A Discriminative Locally-Adaptive Nearest Centroid Classifier for Phoneme Classification

by

Yong-Peng Sun

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2012

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Phoneme classification is a key area of speech recognition. Phonemes are the basic modeling units in modern speech recognition and they are the constructive units of words. Thus, being able to quickly and accurately classify phonemes that are input to a speech-recognition system is a basic and important step towards improving and eventually perfecting speech recognition as a whole.

Many classification approaches currently exist that can be applied to the task of classifying phonemes. These techniques range from simple ones such as the nearest centroid classifier to complex ones such as support vector machine. Amongst the existing classifiers, the simpler ones tend to be quicker to train but have lower accuracy, whereas the more complex ones tend to be higher in accuracy but are slower to train. Because phoneme classification involves very large datasets, it is desirable to have classifiers that are both quick to train and are high in accuracy. The formulation of such classifiers is still an active ongoing research topic in phoneme classification. One paradigm in formulating such classifiers attempts to increase the accuracies of the simpler classifiers with minimal sacrifice to their running times. The opposite paradigm attempts to increase the training speeds of the more complex classifiers with minimal sacrifice to their accuracies.

The objective of this research is to develop a new centroid-based classifier that builds upon the simpler nearest centroid classifier by incorporating a new discriminative locally-adaptive training procedure developed from recent advances in machine learning. This new classifier, which is referred to as the discriminative locally-adaptive nearest centroid (DLANC) classifier, achieves much higher accuracies as compared to the nearest centroid classifier whilst having a relatively low computational complexity and being able to scale up to very large datasets.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Professor Fakhreddine Karray, for guiding me through this research. His enlightening discussions, valuable suggestions, and strict guidance helped me to complete this research. Throughout my term as a graduate student, he has been a source of knowledge and trusted advice. I thank my readers, Professor Douglas Harder and Professor Hamid Tizhoosh, who provided me with invaluable advices and expertise for revising and perfecting my thesis. I also thank Dr. Jiping Sun and his colleagues in Vestec Inc. for their valuable discussions and suggestions during this research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

An important goal of phoneme recognition is to devise a classifier that trains quickly and predicts accurately. In this chapter, the background to this research regarding a novel phoneme classifier is introduced first. This is followed by outlining the motivations behind this research, the objectives and the contributions, along with major items pertinent to the organization of the thesis.

1.1 Background

Phonemes, such as the vowel "aa" and the nasal "m", form the building blocks of words and of speech as a whole. Therefore, the classification or recognition of phonemes is a fundamental component of speech recognition by means of complex automatic speech recognition (ASR) systems. How well phonemes can be classified or recognized is a key factor for determining the performance of ASR systems.

Phoneme classification involves using training samples to construct a suitable model and using this model to predict the class label of any novel sample by analyzing its feature values. Phoneme classification involves a number of steps. The first step is signal processing. In this step, typically each 25 ms windows consisting of 1-dimensional values of a speech audio signal are transformed using Fast Fourier Transform (FFT) and the Mel-Frequency Cepstral Coefficients (MFCC) algorithm into (typically in the case of 8 kHz telephone data) 27-dimensional MFCC feature vectors. Typically the window slides forward with a step size of 10 ms. Starting from the beginning of any given audio signal, every 10 ms of speech audio signal is transformed into a 27-dimensional MFCC feature vector that is referred to as a frame. Figure 1.1 below illustrates the transformation of an audio signal into MFCC feature vectors or frames.

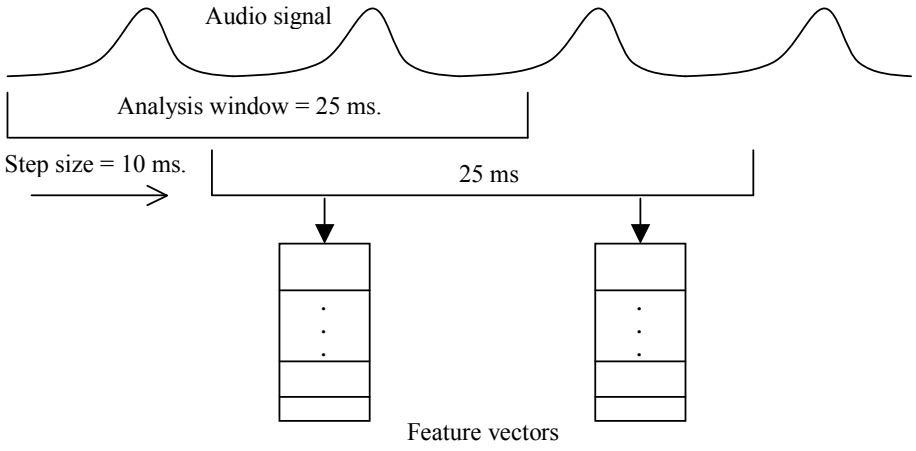


Figure 1.1 The transformation of an audio signal into MFCC feature vectors

On the average, a person speaks about 10 phonemes in a second. The length of a phoneme varies. A phoneme can be as short as 1 frame and as long as 20 frames. Figure 1.2 below illustrates the structure of a phoneme analyzed into 10 frames or feature vectors.

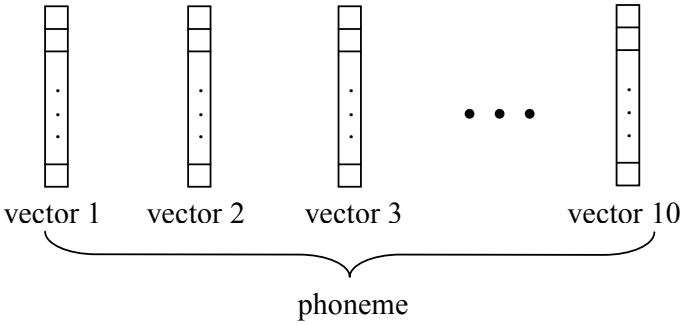


Figure 1.2 The structure of a phoneme sample in terms of frames

In phoneme classification, the boundaries between the frames must be given to demarcate one phoneme sample from the next. These boundaries could be obtained by means of manual labeling as in the case of phoneme databases. These boundaries could also be automatically obtained by an ASR system during the decoding process.

Phoneme classification has many uses in the broader topic of speech recognition. In an ASR system, when the recognition model has been applied to recognize words, how well phonemes have been recognized on the demarcated phonemes could be re-evaluated by a phoneme classification system to provide a measure of confidence [28][29] for the words recognized by the ASR system. In a phoneme recognition system, as the boundaries between phonemes are dynamically searched or independently detected, phoneme classifiers could be used to convert speech audio signals to sequences of phoneme symbols that could in turn be used in areas such as audio search or robotic control.

1.2 Motivations

A broad range of classifiers could be used for phoneme classification. Classifiers such as the nearest centroid classifier and *k*-means are simple in structure and very quick to run but typically result in lower accuracies. Classifiers such as recurrent time-delayed neural network and support vector machine are very complex in structure and take much longer to run but typically result in higher accuracies.

For phoneme classification, because very large numbers of training samples are used for model construction, it is desirable to have an ideal classifier that is quick to run and gives good accuracies. The formulation of such classifiers is still an active ongoing research topic in speech recognition and phoneme classification.

One approach for formulating an ideal classifier is to take a classifier that is complex in structure and reduce its running time while minimally sacrificing its accuracy. A result of this approach is Platt's Sequential Minimal Optimization (SMO), which breaks down the

complex quadratic optimization problem of standard SVM into a number of smaller 2-dimensional sub-problems that could be solved sequentially with analytical techniques.

Another approach for formulating an ideal classifier is to take a classifier that is simple in structure and increase its accuracy while minimally sacrificing its running time. A result of this approach is supervised *k*-means introduced by Al-Harbi et al. in 2006 [2], which is a centroid-based classifier that builds upon *k*-means. Supervised *k*-means adapts the efficient *k*-means algorithm by using the weighted Euclidean metric in place of the Euclidean metric, modifying the objective function of *k*-means, and using simulated annealing to optimize the values of the weights. On real datasets, Al-Harbi et al. have shown that supervised *k*-means obtains very good accuracies and it is also quick to run. The very good performance of supervised *k*-means was the motivation that led me to follow this approach in this research.

In this research, the motivation behind formulating a discriminative locally-adaptive training procedure and incorporating this procedure in a novel centroid-based phoneme classifier comes from the many efficient and useful soft-computing techniques described in Karray and de Silva's 2004 book *Soft-Computing And Intelligent Systems Design* [1]. This centroid-based classifier could be considered a novel soft computing technique due to two reasons. One reason is that each centroid retains all of the features by using a weight to represent the significance of each feature with relation to itself. The other reason is that the mean or the weight vector of each centroid is adjusted during the training process by means of the novel discriminative locally-adaptive training procedure.

1.3 Objectives

The main objective of this research is to follow the second approach towards formulating an ideal classifier that is quick to run and gives good accuracies. The goal was to formulate a novel centroid-based classifier that builds upon the simple nearest centroid classifier.

There are three reasons for choosing a centroid-based structure for this novel classifier. The first reason is that a centroid-based classifier typically has a low computational complexity that results in it being quick to run and being able to scale up to very large datasets. The second reason is that there exist a number of recent advances in machine learning, such as discriminative learning and locally adaptive metrics, which are suitable for improving the accuracies of centroid-based classifiers. The third reason is that a novel centroid-based classifier would be a natural extension of the nearest centroid classifier, because the nearest centroid classifier is a centroid-based classifier whose centroids are comprised of all of the training samples.

An important objective in this research is to adapt some recent advances in machine learning to formulate a novel, mathematically sound, effective, and yet intuitive training procedure for this novel classifier. The objectives for this novel classifier include being able to achieve relatively high accuracies on TIMIT data and having acceptable running times on desktop PCs.

1.4 Contributions

The first contribution of this research involves adapting two recent advances in machine learning, namely discriminative learning explored by Srikanth et al. in 2010 [5] and locally adaptive metrics explored by Domeniconi et al. in 2007 [6], to develop a novel discriminative locally-adaptive training procedure.

The second contribution in this research involves adapting two additional recent advances in machine learning, namely self-learning vector quantization (SLVQ) introduced by Rasanen et al. in 2009 [3] and top-down splitting (TDS) explored by Eick et al. in 2004 [4], to develop a novel centroid-based classifier, which is referred to as the discriminative locally-adaptive nearest centroid (DLANC) classifier. DLANC builds upon the simple nearest centroid classifier and it uses the novel discriminative locally-adaptive training procedure.

The DLANC classifier is simple in terms of structure, has an intuitive training procedure, is very quick to run from start to finish, is able to scale up to large datasets due to its low computational complexity, requires very few input parameters, and has experimentally shown to obtain accuracies that are both relatively high and relatively stable. Figure 1.3 below compares the sequential stages of development that resulted in the extension of the basic nearest centroid classifier to DLANC with those of several other classifiers that also extend the basic nearest centroid classifier[1].

[1] The details are dealt with in the chapter of literature review.

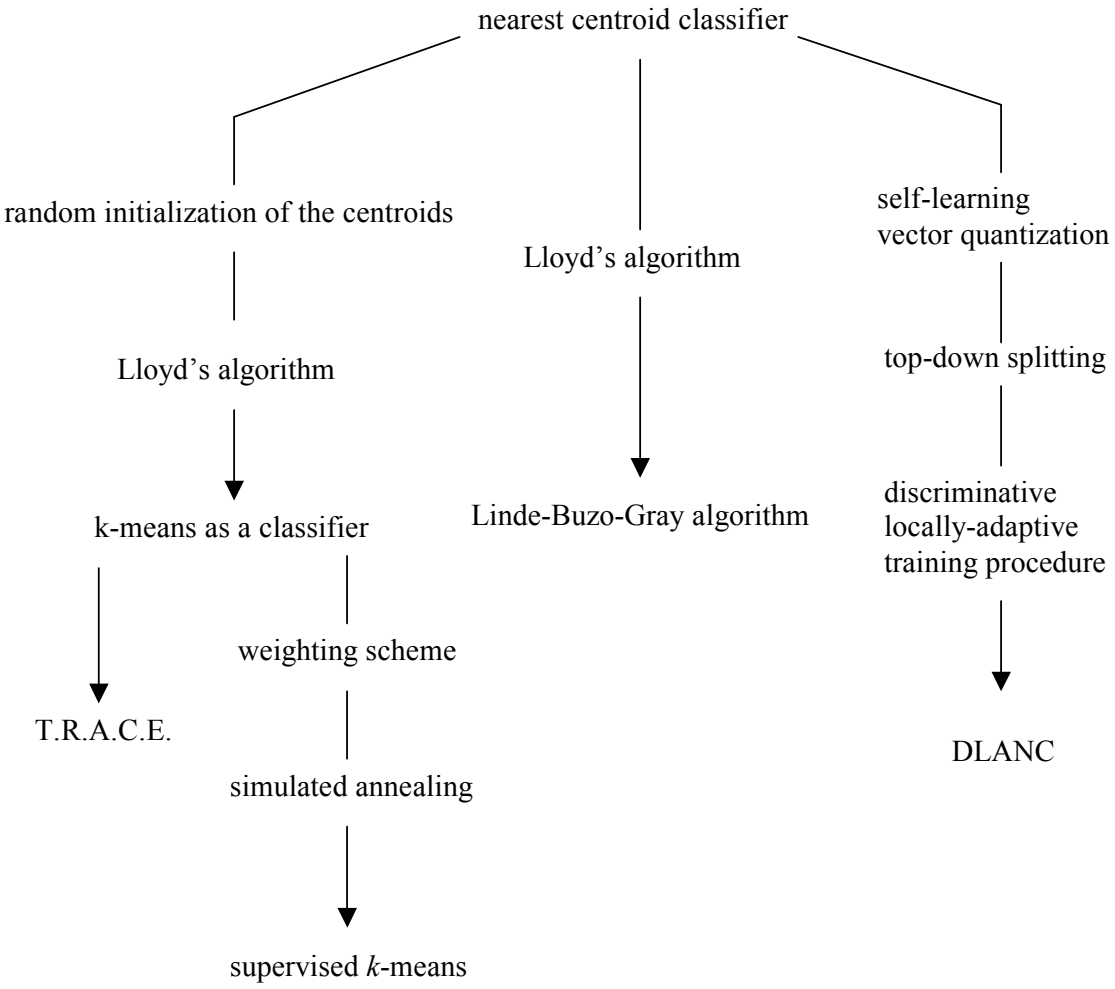


Figure 1.3 How DLANC and several other classifiers extend the nearest centroid classifier

1.5 Organization of the Thesis

In this thesis, some of the common classifiers used for classifying phonemes are reviewed and the recent advances in machine learning that are adapted for formulating the DLANC classifier are discussed first. The proposed approach for classifying phonemes, including the derivation of the DLANC classifier's training algorithm, are then discussed. Following this, the two implementations of the DLANC classifier, the experiments carried out for each implementation to test the performance of this novel classifier and how this novel classifier compares with some of the common classifiers in terms of running time and accuracy for recognizing phonemes are given. Last but not the least, the concluding remarks of the DLANC classifier and some future directions are discussed.

Chapter 2

Literature Review

This thesis reviews several existing methods for classification in detail. Supervised *k*-means is a major motivation behind the direction of this research. Some of the latest works such as self-learning vector quantization and top-down splitting form the basis of the novel DLANC classifier.

2.1 *K*-Means

K-means [27] is a popular clustering algorithm [2, 12]. It partitions a set of N data samples into a set of M clusters. The value of M is specified by the user. Each cluster is associated with a centroid and a set of data samples partitioned to it. The set of centroids is denoted as $Z = \{z_1 , \ldots , z_M\}$, and the centroid of the cluster to which any data sample x_i is assigned is denoted as $z_{A(x_i)} \in Z$, where $A(x_i) \in \{1,...,M\}$ is the index of $z_{A(x_i)}$.

The objective is to minimize the total error in the partitioning of the data samples to the clusters, i.e. the Euclidean distances between the data samples and the associated centroids. In mathematical notation, the objective is

$$\arg\min_{Z} \sum_{i=1}^{N} \| x_i - z_{A(x_i)} \|^2 . \tag{2.1}$$

K-means is carried out with Lloyd's algorithm, which iterates two steps until convergence, i.e. a local optimum is obtained at which the partioning of the data samples to the clusters no longer changes. In the first step, when given the centroids, each data sample is partitioned to the cluster whose centroid is the nearest to it. In the second step, when given the partitioning of the data samples to the clusters, the centroid of each cluster is updated and assigned the mean of the data samples partioned to that cluster. Prior to carrying out Lloyd's algorithm, the centroids of the clusters are typically randomly initialized in such a way that they cover the data samples as best as possible.

A number of developments to *k*-means have been made. One such development is the fuzzy *c*-means algorithm introduced by Dunn in 1973 [13], which assigns each data sample to each cluster with a fuzzy weighting that takes a value between 0 and 1. Another such developoment is the Linde-Buzo-Gray algorithm introduced by Linde et al. in 1980 [14], which begins with a single centroid and, in each iteration, additional centroids are generated by splitting some of the centroids and applying *k*-means to refine the centroids.

2.2 Adaptating *K*-Means for Classification

K-means is typically used as a clustering algorithm. However, when the data samples are labeled, *k*-means can be adapted into a classifier [12]. This classifier could be considered an extension of the simple nearest centroid classifier.

Scheme 1 of using *k*-means as a classifier consists of three steps. In the first step, for each class, *k*-means is applied to cluster the data samples of that class and generate R centroids, where R is specified by the user. In the second step, each centroid is assigned the class label of the data samples from which it was generated. In the third step, each novel data sample is assigned the class label of the centroid nearest to it. Scheme 2 of using *k*-means as a classifier also consists of three steps. In the first step, *k*-means is applied to cluster all of the data samples and generate M centroids, where M is specified by the user. In the second step, each centroid is assigned the most common class label amongst the data samples whose nearest centroid is itself. In the third step, each novel data sample is assigned the class label of the centroid nearest to it.

K-means as a classifer has been successfully applied to a number of areas related to machine learning. One such area is face recognition, where Cifarelli et al. in their 2009 paper *Statistical Face Recognition and Intruder Detection Via a k-means Iterative Algorithm: a Resampling Approach* [15] applied *k*-means in their T.R.A.C.E. (Total Recognition by Adaptive Classification Experiments) algorithm which yielded very good results and which provides an alternative to face recognition using PCA (Principal Component Analysis).

2.3 Supervised *K*-Means

Building upon the adaptation of *k*-means into a classifier, supervised *k*-means is a novel centroid-based classifier introduced by Al-Harbi et al. in their 2006 paper *Adapting K-Means For Supervised Clustering* [2]. It results from the approach of formulating an ideal classifier by means of extending a simple classfier with techniques or algorithms in machine learning so as to increase the accuracy while increasing the running time by as little as possible. It could be considered an extension of the simple nearest centroid classifier.

K-means is efficient due to two reasons. One reason is that it has a computational complexity of $O(nkt)$, where n is the number of data samples, k is the number of clusters with $k \ll n$ and t is the number of iterations. Another reason is that it is guaranteed to converge to a local minimum.

To adapt *k*-means into a classifier, Al-Harbi et al. used the weighted Euclidean metric in place of the Euclidean metric to differentiate the significances of the features in relation to each cluster. A weight is assigned to each feature and the distance between any two data samples $\mathbf{x}$ and $\mathbf{y}$ is calculated as $\sqrt{\sum_{i=1}^{d} w_i \left(x_i - y_i\right)^2}$, where d is the number of features or dimensions. In their paper, Al-Harbi et al. have shown that the use of the weighted Euclidean metric in place of the Euclidean metric does not affect the efficiency of *k*-

means. They used simulated annealing to optimize the weights with the objective of maximizing the confidence of *k*-means' partitioning of the data samples to the clusters. It was found that setting the initial temperature at 1 and using a geometric cooling scheme with a value of 0.99 for the multiplier α and decreasing the temperature once every 300 iterations until the temperature is below 0.005 enables simulated annealing to give good results on training data.

A vector is constructed that contains a weight corresponding to each feature. Each weight is initialized with a random value. Each cluster is associated with a class label. The class label of each cluster is determined to be the most common class label amongst the data samples partitioned to that cluster.

Three steps are carried out in each iteration of the training process. In the first step, at the current values of the weights, the *k*-means clustering algorithm is run using the weighted Euclidean metric to cluster the data samples into k clusters, where k is typically the number of class labels. In the second step, the fitness of the partitioning is calculated as the percentage of the data samples that were partitioned to clusters having the same class labels as themselves. In the third step, the values of the weights are improved using simulated annealing.

A number of iterations are carried out until a local minimum has been obtained. Typically, 20 iterations are sufficient [2]. After the training process is complete, each unlabeled novel sample is assigned the class label associated with the cluster to which that sample is partitioned.

In their experiments, Al-Harbi et al. used the Wisconsin breast cancer, the Pima Indians diabetes, the contraceptive method choice and the auto imports datasets from the UCI repository[2] to compare the performance of supervised *k*-means with that of the C4.5 classification technique. 5-fold cross-validation was used to prevent the models from becoming overfit and to reduce running times. On each dataset, supervised *k*-means resulted in a better accuracy than C4.5 did.

The good performance of supervised *k*-means was the major motivation for me to formulate a novel phoneme classifier by using the approach of extending a simple classifier with techniques or algorithms in machine learning.

2.4 Self-Learning Vector Quantization

Self-learning vector quantization (SLVQ) is a novel type of a family of clustering algorithms known as vector quantization algorithms [16]. It was introduced by Rasanen et al. in their 2009 paper *Self-Learning Vector Quantization for Pattern Discovery from Speech* [3]. It forms the basis of the DLANC classifier's first stage.

SLVQ works in an online manner. It goes over the data samples one at a time to generate a set of centroids. The resulting set of centroids have two benefits. One benefit is that it represents the set of data samples using fewer information, which speeds up computation. Another benefit is that it eliminate as much as possible the undesirable noise

[2] http://archive.ics.uci.edu/ml/

and outliers that are present in the set of data samples.

As with the *k*-means clustering algorithm, the means of the centroids may not be existing data samples. Unlike *k*-means, SLVQ does not have a parameter that specifies the number of centroids to be formed from the data samples. The number of resulting centroids is determined by the values of three parameters, which are the default radius of any centroid, the amount by which the radius of any centroid could be adjusted at any given point in time[3] and the minimum radius that any centroid could have.

SLVQ is carried out on a set of data samples with two steps. In the first step, the first data sample automatically generates the first centroid having the default radius. In the second step, one of two actions is applied to each subsequent data sample. If that sample is not situated within the radius of any existing centroid, then it generates a new centroid having the default radius. If that sample is situated within the radius of some existing centroid, then it is merged with the mean of the centroid nearest to it. In doing so, that mean is moved to a new location.

In the adaptive version of SLVQ, as the algorithm passes over the data samples one at a time, each centroid keeps track of how many data samples are situated within its radius. Taking into account of the mean number of data samples situated within the centroids, each newly-generated centroid or a centroid whose mean is moved to a new location may have its radius adjusted by an amount. This amount is specified as the value of a parameter.

For this research, the adaptive version of SLVQ is modified to generate labeled centroids from labeled data samples. Each newly-generated centroid has the same class label as the data sample that generated it. A data sample could only be merged with a same-labeled centroid; should such a centroid not exist, a new centroid is generated from it. In this modified version of SLVQ, the following steps are carried out:

1. The set of labeled centroids is initialized as the empty set.
2. For each class of data samples, the following steps are carried out:
    a. Using the adaptive version of SLVQ, a set of unlabeled centroids is generated from these data samples.
    b. Each unlabeled centroid is assigned the class label of that class.
    c. These centroids are added to the set of labeled centroids initialized in step 1.

[3] this is a parameter of the adaptive version of SLVQ

Figure 2.1 below shows the merging between a data sample and the mean of a centroid.

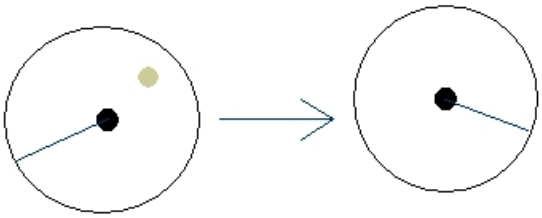


Figure 2.1 SLVQ's merging procedure

2.5 Top-Down Splitting

Top-down splitting (TDS) is an algorithm described in § 4.4 of Eick et al.'s 2004 paper *Supervised Clustering – Algorithms and Benefits* [4]. It forms the basis of the DLANC classifier's second stage. Given a set of centroids that represents a set of data samples, top-down splitting splits the set of centroids into a larger and more representative set of centroids that covers the set of data samples in a more uniform manner.

For this research, top-down splitting was modified to take into account of two points. One point is that the data samples and the centroids have class labels. The other point is that, with its counts, each centroid keeps track of how many data samples of each class are the nearest to it.

A centroid is split into two new centroids whenever two conditions are satisfied for it. One condition is that the value resulting from dividing the second-largest value in its counts by the largest value in its counts exceeds a user-defined threshold. The other condition is that the sum of the two largest values in its counts exceeds a user-defined threshold.

Following the splitting of a centroid, the two new centroids replace the original centroid and they have two properties. One property is that their means are the average of the data samples associated with the largest value in the original centroid's counts and the average of the data samples associated with the second largest value in the original centroid's counts. The other property is that their class labels are those associated with the two largest values in the original centroid's counts.

Following the splitting of a centroid, for each of the two new centroids, its radius is obtained with four steps. In the first step, a sum is initialized with a value of 0. In the second step, the data samples whose nearest centroid is itself are obtained. In the third step, the distances between the data samples obtained in the second step and itself are added to the sum initialized in the first step. In the fourth step, one of two possible actions is made. If the sum initialized in the first step has a value of 0, then its radius is assigned a value of 0. If the sum initialized in the first step has a value greater than 0, then its radius is assigned the average distance between the data samples obtained in the second step and itself.

Figure 2.2 below shows the splitting of a centroid into two new centroids. The centroid being split has the red class label and there are 5 data samples having its own class label and 4 data samples having the blue class label situated within its radius.

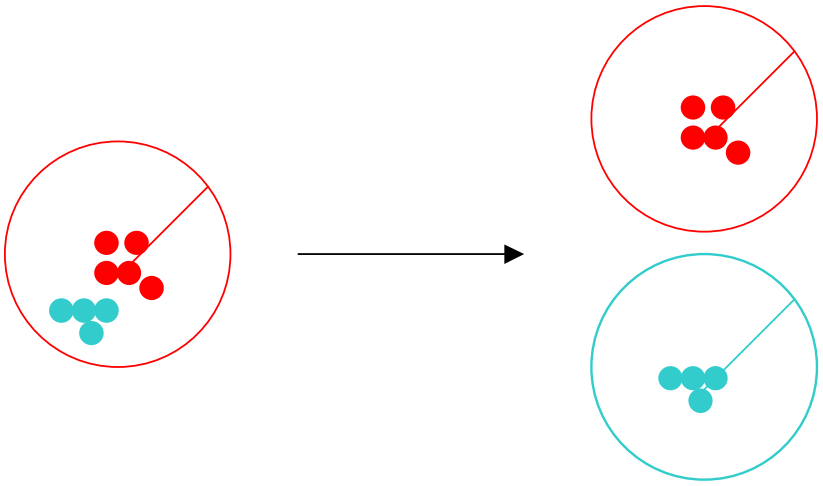


Figure 2.2 The splitting of a centroid into two new centroids

2.6 Discriminative Learning

Srikanth et al. in their 2010 paper *Discriminative Training of Gaussian Mixture Speaker Models: A New Approach* [5] explored the application of discriminative learning to Gaussian mixture speaker models. It is one of two techniques that form the basis of the discriminative locally-adaptive training procedure explored in this research and used in the DLANC classifier's fourth stage.

Traditionally, Gaussian Mixture Model (GMM) is most often used for modeling a speaker's voice using his or her acoustic characteristics. However, GMM does not take into account of a speaker's negative examples, i.e. the data samples that were not spoken by that person. Instead, for any speaker, GMM only takes into account of his or her positive samples, i.e. the data samples that were spoken by that person.

GMM typically gives very good results in terms of speaker identification and it is also very easy to train. To take into account of both the positive samples and the negative samples of any speaker, Srikanth et al. [5] explored the application of a discriminative training procedure to the existing GMM framework.With this new framework, for each speaker, the probabilities of the positive samples and the negative samples are simulataneously maximized and minimized, respectively.

With GMM, for any speaker modeled by a Gaussian $N(\bar{x} \mid \bar{\mu}, \Sigma)$ having the parameters $\bar{\theta} = \{\bar{\mu}, \Sigma, \pi\}$ and having the positive samples denoted by the set D and the negative samples denoted by the set D', the two goals are to maximize the log likelihood of the positive samples with the objective

$$\arg\max_{\bar{\theta}} \ln P(D \mid \bar{\theta}). \tag{2.2}$$

and to minimize the log likelihood of the negative samples with the objective

$$\arg\min_{\bar{\theta}} \ln P(D' \mid \bar{\theta}). \tag{2.3}$$

(2.3) is equivalent to

$$\arg\max_{\bar{\theta}} -\ln P(D' \mid \bar{\theta}). \tag{2.4}$$

Combining (2.2) and (2.4), the objective becomes

$$\arg\max_{\bar{\theta}} \left(\ln P(D \mid \bar{\theta}) - \ln P(D' \mid \bar{\theta})\right). \tag{2.5}$$

A regularization parameter α, where $0 < \alpha \leq 1$, is introduced to remove any imbalance between the number of positive samples $|D|$ and the number of negative samples $|D'|$. As a result, the objective becomes

$$\arg\max_{\bar{\theta}} \left(\alpha \ln P(D \mid \bar{\theta}) - (1-\alpha) \ln P(D' \mid \bar{\theta})\right). \tag{2.6}$$

Denoting $\alpha \ln P(D \mid \bar{\theta}) - (1-\alpha) \ln P(D' \mid \bar{\theta})$ as $l(\theta)$, the optimal solution of the parameters $\bar{\theta} = \{\bar{\mu}, \sum, \pi\}$ is found by finding the partial derivative of $l(\theta)$ with respect to each parameter, setting it to 0 and solving for that parameter.

To test the performance of their new framework for speaker identification, Srikanth et al. [5] used a subset of the NTIMIT database consisting of 200 speakers and a subset of the NIST SRE 2003 corpora consisting of 199 male speakers. On both datasets, a GMM having 32 Gaussian mixtures served as the baseline to which this new framework was applied. In addition, on the NIST SRE 2003 dataset, the baseline GMM with this new framework was compared to a much more complex 1024-mixture UBM-GMM (Universal Background Model – Gaussian Mixture Model). On the NTIMIT dataset, the baseline GMM obtained an accuracy of 41.375 % and the baseline GMM with this new framework obtained a best accuracy of 42.875 %. On the NIST SRE 2003 dataset, the baseline GMM obtained an accuracy of 45.04 % and the baseline GMM with this new framework had a better performance as compared to the baseline GMM and a similar performance as compared to the UBM-GMM. From their experimental results, Srikanth et al. [5] inferred that the baseline GMM with this new framework would have very likely outperformed the UBM-GMM at higher false-alarm probabilities.

2.7 Locally Adaptive Metrics

Domeniconi et al. in their 2006 paper *Locally Adaptive Metrics for Clustering High Dimensional Data* [6] introduced their locally adaptive clustering (LAC) algorithm, which

makes use of locally adaptive metrics. It is one of two techniques that form the basis of the discriminative locally-adaptive training procedure explored in this research and used in the DLANC classifier's fourth stage.

With locally adaptive metrics, the data samples are partitioned to the clusters by taking into account of the relevance of each feature in relation to each cluster. For any centroid, the relevance of a feature to it is inversely related to the variance of the data situated within its radius along that feature.

There are three benefits of locally adaptive metrics. One benefit is the avoidance of losses of information and difficulty in interpreting the features, which are consequences of traditional dimensionality-reduction techniques such as PCA. Another benefit is the avoidance of specifying a model for the distribution of data. Yet another benefit is the avoidance of the curse of dimensionality in high-dimensional data spaces.

Each cluster has a mean and a weight vector that contains a weight corresponding to each feature. Each weight has a value between 0 and 1 and it captures the relevance of the feature associated with it in relation to the cluster associated with it.

Domeniconi et al. [6] used a number of easy-to-understand notations. The number of features is denoted as D. Each data sample is denoted as $\mathbf{x}$. The K clusters to which the data samples are partitioned are denoted as S_j, $j = 1,...,K$, with S_j's mean and weight vector being denoted as $\mathbf{c}_j$ and $\mathbf{w}_j$, respectively.

For any cluster S_j, calculated at its weight vector, the sum of the weighted Euclidean distances between the data samples partitioned to it and its mean must be strictly less than the sum of the weighted Euclidean distances between the data samples partitioned to it and the mean of any other centroid. In mathematical notation, this condition is expressed as

$$S_j = \left\{ \mathbf{x} : \sqrt{\sum_{i=1}^{D} w_{ji} \left(x_i - c_{ji} \right)^2} < \sqrt{\sum_{i=1}^{D} w_{ki} (x_i - c_{ki})^2}, \forall j \neq k \right\}. \tag{2.7}$$

There are four steps for obtaining the optimal solutions of the means and the weights of the clusters. In the first step, the objective is to minimize the sum of the average weighted Euclidean distances between the data samples and the means added up over the K clusters and the D features. In mathematical notation, the objective function that needs to be minimized is

$$E_1(C,W) = \sum_{j=1}^{K} \sum_{i=1}^{D} w_{ji} \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} \left(c_{ji} - x_i \right)^2. \tag{2.8}$$

which is subject to the K constraints $\sum_i w_{ji} = 1, \forall j$. In the second step, denoting the variance of the data samples partitioned to S_j along feature i as X_{ji} and introducing the regularization term $\sum_{i=1}^{D} w_{ji} \log w_{ji}$[4], (2.8) is re-written as

$$E_2(C,W) = \sum_{j=1}^{K} \sum_{i=1}^{D} \left(w_{ji} X_{ji} + h\, w_{ji} \log w_{ji} \right). \tag{2.9}$$

which is subject to the same K constraints. h is a parameter that takes any non-negative value. Its value determines, for any cluster, how the unit weight is distributed amongst the D features. In the third step, introducing a Langrage multiplier λ_j for each of the K constraints, the unconstrained objective function to be minimized is

$$E(C,W) = \sum_{j=1}^{K} \sum_{i=1}^{D} \left(w_{ji} X_{ji} + h\, w_{ji} \log w_{ji} \right) + \sum_{j=1}^{K} \lambda_j \left(1 - \sum_{i=1}^{D} w_{ji} \right). \tag{2.10}$$

In the fourth step, the optimal values of the weights w_{ji} and the means c_{ji}, $j = 1,...,K$ and $i = 1,...,D$, are obtained by finding the partial derivatives of $E(C,W)$ in (2.10) with respect to w_{ji}, λ_j and c_{ji}, setting them to 0 and solving for each variable. Doing so results in

$$w_{ji}^{*} = \frac{\exp\left(-\frac{X_{ji}}{h} \right)}{\sum_{i=1}^{D} \exp\left(-\frac{X_{ji}}{h} \right)}. \tag{2.11}$$

and

$$c_{ji}^{*} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} x_i. \tag{2.12}$$

In LAC, six steps are carried out to cluster a set of data samples into K clusters. In the first step, the K clusters are initialized with the means spread out evenly across the data space and each weight is initialized with a value of $\frac{1}{D}$. In the second step, the data samples are partitioned to the K clusters according to (1). In the third step, the weights are adjusted according to (2). In the fourth step, the data samples are re-partitioned to the K clusters according to (1). In the fifth step, the means are adjusted according to (3). In the sixth step, steps 2 to 5 are repeated until convergence is reached.

The clustering process is relatively quick. Steps 2 to 5 have a computational complexity of $O(KDN)$, where K is the number of clusters, D is the number of features, and N is

[4] According to Friedman and Meulman in 2002, this regularization term represents the negative entropy of the weight distribution for each cluster.

the number of data samples. Due to the exponential weighting scheme in (2), relatively few iterations of steps 2 to 5 are needed to reach convergence.

Domeniconi et al. [6] performed a series of experiments to compare the performance of LAC to those of several other clustering algorithms. First, the comparison was made on five simulated datasets. Each dataset had training and testing samples generated from the multivariate Gaussian distribution, 2 to 3 intrinsic clusters and up to 50 features. When the dimensionality was high, LAC outperformed *k*-means and the EM algorithm that uses full covariance matrices and it performed much better than PROCLUS and DOC. Next, the comparison was made on several real datasets. The OQ-letter, Wisconsin breast cancer, Pima Indians diabetes and Sonar datasets are from the UCI repository[5]. The Image dataset is from the MIT Media Lab[6]. The three high-dimensional text datasets Classic3, Spam2000 and Spam5996 are subsets of the Email-1431 dataset. The two gene expression datasets are a B-cell lymphoma dataset and a microarray dataset. The microarray dataset is the only real dataset that consists of unlabeled data. On six of the real datasets containing labeled data, LAC obtained the best clustering result. On the Spam2000 and Spam5996 datasets, LAC resulted in relatively very low false positive and false negative rates. On the microarray dataset, the biclusters obtained by LAC consistently had relatively very low mean squared residual scores, which was desirable.

[5] http://archive.ics.uci.edu/ml/
[6] www.media.mit.edu/

Chapter 3

The Proposed Approach

In this chapter, an overview of DLANC is provided. The structural components and the various stages for tackling DLANC are discussed. Following this, the discriminative locally-adaptive training procedure, which is used in the learning stage of DLANC, is derived. Lastly, the training and testing procedures of DLANC are given.

3.1 An Overview of the DLANC Classifier for Phoneme Classification

In this research, the formulation of a novel classifier for phoneme classification was explored. This classifier, which extends upon the simple nearest centroid classifier, is referred to as the discriminative locally-adaptive nearest centroid (DLANC) classifier. It makes use of four recent advances in machine learning, which are self-learning vector quantization (SLVQ) introduced by Rasanen et al, top-down splitting (TDS) explored by Eick et al., the application of discriminative learning to classification explored by Srikanth et al. [5] and the application of locally adaptive metrics to clustering explored by Domeniconi et al. [6].

A centroid has four pieces of information, which are its class label, its mean vector that specifies its location, its radius and its counts that specifies how many data samples of each class are the nearest to it.

Figure 3.1 below gives a graphical representation of a centroid. This centroid's class label is coded green. Situated within its radius are 85 data samples having its class label, 10 data samples having the class label coded blue, and 6 data samples having the class label coded orange.

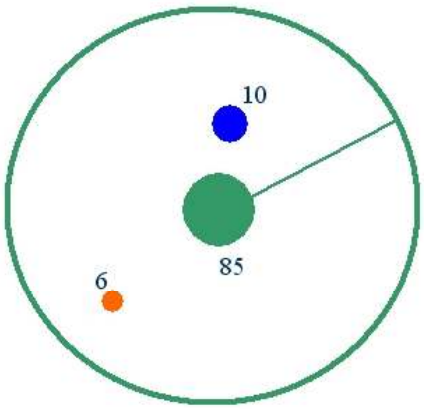


Figure 3.1 A centroid in the framework of DLANC

Figure 3.2 below illustrates the phases and stages of the DLANC classifier.

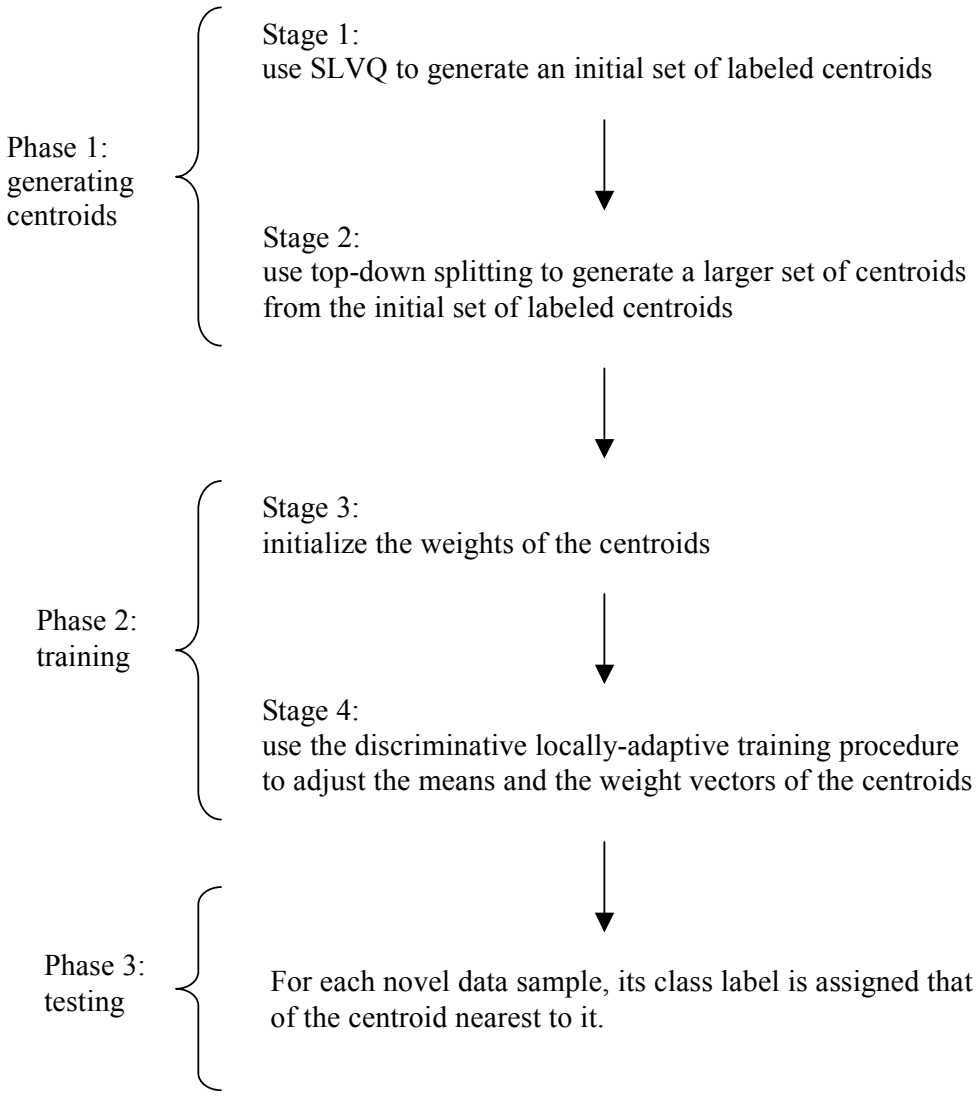


Figure 3.2 The phases and stages of the DLANC classifier

3.2 The Formulation of the Discriminative Locally-Adaptive Training Procedure

The discriminative locally-adaptive training procedure used in the DLANC classifier's fourth stage combines the application of discriminative learning to classification as explored by Srikanth et al. [5] and the application of locally adaptive metrics to clustering

as explored by Domeniconi et al. [6]. It adjusts the means and the weights of the centroids in DLANC's training phase.

A number of easy-to-understand notations were used. The number of features is denoted as D. Each data sample is denoted as $\mathbf{x}$. The K centroids are denoted as S_j, $j = 1,...,K$. The mean and the weight vector of each centroid S_j are denoted as $\mathbf{c}_j$ and $\mathbf{w}_j$, respectively. $\mathbf{x} \in S_j^{=}$ and $\mathbf{x} \in S_j^{\neq}$ represent any training sample situated within the radius of S_j which has the same class label as S_j and any training sample situated within the radius of S_j which does not have the same class label as S_j, respectively.

Given below are the steps with which the optimal values of the means and the weights of the centroids are derived:

1. The objective is to have each centroid being as near as possible to the training samples that have its class label and as far as possible from those that do not. The objective function to be minimized is

$$E = \sum_{j=1}^{K} \sum_{i=1}^{D} w_{ji} \left(\sum_{\mathbf{x} \in S_j^{=}} \left(c_{ji} - x_i \right)^2 - \eta \sum_{\mathbf{x} \in S_j^{\neq}} \left(c_{ji} - x_i \right)^2 \right). \tag{3.1}$$

   subject to the K constraints $\sum_i w_{ji} = 1, \ \forall \ j$.

   η, which has a value between 0 and 1, is a regularization coefficient that automatically eliminates the imbalance between the number of positive samples $|S_j^{=}|$ and the number of negative samples $|S_j^{\neq}|$ for each centroid [6].

2. Denoting $\sum_{\mathbf{x} \in S_j^{=}} \left(c_{ji} - x_i \right)^2 - \eta \sum_{\mathbf{x} \in S_j^{\neq}} \left(c_{ji} - x_i \right)^2$ as X_{ji}, (3.1) is re-written as

$$E = \sum_{j=1}^{K} \sum_{i=1}^{D} w_{ji} X_{ji}. \tag{3.2}$$

3. To prevent the most relevant feature from being associated with a weight value of 1 and every other feature from being associated with a weight value of 0, the regularization term $\sum_{i=1}^{D} w_{ji} \log w_{ji}$ [6], which represents the negative entropy of each centroid's weight distribution, is used to re-write (3.2) as

$$E = \sum_{j=1}^{K} \sum_{i=1}^{D} \left(w_{ji} X_{ji} + h \ w_{ji} \log w_{ji} \right). \tag{3.3}$$

   h, which takes non-negative real values, is a parameter that controls how much the distribution of each centroid's D weights deviate from the uniform distribution [6].

3. Constrained optimization is used to minimize (3.3) by introducing a Lagrange multiplier λ_j for each of the K constraints. The unconstrained objective function that needs to be minimized is

$$E = \sum_{j=1}^{K}\sum_{i=1}^{D}\left(w_{ji}X_{ji} + h\, w_{ji}\log w_{ji}\right) + \sum_{j=1}^{K}\lambda_j\left(1 - \sum_{i=1}^{D} w_{ji}\right). \tag{3.4}$$

4. The optimal values of the weights w_{ji}^{*}, $j = 1,...,K$ and $i = 1,...,D$ are obtained by finding the partial derivatives of E in (3.4) with respect to w_{ji} and λ_j, setting them to 0 and solving for w_{ji}. This is done with the following steps,

   a. $\dfrac{\partial E}{\partial w_{ji}} = X_{ji} + h\log w_{ji} + h - \lambda_j$ and

$$\frac{\partial E}{\partial \lambda_j} = 1 - \sum_{i=1}^{D} w_{ji}\,. \tag{3.5}$$

   b. Setting $\dfrac{\partial E}{\partial w_{ji}}$ to 0 and solving for w_{ji} results in

$$w_{ji} = \frac{\exp\left(\dfrac{-X_{ji}}{h}\right)}{\exp\left(1 - \dfrac{\lambda_j}{h}\right)}. \tag{3.6}$$

   c. Substituting (3.6) into (3.5), setting it to 0 and solving for λ_j results in

$$\lambda_j = -h\log\sum_{i=1}^{D}\exp\left(\frac{-X_{ji}}{h-1}\right). \tag{3.7}$$

   d. Substituting (3.7) back into (3.6) results in

$$w_{ji}^{*} = \frac{\exp\left(\dfrac{-X_{ji}}{h}\right)}{\sum_{d=1}^{D}\exp\left(\dfrac{-X_{jd}}{h}\right)}. \tag{3.8}$$

The optimal values of the means c_{ji}^{*}, $j = 1,...,K$ and $i = 1,...,D$, are obtained by finding the partial derivatives of E in (3.4) with respect to c_{ji}, setting them to 0 and solving for c_{ji}. This is done as follows:

$$\frac{\partial E}{\partial c_{ji}} = \frac{\partial w_{ji} X_{ji}}{\partial c_{ji}}$$

$$= \frac{\partial w_{ji}\left(\sum_{\mathbf{x} \in S_j^{=}} \left(c_{ji} - x_i\right)^2 - \eta \sum_{\mathbf{x} \in S_j^{\neq}} \left(c_{ji} - x_i\right)^2 \right)}{\partial c_{ji}}$$

$$= w_{ji}\left(2 \sum_{\mathbf{x} \in S_j^{=}} \left(c_{ji} - x_i\right) - 2\eta \sum_{\mathbf{x} \in S_j^{\neq}} \left(c_{ji} - x_i\right) \right)$$

$$= 2w_{ji}\left(\sum_{\mathbf{x} \in S_j^{=}} \left(c_{ji} - x_i\right) - \eta \sum_{\mathbf{x} \in S_j^{\neq}} \left(c_{ji} - x_i\right) \right)$$

$$= 2w_{ji}\left(\left(|S_j^{=}| \, c_{ji} - \sum_{\mathbf{x} \in S_j^{=}} x_i \right) - \eta \left(|S_j^{\neq}| \, c_{ji} - \sum_{\mathbf{x} \in S_j^{\neq}} x_i \right) \right)$$

$$= 2w_{ji}\left(|S_j^{=}| \, c_{ji} \; - \; \eta |S_j^{\neq}| \, c_{ji} + \eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i - \sum_{\mathbf{x} \in S_j^{=}} x_i \right)$$

$$= 2w_{ji}\left(\left(|S_j^{=}| \; - \; \eta |S_j^{\neq}| \right) c_{ji} + \eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i - \sum_{\mathbf{x} \in S_j^{=}} x_i \right)$$

$$= 2w_{ji}\left(|S_j^{=}| \; - \; \eta |S_j^{\neq}| \right) c_{ji} + 2w_{ji}\left(\eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i - \sum_{\mathbf{x} \in S_j^{=}} x_i \right).$$

By setting $\frac{\partial E}{\partial c_{ji}}$ to 0, we get

$$\frac{\partial E}{\partial c_{ji}} = 0$$

$$\Rightarrow \; 2w_{ji}\left(|S_j^{=}| \; - \; \eta |S_j^{\neq}| \right) c_{ji}^{*} = 2w_{ji}\left(\sum_{\mathbf{x} \in S_j^{=}} x_i - \eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i \right)$$

$$\Rightarrow \; \left(|S_j^{=}| \; - \; \eta |S_j^{\neq}| \right) c_{ji}^{*} = \sum_{\mathbf{x} \in S_j^{=}} x_i - \eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i$$

$$\Rightarrow \qquad c_{ji}^{*} = \frac{\sum_{\mathbf{x} \in S_j^{=}} x_i - \eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i}{|S_j^{=}| \; - \; \eta |S_j^{\neq}|}. \tag{3.9}$$

3.3 The Training and Testing Processes of the DLANC Classifier

DLANC's training process is composed of four stages. In the first stage, self-learning vector quantization (SLVQ) is used to generate an initial set of centroids from the training samples. In the second stage, top-down splitting (TDS) is used to split the initial set of centroids, often more than once, to obtain a larger set of centroids. In the third stage, for each centroid's D-dimensional weight vector, each of the D weights is initialized with a value of $\frac{1}{D}$, where D is the number of features. In the fourth stage, the discriminative locally-adaptive metrics training procedure is used to adjust the means and the weights of the centroids.

After the training process is complete, DLANC predicts the class label of any novel unlabeled sample in a similar manner as the nearest centroid classifier does by assigning it the class label of the centroid nearest to it in terms of weighted Euclidean distance.

Chapter 4

Implementations and Experiments

In this chapter, the data used in the experiments of DLANC, including how the MFCC vectors are obtained, are discussed. Following this, the two implementation of DLANC, the various experiments and the results are given.

4.1 The TIMIT Data Used in the Experiments

The data used in the experiments consisted of speaker-independent continuous English speech corpus from the TIMIT[7] database. TIMIT was created to provide high quality and low noise speech data. It is the standard database used internationally by researchers for acoustic and phonetic studies and for evaluating automatic speech recognition (ASR) systems. TIMIT contains broadband recordings of 630 speakers who spoke 8 major dialects of American English. Each speaker read 10 sentences that were phonetically rich. The Massachusetts Institute of Technology (MIT) and Texas Instruments, Inc. (TI) were two of the primary parties involved in the making of this popular database in the year 1993. This is how TIMIT's name came about. To ensure correctness and reliability, all transcriptions in the TIMIT corpus have been manually checked.

A total of 138,475 training samples and 2,500 test samples were used. These data covered 8 dialect regions, and they were representative of speech in the real world. These data were down-sampled to 8 kHz and they were filtered by u-law. 5-frame MFCC vectors of 27 dimensions each were concatenated to form 135-dimensional vectors.

MFCC (Mel-Frequency Cepstral Coefficients) are the coefficients that constitute an MFC (Mel-Frequency Cepstrum). They are important in refining raw acoustic signals for areas such as speech recognition. According to Muda et al. in their 2010 paper *Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques* [7], MFCC is useful because "*the extraction of the best parametric representation of acoustic signals is an important task to produce a better recognition performance*". MFCC is sequentially carried out with the following steps:

1. The **pre-emphasis** step. In this step, a speech signal is passed through a filter that increases the energies of signals that have high frequencies. Denoting X as the input and Y as the output, an example of such a filter is $Y[n] = X[n] - 0.95X[n-1]$, where 0.95 means that 95 % of any sample is assumed to come from the previous sample.
2. The **framing** step. In this step, the speech signals obtained from analog to digital conversion (ADC) are each segmented into small frames having lengths in the range of 20 msec to 40 msec. Typically, a voice signal is divided into frames that are each of N samples, where frames that are next to one another are separated by M samples, with $M < N$. Typical values of N and M are 256 and 100, respectively.

[7] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1

3. The **Hamming windowing** step. In the process of extracting signal features, Hamming windowing is used for putting together the closest blocks in terms of frequency. Denoting a Hamming window as $W(n)$, where $0 \le n \le N-1$, applying Hamming windowing results in

$$Y(n) = X(n)W(n), \tag{4.1}$$

where N represents the number of samples contained in each frame, $X(n)$ represents the input signal and $Y(n)$ represents the output signal. An example of $W(n)$ is $W(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right)$.

4. The **Fast Fourier Transform** step. This step is necessary for converting each frame containing N samples from the time domain into the frequency domain. In the time domain, Fast Fourier Transform is applied to convert the convolution of the glottal pulse $U[n]$ and the vocal tract impulse response $H[n]$. Applying Fast Fourier Transform results in

$$Y(w) = \mathrm{FFT}\left[H(t) * U(t)\right] = H(w)U(w), \tag{4.2}$$

where $U(w)$, $H(w)$ and $Y(w)$ are the Fast Fourier Transforms of $U(t)$, $H(t)$ and $Y(t)$, respectively.

5. The **Mel Filter Bank Processing** step. In the output of the Fast Fourier Transform, the voice frequencies have a very wide range and they typically do not follow the linear scale. To correct the voice frequencies resulting from using FFT by approximating them with a Mel scale, a set of triangular filters is used for computing a weighted sum of these voice frequencies' filter spectral components. Denoting f as a given frequency in Hz, $Mel = 2595 \log_{10}(1+f)\ 700$ is a particular equation that could be used for finding the Mel corresponding to f. Essentially, one applies the logarithmic function to any given frequency for obtaining that frequency's Mel.
6. The **Discrete Cosine Transform** step. In this step, Discrete Cosine Transform (DCT) is used for converting the log Mel from the frequency domain into the time domain. The outputs resulting from the DCT conversion process are referred to as Mel Frequency Cepstrum Coefficients (MFCC) and they consist of acoustic vectors.
7. The **Delta Energy and Delta Spectrum** step. The frames of the input voice signal can change over time, which prompts the need to use additional features in this step to correct changes in the cepstral features over time. Typically used in practice are 39 features consisting of 12 cepstral features plus energy followed by 13 delta features plus 13 double delta or acceleration features. In the window from time t_1 to time t_2, the energy in a frame of a signal X is represented as $\sum X^2[t]$. Each of the 13 delta

features represents the change between frames in the corresponding cepstral or energy feature and, subsequently, each of the 13 double delta features represents the change between frames in the corresponding delta features. The delta at time t is calculated with the cepstral coefficients at times $t-1$ and $t+1$ using

$$d(t) = \frac{c_{t+1} - c_{t-1}}{2}. \tag{4.3}$$

Figure 4.1 below shows an example of the MFCC process:

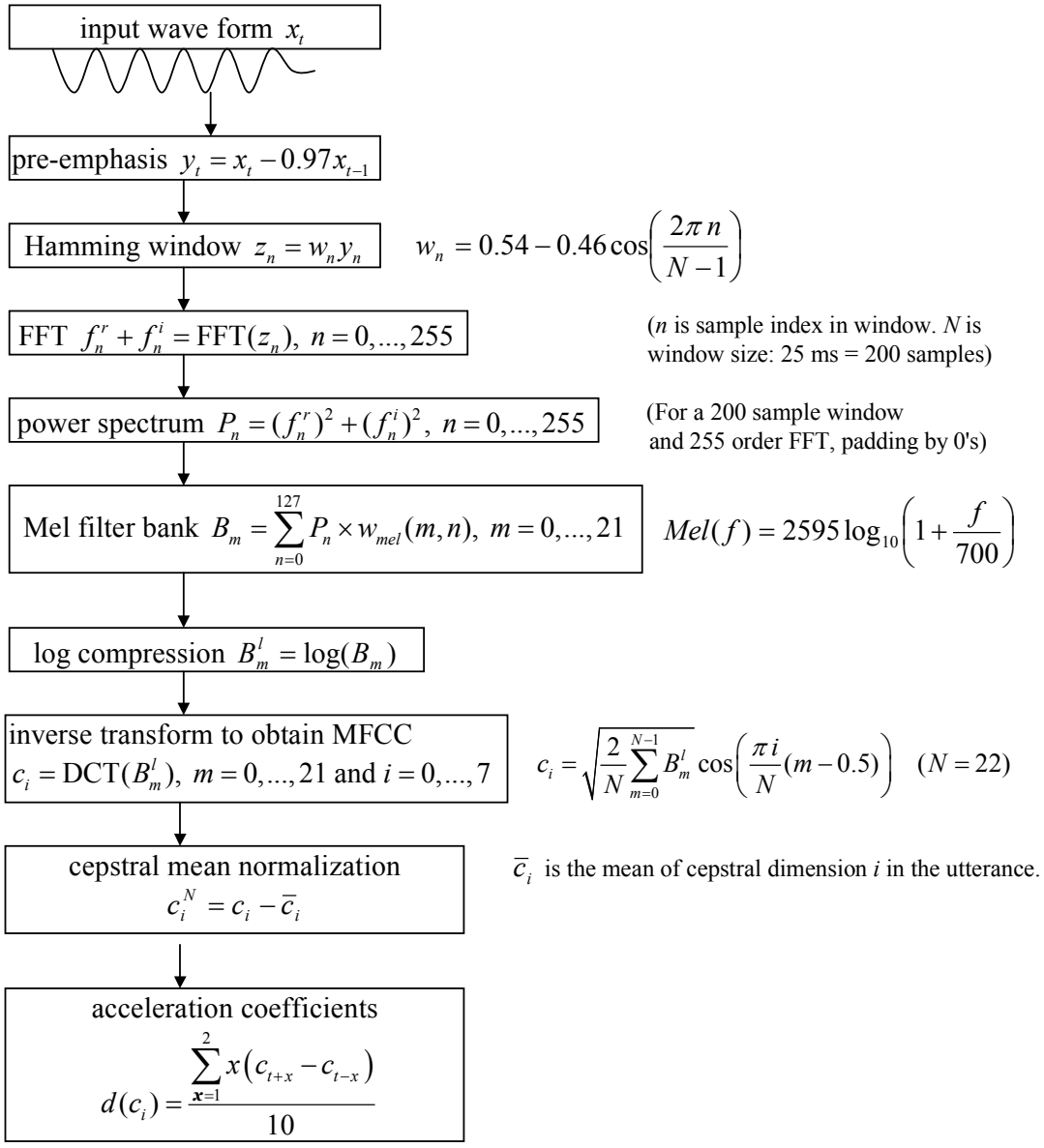


Figure 4.1 How MFCCs can be generated

To obtain segment level feature vectors, the classifier's inputs are the phonemes' feature representations. These feature representations are also referred to as segments. Segments are variable in length when spoken by people at different speeds and by the nature of different kinds of phonemes. For example, a vowel is usually longer than a consonant. To have fixed length vectors for classifier training and testing, the variable durations of segments, i.e. the variable number of frame vectors, are mapped to fixed length vectors. There are different methods of doing this mapping [19]. Common methods include re-sampling and averaging. In averaging, state-aligned frames are averaged for each state and, depending on the HMM structure, either 3 or 5 states are concatenated into a segment level feature vector. In the present work, re-sampling is used. For each segment, 5 frames are chosen and concatenated. Figure 4.2 below gives the 6 rules for selecting 5 frames from a phoneme's MFCC vector sequence. Each rule describes how to map frame vectors of a phoneme to a fixed length vector by means of concatenation. To the left of the $\rightarrow$ symbol is the input pattern and to the right of it is the output pattern. For example, rule 1 says that if the phoneme consists of only 1 frame, then that frame is repeated 5 times.

1. $\mathbf{F}_1 \rightarrow \mathbf{F}_1\ \mathbf{F}_1\ \mathbf{F}_1\ \mathbf{F}_1\ \mathbf{F}_1$

2. $\mathbf{F}_1\ \mathbf{F}_2 \rightarrow \mathbf{F}_1\ \mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_2\ \mathbf{F}_2$

3. $\mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_3 \rightarrow \mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_2\ \mathbf{F}_3\ \mathbf{F}_3\ \mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_2\ \mathbf{F}_2\ \mathbf{F}_3$

4. $\mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_3\ \mathbf{F}_4 \rightarrow \mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_3\ \mathbf{F}_3\ \mathbf{F}_4$

5. $\mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_3\ \mathbf{F}_4\ \mathbf{F}_5 \rightarrow \mathbf{F}_1\ \mathbf{F}_2\ \mathbf{F}_3\ \mathbf{F}_4\ \mathbf{F}_5$

6. >5 frames $\rightarrow$ (middle 5)

Figure 4.2 How 5 frames can be selected from the duration of a segment

To obtain the best possible experimental results, the 135 feature values of each sample were normalized to lie between –1 and 1. In the data text files, each phoneme sample is represented as a single line consisting of a class label between 1 and 41 followed by 135 normalized feature values. Table 4.1 below shows the correspondence between the class labels and the phonemes.

class label	phoneme
1	p
2	t
3	k
4	b
5	d
6	g
7	f
8	s
9	sh
10	th
11	v
12	z
13	zh
14	dh
15	ch
16	jh
17	hh
18	m
19	n
20	ng
21	w
22	y
23	r
24	l
25	ih
26	iy
27	eh
28	ae
29	ey
30	aa
31	ah
32	aw
33	ax
34	er
35	ao
36	aw
37	ow
38	oy
39	uh
40	uw
41	h#

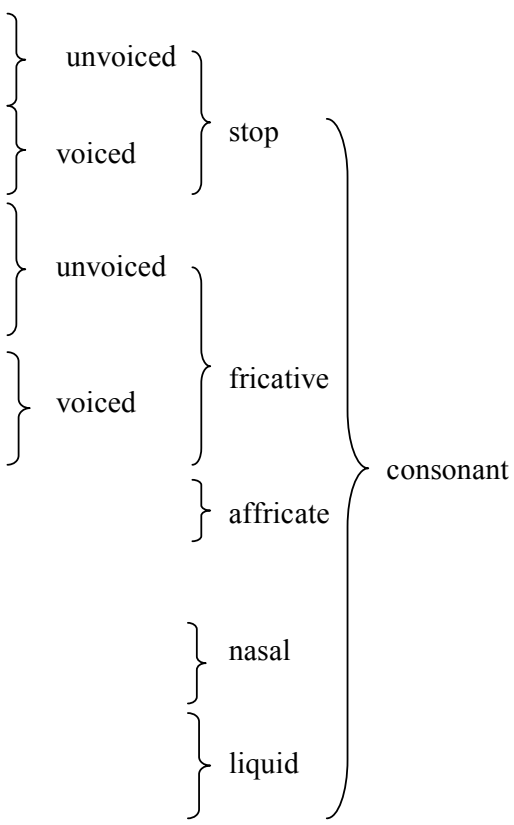


Table 4.1 The 41 phonemes and their class labels

4.2 The First Implementation of the DLANC Classifier

There are five variables that are the most important. The first one is *range*, which specifies, for each training sample, how many of its nearby centroids get their counts updated by it. The second one is *homes*, which, for each training sample, specifies its nearest same-labeled centroid in the case that its nearest centroid does not have its class label. The third one is *scope*, which specifies, for each centroid, how many of its nearest centroids are used in addition to itself for determining its positive and negative samples. The fourth one is *hVals*, which specifies the values of the parameter h. Last but not the least, the fifth one is *ehtaVals*, which specifies the values of the parameter η.

An important function called *fallFlies* uses the training samples to obtain the counts, the class labels of the centroids and the homes of the training samples. This procedure is carried out before each adjustment to the centroids is made. In *fallFlies*, the following steps are done:

1. For each centroid, each of the 41 values in its counts is initialized to 0.
2. For each training sample, the following steps are done:
   a. Its weighted Euclidean distance to the mean of each centroid is obtained.
   b. These distances are sorted from the smallest to the largest using quicksort.
   c. For each of its nearest *range* centroids, the value in the counts corresponding to its class label increases by 1.
3. For each centroid, its class label is the one associated with the largest value in its counts.
4. For each training sample, its *home* is initialized with a value of –1. If its nearest centroid does not have its class label, then the following steps are done to determine its *home*:
   a. Its weighted Euclidean distance to the mean of each same-labeled centroid is obtained.
   b. These distances are sorted from the smallest to the largest using quicksort.
   c. Its *home* is its nearest same-labeled centroid.

Another important function uses the discriminative locally-adaptive training procedure to adjust the weight vector $\mathbf{w}_j$ or the mean $\mathbf{c}_j$ of the generic centroid S_j, $j \in \{1, \ldots, K\}$. In this function, the following steps are done:

1. A temporary weight vector and a temporary mean are initialized with $\mathbf{w}_j$ and $\mathbf{c}_j$, respectively.
2. Calculated at the value of the temporary weight vector, the weighted Euclidean distance between the temporary mean and the mean of each of the other centroids is obtained.
3. These distances are sorted from the smallest to the largest using quicksort.

4. To obtain the positive samples and the negative samples, the following are carried out with each of the pairwise combinations $\{S, \mathbf{x}\}$ of the *scope* centroids and the training samples:

   If S is the nearest centroid of $\mathbf{x}$, then $\mathbf{x}$ becomes a positive sample if S and $\mathbf{x}$ have the same class label or $\mathbf{x}$ becomes a negative sample if otherwise. If S is the not the nearest centroid of $\mathbf{x}$ and S is the *home* of $\mathbf{x}$, then $\mathbf{x}$ becomes a positive sample.

5. A variable *bestGain* is initialized with a value of 0. The average weighted Euclidean distances between the temporary mean and the positive samples and between the temporary mean and the negative samples are calculated at the value of the temporary weight vector and they are stored in the variables *br* and *bw*, respectively.

6. With each of the 48 pairwise combinations of the values of the parameters h and η, where $h \in \{1,5,10,...,25\}$ and $\eta \in \{0.05, 0.1, 0.15, ..., 0.4\}$, the following steps are done:

   a. A 135-dimensional vector X_j is calculated according to

$$X_{ji} = \sum_{\mathbf{x} \in S_j^{=}} \left(c_{ji} - x_i\right)^2 - \eta \sum_{\mathbf{x} \in S_j^{\neq}} \left(c_{ji} - x_i\right)^2, \; i = 1,...,135. \tag{4.4}$$

   b. If the weight vector of S_j is adjusted, then the temporary weight vector is adjusted according to

$$w_{ji}^{*} = \frac{\exp\left(\frac{-X_{ji}}{h}\right)}{\sum_{d=1}^{D} \exp\left(\frac{-X_{jd}}{h}\right)}, \; i = 1,...,135. \tag{4.5}$$

   If the mean of S_j is adjusted, then the temporary mean is adjusted according to

$$c_{ji}^{*} = \frac{\sum_{\mathbf{x} \in S_j^{=}} x_i - \eta \sum_{\mathbf{x} \in S_j^{\neq}} x_i}{|S_j^{=}| - \eta |S_j^{\neq}|}, \; i = 1,...,135. \tag{4.6}$$

   c. The average weighted Euclidean distances between the temporary mean and the positive samples and between the temporary mean and the negative samples are calculated at the value of the temporary weight vector and they are stored in the variables *ar* and *aw*, respectively.

   d. The overall gain from adjusting either the temporary weight vector or the temporary mean is calculated as *br* – *ar* + *aw* – *bw* and it is stored in the variable *gain*.

   e. If *gain* > *bestGain*, then *bestGain* takes the value of *gain* and the following is done:

If the temporary weight vector was adjusted, then $\mathbf{w}_j$ takes the value of the post-adjustment temporary weight vector. If the temporary mean was adjusted, then $\mathbf{c}_j$ takes the value of the post-adjustment temporary mean.

After reading in the training and test samples and the centroids' means and class labels, obtaining the pre-adjustment set of centroids by using SLVQ and TDS and initializing the weight vector of each centroid as 135 values of $\frac{1}{135}$, a while loop is carried out. Each iteration of this loop carries out three steps. In the first step, the user enters values for the variables *adj*, *range* and *scope*. In the second step, if *adj* has a value of –1, 0 or 1, then the loop exits, the weight vector of each centroid is adjusted or the mean of each centroid is adjusted, respectively. In the third step, the accuracy on the test samples is calculated and reported.

4.3 The First Experiment with the First Implementation of DLANC

First, SLVQ was used to generate a set of centroids from the 138,475 training samples. The parameters of SLVQ were selected through two steps. In the first step, the average pairwise Euclidean distance between the first 200 training samples was calculated and stored in the variable *meanPairwiseDistance* and it serves as an estimate of the average pairwise Euclidean distance between the training samples. In the second step, the default radius of the centroids was assigned the value $meanPairwiseDistance/1.50$ and stored in the variable *r*, the minimum radius that any centroid could have was assigned the value $r/24.0$ and stored in the variable *rmin* and by how much the radius of any centroid could be adjusted at any given time was assigned the value $r/14.0$ and stored in the variable *dr*. A set of 4,794 centroids was generated, which took only a few minutes. Figure 4.3 below shows, after using PCA to reduce the dimensionality from 135 to 2, the means of these 4,794 centroids as red dots and the training samples as blue dots.

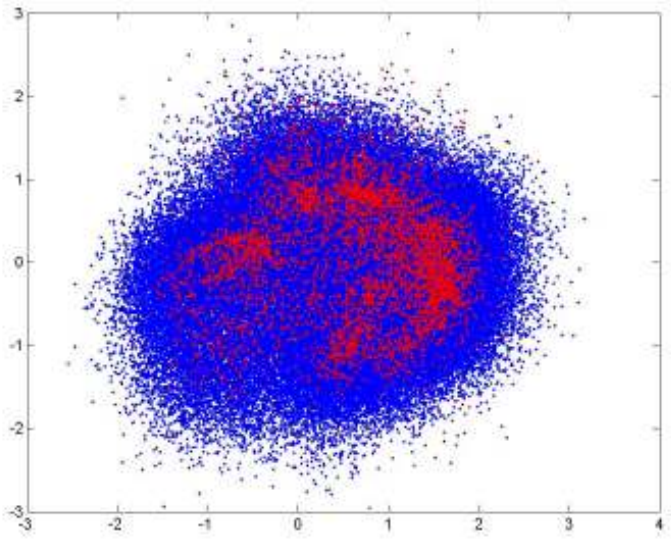


Figure 4.3 The means of the initial centroids in the first experiment

Next, TDS was used to split the initial set of 4,794 centroids.The parameters *ratio* and *sumfLsL*, which correspond to the two considtions[8] for any centroid, were set with the values 0.12 and 8, respectively. The initial set of 4,794 centroids was split into a set of 7,026 centroids. Then, the set of 7,026 centroids was split into a set of 9,368 centroids. Figure 4.4 below shows, after using PCA to reduce the dimensionality from 135 to 2, the means of these 9,368 centroids as red dots and the training samples as blue dots.

8

1. the value resulting from dividing the second-largest value in its counts by the largest value in its counts exceeds a user-defined threshold

2. the sum of the two largest values in its counts exceeds a user-defined threshold

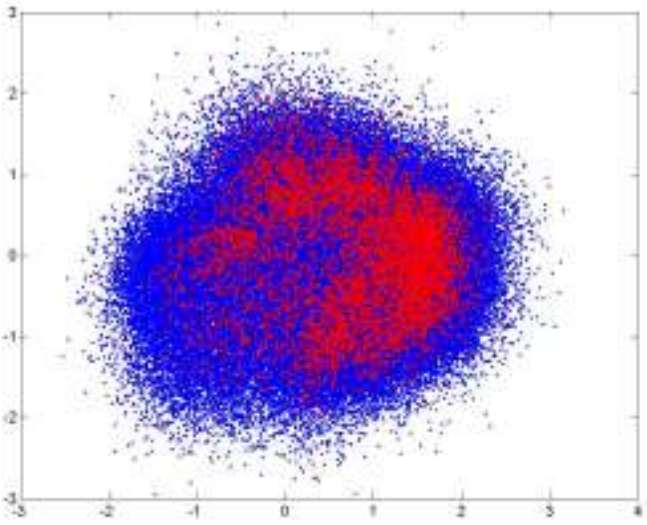


Figure 4.4 The pre-adjustment means of the centroids in the first experiment

Lastly, adjustments to these 9,368 centroids were made. First, the weight vector of each centroid was adjusted at the values of 2 and 1 for the parameters *range* and *scope*, respectively. Then, the mean of each centroid was adjusted at the values of 8 and 3 for the parameters *range* and *scope*, respectively. Figure 4.5 below shows some of the post-adjustment values of the centroids' weights. It could be seen that some of these weights differ significantly from their initial value of $\frac{1}{135} = 0.007407$.

```
0.007543 0.007476 0.007319 0.007157 0.007534 0.007010 0.007521 0.007150
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007482 0.007366 0.007346 0.007368 0.007347 0.007580 0.007450 0.007352
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007068 0.008146 0.008322 0.006536 0.007692 0.007611 0.008240 0.007296
```

Figure 4.5 Some of the post-adjustment values of the centroids' weights in the first experiment

Figure 4.6 below shows, after using PCA to reduce the dimensionality from 135 to 2, the post-adjustment means of these 9,368 centroids as red dots and the training samples as blue dots.

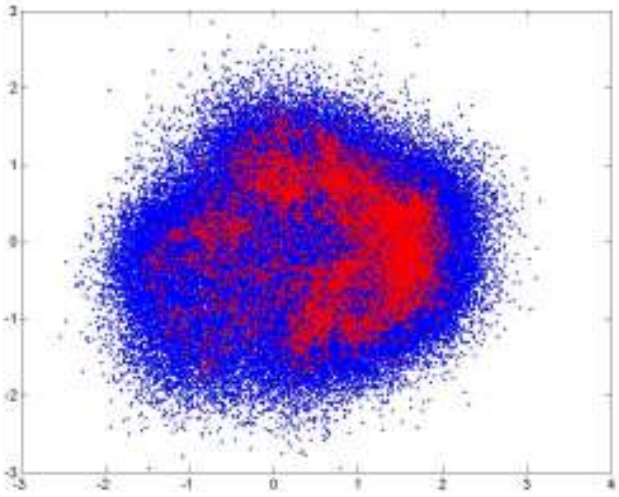


Figure 4.6 The post-adjustment means of the centroids in the first experiment

Table 4.2 below gives the accuracies obtained with the initial set of 4,794 centroids, the set of 7,026 centroids, the pre-adjustment set of 9,368 centroids, the 9,368 centroids after having their weight vectors adjusted and the 9,368 centroids after first having their weight vectors adjusted and then having their means adjusted.

the initial set of 4,794 centroids	61.88 %
the set of 7.026 centroids	65.88 %
the pre-adjustment set of 9,368 centroids	65.48 %
the 9,368 centroids after having their weight vectors adjusted	67.08 %
the 9,368 centroids after first having their weight vectors adjusted and then having their means adjusted	**70.08 %**

Table 4.2 The accuracies obtained in the first experiment

4.4 The Second Experiment with the First Implementation of DLANC

The second experiment explored the possibility of obtaining very good classification results on TIMIT data simply by adjusting the mean of each centroid. The preliminary steps for obtained the pre-adjustment set of centroids were identical to those done in the first experiment with the exception that the default radius of the centroids was set at the value *meanPairwiseDistance* / 1.51 instead of *meanPairwiseDistance* / 1.50 so that a

larger set of initial centroids was generated. An initial set of 5,194 centroids were generated from the 138,475 training samples using SLVQ, which again took only a few minutes. Figure 4.7 below shows, after using PCA to reduce the dimensionality from 135 to 2, the means of the initial 5,194 centroids as red dots and the training samples as blue dots.

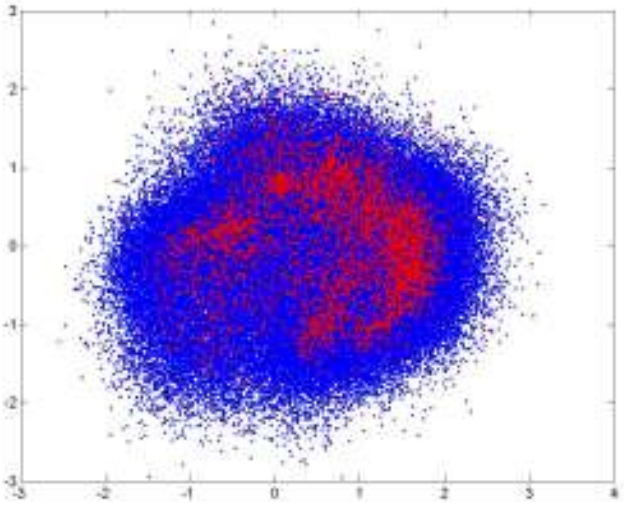


Figure 4.7 The means of the initial centroids in the second experiment

Using TDS, the initial set of 5,194 centroids was split into a set of 7,487 centroids, which was in turn split into a set of 9,891 centroids. Figure 4.8 below shows, after using PCA to reduce the dimensionality from 135 to 2, the means of these 9,891 centroids as red dots and the training samples as blue dots.

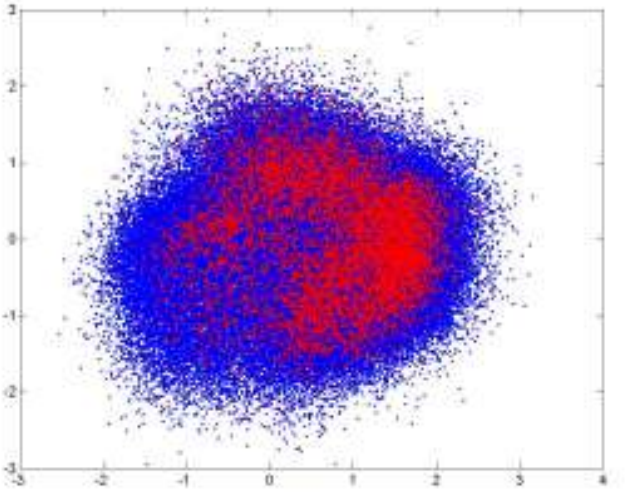


Figure 4.8 The pre-adjustment means of the centroids in the second experiment

The mean of each centroid was adjusted at the values of 8 and 10 for the parameters *range* and *scope*, respectively. Figure 4.9 below shows, after using PCA to reduce the dimensionality from 135 to 2, the post-adjustment means of these 9,891 centroids as red dots and the training samples as blue dots.

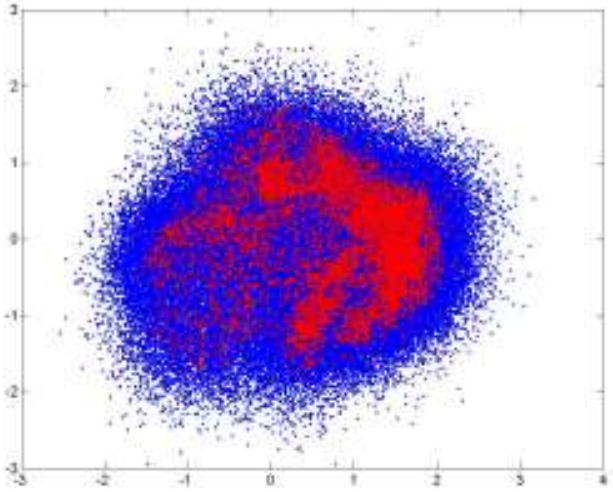


Figure 4.9 The post-adjustment means of the centroids in the second experiment

Table 4.3 below gives the accuracies obtained by the initial set of 5,194 centroids, the set of 7,487 centroids, the pre-adjustment set of 9,891 centroids and the post-adjustment set of 9,891 centroids.

the initial set of 5,194 centroids	60.56 %
the set of 7,487 centroids	65.96 %
the pre-adjustment set of 9,891 centroids	66.68 %
the post-adjustment set of 9,891 centroids	**70.44 %**

Table 4.3 The accuracies obtained in the second experiment

4.5 The Second Implementation of the DLANC Classifier

In the second implementation, the structure was significantly simplified and it consists of the following steps:

1. The training and test samples and the centroids' means and class labels are read in.
2. The value of the parameter *sigma* is obtained by first obtaining the maximum pairwise weighted Euclidean distance between the centroids and then dividing this value by the square root of the number of centroids.
3. The pre-adjustment set of centroids is obtained by using SLVQ and TDS.
4. The weight vector of each centroid is initialized as 135 values of $\frac{1}{135}$.
5. A while loop is carried out. In each iteration of this loop, the following are done:

   The user enters a value for the variable *cont*. If *cont* has a value of –1, then the loop exists. Otherwise, the user enters values for how many positive samples and how many negative samples are used for adjusting the mean of each centroid and how many positive samples and how many negative samples are used for adjusting the weight vector of each centroid. Then, each centroid is adjusted using the discriminative locally-adaptive training procedure. Lastly, the accuracy is calculated and reported along with how many centroids were not adjusted, how many centroids had their means adjusted and how many centroids had their weight vectors adjusted.

An important function uses the discriminative locally-adaptive training procedure to adjust the weight vector $\mathbf{w}_j$ or the mean $\mathbf{c}_j$ of the generic centroid S_j, $j \in \{1,...,K\}$. Six steps are carried out in this function. In the first step, a temporary weight vector and a temporary mean are initialized with $\mathbf{w}_j$ and $\mathbf{c}_j$, respectively. In the second step, calculated at the value of the temporary weight vector, the weighted Euclidean distances between the temporary mean and the training samples are obtained. In the third step, these distances are sorted from the smallest to the largest using quicksort. In the fourth step, using the sorted distances and the values provided by the user in step 5 of this implementation, the positive and negative samples for adjusting $\mathbf{c}_j$ and the positive and negative samples for adjusting $\mathbf{w}_j$ are both obtained as the nearest training samples that have the same class label as S_j and the nearest training samples that do not have the same class label as S_j, respectively. In the fifth step, for both $\mathbf{c}_j$ and $\mathbf{w}_j$, the value of a variable *bestGain* is obtained in a manner analogous to the procedure for obtaining the value of the variable *bestGain* in the second function of the first implementation[9]. In the sixth step, if the value of *bestGain* resulting from adjusting $\mathbf{c}_j$ is the same as, greater than, or less than the value of *bestGain* resulting from adjusting $\mathbf{w}_j$, then S_j is not adjusted, $\mathbf{c}_j$ is adjusted, or $\mathbf{w}_j$ is adjusted, respectively.

[9] please refer to § 4.2 The First Implementation of the DLANC Classifier

4.6 The Experiment with the Second Implementation of DLANC

The pre-adjustment set of 9,891 centroids from the second experiment was used.

The first goal was to observe the result from adjusting only the means of the centroids. The number of positive samples was varied over the values 40, 45 and 50. The number of negative samples was varied over the values 20, 25, 30, 35 and 40. Three steps were carried out at each combination of the number of positive samples and the number of negative samples. In the first step, the mean of each centroid is adjusted with the discriminative locally-adaptive training procedure. In the second step, PCA is used to reduce the dimensionality from 135 to 2 and the post-adjustment means of the centroids are plotted as red dots and the training samples are plotted as blue dots. In the third step, the accuracy on the test samples is calculated and reported.

Table 4.4 below pertains to the 3 combinations with 20 negative samples.

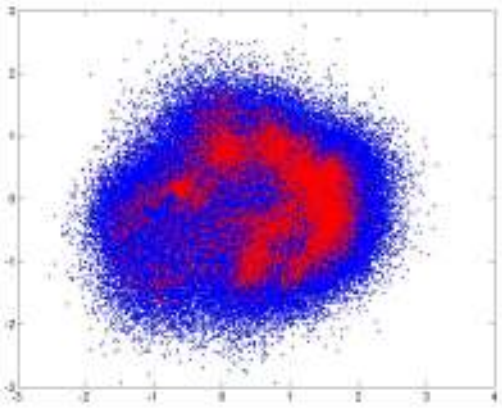
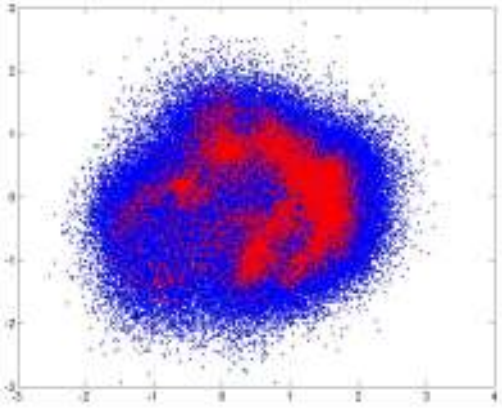
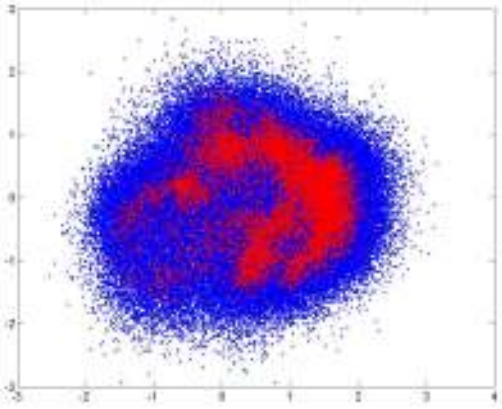
Number of positive samples	40		70.12 %
	45		70.40 %
	50		70.20 %

Table 4.4 The results with 20 negative samples relating to the first goal of the third experiment

Table 4.5 below pertains to the 3 combinations with 25 negative samples.

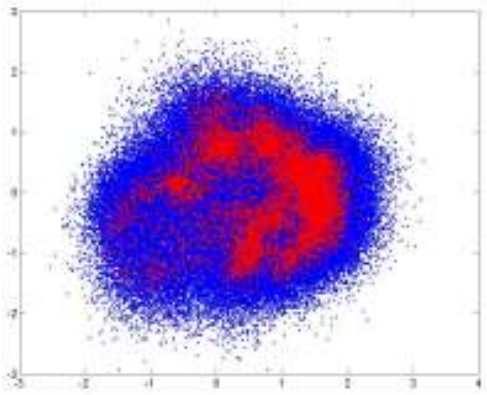
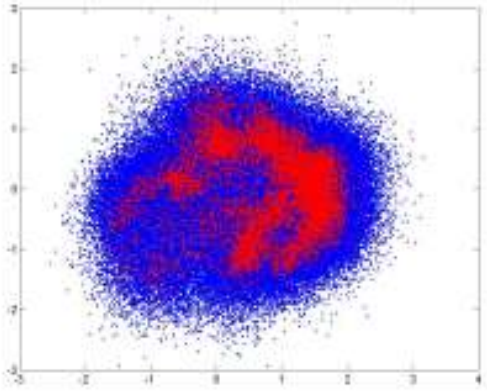
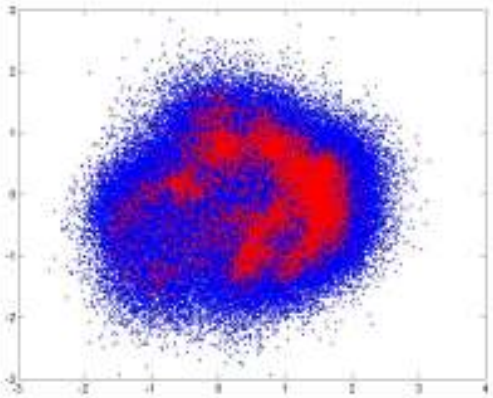
Number of positive samples			
	40		69.88 %
	45		69.76 %
	50		69.92 %

Table 4.5 The results with 25 negative samples relating to the first goal of the third experiment

Table 4.6 below pertains to the 3 combinations with 30 negative samples.

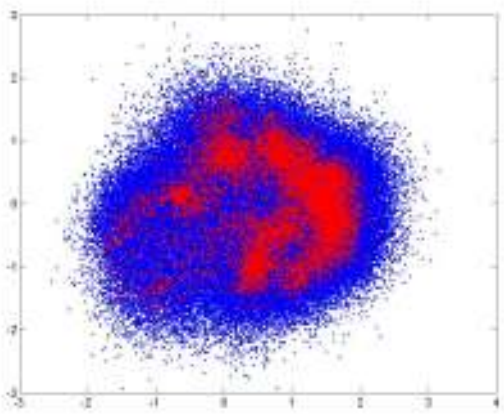
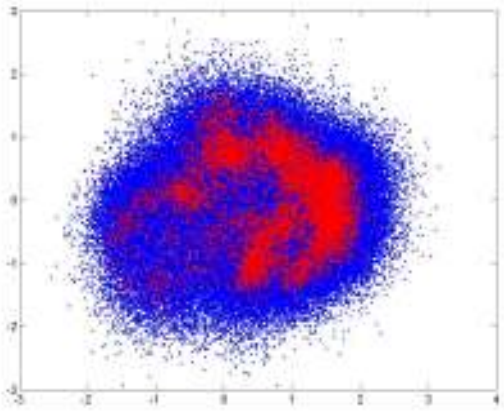
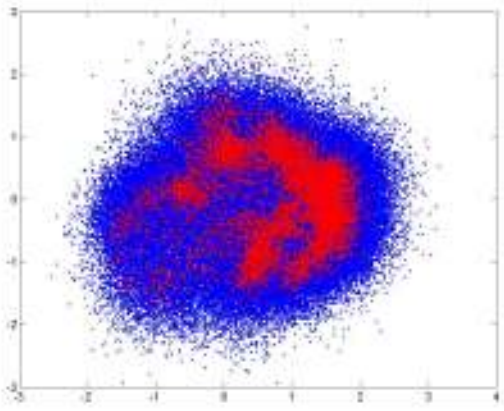
Number of positive samples		
40		69.76 %
45		70.00 %
50		70.52 %

Table 4.6 The results with 30 negative samples relating to the first goal of the third experiment

Table 4.7 below pertains to the 3 combinations with 35 negative samples.

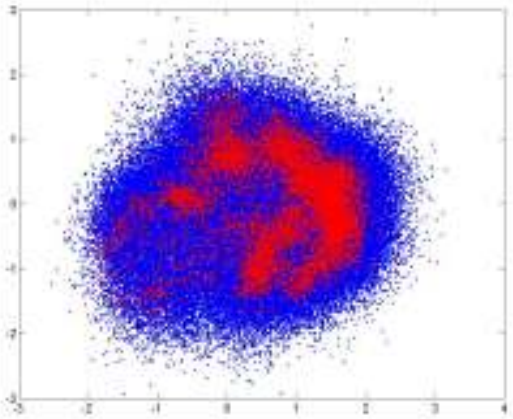
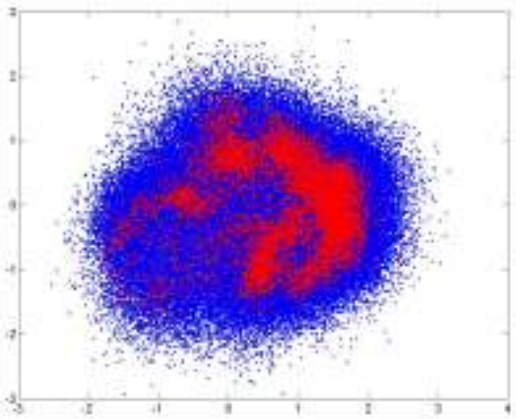
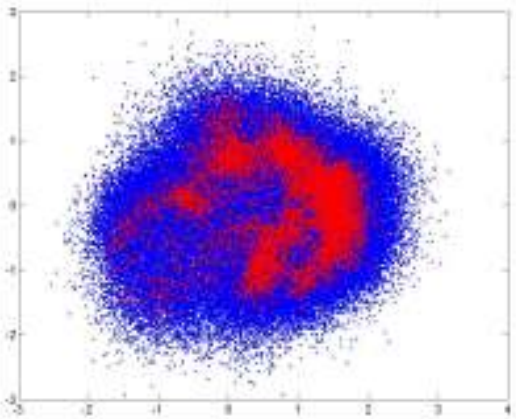
Number of positive samples	40		69.92 %
	45		70.12 %
	50		70.52 %

Table 4.7 The results with 35 negative samples relating to the first goal of the third experiment

Table 4.8 below pertains to the 3 combinations with 40 negative samples.

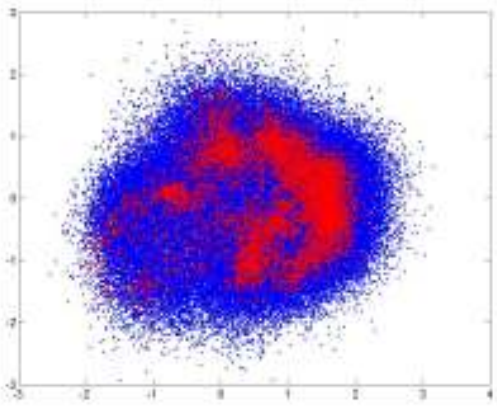
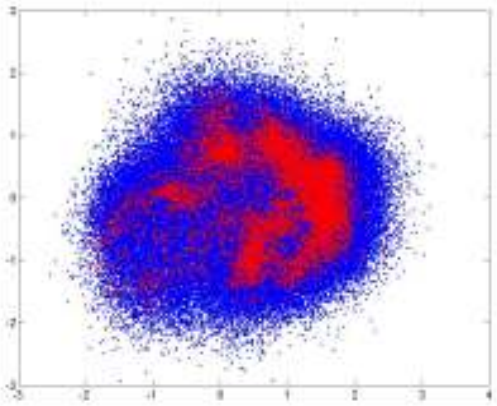
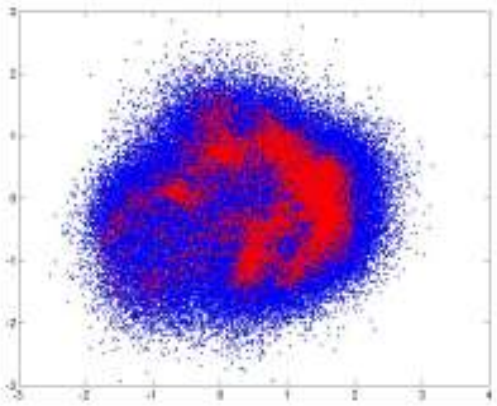
Number of positive samples	40		69.36 %
	45		69.84 %
	50		70.56 %

Table 4.8 The results with 40 negative samples relating to the first goal of the third experiment

The second goal was to observe the result from adjusting only the weight vectors of the centroids. The number of positive samples was varied over the values 2, 3, 4 and 6. The number of negative samples was varied over the values 1, 2 and 3. Table 4.9 below gives DLANC's accuracy after the adjustment at each combination of the number of positive samples and the number of negative samples.

Number of negative samples	Number of positive samples: 6	4	3	2
1	66.88 %	67.04 %	67.20 %	67.44 %
2	65.68 %	67.04 %	67.08 %	67.36 %
3	66.84 %	66.84 %	66.72 %	67.20 %

Table 4.9 The results relating to the second goal of the third experiment

Figure 4.10 below shows some of the post-adjustment values of the centroids' weights that resulted in the best accuracy of 67.44 % after adjusting the weight vector of each centroid with 2 positive samples and 1 negative sample. It could be seen that some of these weights differ significantly from their initial value of $\frac{1}{135} = 0.007407$. For each centroid, the values of its 135 weights are organized as a single line in the output file.

```
0.007511 0.006747 0.006478 0.007272 0.007575
0.007605 0.007174 0.006272 0.007427 0.007468
0.007467 0.007273 0.007821 0.007977 0.007626
0.006989 0.008038 0.007054 0.007664 0.007001
0.007187 0.007503 0.007414 0.007622 0.007206
0.007335 0.007185 0.008070 0.007702 0.007452
0.006990 0.007816 0.007467 0.007080 0.007208
0.007464 0.007367 0.006358 0.006702 0.007207
0.007150 0.007090 0.007316 0.007461 0.007102
0.007419 0.007537 0.007646 0.007542 0.007138
0.007356 0.007682 0.007487 0.006914 0.006991
0.007921 0.006905 0.007184 0.007487 0.007524
```

Figure 4.10 Some of the best post-adjustment values of the centroids' weights relating to the second goal of the third experiment

The third goal was to observe the result from adjusting either the mean or the weight vector of each centroid, depending on which adjustment results in a better improvement as measured by $br - ar + aw - bw$, where *br*, *ar*, *bw* and *aw* are the average distance from that centroid to the positive samples prior to the adjustment, the average distance from that centroid to the positive samples after the adjustment, the average distance from that centroid to the negative samples prior to the adjustment and the average distance from that centroid to the negative samples after the adjustment, respectively. For any centroid, it is not adjusted if adjusting either its mean or its weight vector results in the same amount of improvement. Table 4.10 below gives the accuracy at varying numbers of positive and negative samples. How many positive and negative samples were used for adjusting the mean of any centroid are on the horizontal axis. How many positive and negative samples were used for adjusting the weight vector of any centroid are on the vertical axis.

	10 & 4	20 & 10	30 & 15	40 & 20	40 & 20	52 & 40	50 & 40
5 & 1	66.64 %						
5 & 1		67.24 %					
5 & 1			68.44 %				
7 & 1				68.96 %			
10 & 2					69.04 %		
2 & 1						69.64 %	
1 & 1							70.56

Table 4.10 The results relating to the third goal of the third experiment

The adjustment to the centroids with 52 positive samples and 40 negative samples for adjusting the mean of each centroid and 1 positive sample and 1 negative sample for adjusting the weight vector of each centroid resulted in the best accuracy of 70.56 %. This adjustment resulted in every centroid having either its mean or its weight vector adjusted. 7,670 centroids had their means adjusted and 2,221 centroids had their weight vectors

adjusted. Figure 4.11 below shows, after applying this adjustment, some of the post-adjustment values of the centroids' weights. It could be seen that some of these weights differ significantly from their initial value of $\frac{1}{135} = 0.007407$. For each centroid, the values of its 135 weights are organized as a single line in the output file. Figure 4.12 below shows, after applying this adjustment and using PCA to reduce the dimensionality from 135 to 2, the post-adjustment means of the 9,891 centroids as red dots and the training samples as blue dots.

```
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007434 0.007335 0.007234 0.007362 0.007335 0.007825
0.007343 0.006742 0.008658 0.007863 0.007576 0.007534
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007407 0.007407 0.007407 0.007407 0.007407 0.007407
0.007562 0.006278 0.007293 0.007369 0.007110 0.007262
0.007505 0.006897 0.007563 0.007547 0.007630 0.007670
0.007251 0.004813 0.007273 0.007665 0.007518 0.008693
```

Figure 4.11 Some of the best post-adjustment values of the centroids' weights relating to the third goal of the third experiment

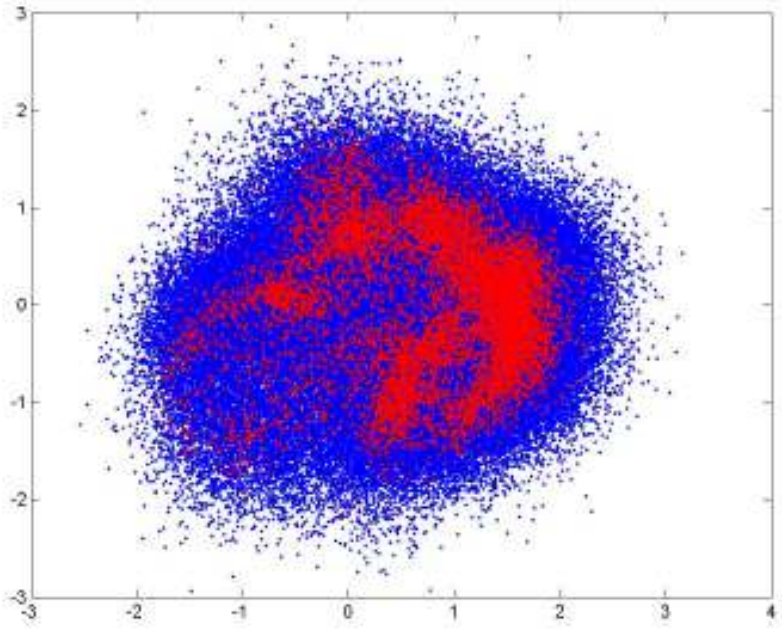


Figure 4.12 The best post-adjustment means of the centroids relating to the third goal of the third experiment

Chapter 5

Comparisons Between DLANC and Other Classifiers

In this chapter, DLANC is compared to a number of existing classifiers used for phoneme classification. First, DLANC is compared to GMM [26], HMM [34] and ANN [35] using results obtained by other researchers. Following this, DLANC is compared to the NC classifier, SVM, PNN and *k*-means using results obtained as part of this research.

5.1 Comparing DLANC with GMM and HMM

Antal et al. in their 2004 paper *Speaker Independent Phoneme Classification In Continuous Speech* [8] used Gaussian Mixture Model (GMM) to classify phonemes. A GMM is a special type of Hidden Markov Model (HMM) with a single state. The performance of GMM was compared to that of the common classical left-to-right 3-state HMM. 6,300 utterances from the TIMIT corpus were used, and these were split into 4,620 for training and 1,680 for testing.

The number of Gaussian mixtures was varied, the maximum-likelihood approach was used for parameter estimation, and only diagonal covariance matrices were used to speed up training. The best accuracy of 60.43 % was obtained with 256 Gaussians and 1,000 occurrences per phone. The best result on TIMIT data obtained by other authors using the common classical left-to-right 3-state HMM was 63.00 %. To achieve this result, full covariance matrices were used, 20 ms / 10 ms was set as the value of frame size per shift, and the features used were *MFCC*-18 + *Delta*.

DLANC vastly outperformed both GMM and HMM, achieving a best accuracy of 70.56% on TIMIT data.

5.2 Comparing DLANC with ANN

Anies et al. in their 2004 paper *Robust Speech Recognition with an Auditory Model* [9] used an advanced version of Artificial Neural Network (ANN), namely Recurrent Time-Delayed Neural Network (RTDNN), to classify phonemes at the frame level. This network was built using the NICO toolkit. The cepstral recognition system was used as the reference system.

The number of hidden neurons was set at 400 to match the cepstral recognition system. The standard tangent hyperbolicus function was used as the activation function. The weights were initialized with random values between –0.1 and 0.1. The training data was split into a training set and a smaller validation set. Using the NICO toolkit's *Backprop* function, the value of the gain parameter was decreased by a factor of 0.8 whenever the objective function failed to decrease on the validation data during the training process. The value of the momentum parameter was empirically set at a value of 0.7. To avoid over-

training this network, the error on the validation set was monitored during the training process. This network was used for training acoustic-based 4 kHz and 8 kHz models. Each model was trained over 30 iterations or epochs and took 10 days to run on a 2.66 GHz standard PC.

The best accuracy of approximately 67 % was obtained on the TIMIT 39 dataset using the auditory-based 8 kHz model. Not only did DLANC obtain a higher accuracy of 70.56 % on TIMIT data, it took only a few hours to run from start to finish on a standard PC.

5.3 Comparing DLANC with the NC Classifier, SVM, PNN and *K*-Means

The nearest centroid classifier, which is the simplest classifier and which serves as the structural foundation of the DLANC classifier, obtains an accuracy of 61.92 % on the 2,500 test samples when the 138,475 training samples were used as the centroids.

SVM [22][23][24] was introduced by Vapnik in 1963. It classifies data samples that are typically not linearly-separable by mapping them to higher-dimensional spaces using a class of one-to-one mappings known as kernel functions. Possible kernel functions include the linear kernel, the radial basis function (RBF) kernel and the polynomial kernel. In higher-dimensional spaces, where the mapped data samples are much more linearly-separable, a maximum-margin hyperplane is constructed to linearly separate any two groups of mapped data samples as best as possible by being situated at the maximal equalized distance from the nearest data sample(s) of either group. These nearest data samples are referred to as support vectors. The maximum-margin hyperplanes become nonlinear boundaries when they are mapped back to the original data spaces, in which they effectively separate the different groups of data samples.

Using the LIBSVM package to implement SVM, an accuracy of 78.5 % was obtained. Though DLANC resulted in a best accuracy of 70.56 %, the complexity of its training process is $O(nck)$ whereas the training process of the standard version of SVM has a complexity of $O(n^3)$ [10], where n is the number of training samples, c is the number of centroids and k is the number of adjustments to the centroids. With DLANC, typically the centroids are adjusted only once to prevent over-fitting.

Probabilistic neural network (PNN) [30][31][32] was introduced by Specht in 1990. It improves upon the back-propagation artificial neural network by making use of a statistically- derived activation function and a statistical technique known as the *Parzen-Rosenblatt window* method [17][18]. Unlike back-propagation ANN, PNN has a single pattern layer whose pattern nodes apply a Gaussian function to each training sample, has very quick running times and its decision boundary asymptotically approaches the Bayes optimality as the number of training samples increases. Using the 138,475 training samples to construct the pattern layer, PNN resulted in an accuracy of 61.7 % on the 2,500 test samples.

Scheme 1 of the application of *k*-means to classification [12] was used for classifying TIMIT data. The same 138,475 training samples from TIMIT were used to generate the centroids and the same 2,500 test samples from TIMIT were used for the testing phase. Table 5.1 below lists the accuracies obtained at several choices of the number of centroids.

Number of centroids	Accuracy
41	49.6 %
4,790	61.8 %
9,600	64.7 %
138,475	60.7 %

Table 5.1 The results of k-means (scheme 1)

Figure 5.1 below shows the accuracy as the number of centroids increases.

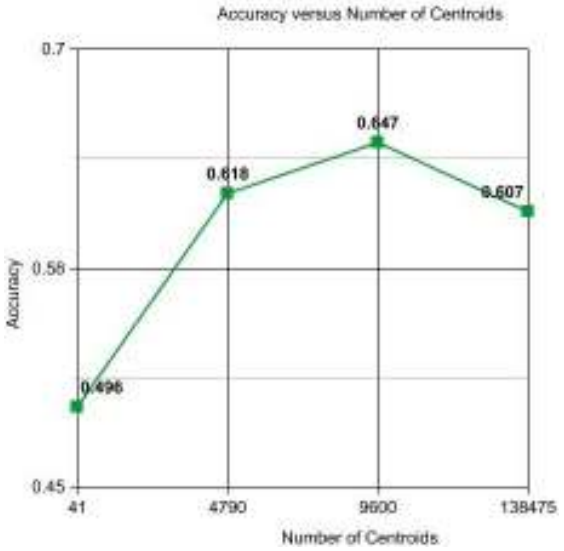


Figure 5.1 the effect on the accuracy as the number of centroids grows

Scheme 2 of the application of *k*-means to classification [12] assigns the class label of each centroid to be the most common class label amongst the data samples nearest to it. It is very similar to the first implementation of the DLANC classifier prior to making any adjustment to the centroids[10], which assigns the class label of each centroid as the largest value in that centroid's counts. This is because the counts of each centroid keeps track of

[10] please refer to § 4.2 The First Implementation of the DLANC Classifier

how many data samples of each class are the nearest to it. With 9,368 centroids, the first implementation of the DLANC classifier resulted in an accuracy of 65.48 % prior to making any adjustment to the centroids. Table 5.2 below lists the best accuracies obtained by DLANC, the nearest centroid classifier, PNN, *k*-means (scheme 1) and SVM.

Algorithm	% Correct
nearest centroid classifier	61.9 %
probabilistic neural network	61.7 %
k-means (scheme 1)	64.7 %
support vector machine	78.5 %
DLANC	70.56 %

Table 5.2 The best accuracies of DLANC and several other classifiers

From Table 5.2 we can see that even though SVM results in the highest accuracy, it takes much longer to train as compared to DLANC. On the 138,475 TIMIT training data, SVM took 12 hours and 3 minutes to train even though Platt's Sequential Minimal Optimization (SMO) was used [25]. On the same training data, DLANC took only 1 hour and 27 minutes to train. In practical applications of speech recognition, where the size of the training data is usually in the millions or more, SVM could not scale up, whereas DLANC could do so due to its linear complexity.

Chapter 6

Conclusion and Future Directions

In this research, I successfully formulated the novel discriminative locally-adaptive nearest centroid (DLANC) classifier and applied it to the task of phoneme classification. DLANC has many good properties. It is accurate, quick to run, has very few parameters, gives stable results when the values of its parameters are moderately varied and it is able to scale up to very large datasets. DLANC is fast to construct, train and run. Consequently, it is easy in practice to find values for DLANC's parameters that result in good accuracies.

In the near future, I would like to modify the DLANC classifier by having it being extended to the k nearest centroids classifier. For example, DLANC's testing phase could be modified by using the k nearest centroids classifier in place of the nearest centroid classifier for obtaining accuracies on test samples. I would also like to apply the DLANC classifier to other real datasets to further test its performance. Furthermore, for future work, I would like to combine the DLANC classifier and PNN by using DLANC to generate the centroids and the values of the weights.

Another future application of DLANC is towards Discriminative Learning for Quantized Time Series [11]. In a quantized time series model, rather than being a concatenation of selected frames from a segment, a phoneme is a set of centroids whose dimensionality is the same as that of a frame vector. The quantized time series model is different from the segmentation centroids model in that each centroid has a time dimension that reflects the temporal locations of its data with which the centroids in a quantized time series could be sorted into a sequence. In the testing phase, each frame vector of the test sequence is matched only to the centroids of similar temporal locations to increase the efficiency of search.

Vector matching is more structured, which results in reduced decoding time and better memory organization. Discriminative learning is carried out by testing each training sample with a test mode, i.e. as if the labels of the training samples are unknown and recording the error ratios of the training samples by comparing the labels of the training samples with the winner template. A correct classification is made if the labels are equal and an incorrect classification is made if otherwise. For each training sample, the winner template has the number of positive samples it has or the number of negative samples it has increased by 1 depending on whether a correct classification or an incorrect classification is made. Each centroid uses its positive and negative samples to adjust its means and weights in the same way DLANC does.

After the test samples are matched to the templates, the frame vectors in the samples and the centroids in the templates are interlocked. This is equivalent to the training phase of DLANC, where each centroid receives a set of positive and negative samples. As a result, the discriminative locally-adaptive training procedure used in the DLANC classifier can be readily applied to discriminatively train a quantized time series model.

In terms of practical applications, phoneme classification system can be used in audio search based on audio to phoneme conversion. Phoneme recognition can be carried out first either by a phoneme recognizer alone [20] or in combination with a word recognizer [21]. After converting audio into phonemes, vocabulary independent audio search can be executed: from a dictionary, a word can be matched into a phoneme string, which can then be searched from converted audio. Phoneme classifier can be used in achieving confidence measure, which is an important component of a speech recognition system. In [28] [29] [33], phoneme recognizer provides the acoustic information in confidence measurement. The extended time series model can perform phoneme classification when given phoneme boundaries by the ASR system. By applying a Viterbi algorithm, a phoneme recognition system can be constructed, in which case a phoneme sequence can be generated by the system with posterior probabilities for confidence measurement.

Bibliography

[1] Karray F.O., de Silva C., "Soft Computing and Intelligent Systems Design", Pearson Education Limited, 2004.

[2] Al-Harbi S.H., Rayward-Smith V.J., "Adapting *K*-means for Supervised Clustering", Applied Intelligence, 24 (3). pp. 219-226, ISSN 0924-669X, 2006.

[3] Rasanen O.J., Laine U.K., Altosaar T., "Self-Learning Vector Quantization for Pattern Discovery from Speech", Brighton, England, 2009.

[4] Eick C.F., Zeidat N., Zhao Z., "Supervised Clustering – Algorithms and Benefits", Department of Computer Science, University of Houston, In proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI04), Boca, 2004.

[5] Srikanth M.R., Murphy H.A., "Discriminative Training of Gaussian Mixture Speaker Models: A New Approach", Department of Computer Science And Engineering, Indian Institute of Technology, 2010 National Conference on Communications (NCC), 29-31 January, 2010.

[6] Domeniconi C., Gunopulos D., Ma S., Yan B., Al-Razgan M., Papadopoulos D., "Locally Adaptive Metrics for Clustering High Dimensional Data", Data Min Knowl Disc (2007) 14:63-97

[7] Lindasalwa M., Mumtaj B., Elamvazuthi I., "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques", Journal of Computing, Volume 2, Issue 3, March 2010.

[8] Antal M., "Speaker Independent Phoneme Classification In Continuous Speech", Studia Univ. Babes-Bolyai Informatica, 2004.

[9] Anies M.T., "Robust Speech Recognition with an Auditory Model", Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, 2004.

[10] Tsang I.W., Kwok J.T., Cheung P., "Core Vector Machines: Fast SVM Training on Very Large Data Sets", Department of Computer Science, The Hong Kong University of Science and Technology, Journal of Machine Learning Research 6 (2005) 363–392, April, 2005.

[11] Sun J., Sun Y., Abida K., Karray F., "A Novel Template Matching Approach To Speaker-Independent Arabic Spoken Digit Recognition", International Conference on Autonomous and Intelligent Systems (AIS), Portugal, 2012.

[12] STAT 557 – Data Mining, The Pennsylvania State University.

[13] Dunn J. C., "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", Journal of Cybernetics 3: 32-57, 1973.

[14] LindeY., Buzo A., Gray R. M., ”An Algorithm for Vector Quantizer Design,” IEEE Trans. on Communication, Vol. COM-28, pp. 84-95, Jan. 1980.

[15] Cifarelli C., Manfredi G., Nieddu L., “Statistical Face Recognition and Intruder Detection via a *K*-means Iterative Algorithm: a Resampling Approach”, Proceedings of the 13th Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2009), Vol. II, pagg. 47-52, Orlando (Florida), July 10-13, 2009.

[16] Tran D.T., Ma W., Sharma D., Bui L., Le T., “A Generic Framework for Soft Subspace Pattern Recognition.”, Theory and Novel Applications of Machine Learning, In-Teh (Croatia), 197-208, 2009.

[17] Parzen E.,"On estimation of a probability density function and mode". *Annals of Mathematical Statistics* **33**: 1065–1076, 1962.

[18] Rosenblatt M., "Remarks on some nonparametric estimates of a density function", *Annals of Mathematical Statistics* **27**: 832–837, 1956.

[19] Ganapathiraju A., Hamaker J., Picone J., "Hybrid SVM/HMM Architectures for Speech Recognition", *ICSLP-2000*, vol.4, 504-507, 2000.

[20] Seide F., Yu P. et al., “Vocabulary-Independent Search in Spontaneous Speech.”, Proc. ICASSP, Montreal, 2004.

[21] Seide F., Yu P., “A Hybrid Word / Phoneme-Based Approach For Improved Vocabulary-Independent Search In Spontaneous Speech”, Proceedings of Interspeech., 2004.

[22] Burges C.J.C., “A Tutorial on Support Vector Machines for Pattern Recognition”, Data Mining and Knowledge Discovery, 2:121-167, 1998.

[23] Vapnik V.N., “Statistical Learning Theory”, John Viley and Sons, 1998.

[24] Lin H.T., Lin C.J., Weng R.C., “A Note on Platt’s Probabilistic Outputs for Support Vector Machines”, Machine Learning, Vol. 68 Issue 3, October 2007.

[25] Chang C., Lin C., “LIBSVM : A Library for Support Vector Machines.”, ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

[26] Reynolds, D.A., "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models", IEEE Transactions on Speech and Audio Processing, Vol. 3, No. 1, 1995.

[27] Lloyd S. P., "Least Squares Quantization in PCM", IEEE Transactions on Information Theory 28 (2): 129–137, 1982.

[28] Cox S., “High-Level Approaches to Confidence Estimation in Speech Recognition”, IEEE Transactions on Speech and Audio Processing, Vol. 10, No. 7, October 2002.

[29] Benitez M.C. et al., “Word Verification Using Confidence Measures in Speech Recognition,” Proc. 5th Int. Conf. Speech Communication and Technology, pp. 1082–1085, Nov. 1998.

[30] Specht D.F., “Probabilistic Neural Networks”, Neural Networks, Vol. 3, No.1, pp. 109–118, 1990.

[31] Montana D., “A Weighted Probabilistic Neural Network”, Advances in Neural Information Processing Systems, Vol. 4, pp. 1110-1117, published by Morgan Kaufmann, 1992.

[32] Low R., Togneri R., "Speech Recognition Using the Probabilistic Neural Network", Proceedings of ICSLP98 (SST Student Day), Paper No. 645, Sydney, Australia, December 1998.

[33] Abida K., “Fuzzy GMM-Based Confidence Measure Towards Keywords Spotting Application”, Master's Thesis, University of Waterloo, 2007.

[34] Gales M.,Young S., "The Application of Hidden Markov Models in Speech Recognition.", Foundations and Trends in Signal Processing 1(3): 195-304, 2007.

[35] Masters T., *Signal and Image Processing with Neural Networks*, John Wiley & Sons, Inc., ISBN 0-471-04963-8, 1994.