our feature vector for speaker similarity score algorithm. As we decrease the number of neurons from 800 to 25 in the last hidden layer, frame error rate also increases. But, there is a very slight increase in frame error performance if we use 50 neurons in the last hidden layer with a higher number of layers in the DNN. The value of "k" used for speaker similarity score and adaptive phoneme classification in this experiment was 30 and 30 respectively.

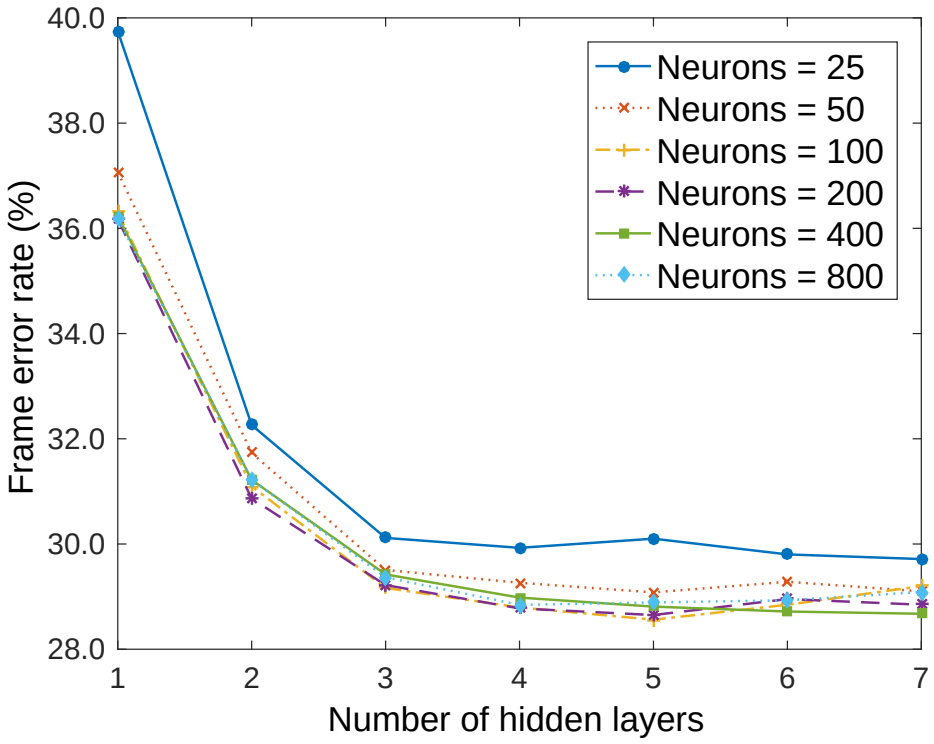

Figure 4.10: Comparison of deep neural network architecture

4.5.3 Value of 'k' for speaker similarity score and phoneme classification

In this experiment we varied the value of 'k' in k-NN for learning speaker similarity score and doing adaptive phoneme classification. Finding the optimal value of 'k' in k-NN depends on the structure of data in the instance space. In general, a large value 'k' gives better classification performance. But, very large value of 'k' may result in an over-smooth

decision boundaries. Using very small value of 'k' results in a noisy decision boundary. To manage these tradeoffs, 'k' is learned over the validation data by varying value of 'k' incrementally. Fig. 4.12 shows performance of the speaker similarity score algorithm with variations in value of 'k' for speaker similarity score (k_s) and adaptive phoneme classification (k_{prd}). The optimal value of 'k' for speaker similarity score and phoneme classification

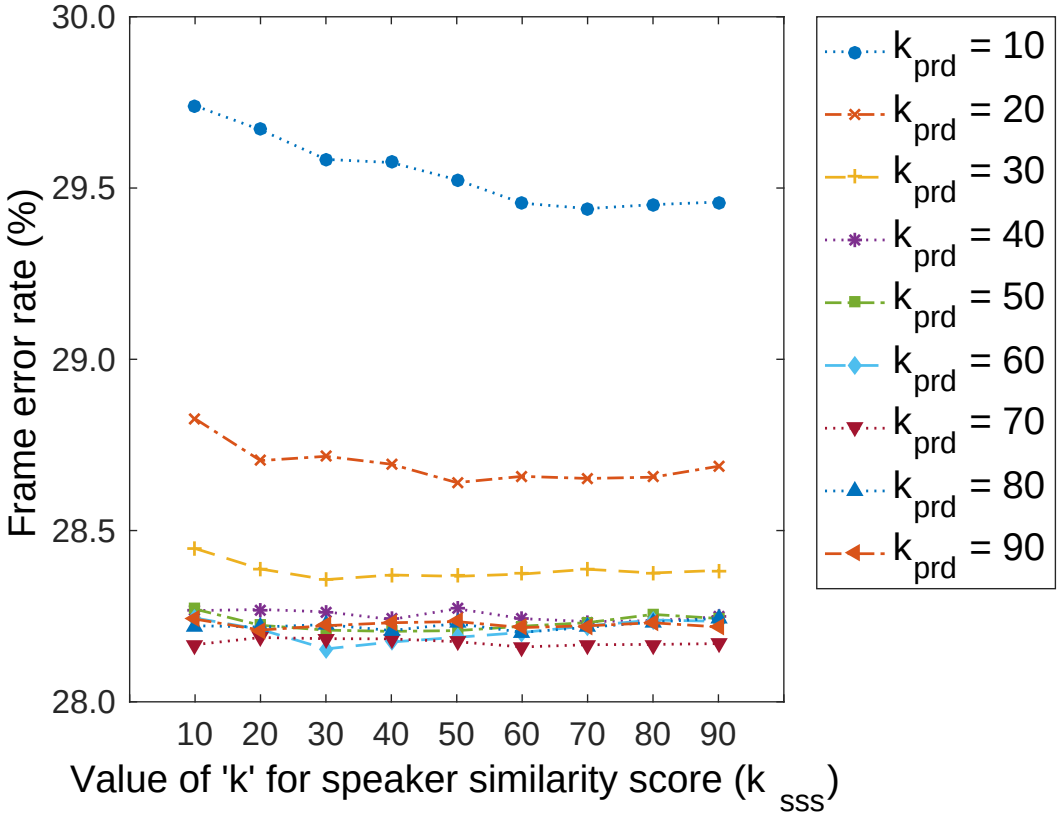

Figure 4.11: Variation with 'k' for speaker similarity score

was 30 and 60 respectively. Fig. 4.11 shows phoneme frame error rate performance of the speaker similarity score algorithm with variations in value of 'k' for speaker similarity score (k_s) calculation. Similarly, we compared phoneme frame error rate with variation in the value of 'k' for phoneme prediction (k_{prd}) for various values of 'k' for speaker similarity score calculation (k_s).

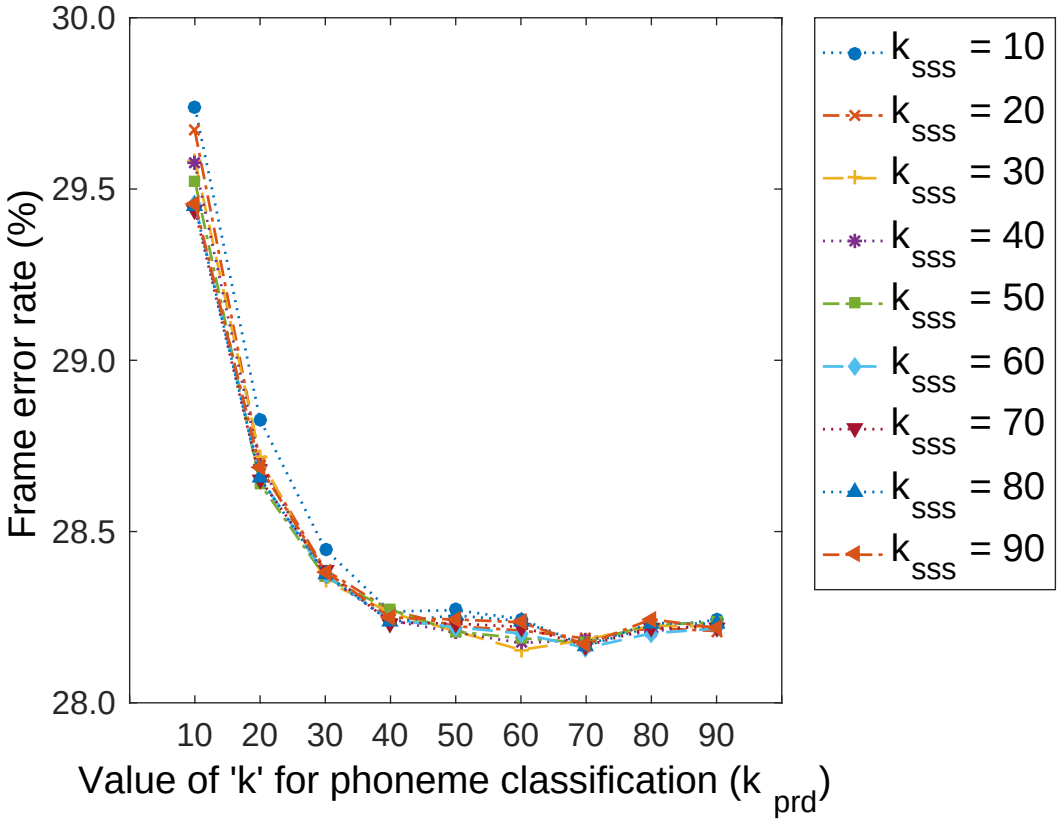

Figure 4.12: Variation with 'k' for phoneme classification

4.5.4 Comparison with number of sentences

In this experiment we varied the number of sentences for learning the speaker similarity score. As we increase the number of sentences form 1 to 4, phoneme frame error rate reduces from 30.2% to 28.8%. The value of "k" used for speaker similarity score and adaptive phoneme classification in this experiment were 30 and 60 respectively. We used DNN architecture "F".

4.5.5 Speaker-wise comparison

We compared speaker-wise phoneme frame error rate by using optimum values of 'k' for speaker similarity score (k_s) calculation and phoneme prediction (k_{prd}). Fig. 4.13 shows

the phoneme frame error rate for the speakers in our testing data.

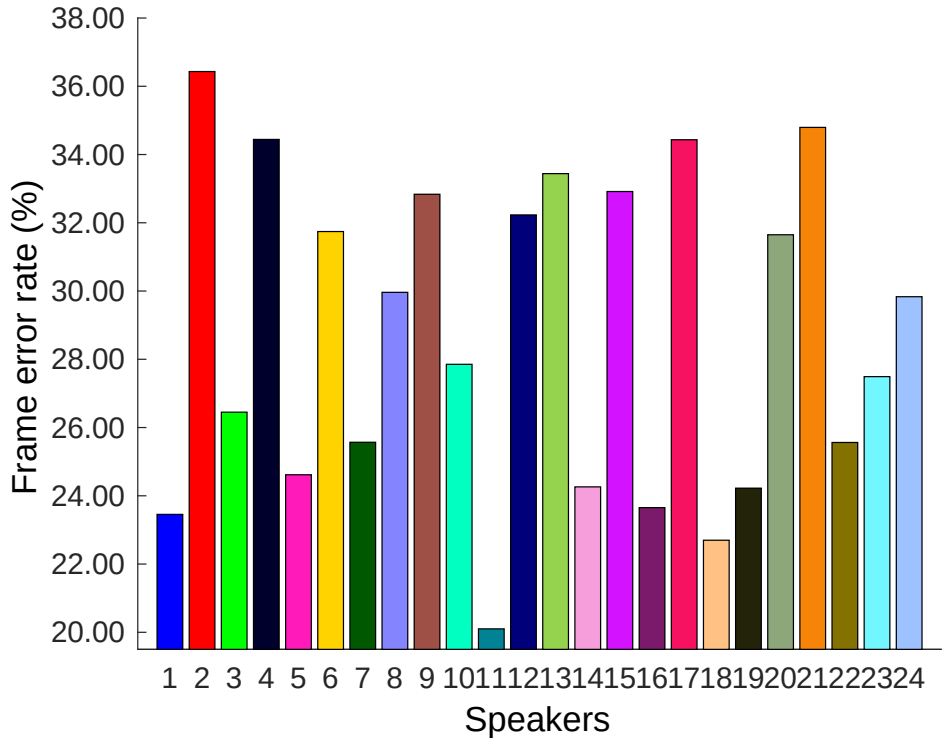

Figure 4.13: Speaker-wise comparison

4.5.6 Comparison of Results

We repeated our experiments twenty times by changing the sentences used for learning the speaker similarity score and phoneme prediction. The average phoneme frame error obtained on these twenty experiments is 29.17% with a standard deviation of 0.3%. The minimum phoneme frame error obtained in these experiments is 28.75%. A. Graves et al. trained bidirectional Long Short Term Memory (LSTM) networks on the TIMIT dataset by modifying full gradient version of the LSTM learning algorithm [127]. They compared various neural network architectures for phoneme frame error rate. Based on their findings, bidirectional networks provide better performance as compared with unidirectional networks. LSTMs are quick to train as compared with recurrent neural networks (RNNs) and also give better performance. They achieved a phoneme frame error rate of 30.2%, 31.0%,

34.0%, 34.8%, and 36.9% for bidirectional LSTM (BLSTM), bidirection RNN (BRNN), LSTM, RNN, and multilayer perceptron (MLP) respectively. J. Labiak et al. compared distance metric in k-NN for phonetic frame error rate [115]. The authors compared a standard Euclidean distance metric with two learned Mahalanobis distance metric based on large-margin nearest neighbors (LMNN) and locality pre-serving projections (LPP). Locality sensitive hashing was used for approximate nearest neighbor search in order to decrease the computational time of k-NN classifier. The phonetic frame error rate of 36.92% and 36.72% were obtained with LMNN (k=30) and LPP (k=38) respectively. The dimensions of phoneme sample were 195 and 130 for LMNN and LPP respectively. A. Dhaka et al. used semi-supervised learning based on sparse autoencoders. The authors tested their approach with varying proportions of labelled and unlabelled data for phoneme frame error rate on the TIMIT dataset. They obtained a phoneme frame error rate of 30.35% [128].

The novel speaker similarity score algorithm presented in this chapter can reduce phoneme frame error rate on features learned through DNNs. The algorithm requires a small amount of adaptation data from the target speaker. The adaptation algorithm calculates speaker similarity score for the target speaker using the training data. On the basis of a speaker similarity score of the target speaker with speakers in instance space, it uses the nearest neighbor classifier for phoneme prediction.

CHAPTER 5
ROBUST PHONEME CLASSIFICATION

K-nearest neighbor (k-NN) classifier can learn a nonlinear decision surface and requires only one hyper parameter (i.e. the value of "k") for training. The classification performance improves as we increase the amount of training data. With an increase in the amount of training data, computational and memory requirements also increase as it has to store and search through the entire training data for classification of one test point. This chapter in- vestigates multiple methods to reduce the computational time of speaker similarity score algorithm for phoneme classification. The chapter is organized as follows: Section 5.1 reviews methods to reduce the computational cost of k-nearest neighbor, Section 5.2 discusses neighborhood components analysis, Section 5.3 discusses a novel approach for reducing the computation time of k-NN by reducing samples in instance space, and Section 5.4 presents results and comparison of these methods.

5.1 Related Work

The performance of k-NN classifier improves as we increase the amount of training data. For a huge amount of data and large value of "k," error rate of k-NN approaches Bayes error rate [129]. Also, using a large value of "k" reduces over-fitting. The computational cost of k-NN is high as it stores all the samples from the training data for classification of a single test sample. Features learned through DNNs lie in a high dimensional space [124]. Using k-NN in DNN feature space results in high computational cost. One such method to reduce computational cost of k-NN in a high dimensional space is to apply dimensionality reduction techniques. Various techniques have evolved such as: principal component analysis (PCA) which finds the linear projection of data that captures maximum variability in an unsupervised manner [130], linear discriminant analysis (LDA) that learns

linear combination of features in such a way that preserves class labels discriminatory information [131], and locally linear embedding (LLE) which preserves local symmetries [132]. For details of dimensionality reduction techniques, readers are referred to [133, 134].

Another method to reduce the computational cost of k-NN is to reduce the number of samples in the training data (instance space). Instance space reduction methods use the same approach as being used in k-NN but work on a subset of training set examples (instance space). It is a data reduction framework in which the goal is to find the most important training set examples which could be used to classify any new observation [135]. This results in a significant reduction in computation time, as the number of comparisons are reduced because of fewer training set examples in the instance space. The instance space reduction method may result in a slight increase in phoneme frame error rate.

Instance space reduction techniques not only provide a reduction in computation time but they also provide a reduction in memory requirements, better generalization capability, and tolerance to noise. The seminal work was proposed by Hart and is known as condensed nearest neighbor [136]. Condensed nearest neighbor finds a subset S of the training set T in such a way that each instance in T is nearer to the instances of the same class in S than to the instances with a different class in S. This approach is computationally expensive and is very sensitive to noisy instances. These noisy instances cause obstruction in the instance space reduction size. The reduced nearest neighbor rule proposed by Gates uses decremental search approach [137]. It begins with $S = T$ and searches for instance to remove from S in a way such that the removal of such instance from S does not result in any misclassification with the instances present in T. The reduced nearest neighbor is computationally more expensive than condensed nearest neighbor but guarantees a subset which is smaller than that obtained with condensed nearest neighbor. It also removes noisy instances and internal points but retains border points. The edited nearest neighbor rule [138] also begins with $S = T$ and removes each instance of S if a majority of its nearest

neighbors does not match with it. The edited nearest neighbor rule also removes noisy instances as well as border point instances. However, it does not remove much of the internal points. As a result, this approach provides smooth decision boundaries but results in more memory requirements as compared with other instance space reduction techniques.

Instance space reduction techniques differ in terms of their approach to keep border points, internal points, noise instances, or some other set of points [139]. Boundary points play a significant role in defining decision boundaries as compared with internal points. Removing internal points from our instance space will have little impact on classification performance. Condensed nearest neighbor and reduced nearest neighbor are based on this intuition. Edited nearest neighbor removes border points or noisy instances which do not agree with their neighbors. This results in a smooth decision boundary. The instance space reduction techniques mentioned above do not work well with speech utterances, as speech signals intrinsically exhibit many variations.

Another method to reduce the computational time for k-NN search is KD-tree. In KD-tree [140, 141], the whole space is partitioned into k-dimensional space by making binary trees recursively. The search is based on nearest query region. The KD-tree is not efficient in high dimensional space. Local sensitivity hashing [142] works well in high dimensional space by reducing the dimensionality of data by mapping data points using hash functions. Hash functions are used for similarity search. Although these techniques reduce the computational time of k-NN search, they do not reduce memory requirements.

5.2 Dimensionality reduction using neighborhood component analysis

Neighborhood components analysis is a non-parametric approach to learn low-dimensional linear embeddings from the labeled data. Low-dimensional linear embeddings results in fast classification for k-NN. It does not assume any information regarding distribution of various classes present in the labeled data, nor the decision boundaries between these classes [143]. Let instance space (training data) consists of $\mathbf{N}$ labeled phoneme samples

given by Eqs. 5.1 and 5.2 in "M" dimensional space.

$$\text{Phoneme samples} = \mathbf{X} = \Big\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_N}\} \tag{5.1}$$

$$\text{Phoneme labels} = \mathbf{Y} = \Big\{\mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}, ..., \mathbf{y_N}\} \tag{5.2}$$

Neighborhood components analysis selects a single neighbor stochastically and looks at the expected votes for each class. For instance, each point "$\mathbf{i}$" selects other points "$\mathbf{j}$" as its neighbor with a probability $\mathbf{p_{ij}}$ by using softmax over the distance metric given by:

$$\mathbf{p_{ij}} = \frac{\mathbf{e^{-d_{ij}}}}{\sum_{\mathbf{k \neq i}} \mathbf{e^{-d_{ik}}}} \tag{5.3}$$

where $\mathbf{p_{ii}} = \mathbf{0}$. Probability that the point "$\mathbf{i}$" will be correctly classified is:

$$\mathbf{p_i^*} = \sum_{\mathbf{j \in C_i}} \mathbf{p_{ij}} \tag{5.4}$$

where $\mathbf{C_i}$ represents set of points in the same class.

The objective of the neighborhood components analysis is to maximize the expected number of points correctly classified which is given by expected leave-one-out classification performance. The objective function is given by Eq. 5.5:

$$\begin{aligned} \mathbf{f(A)} &= \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i}} \mathbf{p_i^*} = \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i}} \sum_{\mathbf{j \in C_i}} \mathbf{p_{ij}} \\ &= \frac{\mathbf{1}}{\mathbf{N}} \sum_{\mathbf{i}} \sum_{\mathbf{j \in C_i}} \frac{\mathbf{e^{-d_{ij}}}}{\sum_{\mathbf{k \neq i}} \mathbf{e^{-d_{ik}}}} \end{aligned} \tag{5.5}$$

For quadratic (Mahalanobis) distance metric we can write the expression as:

$$\mathbf{d_{ij}} = \mathbf{(x_i - x_j)^T Q (x_i - x_j)} \tag{5.6}$$

where $\mathbf{Q}$ is a positive semi-definite matrix which can be decomposed using Eigen decomposition $\mathbf{Q} = \mathbf{A^T A}$ as:

$$\begin{aligned} \mathbf{d_{ij}} &= (\mathbf{x_i} - \mathbf{x_j})^{\mathbf{T}} \mathbf{A^T A} (\mathbf{x_i} - \mathbf{x_j}) \\ &= (\mathbf{A x_i} - \mathbf{A x_j})^{\mathbf{T}} (\mathbf{A x_i} - \mathbf{A x_j}) \end{aligned} \tag{5.7}$$

Here $\mathbf{A}$ is the transformation matrix that we want to learn. We can restrict matrix $\mathbf{A}$ of size $(\mathbf{d} \times \mathbf{M})$ to be a rectangular matrix of low rank in our optimization procedure. Gradient with respect to the transformation matrix A is given by:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{A}} = -\mathbf{2A} \sum_{\mathbf{i}} \sum_{\mathbf{j} \in \mathbf{C_i}} \mathbf{p_{ij}} \left[\mathbf{x_{ij}} \mathbf{x_{ij}^T} - \sum_{\mathbf{k}} \mathbf{p_{ik}} \mathbf{x_{ik}} \mathbf{x_{ik}^T} \right] \tag{5.8}$$

Equivalent and more efficient computed expression is:

$$\frac{\partial \mathbf{f}}{\partial \mathbf{A}} = \mathbf{2A} \sum_{\mathbf{i}} \left[\mathbf{p_i^*} \sum_{\mathbf{k}} \mathbf{p_{ik}} \mathbf{x_{ik}} \mathbf{x_{ik}^T} - \sum_{\mathbf{j} \in \mathbf{C_i}} \mathbf{p_{ij}} \mathbf{x_{ij}} \mathbf{x_{ij}^T} \right] \tag{5.9}$$

By choosing $\mathbf{d} \ll \mathbf{M}$, transformation matrix $\mathbf{A}$ will map the training phoneme samples from $\mathbf{M}$ dimensional space to a low dimensional space $\mathbf{d}$.

$$\text{Low dimensional phoneme samples} = \mathbf{Z} = \mathbf{A} \cdot \mathbf{X} \tag{5.10}$$

Similarly for test point, we can find its projections in low dimensional "$\mathbf{d}$" space by using transformation matrix $\mathbf{A}$.

$$\text{Test phoneme sample} = \mathbf{z_{test}} = \mathbf{A} \cdot \mathbf{x_{test}} \tag{5.11}$$

Thus by learning optimal transformation matrix $\mathbf{A}$ we reduce the dimensions of our training and testing phoneme samples to low dimensional space "$\mathbf{d}$". Using k-NN classifier with Euclidean distance metric we can get significant reduction in memory and computation

cost [143].

5.3 Instance space reduction using adaptive data condensation

Adaptive data condensation method uses speaker characteristics to reduced the instance space. It learns the speaker similarity of the target speaker with speakers in our instance space. Based on the speaker similarity score, it sorts the speakers in the instance space. The speaker with the highest similarity score gets the rank 1, and so on. It picks the phoneme samples of the first "k" speakers in the instance space based on the speaker ranking. Some portion of the target speaker utterance data is used to learn a speaker similarity score with the speakers in the instance space. The instance space comprises phonemes from various sentences and speakers. For each phoneme instance, we have phoneme label and speaker label information available. For each correct match of the phoneme from the target speaker, we will find corresponding speakers for that correct match in our instance space. We then increment the score of that particular speaker by one. In this way we use all the available phoneme-labeled utterances from the target speaker to find the score of the speakers in our instance space. Once we learn the speakers' scores for speakers in our instance space, we then sort the scores in decreasing order of speaker similarity score. This gives us the ranking of the speakers in our instance space. A higher speaker similarity score will result in a lower rank. A lower rank means a speaker from the instance space is more similar to the target speaker. Let n be the number of speakers in our instance space and let k be the number of speakers we want our instance space to be reduced to. Based on the value of k we pick all the phoneme instances of first k speakers in our instance space and remove phoneme instances from all other speakers. Reducing our instance space result in a reduction of computational time as well as memory space for k-NN. This approach of instance space reduction is adaptive with the speaking characteristics of the target speaker. For phoneme classification, we use the reduced instance space based on the speaker ranking and weight our decision by speaker similarity score. Using phoneme acoustic frame from

the target speaker we found the k-nearest neighbors in the reduced instance space using the Euclidean distance metric. From these "k" nearest neighbor phonemes, we find the corresponding speakers. For each phoneme represented among the k-nearest neighbors, the corresponding speaker similarity scores are added. The classifier then assigns a label to the acoustic frame according to which represented phonemes has the highest score.

5.4 Results

5.4.1 Phoneme classification error with reduced feature dimension using neighborhood component analysis

Size of the transformation matrix is varied $\mathbf{A}$ $(\mathbf{d} \times \mathbf{M})$ to learn low dimensional embeddings of features learned from DNN. The dimension of features learned from DNN is 50. Size of dimension $\mathbf{d}$ is varied from 50 to 1 and compared it with phoneme frame error rate of speaker similarity score algorithm with dimension size of 50. As dimension "$\mathbf{d}$" is varied from 50 to 1, there is a very slight increase in phoneme frame error rate till $\mathbf{d}$=10. After that frame error rate increases at a much higher rate as shown in Fig. 5.1.

5.4.2 Computation time with reduced feature dimension using neighborhood component analysis

Computational time of the speaker similarity score algorithm using neighborhood components analysis is compared with the speaker similarity score agorithm with dimension $\mathbf{d}$=50. Fig. 5.2 shows the reduction in computational time with reduced dimensions obtained through neighborhood components analysis with DNN features. As dimensions of feature vectors are decreased, it results in reduced computational time for phoneme classification.

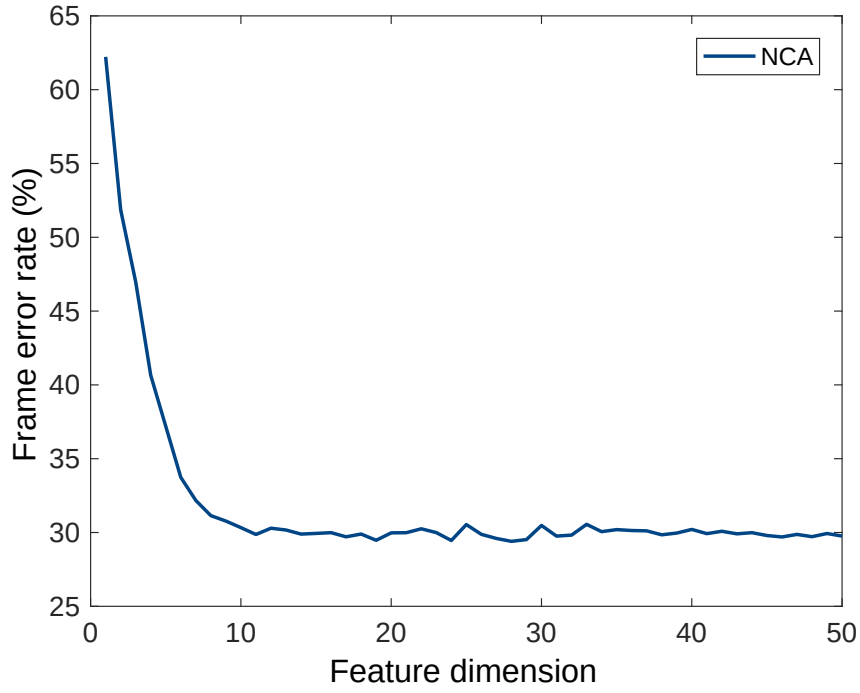

Figure 5.1: Phoneme frame error rate with reduced feature dimension using neighborhood component analysis (NCA)

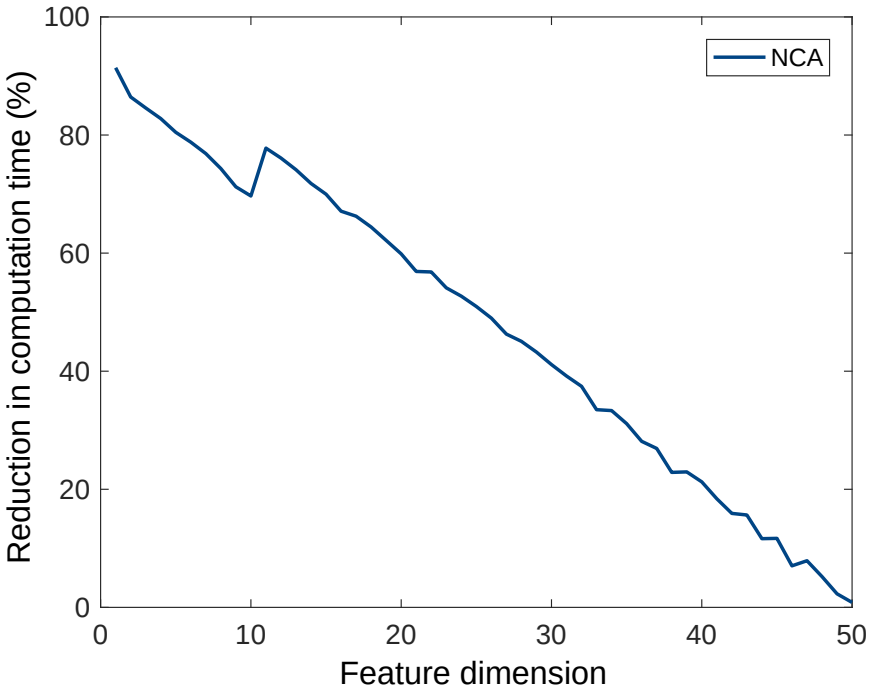

Figure 5.2: Reduction in computation time with reduced feature dimension using neighborhood component analysis

5.4.3 Speaker-wise comparison with reduced feature dimension using neighborhood component analysis with baseline

Figs. 5.3 shows the comparison of speaker similarity score (SSS) algorithm on DNN features with low dimensional features (**d**=10) obtained using neighborhood components analysis (NCA). Phoneme frame error rate of 12 speakers using reduced dimensions are compared with baseline speaker similarity score algorithm. The value of **d** (number of dimensions) used for learning transformation matrix in neighborhood component analysis is 10. The average reduction in computational time obtained using neighborhood components analysis on DNN features is 69.7%. Using dimensionality reduction resulted only in 1.55% increase in phoneme frame error rate on an average.

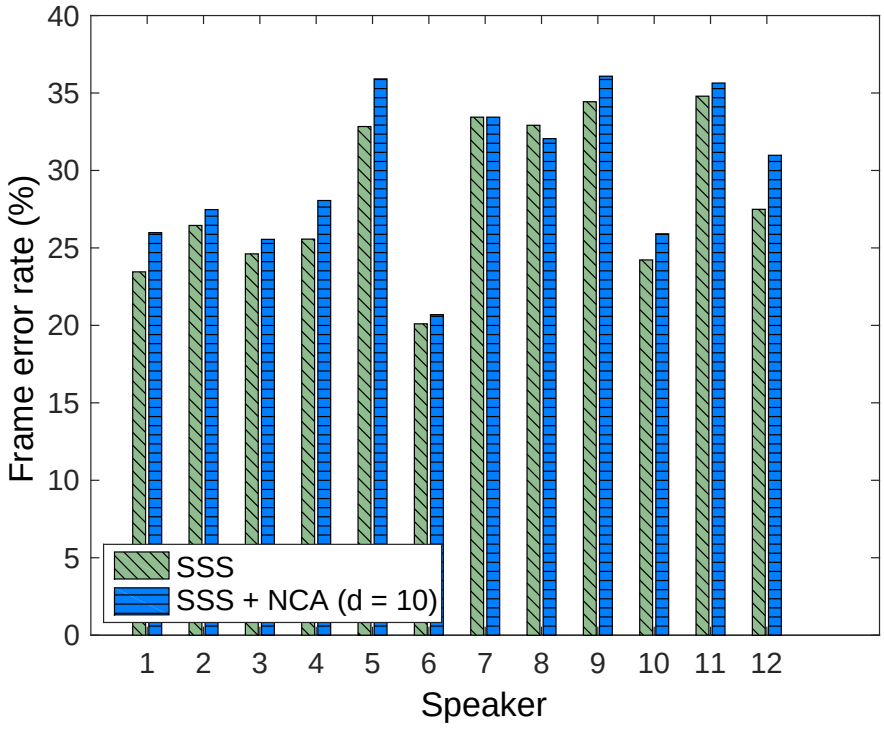

Figure 5.3: Phoneme frame error rate per speaker

5.4.4 Comparison of neighborhood component analysis with PCA and LDA

Phoneme frame error rate with reduced dimension using neighborhood component analysis is compared with the principal component analysis (PCA) and linear discriminant analysis (LDA) as shown in Fig. 5.4 below. NCA gives the best performance for a feature dimension size of 10. At d=10 NCA, PCA, and LDA has a phoneme frame error rate of 30.33%, 31.32%, and 31.83% respectively.

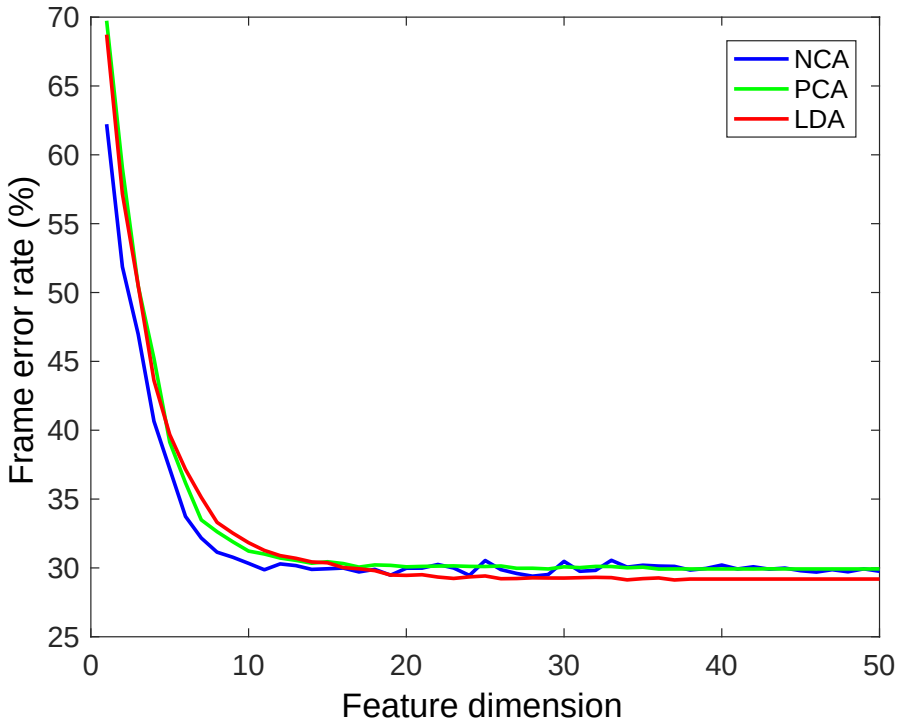

Figure 5.4: Comparison of neighborhood component analysis with principal component analysis and linear discriminant analysis

5.4.5 Copmarison of phoneme classification error with reduced instance space

After learning speaker similarity score for a target speaker (Section 4.3.1), the speakers present in the instance space are ranked in decreasing order. For example, the speaker with the highest score is assigned rank 1, which means that the target speaker more closely matches this speaker in terms of speaking style, accent, etc. The number of phoneme sam-

ples in the instance space is reduced by varying the number of speakers in the instance space. Fig. 5.5 shows the variation of phoneme frame error rate of the target speaker with the number of speakers in the instance space. The number of speakers in the instance space are reduced from 496 to 10. The reduced instance space results in a low memory requirements at the cost of small increase in phoneme frame error rate. Significant reduction in the size of instance space is achieved using 100 speakers in the instance space with an absolute increase in phoneme frame error rate by 1.15%. In general, more the phoneme samples we have from different speakers in the instance space, the better the performance we get (at the cost of large memory requirements and computational cost).

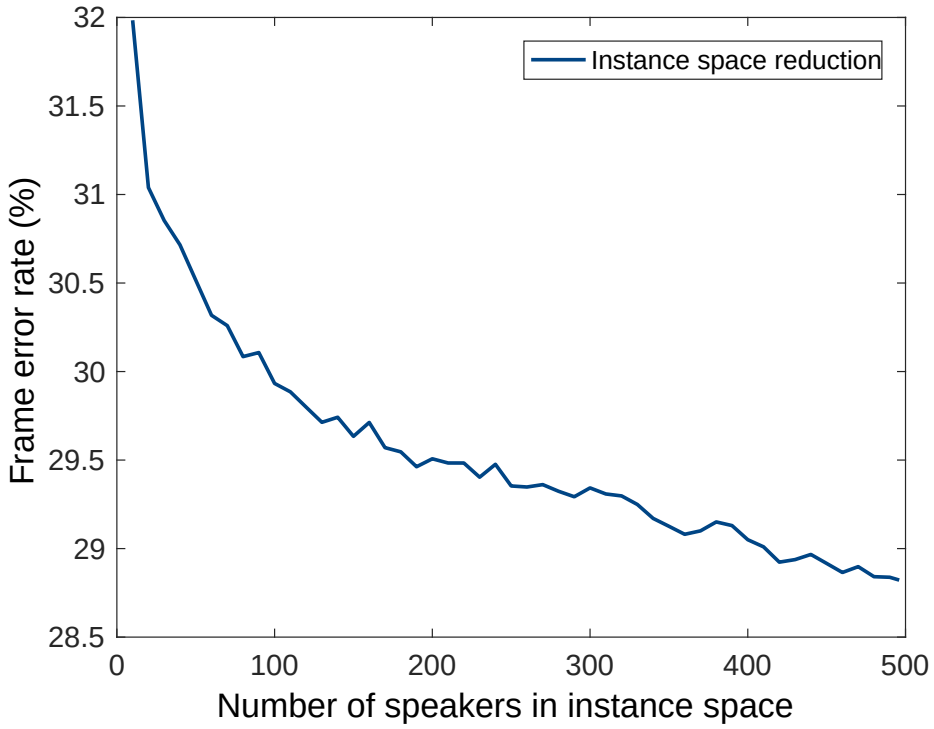

Figure 5.5: Instance space size vs. phoneme frame error rate

5.4.6 Improvement in computation time with reduced instance space

Fig. 5.6 shows the impact of reduced instance space on improvement in computation cost. Using phoneme samples from fewer speakers not only results in low memory requirements

but also helps to speed up the search for nearest neighbors. As we decrease phoneme frame samples from 450 speakers to 10 speakers in the instance space we get a significant improvement in the computational time. For 100 speakers in the instance space we get an improvement in computational time by 77.05% at the cost of 1.15% decrease in the absolute phoneme frame error rate.

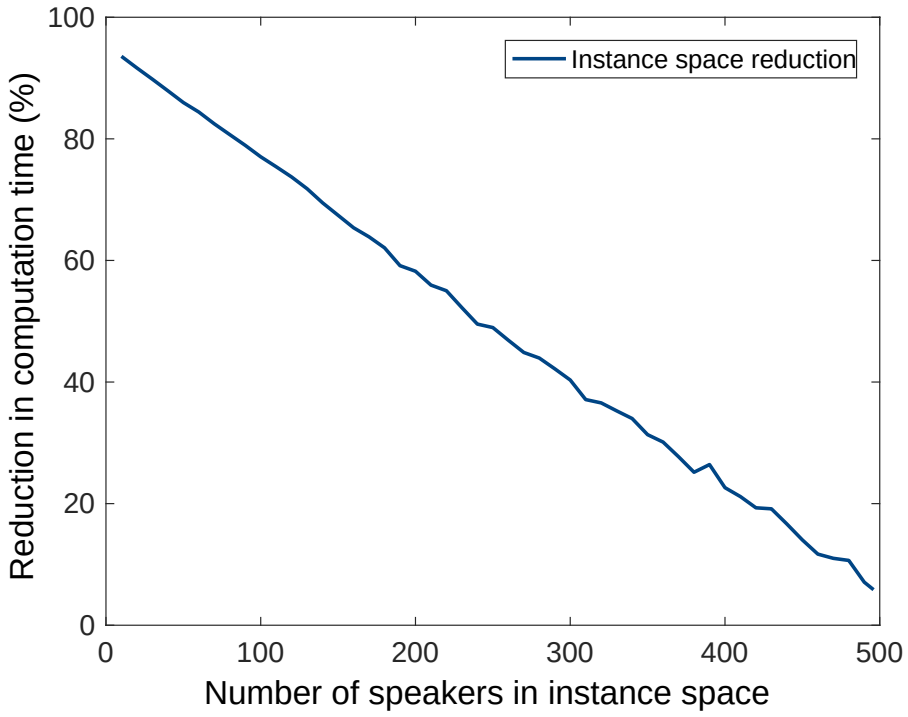

Figure 5.6: Instance space size vs. improvement in computation time

5.4.7 Speaker-wise comparison of reduced instance space with baseline

Fig. 5.7 shows the comparison of phoneme frame error rate of using complete instance space that comprises phoneme frame samples from 496 speakers (complete instance space) with 100 speakers (reduced instance space) for 12 different speakers. The maximum absolute increase in phoneme frame error rate across different speaker was 2.2%.

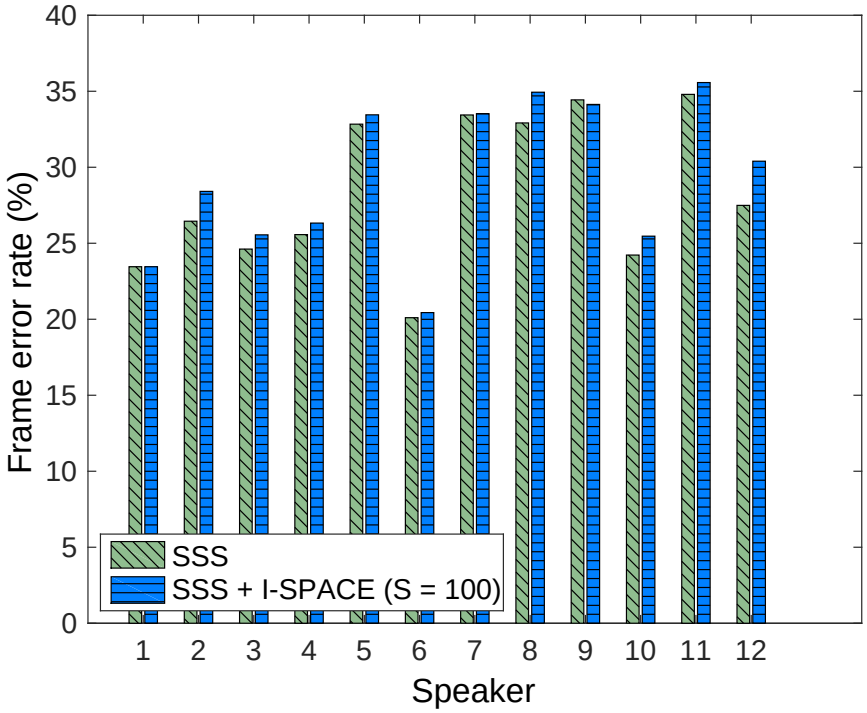

Figure 5.7: Complete vs. reduced instance space-Speakerwise comparison

5.4.8 Approximate nearest neighbor based search

Approximate nearest neighbor methods provide another approach to reduce the computational time of nearest neighbor search [144]. The key idea of approximate nearest neighbor is to pre-process the training data in such a way that when test instance comes, it can quickly search and find the nearest neighbor from the training data for the given test instance. The training data is preprocessed using a data structure for an efficient and quick search for a given test point. Approximate nearest neighbor wrapper for Matlab was used, but it resulted in an increase in computational time with almost the same phoneme frame error rate.

CHAPTER 6

SPEAKER ADAPTATION OF SPEECH RECOGNITION SYTEM

Speech recognition systems use acoustic models, language models, and lexicon to decode words for a given speech signal from a test speaker. There is always a mismatch between trained acoustic model and test speaker. Speaker adaptation techniques can minimize the difference between acoustic model and test speaker. Speaker adaptation techniques require adaptation data from the test speaker to optimize system performance. In most cases only a limited amount of adaptation data from the test speaker is available. This chapter discussed two methods for speaker adaptation. The first method is based on the speaker similarity score algorithm. The second method extracts speaker features and appends these with speech features for speaker adaptation of speech recognition system.

6.1 Speaker similarity based speaker adaptation

The adaptive phoneme classification method discussed in Chapter 4 is extended for the speaker adaptation of speech recognition systems. The main idea is that ASR system can be adapted by using speaker similarity score information and finding a similar frame from the training data using k-NN. The intuition is that since DNN has already seen the similar frame sample, it will be able to give a better estimate of the probability across phoneme classes. Replacement with the similar frame can reduce the mismatch between the training and testing data. Figs. 6.1(a) and 6.1(b) show the block diagram of the conventional hybrid DNN-HMM system and speaker similarity score based speaker adaptation of the hybrid DNN-HMM system. The speaker similarity score based speaker adaptation can be robust as it does not require modification and retraining of the DNN for speaker adaptation.

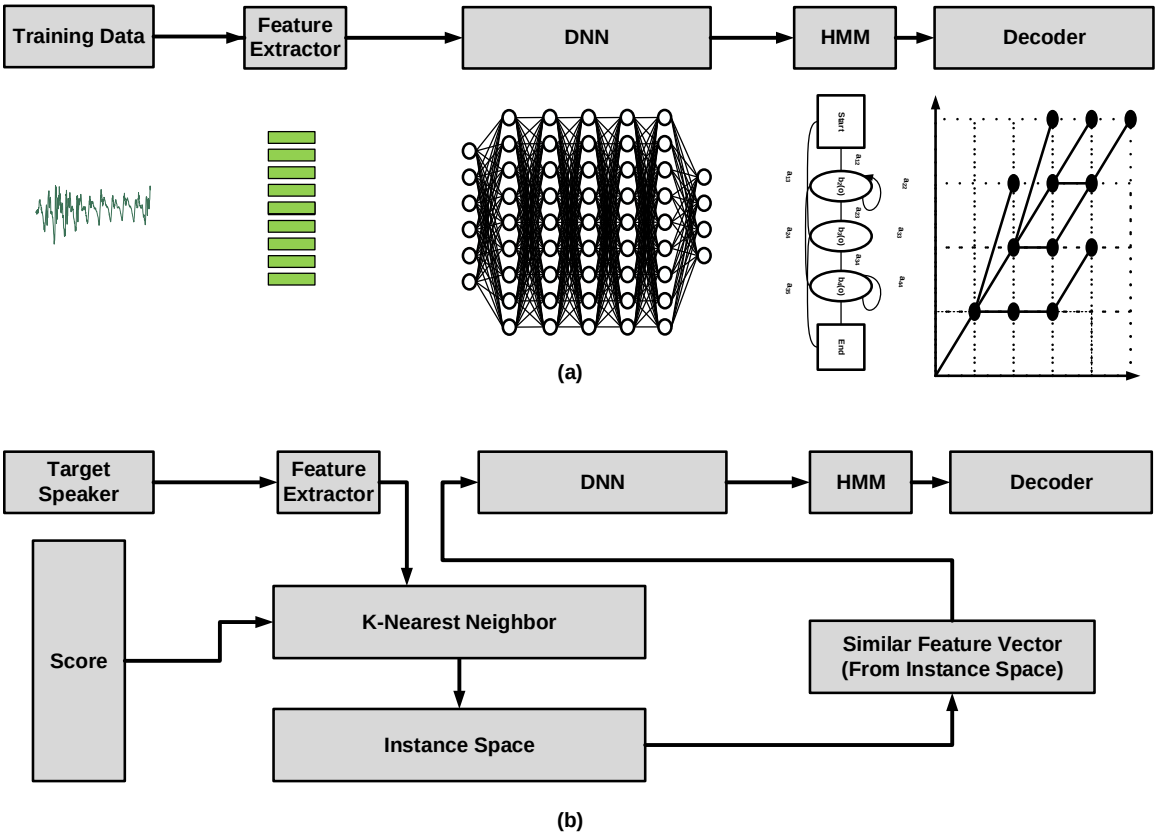

Figure 6.1: Speaker adaptation using speaker similarity score

The speaker similarity score based speaker adaptation method consists of the following three steps.

Step 1

Th speaker similarity score of the target speaker is learned using the k-NN. The instance space for the k-NN comprises speech frames from the training data with their corresponding phoneme labels and speaker information. K-NN classifier is used to learn the speaker similarity score for a given target speaker. For details regarding speaker similarity score calculation, readers are referred to Section 4.3.1 of this thesis. All the available adaptation data from the target speaker is used to learn the speaker similarity score information. Greater the score of a particular speaker in the training data, the more similar it is to the

given target speaker.

Step 2

In the second step, feature vectors of the test speaker are replaced with the similar feature vectors from the instance space. K-nearest neighbor is used for searching phoneme frames in the instance space using the Euclidean distance metric. The k-nearest phoneme frames are weighted by their corresponding speaker score information to find the most similar feature vector frame in the instance space. The similar feature vectors from the instance space corresponding to the test speaker utterances are given as an input to the DNN. The DNN provides posteriori probabilities for all states in the HMM. These posteriori probabilities are converted to scaled likelihoods by using the state prior probability information. The likelihoods are given as an input to the decoder for recognition.

We used the same experimental setup discussed in Section 4.4. The baseline hybrid DNN-HMM system consists of five hidden layers with monophone HMMs. The hidden layer comprises 1000 neurons with sigmoid as an activation function. Two sets of experiments were conducted using this method. In the first approach, all the frames from the target speaker testing data are replaced with the frames from the instance space. This approach resulted in an increase of the phoneme error rate as compared with the baseline system. In the second approach, only selected frames from the testing data of the target speaker are replaced with the similar frames from the instance space. The selection of the frames from the target speaker testing data was based on probability values obtained at the softmax layer. This approach does not provide any significant reduction in the phoneme error rate as compared with the baseline system.

6.2 Sparse coding based speaker adaptation

Sparse coding has attracted a lot of attention in many speech processing applications. Sparse coding maps original features to sparse representations. Recently, universal background sparse coding is proposed for the speaker verification task [145, 146]. The univer-

sal background sparse coding uses multilayer bootstrap network in a supervised manner to learn high dimensional sparse features for each speaker termed as a super vector. The super vector dimensionality is reduced by the multilayer bootstrap network. The multilayer bootstrap network trains an ensemble of clusters and applies one-nearest-neighbor optimization and binarization to produce sparse codes. Fig. 6.2 shows the overall block diagram for learning speaker features. The speaker features learned are augmented with the speech features for the speaker adaptation of the ASR system.

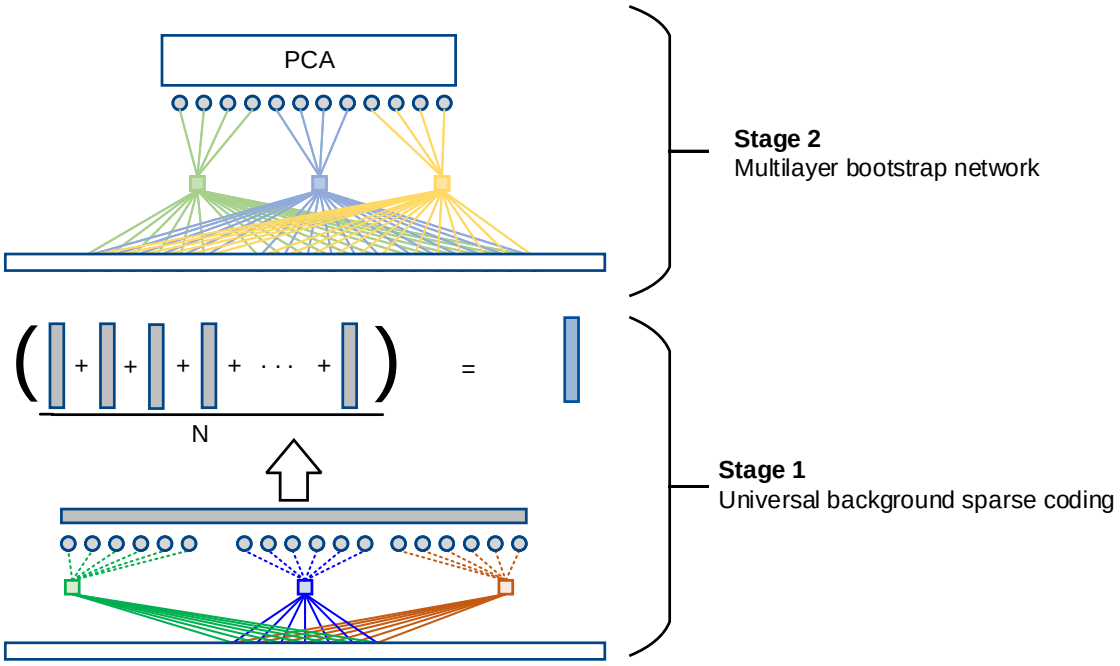

Figure 6.2: Sparse coding based speaker features

Learning speaker features involve two stages. In the first stage, the universal background sparse coding is used to generate high-dimensional sparse features for each speaker termed as a super vector. In the second stage, the dimensionality of the speaker super vector is reduced by multilayer bootstrap network.

Stage 1: Universal background sparse coding-based speaker super vector

The universal background sparse coding learns a data distribution in a discrete space. The

first step comprises MFCC feature extraction at the frame level from the training data. The second step randomly selects k frame samples from the whole training data as centers for the cluster to build random models. Each layer comprises M random models with k frames randomly selected without replacement from the training data as centers. After this step, frame samples from the given data are assigned to one of the k clusters based on the minimum squared Euclidean distance. This step is repeated for all M models. Based on the cluster assignments, it gives an output indicator vector for each model $\mathbf{i} = [i_1, i_2, ..., i_k]$. For example, if the frame sample is assigned to cluster 1 then the indicator vector is given by $\mathbf{i} = [1, 0, ..., 0]$. The indicator vectors for all the models are combined to provide frame level binary sparse features. Finally, the frame level binary sparse features are combined and normalized for all the frame samples from each speaker to give a d dimensional speaker super vector.

Stage 2: Multilayer bootstrap network based dimensionality reduction

Multilayer bootstrap network is applied to reduce the dimensions of the speaker super vector from d to d^*. The multilayer bootstrap network is trained layer-by-layer. Each layer of the multilayer bootstrap network consists of mutually independent clusters that are randomly sampled from the training data after random feature selection. Each cluster consists of k output units. The output units of all clusters are concatenated as the input to the next layer. The last layer of multilayer bootstrap network applies principal component analysis (PCA).

The same experiment setup is used as mentioned in Section 4.4. The baseline hybrid DNN-HMM system consists of five hidden layers with triphone HMMs. Each hidden layer contained 1000 neurons and used sigmoid as an activation function. We used five words from each speaker to learn speaker features using the universal background sparse coding. A number of experiments were conducted to determine the dimension of the speaker features. Using speaker features with small dimension did not have any impact on the phoneme error rate. And, using speaker features with large dimension distorted the speech

features that contain phoneme information. Fig. 6.3 shows the comparison of the baseline system with the sparse coding based speaker adaptation for 24 speakers. An absolute reduction of 0.8% in phoneme error rate is achieved using 39-dimensional MFCCs as speech features and 11-dimensional speaker features learned with the universal background sparse coding.

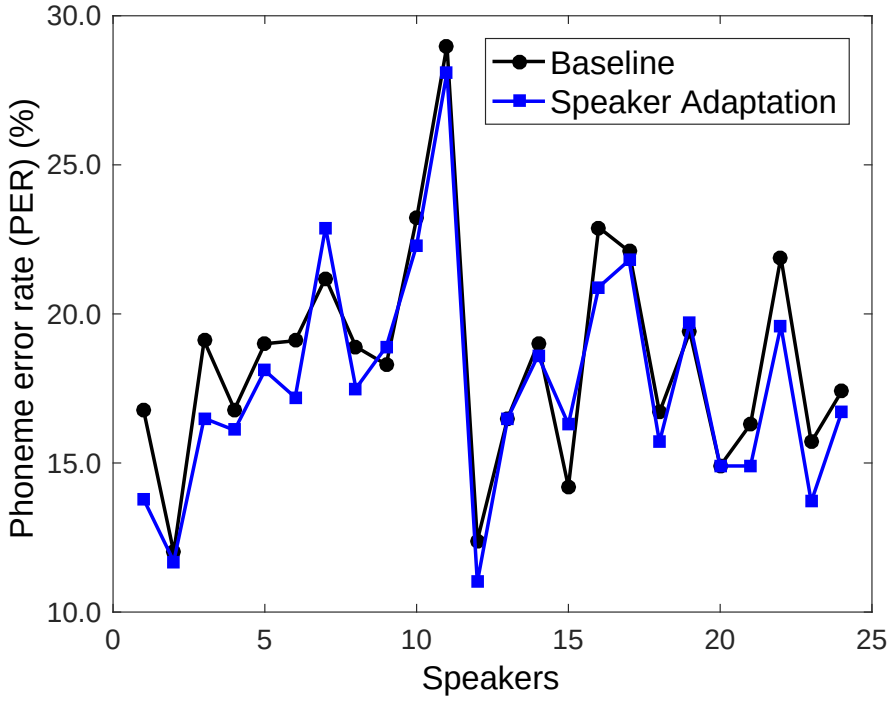

Figure 6.3: Speaker adaptation using speaker features

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion

This dissertation presented multiple methods for the adaptation of speech recognition systems.

Chapter 3 presented a novel architecture for accent classification based on extreme learning machines. Using a single word from a speaker is effective for the accent classification and significant improvement in accent classification accuracy can be obtained by incorporating multiple words from a speaker. Different words were analyzed for accent classification. Also, extreme learning machines are effective for accent classification tasks as ELMs do not require a significant amount of training data for learning neural network weights. However, the proposed method requires specific words from a test speaker for accent classification.

Chapter 4 presented a novel speaker similarity score algorithm for adaptive phoneme classification. The activations of the last hidden layer were used as features for the speaker similarity score algorithm. We found that DNNs last hidden layer activations are more effective in identifying phoneme classes of frames as compared with traditional raw MFCC features. The number of neurons in the last hidden layer and number of hidden layers were varied to investigate the impact of these on phoneme frame error rate performance. Based on our experiments 50 neurons in the last hidden layer and five layers is a nice compromise between frame error rate and computational cost. The proposed speaker similarity score algorithm based on DNN features outperformed both the k-NN based method on raw MFCC features and neural networks using soft-max for phoneme classification.

Chapter 5 presented two methods for reducing the computational time of the speaker

similarity score algorithm. The first method applies neighborhood component analysis for dimensionality reduction. The dimensions of features were reduced to 10. The second method discusses adaptive data condensation that uses speaker similarity score information to reduce the number of phoneme frame samples in the instance space. In our experiments, the number of speakers in the instance space were reduced to 100. We discovered based on our experiments that both dimensionality reduction using neighborhood component analysis and adaptive data condensation provide a significant decrease in computation time for speaker similarity score algorithm with a slight increase in phoneme frame error rate.

Chapter 6 investigates methods for speaker adaptation. In our first method based on speaker similarity score, each frame of the target speaker was replaced with most similar frame from the instance space. This resulted in an increase in phoneme error rate as this replacement lost the contextual information. In the second method, speech features are augmented with the speaker features that contain speaker information. Speaker features learned using the universal background sparse coding to be useful for speaker adaptation. We discovered that the dimension of speaker features augmented with speech feature is critical. Using few speaker features does not have any impact and using too many speaker features will distort the speech features that contain phoneme information. Based on our experiments, we found 39-dimensional MFCCs and 11-dimensional sparse features learned with the universal background sparse coding gives an absolute reduction in phoneme error rate by 0.8%.

7.2 Future Work

The work presented in this dissertation may lead to some new opportunities for research. The following list includes a number of promising ideas that need further investigation.

Accent classification using multiple words

- Identification of distinguishing words that can help in accent classification

- Extension of the proposed method to non-native speakers

Adaptive phoneme classification

- Combining hidden layer activations from multiple layers for feature extraction
- Identification of hidden layer activations that contain speaker specific information using information visualization techniques

Robust phoneme classification

- Using the adaptive data condensation method on other speech tasks such as speaker identification and language identification

Speaker adaptation of ASR

- Extension of the work to a large vocabulary continuous speech recognition dataset
- Extracting speaker specific features from the activations of DNNs

REFERENCES

[1] M. Gales and S. Young, "The application of hidden Markov models in speech recognition," *Foundations and Trends in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2008.

[2] D. Yu and L. Deng, *Automatic speech recognition: A deep learning approach*. Springer, 2014.

[3] O. Abdel-Hamid, "Automatic speech recognition using deep neural networks: New possibilities," PhD thesis, York University, 2014.

[4] T. N. Sainath, "Applications of broad class knowledge for noise robust speech recognition," PhD thesis, Massachusetts Institute of Technology, 2009.

[5] Y. Miao, "Incorporating context information into deep neural network acoustic models," PhD thesis, Carnegie Mellon University, 2016.

[6] M. Rizwan, B. O. Odelowo, and D. V. Anderson, "Word based dialect classification using extreme learning machines," in *IEEE International Joint Conference on Neural Networks*, 2016, pp. 2625–2629.

[7] M. Rizwan and D. V. Anderson, "Using k-nearest neighbor and speaker ranking for phoneme prediction," in *IEEE International Conference on Machine Learning and Applications*, 2014, pp. 383–387.

[8] ——, "Speaker adaptation using speaker similarity score on DNN features," in *IEEE International Conference on Machine Learning and Applications*, 2015, pp. 877–882.

[9] ——, "Comparison of distance metrics for phoneme classification based on deep neural network features and weighted k-NN classifier," in *Workshop on Machine Learning in Speech and Language Processing*, 2016.

[10] ——, "Speaker similarity score based fast phoneme classification using neighborhood components analysis," in *IEEE Global Conference on Signal and Information Processing*, 2016, pp. 50–54.

[11] X. Huang, A. Acero, and H. Hon, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice Hall, 2001.

[12] J. R. Deller, J. H. L. Hansen, and J. G. Proakis, *Discrete time processing of speech signals*. Prentice Hall, 1993.

[13] A. R. Mohamed, G. Dahl, and G. E. Hinton, "Deep belief networks for phone recognition," in *NIPS workshop on deep learning for speech recognition and related applications*, vol. 1, 2009, p. 39.

[14] X. Huang and K. F. Lee, "On speaker-independent, speaker-dependent, and speaker-adaptive speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 2, pp. 150–157, 1993.

[15] L. Rabiner and B. Juang, *Fundamentals of speech recognition*. Prentice Hall, 1993.

[16] S. Furui, "Unsupervised speaker adaptation based on hierarchical spectral clustering," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 1923–1930, 1989.

[17] P. C. Woodland, "Speaker adaptation for continuous density HMMs: A review," in *ISCA Tutorial and Research Workshop on Adaptation Methods for Speech Recognition*, 2001.

[18] C. H. Lee and Q. Huo, "On adaptive decision rules and decision parameter adaptation for automatic speech recognition," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1241–1269, 2000.

[19] M. J. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Elsevier Computer Speech & Language*, vol. 12, no. 2, pp. 75–98, 1998.

[20] S. Furui, "Generalization problem in ASR acoustic model training and adaptation," in *IEEE Workshop on Automatic Speech Recognition & Understanding*, 2009, pp. 1–10.

[21] J. L. Gauvain and C. H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994.

[22] S. Ahadi and P. C. Woodland, "Rapid speaker adaptation using model prediction," in *International Conference on Acoustics, Speech, and Signal Processing*, 1995, pp. 684–687.

[23] K. Shinoda and C. H. Lee, "A structural Bayes approach to speaker adaptation," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 3, pp. 276–287, 2001.

[24] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Elsevier Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995.

[25] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, "A compact model for speaker-adaptive training," in *IEEE International Conference on Spoken Language*, vol. 2, 1996, pp. 1137–1140.

[26] M. Gales, "Cluster adaptive training of hidden Markov models," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 4, pp. 417–428, 2000.

[27] R. Kuhn, J. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 6, pp. 695–707, 2000.

[28] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, Luis, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," in *Eurospeech*, 1995.

[29] V. Abrash, H. Franco, A. Sankar, and M. Cohen, "Connectionist speaker normalization and adaptation," in *Eurospeech*, 1995.

[30] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011, pp. 24–29.

[31] R. Gemello, F. Mana, S. Scanzio, P. Laface, and R. D. Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Elsevier Speech Communication*, vol. 49, no. 10, pp. 827–835, 2007.

[32] K. Yao, D. Yu, F. Seide, H. Su, L. Deng, and Y. Gong, "Adaptation of context-dependent deep neural networks for automatic speech recognition," in *IEEE Spoken Language Technology Workshop*, 2012, pp. 366–369.

[33] S. M. Siniscalchi, T. Svendsen, F. Sorbello, and C. H. Lee, "Experimental studies on continuous speech recognition using neural architectures with adaptive hidden activation functions," in *IEEE International Conference on Acoustics Speech and Signal Processing*, 2010, pp. 4882–4885.

[34] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 7893–7897.

[35] S. Xue, H. Jiang, L. Dai, and Q. Liu, "Speaker adaptation of hybrid NN/HMM model for speech recognition based on singular value decomposition," *Springer Journal of Signal Processing Systems*, vol. 82, no. 2, pp. 175–185, 2016.

[36] Y. Zhao, J. Li, J. Xue, and Y. Gong, "Investigating online low-footprint speaker adaptation using generalized linear regression and click-through data," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4310–4314.

[37] R. Price, K. Iso, and K. Shinoda, "Speaker adaptation of deep neural networks using a hierarchy of output layers," in *IEEE Spoken Language Technology Workshop*, 2014, pp. 153–158.

[38] Z. Huang, J. Li, S. M. Siniscalchi, I. F. Chen, J. Wu, and C. H. Lee, "Rapid adaptation for deep neural networks through multi-task learning," in *Interspeech*, 2015, pp. 3625–3629.

[39] Z. Huang, S. M. Siniscalchi, I. F. Chen, J. Wu, and C. H. Lee, "Maximum a posteriori adaptation of network parameters in deep models," *ArXiv preprint arXiv:1503.02108*, 2015.

[40] C. Zhang and P. C. Woodland, "DNN speaker adaptation using parameterised sigmoid and ReLU hidden activation functions," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 5300–5304.

[41] Y. Miao, H. Zhang, and F. Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 23, no. 11, pp. 1938–1949, 2015.

[42] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 7942–7946.

[43] ——, "Rapid and effective speaker adaptation of convolutional neural network based models for speech recognition," in *Interspeech*, 2013, pp. 1248–1252.

[44] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1713–1725, 2014.

[45] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Elsevier Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[46] G. B. Huang, E. Cambria, E. K. A. Toh, B. Widrow, and Z. Xu, "New trends of learning in computational intelligence [guest editorial]," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 16–17, 2015.

[47] L. M. Arslan and J. H. L. Hansen, "Language accent classification in American English," *Elsevier Speech Communication*, vol. 18, no. 4, pp. 353–367, 1996.

[48] J. J. Humphries, "Accent modelling and adaptation in automatic speech recognition," PhD thesis, University of Cambridge, 1998.

[49] R. Huang, J. H. L. Hansen, and P. Angkititrakul, "Dialect/accent classification using unrestricted audio," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 2, pp. 453–464, 2007.

[50] S. Goronzy, *Robust adaptation to non-native accents in automatic speech recognition*. Springer Science & Business Media, 2002.

[51] G. Choueiter, G. Zweig, and P. Nguyen, "An empirical study of automatic accent classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008, pp. 4265–4268.

[52] P. Angkititrakul and J. H. L. Hansen, "Advances in phone-based modeling for automatic accent classification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 2, pp. 634–646, 2006.

[53] J. Macías-Guarasa, "Acoustic adaptation and accent identification in the ICSI MR and FAE corpora," in *ICSI Meeting slides*, 2003.

[54] C. G. Clopper, D. B. Pisoni, and K. D. Jong, "Acoustic characteristics of the vowel systems of six regional varieties of American English," *The Journal of the Acoustical Society of America*, vol. 118, no. 3, pp. 1661–1676, 2005.

[55] J. H. L. Hansen, U. H. Yapanel, R. Huang, and A. Ikeno, "Dialect analysis and modeling for automatic classification," in *Interspeech*, 2004.

[56] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *IEEE International Joint Conference on Neural Networks*, vol. 2, 2004, pp. 985–990.

[57] G. Huang, S. Song, J. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2405–2417, 2014.

[58] G. B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Springer International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 107–122, 2011.

[59] E. Cambria, N. Howard, Y. Xia, and T. S. Chua, "Computational intelligence for big social data analysis [guest editorial]," *IEEE Computational Intelligence Magazine*, vol. 11, no. 3, pp. 8–9, 2016.

[60] G. B. Huang, "An insight into extreme learning machines: Random neurons, random features and kernels," *Springer Cognitive Computation*, vol. 6, no. 3, pp. 376–390, 2014.

[61] ——, "What are extreme learning machines? Filling the gap between Frank Rosenblatt's dream and John von Neumann's puzzle," *Springer Cognitive Computation*, vol. 7, no. 3, pp. 263–278, 2015.

[62] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–821, 2016.

[63] G. B. Huang, L. Chen, and C. K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[64] Y. Lan, Y. C. Soh, and G. B. Huang, "Ensemble of online sequential extreme learning machine," *Elsevier Neurocomputing*, vol. 72, no. 13, pp. 3391–3395, 2009.

[65] G. B. Huang, Z. Bai, K. L. Chamara, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 18–29, 2015.

[66] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[67] G. B. Huang and L. Chen, "Convex incremental extreme learning machine," *Elsevier Neurocomputing*, vol. 70, no. 16, pp. 3056–3062, 2007.

[68] W. Schmidt, M. Kraaijveld, and R. Duin, "Feedforward neural networks with random weights," in *IEEE International Conference on Pattern Recognition Methodology and Systems*, 1992, pp. 1–4.

[69] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 525–536, 1998.

[70] P. Lancaster and M. Tismenetsky, *The theory of matrices: With applications*. Elsevier, 1985.

[71] N. Draper, H. Smith, and E. Pownell, *Applied regression analysis*. Wiley New York, 1966.

[72] C. Cortes and V. Vapnik, "Support-vector networks," *Springer Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[73] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems*, 1997, pp. 281–287.

[74] S. Schölkopf, V. Vapnik, and A. Smola, "Improving the accuracy and speed of support vector machines," in *Advances in Neural Information Processing Systems*, 1997, pp. 375–381.

[75] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Springer Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[76] A. Aizerman, M. E. Braverman, and L. Rozoner, "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.

[77] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[78] B. Frénay and M. Verleysen, "Using SVMs with randomised feature spaces: An extreme learning approach," in *ESANN*, 2010.

[79] L. Zhang, D. Zhang, and F. Tian, "SVM and ELM: Who wins? Object recognition with deep convolutional features from ImageNet," in *Proceedings of Extreme Learning Machines*, Springer, 2016, pp. 249–263.

[80] O. Chapelle, V. Vapnik, Vladimir, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines," *Springer Machine Learning*, vol. 46, no. 1-3, pp. 131–159, 2002.

[81] J. Chorowski, J. Wang, and J. M. Zurada, "Review and performance comparison of SVM-and ELM-based classifiers," *Elsevier Neurocomputing*, vol. 128, pp. 507–516, 2014.

[82] X. Liu, C. Gao, and P. Li, "A comparative analysis of support vector machines and extreme learning machines," *Elsevier Neural Networks*, vol. 33, pp. 58–66, 2012.

[83] B. Scholkopf and A. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[84] G. B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Elsevier Neurocomputing*, vol. 74, no. 1, pp. 155–163, 2010.

[85] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[86] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *ACM Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[87] C. W. Hsu and C. J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[88] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," *NASA Technical Report*, vol. 93, p. 27 403, 1993.

[89] V. Zue, S. Seneff, and J. Glass, "Speech database development at MIT: TIMIT and beyond," *Elsevier Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.

[90] L. Rabiner and R. Schafer, *Digital processing of speech signals*. Prentice Hall, 1978.

[91] M. Slaney, "Auditory toolbox: A Matlab toolbox for auditory modeling," *Interval Research Corporation Work Technical Report*, pp. 29–32, 1998.

[92] V. Tiwari, "MFCC and its applications in speaker recognition," *International Journal on Emerging Technologies*, vol. 1, no. 1, pp. 19–22, 2010.

[93] C. W. Hsu, C. C. Chang, and C. J. Lin, *A practical guide to support vector classification*, 2010.

[94] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, p. 27, 2011.

[95] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 10, pp. 1042–1052, 1993.

[96] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

[97] M. Ankerst, G. Kastenmüller, H. Kriegel, and T. Seidl, "Nearest neighbor classification in 3D protein databases," in *International Conference on Intelligent Systems for Molecular Biology*, vol. 99, 1999, pp. 34–43.

[98] Y. Lee, T. Hara, H. Fujita, S. Itoh, and T. Ishigaki, "Automated detection of pulmonary nodules in helical CT images based on an improved template-matching technique," *IEEE Transactions on Medical Imaging*, vol. 20, no. 7, pp. 595–604, 2001.

[99] E. Wold, T. Blum, D. Keislar, and J. Wheaten, "Content-based classification, search, and retrieval of audio," *IEEE MultiMedia*, vol. 3, no. 3, pp. 27–36, 1996.

[100] S. Z. Li, "Content-based audio classification and retrieval using the nearest feature line method," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 5, pp. 619–625, 2000.

[101] L. Lu, H. Jiang, and H. J. Zhang, "A robust audio classification and segmentation method," in *ACM International Conference on Multimedia*, 2001, pp. 203–211.

[102] J. C. Wang, J. F. Wang, K. W. He, and C. S. Hsu, "Environmental sound classification using hybrid SVM/K-NN classifier and MPEG-7 audio low-level descriptor," in *IEEE International Joint Conference on Neural Networks*, 2006, pp. 1731–1735.

[103] K. Lee and D. P. Ellis, "Audio-based semantic concept classification for consumer video," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1406–1416, 2010.

[104] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification," in *International Society for Music Information Retrieval Conference*, vol. 11, 2011.

[105] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[106] Y. Yaslan and Z. Cataltepe, "Audio music genre classification using different classifiers and feature selection methods," in *IEEE International Conference on Pattern Recognition*, vol. 2, 2006, pp. 573–576.

[107] Y. Panagakis, C. Kotropoulos, and G. R. Arce, "Music genre classification via sparse representations of auditory temporal modulations," in *IEEE European Signal Processing Conference*, 2009, pp. 1–5.

[108] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, 2010.

[109] Y. Li, A. Cichocki, and S. Amari, "Analysis of sparse representation and blind source separation," *Neural Computation*, vol. 16, no. 6, pp. 1193–1234, 2004.

[110] J. Shah, B. Smolenski, R. Yantorno, and A. Iyer, "Sequential k-nearest neighbor pattern recognition for usable speech classification," in *IEEE European Signal Processing Conference*, 2004, pp. 741–744.

[111] I. Naseem, R. Togneri, and M. Bennamoun, "Sparse representation for speaker identification," in *IEEE International Conference on Pattern Recognition*, 2010, pp. 4460–4463.

[112] T. Deselaers, G. Heigold, and H. Ney, "Speech recognition with state-based nearest neighbour classifiers," in *Interspeech*, 2007, pp. 2093–2096.

[113] L. Golipour and D. O. Shaughnessy, "Context-independent phoneme recognition using a k-nearest neighbour classification approach," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1341–1344.

[114] ——, "Phoneme classification and lattice rescoring based on a k-NN approach," in *Interspeech*, 2010, pp. 1954–1957.

[115] J. Labiak and K. Livescu, "Nearest neighbors with learned distances for phonetic frame classification," in *Interspeech*, 2011, pp. 2337–2340.

[116] J. Gemmeke, L. Bosch, L. Boves, and B. Cranen, "Using sparse representations for exemplar based continuous digit recognition," in *IEEE European Signal Processing Conference*, 2009, pp. 1755–1759.

[117] T. N. Sainath, B. Ramabhadran, D. Nahamoo, D. Kanevsky, Dimitri, and A. Sethy, "Sparse representation features for speech recognition," in *Interspeech*, 2010, pp. 2254–2257.

[118] J. F. Gemmeke, T. Virtanen, and A. Hurmalainen, "Exemplar-based sparse representations for noise robust automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2067–2080, 2011.

[119] M. D. Wachter, K. Demuynck, D. V. Compernolle, and P. Wambacq, "Data driven example based continuous speech recognition," in *Interspeech*, 2003.

[120] M. D. Wachter, K. D. M. Matton, P. Wambacq, R. Cools, and D. V. Compernolle, "Template-based continuous speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, pp. 1377–1390, 2007.

[121] S. Demange and D. V. Compernolle, "HEAR: An hybrid episodic-abstract speech recognizer," in *Interspeech*, 2009, pp. 3067–3070.

[122] T. N. Sainath, B. Ramabhadran, D. K. D. Nahamoo, D. Compernolle, K. Demuynck, J.Gemmeke, J. Bellegarda, and S. Sundaram, "Exemplar-based processing for speech recognition: An overview," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 98–113, 2012.

[123] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[124] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[125] R. Michalski, J. Carbonell, and T. Mitchell, *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[126] T. Robinson, *Several improvements to a recurrent error propagation network phone recognition system*. University of Cambridge, 1991.

[127] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *IEEE International Joint Conference on Neural Networks*, vol. 4, 2005, pp. 2047–2052.

[128] A. K. Dhaka and G. Salvi, "Semi-supervised learning with sparse autoencoders in phone classification," *ArXiv preprint arXiv:1610.00520*, 2016.

[129] R. Duda, P. Hart, and D. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[130] H. Abdi and L. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[131] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-A brief tutorial," *Institute for Signal and Information Processing*, vol. 18, 1998.

[132] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[133] I. K. Fodor, *A survey of dimension reduction techniques*, 2002.

[134] F. Camastra, "Data dimensionality estimation methods: A survey," *Elsevier Pattern Recognition*, vol. 36, no. 12, pp. 2945–2954, 2003.

[135] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Springer Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[136] K. C. Gowda and G. Krishna, "The condensed nearest neighbor rule using the concept of mutual nearest neighborhood," *IEEE Transactions on Information Theory*, vol. 25, no. 4, pp. 488–490, 1979.

[137] G. W. Gates, "The reduced nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 431–433, 1972.

[138] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 408–421, 1972.

[139] D. R. Wilson and T. R. Martinez, "Instance pruning techniques," in *International Conference on Machine Learning*, vol. 97, 1997, pp. 403–411.

[140] J. L. Bentley, "Multidimensional divide-and-conquer," *Communications of the ACM*, vol. 23, no. 4, pp. 214–229, 1980.

[141] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, 1977.

[142] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *ACM symposium on Theory of Computing*, 1998, pp. 604–613.

[143] S. Roweis, G. E. Hinton, and R. Salakhutdinov, "Neighbourhood component analysis," in *Advances in Neural Information Processing Systems*, 2004, pp. 513–520.

[144] S. Arya and D. M. Mount, "Approximate nearest neighbor queries in fixed dimensions," in *ACM-SIAM Symposium on Discrete Algorithms*, vol. 93, 1993, pp. 271–280.

[145] X. L. Zhang, "Universal background sparse coding and multilayer bootstrap network for speaker clustering," in *Interspeech*, 2016, pp. 1858–1862.

[146] ——, "Multilayer bootstrap networks," *ArXiv preprint arXiv:1408.0848*, 2014.