CHAPTER 6

VECTOR QUANTIZATION TECHNIQUES FOR
GMM TRAINING

6.1 INTRODUCTION

Gaussian distribution is a good approximation for many naturally occurring data. Gaussian Mixture Model (GMM) is the powerful tool for modeling the unique characteristics of speaker (Bhattacharyya et al 2001). Single Gaussian distribution is not enough to represent the multiple underlying characteristics of speech. Gaussian mixture density is parameterized by mean, covariance and mixture weights from all mixture component densities. Training of GMM is accomplished by EM algorithm. Though the expression of Gaussian distribution is simple to implement, the training algorithm is iterative, complex and which guarantees convergence to local maximum (Gurmeet singh et al 2003). In order to reduce the complexity of the training algorithm, Gurmeet singh et al (2003) proposed that unsupervised clustering algorithms such as k-means and LBG can be used to train the GMM. They achieved 99% of EM accuracy in performance with half the number of computations to that of GMM-EM.

In this chapter, we discuss the application of various Vector Quantization techniques for training the Gaussian Mixture Model. Here, GMM is used for modelling the speaker in speaker verification system. The various VQ algorithms used in this research are LBG (Linde, Buzo, Gray) algorithm, k-means, Fuzzy c-means, and Learning Vector Quantization
The success of each algorithm depends on the principle employed for clustering the data and initialization of cluster centers. State of art speaker verification system employs various modelling techniques. Among those, the widely used one is GMM. The success of GMM is due to the availability of sufficient data for speaker modelling (Angkititrakul & Hansen 2007).

In order to improve the convergence rate and speed of learning process, Kohenen (1990) proposed the algorithm LVQ. As its name indicates, the LVQ is used to classify patterns through learning. Essentially, learning will be used to find representative prototypes called ‘reference vectors’ so that a correct category of input vector can be obtained after giving a pattern into the network. The clustering approach by LVQ algorithm is used for developing a Gaussian Mixture Model. This proposed method is conceptually simple but effective replacement of standard training method that yields the system performance comparable to that of conventional training method.

LVQ developed by Kohenen (1990) is used to globally optimize the code book generated by k-means or any unsupervised clustering algorithms. LVQ is a supervised learning technique which uses the class information to optimize the positions of code vectors, so as to improve the quality of the classifier decision regions. An input vector is picked at random from the input space. When the class label of the input vector and the code vector are same, then the code vector is updated in the direction of input vector. Otherwise the code vector is moved away from the input vector. The code vector which is closest to the input vector is the winning prototype (Karayiannis 1997). Clustering by LVQ is therefore influenced by the actual distribution of feature vectors and hence the modeling of GMM through LVQ algorithm provides better verification rate.

In our proposed method, the parameters of Gaussian Mixture Model are determined from the clusters formed during LVQ learning. The mean of
the GMM is determined by calculating the mean of all vectors which are assigned to that particular cluster. The weights are determined by calculating the proportion of the vectors assigned to the cluster and the covariance matrix is the covariance of the assigned vectors. The new cluster centers are determined by calculating the Euclidean distance between the cluster centers and the input vectors. Then during learning the parameters of GMM are updated according to the new clusters.

### 6.2 VECTOR QUANTIZATION

Vector Quantization is a process of dividing the large vector space into discrete space. The input of the system is a set of n-dimensional vectors and the output is a discrete representation of that input vector space. The system has to find optimal discrete representation of the input space. It can also be viewed as mapping of large vector space to a finite number of vectors which are cluster centers. These cluster centers are the representation of each cluster and it is represented by the coordinates of that vector space. These cluster centers are named as code vectors. The set of code vectors form a codebook.

Let $X$ be an input vector space consists of a set of feature vectors $x \in X$, $X = \{x_1, x_2, \ldots, x_M\}$. These vectors are grouped into clusters based on their positioning in the space and their distance with the cluster center. Let the cluster centers are represented by a set of vectors $Y = \{y_1, y_2, \ldots, y_k\}$, where $k$ is the number of reference vectors. $Y$ is called the codebook and its elements are code vectors. The vectors of $X$ are input patterns. So, VQ can be represented as a function: $q : X \rightarrow Y$. The knowledge of $q$ permits us to obtain a partition $S$ of $X$ constituted by the $N$ subsets $S_i$ as shown in (1).

$$S_i = \{x \in X : q(x) = y_i\} \text{ where } i = 1, 2, \ldots, k$$

(6.1)
Codebook is a look up table of code vectors. The code vectors act as the representatives of subset of feature vectors. Codebook can be built by any clustering algorithms such as K-means (Gersho & Gray 1992) and Fuzzy C-Means (FCM) (Bezdek 1981). They are the keys behind the effective code book design. K-means is an iterative approach which updates the cluster center by minimizing the distortion in each iteration. But it is a crisp clustering technique. However LBG determines the required number of clusters by using splitting technique. The limitation of LBG is that the number of clusters must be a power of 2 (Gurmeet Singh et al. 2003). These two techniques identify the code vectors by learning in an unsupervised way. The codebook generated by unsupervised learning algorithm can be optimized globally through Learning Vector Quantization.

6.2.1 K-Means Clustering

The K-means (Rabiner & Juang 1993) algorithm clusters data based on attributes or features into ‘k’ groups where k is a positive integer. It aims to partition ‘n’ observations into ‘k’ clusters in which each observation belongs to a cluster with the nearest mean, serving as a prototype of the cluster. Nearest mean means the clustering is achieved by minimizing the squared Euclidean distance between vectors x_i and the corresponding cluster centroid prototype vector \( \theta_j \). The centroid vector represents each cluster as a mean vector of the cluster. Let us assume that a set of T vectors \( X=\{x_1,x_2,x_3,\ldots,x_T\} \) is to be divided into K clusters represented by their mean vectors \( \theta=\{\theta_1,\theta_2,\theta_3,\ldots, \theta_K\} \). The objective of the K-means algorithm is to minimize the total distortion given by (Gray 1984)

\[
D = \sum_{i=1}^{T} \min_{j \in \{1,2,\ldots, K\}} \|x_i - \theta_j\|^2
\]

The algorithm is summarized as:
1) Select randomly the cluster centers $\theta_j$, $j=1,2,\ldots,K$. Calculate the initial value of the distortion $D(0)$.

2) Determine the distance between the input vector and all the cluster centers. Select the cluster center for which the distance is less and assign the input vector to that cluster. Do this for all the input vectors.

3) Now, calculate the new centroids $\theta_j$ as the mean of the vectors $x_i \in X$ with centroid($i$)=$j$. Calculate the distortion value $D(i)$.

4) Repeat Step 2 & 3 until either a maximum number of iterations is reached or the distortion value $D(i)$ falls below a preset threshold or until no change in $\theta_j$'s occurs between a few successive iterations.

The above algorithm iteratively moves the cluster boundaries. When the distortion $D$ is minimized, subsequent iterations do not result in any movement of vectors between clusters and the cluster boundaries.

6.2.2 LBG Algorithm

LBG algorithm (Linde et al 1980) is like a K-means clustering algorithm which takes a set of input vectors $S = \{x_i \in R^d \mid i=1,2,\ldots,n\}$ as input and generates a representative subset of vectors $C = \{c_j \in R^d \mid j=1,2,\ldots,K\}$ with a user specified $K << n$ as output according to the similarity measure and $d$ is the dimension of the vector. For the application of Vector Quantization (VQ), $d=16$, $K=256$ or $512$ are commonly used.

The LBG algorithm requires the user to provide an initial estimate of the codebook and to specify the desired number of clusters. Due to the
nature of the conventional LBG algorithm, which usually generates the initial codebook by randomly splitting code vector into two new code vectors by a small random perturbation, the desired number of clusters needs to be a power of 2. The convergence of LBG algorithm depends on the initial codebook $C$, the distortion $D_i$, and the threshold $\varepsilon$, in implementation, we need to provide a maximum number of iterations to guarantee the convergence.

### 6.2.3 Fuzzy C-means algorithm

Fuzzy c-means (FCM) (Dunn 1973; Bezdek 1981) is a data clustering technique in which a dataset is grouped into n clusters with every data point in the dataset belonging to every cluster to a certain degree. Every data point will be assigned with membership values representing the degree of membership of each cluster. The degree of membership will be high if the data point is closer to the cluster; otherwise the degree of membership will be low if the data point is far away from the particular cluster.

This algorithm starts with an initial guess for the cluster centers, which are intended to mark the mean location of each cluster. The initial guess for these cluster centers is most likely incorrect. Next, it assigns every data point a membership grade for each cluster. By iteratively updating the cluster centers and the membership grades for each data point, fuzzy c-means algorithm iteratively moves the cluster centers to the right location within a data set. This iteration is based on minimizing an objective function that represents the distance from any given data point to a cluster center weighted by that data point's membership grade.

It is based on minimization of the following objective function (Karayiannis 1997)
\[
J_m = \sum_{j=1}^{N} \sum_{i=1}^{C} u_{ij}^m \|x_j - c_i\|^2, \quad 1 \leq m \leq \infty
\]  

subject to the constraints \(\sum_{i=1}^{C} u_{ij} = 1 \quad \forall j; \quad 0 \leq \sum_{j=1}^{N} u_{ij} \leq N \quad \forall i \) and \(0 \leq u_{ij} \leq 1 \quad \forall i, j\)

where \(m\) is any real number greater than 1, \(u_{ij}\) is the degree of membership of \(x_j\) in the cluster \(i\), \(x_j\) is the \(j\)th component of \(d\)-dimensional measured data, \(c_i\) is the \(d\)-dimension center of the cluster, and \(\|\cdot\|\) is any norm expressing the similarity between any measured data and the center. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership \(u_{ij}\) and the cluster centers \(c_i\) by

\[
u_{ij} = \frac{1}{\sum_{i=1}^{C} \left( \frac{\|x_j - c_i\|}{\|x_j - c_i\|^m} \right)^2}, \quad c_i = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}^m}
\]

This iteration will stop when

\[
\max_{ij} \left| u_{ij}^{k+1} - u_{ij}^k \right| < \varepsilon
\]

where \(\varepsilon\) is a termination criterion between 0 and 1, whereas \(k\) are the iteration steps. This procedure converges to a local minimum or a saddle point of \(J_m\).

The algorithm is composed of the following steps:

### 6.2.4 LVQ Algorithm

The Learning Vector Quantization (LVQ) is an algorithm for learning classifiers from labeled data samples. Instead of modeling the class densities, it models the discrimination function defined by the set of labeled codebook vectors and the nearest neighborhood search between the codebook and data. In classification, a data point \(x_i\) is assigned to a class according to
the class label of the closest codebook vector. The training algorithm involves an iterative gradient update of the winner unit. The winner unit $m^c$ is defined by (Karayiannis 1997)

$$m^c = \min_k \| x_i - m^k \| \quad \text{where} \; k = 1,2,\ldots,c,\ldots,C \quad (6.12)$$

The direction of the gradient update depends on the correctness of the classification using a nearest neighborhood rule in Euclidean space. If a data sample is correctly classified (the labels of the winner unit and the data sample are the same), the model vector closest to the data sample is attracted towards the sample; if incorrectly classified, data sample has a repulsive effect on the model vector. The update equation for the winner unit $m^c$ defined by the nearest-neighbor rule and a data sample $x(t)$ are

$$m^c(t+1) := m^c(t) \pm \alpha(t) [ x(t) - m^c(t) ] \quad (6.13)$$

where the sign depends on whether the data sample is correctly classified (+) or misclassified (−). The learning rate $\alpha(t) \in ]0,1[$ must decrease monotonically in time. For different picks of data samples from our training set, this procedure is repeated iteratively until convergence. Kohonen also presents optimized learning-rate LVQ, where the learning-rate is optimized for each codebook individually.

### 6.3 LVQ BASED TRAINING ALGORITHM FOR GMM

Learning Vector Quantization is a supervised learning algorithm which is used in competitive learning networks (Yair et al 1992). This algorithm clusters the vectors through learning. In this work, we have proposed training of GMM by LVQ algorithm. The proposed method is compared with the popular EM algorithm training process. The centroids of the clusters formed by LVQ algorithm are used to obtain the mean vectors of
GMM. Let \( x = \{x_1, x_2, x_3, \ldots, x_n\} \) be the input vector and \( w_j = \{w_{1j}, w_{2j}, w_{3j}, \ldots, w_{nj}\} \) be the codebook vector of the \( j^{th} \) cluster. Let ‘\( C \)’ be the total number of clusters. The steps involved in training of GMM through LVQ algorithm is given below.

1. The codebook vectors \( w_j, j=1\) to \( C \) (by means of k-Means) are initialized from the training vectors, where \( C \) is the number of different categories and the learning rate is set as \( \alpha \).

2. For each training input vector \( x \), the codebook vector (winning code vector) which is closest to the input vector is obtained by calculating the Euclidean distance between the input vector and all the codebook vectors.

3. The winning code vector \( w_j \) is updated (moved towards the input vector) as

\[
\begin{align*}
\text{when the input vector } x \text{ belong to the same class } j \text{ of the winning code vector.}
\end{align*}
\]

4. Otherwise the winning code vector is moved away from the input vector by

\[
\begin{align*}
\text{If the winning cluster } C_j \text{ is the same as the defined target } T, \\
\text{the winning code vector is updated with the positive learning rate; otherwise it is updated with negative learning rate.}
\end{align*}
\]
5. The mean of GMM $\mu_i$ (where ‘$i$’ denotes the number of mixture in a GMM; $i$ or $j=1$ to $C$) is obtained by calculating the mean of all the vectors assigned to that particular cluster and the covariance matrix $\Sigma_i$ is the covariance of the assigned vectors.

6. The weights $P_i$ are determined by calculating the proportion of the vectors assigned to the cluster to the total number of input vectors.

7. The stopping condition is tested. This may be a fixed number of iterations or the learning rate reaching a sufficiently small value. The steps 2, 3, 4 are repeated until the stopping condition is met.
Figure 6.1 Flow chart for LVQ based GMM training

Thus the parameters of a GMM (mean, covariance and mixture weights) represent the model of a speaker $\hat{\lambda}_i = \hat{\mu}_i, \hat{\Sigma}_i, p_i$. 
6.4 EXPERIMENTAL SETUP

Training of GMM using LVQ algorithm has been evaluated with the TIMIT database. Randomly 50 speakers are selected from the database as target speakers. 100 speakers are used as background speakers including 50 speakers as imposters. Eight sentences (three "si" and five "sx" sentences) of each speaker are used for training, and the rest two "sa" sentences are used for testing. All the training sentences are concatenated and segmented into frames of 30 ms duration with 10 ms overlapping. After preprocessing, MFCC coefficients and its first and second derivatives are extracted from each frame to compose a feature vector. The feature vector includes 13 MFCC coefficients, 13 first derivative and 13 second derivative, totally 39 coefficients. Cepstral Mean Subtraction (CMS) is applied to remove the linear channel effect. Silence and low energy speech frames are removed using an energy-based frame selection technique. The threshold used for selection of the speech frames is 0·1 times the average frame energy. The extracted features are used for modeling the speakers by GMM trained with different clustering algorithms.

Gaussian mixture models are developed for every target speaker through LVQ algorithm. Experiments are conducted for LVQ-GMM with $M=16$, 32 & 64 mixture components. Models are also trained using LBG, k-Means, fuzzy c-means, and EM algorithm in order to compare the performance of the system. The Universal Background model with $M$ Gaussian components is obtained using hours of speech data from all non target speakers. For each target speaker, a GMM is created. The speaker verification task is a hypothesis testing problem. The decision between whether the test utterance is generated by the target speaker or by someone else is based on the average frame log-likelihood ratio between the target GMM and the UBM. The Log Likelihood Ratio is calculated as
where $\lambda_t$ is parameter set of target GMM and $\lambda_{UBM}$ is the parameter set of UBM. The decision criterion is to verify the claimed speaker identity, according to the log likelihood ratio obtained from the model evaluated with the test utterance. By thresholding the average frame log-likelihood ratio, a decision can be made to either accept or reject the target speaker. In the testing phase, the decision is made based on the log likelihood of a model $\lambda$ for a sequence of test feature vectors, $x = \{x_1, x_2, \ldots, x_t\}$.

### 6.5 SUMMARY

A training algorithm based on vector quantization technique for Gaussian Mixture model was proposed in this chapter. The computation time for training the GMM and the performance of the system for various VQ algorithms have been analysed. Based on result, it is shown that the LVQ based GMM performs better than other VQ algorithms like K-means, LBG and FCM algorithms.