Chapter 2
Multivariate Statistical Analysis and One-Pass Vector Quantization

Current speaker authentication algorithms are largely based on multivariate statistical theory. In this chapter, we introduce the most important technical components and concepts of multivariate analysis as they apply to speaker authentication: the multivariate Gaussian (also called normal) distribution, principal component analysis (PCA), vector quantization (VQ), and segmental K-means. These fundamental techniques have been used for statistical pattern recognition and will be used in our further discussions throughout this book. Understanding the basic concepts of these techniques is essential for understanding and developing speaker authentication algorithms.

Those readers who are already familiar with multivariate analysis can skip most of the sections in this chapter; however, the one-pass VQ algorithm presented in Section 2.4 is from the author and Swaszek's research [8] and may be unknown to the reader. The algorithm is useful in processing very large datasets. For example, when training a background model in speaker recognition, the dataset can be huge; the one-pass VQ algorithm can speed up the initialization of the training procedure. It can be used to initialize the centroids during Gaussian mixture model (GMM) or hidden Markov model (HMM) training for large datasets. Also, the concept will be applied to Chapter 3 in sequential design of a classifier and to Chapter 14 in sequential design of speaker authentication systems.

2.1 Multivariate Gaussian Distribution

Multivariate Gaussian distribution plays a fundamental role in multivariate analysis and many real-world problems fall naturally within the framework of Gaussian theory. It is also important and popular in speaker recognition. The importance of the Gaussian distribution in speaker authentication rests on its extension to mixture Gaussian distribution or Gaussian mixture model (GMM). A mixture Gaussian distribution with enough Gaussian components

Q. Li, *Speaker Authentication,* Signals and Communication Technology,
DOI: 10.1007/978-3-642-23731-7_2, © Springer-Verlag Berlin Heidelberg 2012

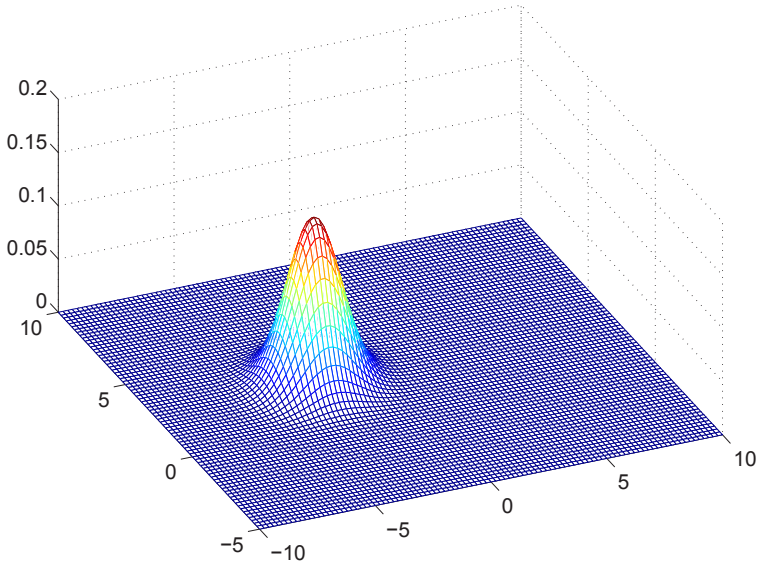

Fig. 2.1. An example of bivariate Gaussian distribution: $\rho_{11} = 1.23$, $\rho_{12} = \rho_{21} = 0.45$, and $\rho_{22} = 0.89$.

can approximate the "true" population distribution of speech data. In addition, the EM (Expectation-Maximization) algorithm provides a convenient training algorithm for the GMM. The algorithm is fast and guarantees for convergence at every iteration. Third, based on the GMM, we can build a hidden Morkov model (HMM) for speaker verification and verbal information verification.

In this section we discuss the single Gaussian distribution, which is the basic component in a mixture Gaussian distribution. The mixture Gaussian distribution will be discussed with models for speaker recognition in the following sections.

A p-dimensional Gaussian density for the random vector $\mathbf{X}' = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_p]$ can be presents as:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)\right] \tag{2.1}$$

where $-\infty < \mathrm{x}_i < \infty$, $i = 1, 2, \ldots, p$. The p-dimensional normal density can be denoted as $N_p(\mu, \Sigma)$ [6].

For example, when $p = 2$, the bivariate Gaussian density has the following individual parameters: $\mu_1 = E(X_1)$, $\mu_2 = E(X_2)$, $\sigma_{11} = \mathrm{Var}(X_1)$, $\sigma_{22} = \mathrm{Var}(X_2)$, the covariance matrix is:

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} \tag{2.2}$$

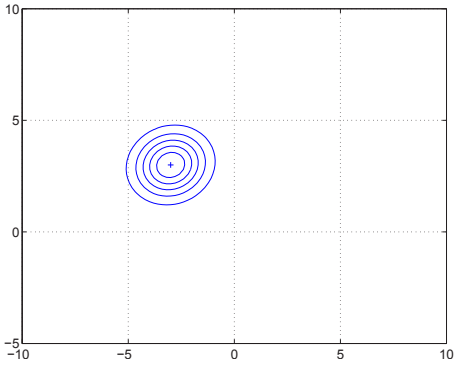

Fig. 2.2. The contour of the Gaussian distribution in Fig. 2.1.

and the inverse of the covariance is

$$\Sigma^{-1} = \frac{1}{\sigma_{11}\sigma_{22} - \sigma_{12}^2} \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix}. \tag{2.3}$$

Introducing the correlation coefficient $\rho_{12} = \sigma_{12}\sqrt{\sigma_{11}}\sqrt{\sigma_{22}}$, we have the expression for the bivariate ($p = 2$) Gaussian density as:

$$\begin{aligned} f(x_1, x_2) = \frac{1}{2\pi\sqrt{\sigma_{11}\sigma_{22}(1-\rho_{12}^2)}} \exp\Bigg\{ -\frac{1}{2(1-\rho_{12}^2)} \\ \Bigg[(\frac{x_1-\mu_1}{\sqrt{\sigma}_{11}})^2 + (\frac{x_2-\mu_2}{\sqrt{\sigma}_{22}})^2 \\ -2\rho_{12}(\frac{x_1-\mu_1}{\sqrt{\sigma}_{11}})(\frac{x_2-\mu_2}{\sqrt{\sigma}_{22}}) \Bigg] \Bigg\} \end{aligned} \tag{2.4}$$

For illustration, we plot two bivariate Gaussian distributions and its contour as in Figs. 2.1 and 2.2.

2.2 Principal Component Analysis

Principal component analysis (PCA) is concerned with explaining the variance-covariance structure through the most important linear combinations of the original variable [6]. In speaker authentication, the principal component analysis is a useful tool for data interpretation, reduction, analysis, and visualization.

When a feature or data set represents in a N dimensional data space, the actual data variability may be largely in a small number of k dimension, where $k < p$; thus the k dimension can represent the data in p dimension and the original data set can be reduced from a $n \times p$ matrix to a $n \times k$ matrix, where k is the number of principal components.

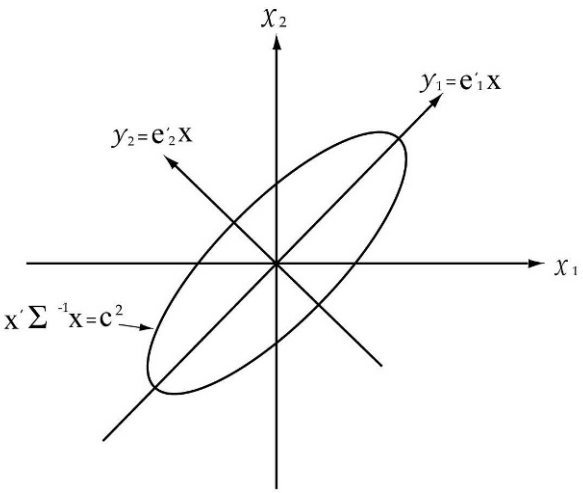

Fig. 2.3. An illustration of a constant density ellipse and the principal components for a normal random vector $\mathbf{X}$. The largest eigenvalue associates with the long axis of the ellipse and the second eigenvalue associates with the short axis. The eigenvectors associate with the axes.

When a random vector, $\mathbf{X}' = [X_1, X_2, \ldots, X_p]$, has a covariance matrix Σ, which has the eigenvalue λ_i and eigenvector $\mathbf{e}_i$ pairs:

$$(\lambda_1, \mathbf{e}_1), (\lambda_2, \mathbf{e}_2), \ldots, (\lambda_p, \mathbf{e}_p),$$

and

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0.$$

The ith principal component is represented as:

$$Y_i = \mathbf{e}_i'\mathbf{X} = e_{1i}X_1 + e_{2i}X_2 + \ldots + e_{pi}X_p, \quad i = 1, 2, \ldots, p \tag{2.5}$$

and then the variance and covariance of Y_i are:

$$\mathrm{Var}(Y_i) = \mathbf{e}_i'\Sigma\mathbf{e}_i \quad i = 1, 2, \ldots, p \tag{2.6}$$

A proof of this property can be found in [6]. An illustration of the concept of the principal components is given in Fig. 2.3. The principal components coincide with the axes of the constant density ellipse. The largest eigenvalue is associated with the long axis and the second eigenvalue is associated with the short axis. The eigenvectors are associated with directions of the axes.

In speaker authentication, the PCA can be used to reduce the feature space for efficient computation without much affecting the recognition or classification accuracy. Also, it can be used to project multidimensional data samples

onto a selected two-dimensional space, so the data is visible for analysis. Readers can refer to [6] to gain more knowledge on multivariate statistical analysis.

2.3 Vector Quantization

Vector quantization (VQ) is a fundamental tool for pattern recognition, classification, and data compression. It has been applied widely to speech and image processing. VQ has become the first step in training the GMM and HMM. Both are the most popular models used in speaker authentication.

The task of VQ is to partition m-dimensional space into multiple cells represented by quantized vectors. The vectors are all called centroids, codebook vectors, and codewords. The VQ training criterion is to minimize overall average distortions over all cells when using centroids to represent the data in cells.

The most popular algorithm for VQ training is the K-means algorithm. In a quick overview, the K-means algorithm is an iteration process with the following steps: First, initialize the centriods by an adequate method, such as the one which will be discussed in the next section. Second, partition each training data vector into one of the cells by looking for the nearest centroids. Third, update the centroids by the data grouped into the corresponding cell. Last, repeat the second and third steps until the values of centroids converges to required ranges.

We note that the K-means algorithm can only converge to a local optimum [12]. Different initial centroids may lead to different local optimum. This will also affect the HMM training results when using VQ for HMM training initialization. When using HMM for speaker recognition, it can be observed each time the recognizer is retrained. It may have slightly different recognition accuracies.

The LBG (Linde, Buzo, and Gray) algorithm [12] is another popular algorithm for VQ. Instead of selecting all initial centroids at one time, the LBG algorithm determines the initial centroids through a splitting process.

The vector quantization (VQ) method can be used directly for speaker identification. In [17], Soong, Rosenberg and Juang used a speaker-dependent codebook to represent characteristics of a speaker's voice. The codebook is generated by a clustering procedure based upon a predefined objective distortion measure, which computes the dissimilarity between any two given vectors [17]. The codebook can also be considered an implicit representation of a mixture distribution used to describe the statistical properties of the source, i.e., the particular talker. In the test session, input vectors from the unknown talker are compared with the nearest codebook entry and the corresponding distortions are accumulated to form the basis for a classification decision.

2.4 One-Pass VQ

The initial centroids or codebook is critical to the final VQ results. To achieve a high performance, the high computational complexity in both the VQ encoding and codebook design are usually required. The methods most often employed for designing the codebook, such as the K-means and the LBG algorithms, are iterative algorithms and require a large amount of CPU time to achieve a locally optimum solution.

In speaker recognition, it is often required to pull all available data together to train a background model or cohort model. The first step in the training is to initialize the centroids for VQ. The traditional initialization algorithm just randomly selects the centroids, which is not efficient and wastes a lot of computation time when a dataset is huge. To speed up the model training procedure, an algorithm which can provide better initial centroids and needs less iteration is required. In [8], we proposed a *one-pass VQ* algorithm for the purpose. We introduce it in this section in detail.

2.4.1 The One-Pass VQ Algorithm

The one-pass VQ algorithm is based on a sequential statistical analysis of the local characteristics of the training data and a sequential pruning technique. The work was originaly proposed by the author and Swaszek in [8]. It was inspired by a constructive neural network design concept for classification and pattern recognition as described in Chapter 3 [10, 11, 9, 22].

The one-pass VQ algorithm sequentially selects a subset of the training vectors in a high density area of the training data space and computes a VQ codebook vector. Next, a sphere is constructed about the code vector whose radius is determined such that the encoding error for points within the sphere is acceptable. In the next stage, the data within the sphere is pruned (deleted) from the training data set. This procedure continues on the remainder of the training set until the desired number of centroids is located. To achieve a local optimum, one or a few iterations like in the K-means algorithm can be further applied.

The basic steps of the one-pass VQ algorithm are illustrated in Figure 2.4 which shows how a code vector is determined for a two-dimensional training data set (the algorithm is developed and implemented in N dimensions). We refer to this figure in the following discussion.

The one-pass VQ algorithm is compared with several benchmark results for uncorrelated Gaussian, correlated Gaussian, and Laplace sources; its performance is seen to be as good or better than the benchmark results with much less computation. Furthermore, the one-pass initialization needs only slightly more computation than a single iteration of the K-means algorithm when the data set is large with high dimension. The algorithm also has a robust property and can be invariant to outliers. The algorithm can be applied

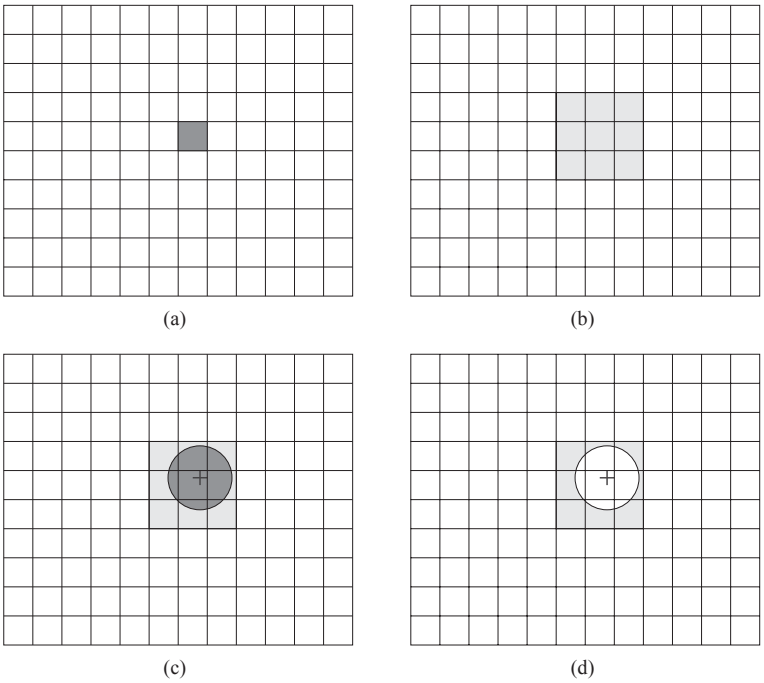

Fig. 2.4. The method to determine a code vector: (a) select the highest density cell; (b) examine a group of cells around the selected one; (c) estimate the center of the data subset; (d) cut a "hole" in the training data set.

directly in classification, pattern recognition and data compression, or indirectly as the first step in training Gaussian mixture models or hidden Markov models.

Design Specifications

The training data set X is assumed to consist of M N-dimensional training vectors. The total number of regions R is determined by the desired bit-rate. Let D represent the *maximum allowable distortion*

$$||x_m - y_c||^2 \leq D \tag{2.7}$$

where $x_m \in X$ is a data vector and y_c is the nearest code vector to x_m. The value for D can be determined either from the application (such as from human vision experiments, etc.) or estimated from the required data compression ratio.

Source Histogram

The one-pass design first partitions the input space by a cubic lattice to compute a histogram. Each histogram cell is an N-dimensional hypercube.

(In general, we could allow different side lengths for the cells.) The number of cells in each dimension is calculated based on the range of the data in that dimension and the allowable maximal distortion D. The employed method for calculating the number of cells in the jth-dimension, L_j, is

$$L_j = \lceil \frac{x_{j,\max} - x_{j,\min}}{2D/3} \rceil \tag{2.8}$$

where $x_{j,\max}$ and $x_{j,\min}$ are, respectively, the maximal and minimal values of X in dimension j. The term $2D/3$ is the recommended size of the cell in that dimension. The probability mass function $f(k_1, k_2, \ldots, k_N)$ with $k_i \in \{1, 2 \ldots L_i\}$ is determined by counting the number of training data vectors within each cell.

The sequential design algorithm starts from that cell which currently has the largest number of training vectors (the maximum of $f(\cdots)$ over the k_i). In Figure 2.4(a) we assume that the highlighted region is that cell. Then, as shown in Figure 2.4(b), a group of contiguous cells around the selected one is chosen. This region $X_s \subset X$ (highlighted) will be used to locate a VQ code vector.

Locating a Code Vector

Two algorithms are employed in locating a code vector: a principal component method and a direct median method. For the results presented here, the medians of X_s in each of the N dimensions are computed as the components of the code vector. Medians are employed to provide some robustness with respect to variation in the data subset X_s. The median is marked in Figure 2.4(c) by the "+" symbol.

Principal Component (PC) Method: As shown in Fig. 2.5, this method first solves an eigenvector problem on the data set X_s,

$$\mathbf{R}_{X_s}\mathbf{E} = \lambda\mathbf{E}, \tag{2.9}$$

where $\mathbf{R}_{X_s}$ is the covariance matrix of X_s, λ is a diagonal matrix of eigenvalues, and $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N]$ are the eigenvectors.

Then, the X_s is projected onto the directions of each of the eigenvectors.

$$Y_j = X_s\mathbf{e}_j. \tag{2.10}$$

Medians are further determined from each of the Y_j. The centroid is obtained by taking the inverse transforms of the median of Y_j in each dimension.

Direct-Median Method: This method is straight forward. It just uses the medians of X_s in each of the original dimensions as the estimated centroid.

It is clear that the second method can save the eigenvector calculation, so it is faster than the first one. For evaluation, both methods have been tested on the codebook design of Gaussian and Laplace sources. The mean square errors from both methods are very close, while the PC method is slightly better for the Laplace source and the direct median method is better for the Gaussian source.

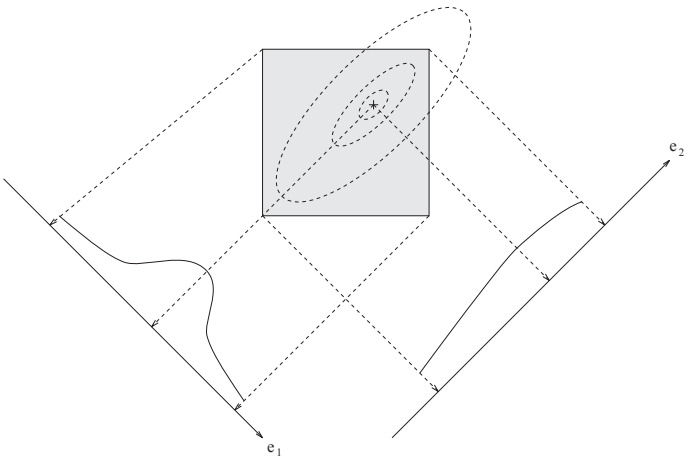

Fig. 2.5. The Principal Component (PC) method to determine a centroid.

Pruning the Training Data

Once the code vector is located, the next step is to determine a sphere around it as shown in Figure 2.4(c). (We note that for classification applications it could be an ellipse.) One way to determine the size of the sphere is to estimate the maximal number of data vectors which will be included inside the sphere. The set of vectors within the sphere X_c is a subset of X_s, $X_c \subset X_s$. To determine a sphere for the cth code vector, the total number of data vectors T^c within the sphere is estimated by

$$T^c = W^c \frac{M^c}{R+1-c} \tag{2.11}$$

where M^c is the total number of data vectors in the current training data set ($M^1 = M$ and $M^c < M$ when $c > 1$) and $R + 1 - c$ is the number of code vectors which have yet to be located. The term W^c is a variable weight

$$W^c = W^{c-1} - \triangle W \tag{2.12}$$

where $\triangle W$ is the change of the weight variable between each of the algorithm's R cycles. The weight is employed to keep the individual spheres from growing too large. From the experience of the design examples in this chapter, we note that the resulting performance is not very sensitive to either W^1 or $\triangle W$.

After the number of vectors of the sphere X_c is determined, the data subset X_c is pruned from the current training data set. As shown in Figure 2.4(d) a "hole" is cut and the data vectors within the hole are pruned. The entries of the mass function $f(\cdots)$ associated with the highlighted cells are then updated. The design for the next code vector starts by selecting that cell which currently contains the largest number of training vectors.

Due to the nature of the design procedure it can be imagined that the diameters of the spheres might get larger and larger while the design is in progress. To alleviate this, each sphere's diameter is limited to $2D$. This value is chosen to avoid the situation in which one sphere entirely becomes a subset of another sphere (some overlap is common).

Updating the Designed Code Vectors Once

After locating all R code vectors and cutting the R "holes" in X there often remains a residual data set $X_l \subset X$ from the training set. The last step of our one-pass design is to update the designed code vectors by calculating the centroids of the nearest vectors around each code vector. This is equal to one iteration of the K-means algorithm.

2.4.2 Steps of the One-Pass VQ Algorithm

The one-pass VQ algorithm is summarized below:

Step 1. Initialization
- 1.1 Give the training data set X (an $M \times N$ data matrix), the number of designed centroid C (or the number of regions since $R = C$), and the Allowed Maximal Distortion D which can be either given or estimated from the X and C.
- 1.2 Set weight value W^1 and $\triangle W$.

Step 2. Computing the Rectangular Histogram
- 2.1 Determine the number of cells in each dimension n, $n = 1, 2, ..., N$ as in the formula (2.8).
- 2.2 Compute the probability mass function f by counting the training vectors within each of the cells of the histogram.

Step 3. Sequential codebook design
- 3.1 From $f(\cdot)$, select the histogram cell which currently contains the most training vectors.
- 3.2 Group histogram cells adjacent to the selected one to obtain the data set X_s.
- 3.3 Determine a centroid for X_s by the PC or direct median method.
- 3.4 Calculate the maximal number of vectors T^c for the cth "hole"as in (2.11).
- 3.5 Determine and prune (delete) total T^c vectors from X_s. Update the entries of $f(\cdot)$ associated with the cells of X_s only.
- 3.6 Update the parameters W^{c+1} as in (2.12) and $R^{c+1} = R^c - 1$.
- 3.7 $c = c + 1$ and goto Step 3.1 if $R^{c+1} > 0$.

Step 4. Improve the designed centroids

4.1 Update the VQ's centroids once by re-calculating the means of those vectors from X nearest to each centroid (one iteration of the K-means algorithm).

Step 5. Stop

2.4.3 Complexity Analysis

This section is concerned with complexity analysis on the *sequential selection* of codebook vectors (principal features). The one-pass VQ includes one sequential selection and one LBG iteration.

Complexity is measured by counting the number of real number operations required by the sequential selection. We define the following notations for the analysis.

- R — the number of VQ regions
- M — the number of training vectors
- N — the data dimensions
- L — the number of histogram bins per dimension
- k — the number of data vectors in a highlight window is k times larger than the average number, M/R.
- B_{max} — The max number of histogram cells (boxes) with nonzero count of training data vectors.

The VQ design requires some initialization:

1) Computing the initial histogram requires $NM \log_2(L)$ ops

This is followed by R repetitions of finding a centroid of the local window about the histogram maximum and pruning the data set:

2) Sorting the histogram counts requires (to find a high density cell)

$$B_{max} \log_2(B_{max}) \text{ ops}$$

3) Finding a centroid by calculating median

$$kN\frac{M}{R} \log_2(k\frac{M}{R}) \text{ ops}$$

4) Computing the distance of each training vector in the window to the centroid (N multiplications plus 2N-1 additions/subtractions for each vector)

$$k\frac{M}{R}(3N-1) \text{ ops}$$

5) Sorting the distance to determine the cut-off point

$$k\frac{M}{R} \log_2(k\frac{M}{R}) \text{ ops}$$

6) Updating the histogram

$$k\frac{M}{R} \text{ ops}$$

Summing all this yields a total of

$$NM\log_2(L) + R[B_{max}\log_2(B_{max}) + kN\frac{M}{R}\log_2(k\frac{M}{R}) + k\frac{M}{R}(3N-1) + k\frac{M}{R}\log_2(k\frac{M}{R}) + k\frac{M}{R}] \tag{2.13}$$

operations. It is equal to

$$NM\log_2(L) + R[B_{max}\log_2(B_{max}) + 3Nk\frac{M}{R} + (N+1)k\frac{M}{R}\log_2(k\frac{M}{R})] \tag{2.14}$$

For comparison, the LBG algorithm needs $3NMR$ operations.

Suppose a data source is a 512×512 image and the codebook size is 64 ($M = 16384$, $R = 64$, and $N = 16$). If we use the parameters in a worst case for the one-pass algorithm, $K = 4$, $L = 8$, and $B_{max} = M/2$, the sequential selection will need 21.89 Mops and one LBG iteration will need 50.33 Mops.

2.4.4 Codebook Design Examples

In this section the one-pass VQ algorithm is tested by designing codebooks for uncorrelated Gaussian, correlated Gaussian, and independent Laplace sources since many benchmark results on these sources have been reported [2, 3, 14, 15, 21, 19, 20, 18, 24].

To facilitate the comparison of CPU time and Flops (floating point operations) from different systems we use the well-known LBG algorithm as a common measurement and define the following two kinds of speed-up

$$\text{Speed-up-in-time} = \frac{\text{CPU time for LBG}}{\text{CPU time for algorithm}} \tag{2.15}$$

$$\text{Speed-up-in-flops} = \frac{\text{Flops for LBG}}{\text{Flops for algorithm}}. \tag{2.16}$$

Since we use a high-level, interpreted language for simulation on a multi-user system, we prefer the Speed-up-in-flops as a comparison measurement.

Two dimensional Uncorrelated Gaussian source

In this example, the data vectors are from a zero-mean, unit-variance, independent Gaussian distribution. The joint density is given by

$$f(x_1, x_2) = \frac{1}{2\pi}\exp[-(x_1^2 + x_2^2)/2)], \quad -\infty < x_1,\ x_2 < \infty. \tag{2.17}$$

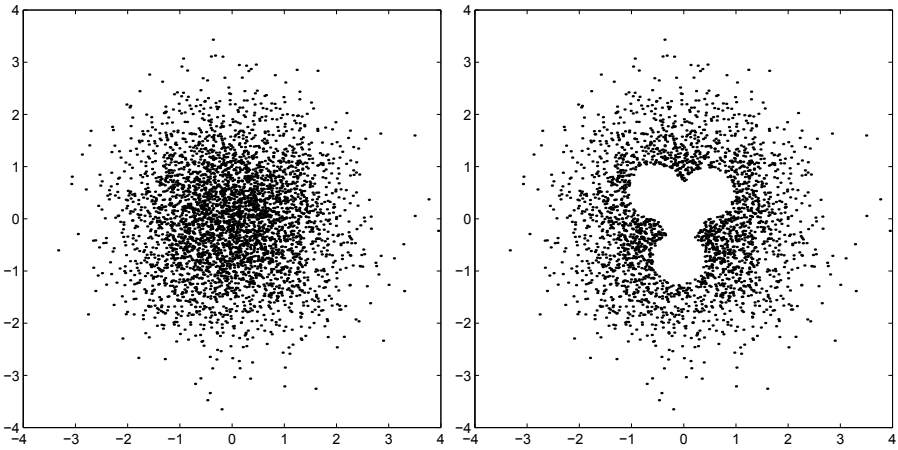

Fig. 2.6. Left: Uncorrelated Gaussian source training data. Right: The residual data after four code vectors have been located.

The training set X consisted of 4,000 length two vectors as shown in Figure 2.6. The goal is to design a size $R = 16$ codebook to make the MSE (mean-squared error) as small as possible.

To set the design parameters we estimate D by $D = 5.2/(2 \times \log_2(16)) = 0.65$ since most of the data vectors are within a circle region of diameter 5.2. We used the weight $W^1 = 1.4$ and $\triangle W = 0.005$.

The simulation showed that the first "hole" was selected and cut in the center of the Gaussian distribution; the 2nd to the 4th holes were selected around the first one as shown in Figure 2.6. The algorithm continued until all 16 code vectors had been located and 16 holes had been cut. Figure 2.7 shows these code vectors and the residual data. Then the code vectors were updated by the nearest vectors of X. The "+" signs in the Figure 2.7 are the final centroids generated by the one-pass algorithm.

Table 2.1. Quantizer MSE Performance

1	One-pass VQ (introduced method)	0.218
3	One-pass VQ + 2 LBG (introduced)	0.211
2	Linde-Buzo-Gray (LBG, 20 iterations)	0.211
4	Strictly polar quantizer (from [24])	0.240
5	Unrestricted polar quantizer (from [24])	0.218
6	Scalar (Max) quantizer (from [14])	0.236
7	Dirichlet polar VQ (from [21])	0.239
8	Dirichlet rotated polar VQ (from [21])	0.222

In order to further compare the one-pass algorithm with the LBG algorithm we used the centroids designed by the one-pass algorithm as the initial

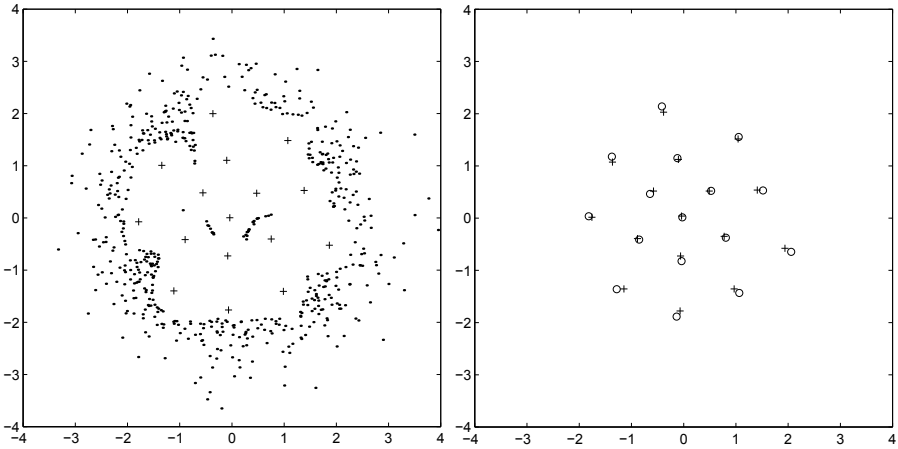

Fig. 2.7. Left: The residual data after all 16 code vectors have been located. Right: The "+" and "○" are the centroids after one and three iterations of the LBG algorithm, respectively.

Table 2.2. Comparison of One-Pass and LBG Algorithms

Type	Iterations	MFLOPS	CPU Time (seconds)	MSE
One-Pass	1	1.5	35	0.218
One-Pass +2LBG	1 + 2	3.0	67	0.211
LBG	9	7.2	166	0.218
LBG	20	15.1	333	0.211

centroids for the LBG algorithm, then ran two iterations of LBG (called "one-pass+2LBG"). The centroids designed by the one-pass+ 2LBG are shown in Figure 2.7(Right) denoted by the "○"s. They are very close to the one-pass centroids ("+"s). This suggests another application of the one-pass algorithm: it can be used to determine the initial centroids for the LBG algorithm for a fast and high-performance design.

The MSE of the one-pass design is compared in Table 2.1 with the MSE from other methods on an uncorrelated Gaussian source with 16 centroids. The MSE of the one-pass algorithm is equal to or better than the benchmark results. Table 2.2 shows the CPU time and Mflops used by the one-pass and LBG algorithms. For the same MSE, 0.218, the one-pass has a speed-up-in-flops of $7.2/1.5 = 4.8$.

Multidimensional Uncorrelated Gaussian Source

As shown in Table 2.3, the one-pass VQ algorithm was compared with five other algorithms on an $N = 4$ *i.i.d.* Gaussian source of 1,600 training vectors

and codebook size of $R = 16$. The speed-up-in-flops in Table 2.3, items 1 and 2, are from our simulations. The speed-up-in-time in items 3 to 7 were calculated based on the training time provided by Huang and Harris in [4]. Again, the one-pass algorithm shows a higher speed-up.

Table 2.3. Comparison of Different VQ Design Approaches

	VQ Design Approaches	MSE per dimension	Speed-up
1	One-pass VQ (introduced)	0.35054	2.12
2	LBG with 5 iterations	0.34919	1.00
3	LBG (from [4])	0.41791	1.00
4	Directed-search binary-splitting [4]	0.42202	1.50
5	Pairwise nearest neighbor[4]	0.42975	2.00
6	Simulated annealing [4]	0.41166	0.0023
7	Fuzzy c-mean clustering [4]	0.51628	0.22

The speed-ups in item 1 and 2 are in-flops. All others are in-time.

Correlated Gaussian Source

The training set is 4,000 dimension two Gaussian distributed vectors with zero means, unit variances, and correlation coefficient 0.8. The codebook size is $R = 16$. The results are compared with a benchmark result in Table 2.4. The one-pass VQ algorithm yields lower MSE.

Table 2.4. Comparison for the Correlated Gaussian Source

Types	Iterations	Mflops	CPU (seconds)	MSE
One-Pass	1	1.51	58	0.1351
One-Pass + LBG	1 + 2	2.95	110	0.1279
Block VQ (from [20])				0.1478

Laplace Source

In this example, the 4,000 dimension two training vectors have an independent Laplace distribution

$$f(x_1, x_2) = \frac{1}{2} e^{-\sqrt{2}|x_1|} e^{-\sqrt{2}|x_2|} \tag{2.18}$$

The training data is shown in Figure 2.8(Left); the one-pass VQ's centroids ("+"s) as well as the residual data are shown in Figure 2.8(Right). The improved centroids by one-pass+2LBG are denoted as "○"s in Figure 2.8(Right) also.

Table 2.5 contains a comparison of the one-pass algorithm with several other algorithms on independent Laplace sources. The one-pass algorithm has a speed-up-in-flops of $5.8/1.5 = 3.9$ and lower MSE than the benchmark results.

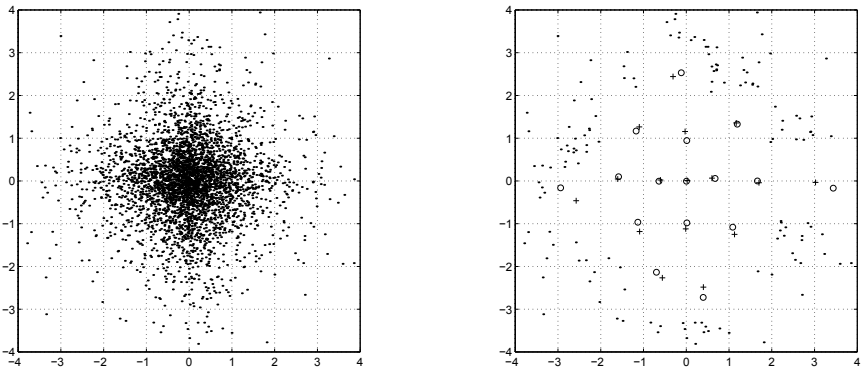

Fig. 2.8. Left: The Laplace source training data. Right: The residual data, one-pass designed centroids "+", and one-pass+2LBG centroids "○".

Table 2.5. Comparison on the Laplace Source

Types	Iterations.	Mflops	CPU (sec.)	MSE
One-Pass	1	1.5	60	0.262
One-Pass + LBG	1 + 2	3.0	111	0.259
LBG (this chapter)	7	5.8	208	0.260
UDQ (from [18])				0.302
MAX (from [14])				0.352
LBG (from [18])				0.264

2.4.5 Robustness Analysis

For robust speaker and speech recognition, the selected VQ algorithm needs to be robust. What this means is that the outliers in the training data should

have little effect on the VQ training results. The one-pass VQ algorithm has the necessary robust property, because the sequential process can be invariant to outliers.

Let us assume that the residual data vectors X_o in Figure 2.8(Right) are outliers. The entire training data set X is a union of the removed set X_r and outliers set X_o, $X = X_r \cup X_o$. Due to the low density of the outlier area, the one-pass algorithm would not assign codebook vectors to them. If we don't want to include the outliers in our training in order to improve the robustness of the designed codebook, we can only use the data set X_r in the last step of centroid update (Step 4 in the List of the Algorithm.) Thus, the outliers X_o are not included in the VQ design and the designed codebook is therefore robust with regard to these outliers.

In summary, the experimental results for different data sources demonstrate that the one-pass algorithm results in near-optimal MSE while the CPU time and Mflops are only slightly more than that of one single iteration of the LBG algorithm. High performance, fast training time, and robustness are the advantages of the one-pass algorithm.

2.5 Segmental K-Means

In the above discussions, we addressed the segmentation problem for a stationary process, where the joint probability distribution does not change when shifting in time. However, a speech signal is a non-stationary process, where the joint probability distribution of observed data changes when shifting in time. The current approach to addressing segmentation in a speech signal is to segment the non-stationary speech sequence into a sequence of small segmentations. Within each small segmentation, we can assume the data is stationary. Segmental K-mans was developed for this purpose. It has been applied to Markov chain modeling or hidden Markov model (HMM) parameter estimation [13, 16, 7]. It is one of the fundamental algorithms for HMM training.

The K-mean algorithm involves iteration of two kinds of computations: segmentation and optimization. In the beginning, the model parameters such as the centroids or means are initialized by random numbers in meaningful ranges. In the segmentation stage, the sequentially observed non-stationary data are partitioned into multiple sequential states. There is no overlap among the states. Within each state, the joint probability distribution is assumed to be stationary; thus, the optimization process can be followed to estimate the model parameters for the data within each state. The segmentation process is equivalent to a sequential decoding procedure and can be performed by using the Viterbi [23] algorithm. The iteration between the optimization and segmentation may need to repeat for several times. The details of the segmental K-mean algorithms are available in [13, 16, 7] while the details of the decoding process will be discussed in Chapter 6.

2.6 Conclusions

In this chapter, we introduced multivariate Gaussian distribution because it will be used often in the rest of this book. We also introduced the concept of principal component analysis because this concept is helpful for reading the rest of this book and to intuitively think about data representations when developing new algorithms. We briefly reviewed the popular K-means and LBG algorithms for VQ. Readers can get more detailed information from other textbooks (e.g. [6, 1, 5]) to understand these traditional algorithms in more detail. We presented the one-pass VQ algorithm in detail. The one-pass VQ algorithm is useful in training background models with very large datasets such as those used in speaker recognition. The concept and method of sequential data processing and pruning used in the one-pass VQ will be used in Chapter 3 and Chapter 14 for pattern recognition and speaker authentication. Finally, we discussed the segmental K-mean algorithm, which will be used in HMM training throughout the book.

References

1. Duda, R. O., Hart, P. E., and Stork, D. G., *Pattern Classification, Second Edition*. New York: John & Wiley, 2001.
2. Fischer, T. R. and Dicharry, R. M., "Vector quantizer design for gaussian, gamma, and laplacian sources," *IEEE Transactions on Communications*, vol. COM-32, pp. 1065–1069, September 1984.
3. Gray, R. M. and Linde, Y., "Vector quantizers and predictive quantizers for Gauss-Markov sources," *IEEE Transactions on Communications*, vol. COM-30, pp. 381–389, September 1982.
4. Huang, C. M. and Harris, R. W., "A comparison of several vector quantization codebook generation approaches," *IEEE Transactions on Image Processing*, vol. 2, pp. 108–112, January 1993.
5. Huang, X., Acero, A., and Hon, H.-W., *Spoken language processing*. NJ: Prentice Hall PTR, 2001.
6. Johnson, R. A. and Wichern, D. W., *Applied Multivariate Statistical Analysis*. New Jersey: Prentice Hall, 1988.
7. Juang, B.-H. and Rabiner, L. R., "The segmental k-means algorithm for estimating parameters of hidden Markove models," *IEEE Trans. Acoustics, speech and Signal Processing*, vol. 38, pp. 1639–1641, Sept. 1990.
8. Li, Q. and Swaszek, P. F., "One-pass vector quantizer design by sequential pruning of the training data," in *Proceedings of International Conference on Image Processing*, (Washington DC), October 1995.
9. Li, Q. and Tufts, D. W., "Improving discriminant neural network (DNN) design by the use of principal component analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, (Detroit MI), pp. 3375–3379, May 1995.
10. Li, Q. and Tufts, D. W., "Synthesizing neural networks by sequential addition of hidden nodes," in *Proceedings of the IEEE International Conference on Neural Networks*, (Orlando FL), pp. 708–713, June 1994.

11. Li, Q., Tufts, D. W., Duhaime, R., and August, P., "Fast training algorithms for large data sets with application to classification of multispectral images," in *Proceedings of the IEEE 28th Asilomar Conference*, (Pacific Grove), October 1994.
12. Linde, Y., Buzo, A., , and Gray, R. M., "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, vol. COM-28, pp. 84–95, 1980.
13. MacQueen, J., "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Stat., Prob.*, pp. 281–296, 1967.
14. Max, J., "Quantizing for minimum distortion," *IEEE Transactions on Information Theory*, vol. IT-6, pp. 7–12, March 1960.
15. Paez, M. D. and Glisson, T. H., "Minimum mean-square-error quantization in speech pcm and dpcm systems," *IEEE Transactions on Communications*, vol. COM-20, pp. 225–230, April 1972.
16. Rabiner, L. R., Wilpon, J. G., and Juang, B.-H., "A segmental k-means training procedure for connected word recognition," *AT&T Technical Journal*, vol. 65, pp. 21–31, May/June 1986.
17. Soong, F. K., Rosenberg, A. E., and Juang, B.-H., "A vector quantization approach to speaker recognition," *AT&T Technical Journal*, vol. 66, pp. 14–26, March/April 1987.
18. Swaszek, P. F., "Low dimension / moderate bitrate vector quantizers for the laplace source," in *Abstracts of IEEE International Symposium on Information Theory*, p. 74, 1990.
19. Swaszek, P. F., "Vector quantization for image compression," in *Proceedings of Princeton Conference on Information Sciences and Systems*, (Princeton NJ), pp. 254–259, March 1986.
20. Swaszek, P. F. and Narasimhan, A., "Quantization of the correlated gaussian source," in *Proceedings of Princeton Conference on Information Sciences and Systems*, (Princeton NJ), pp. 784–789, March 1988.
21. Swaszek, P. F. and Thomas, J. B., "Optimal circularly symmetric quantizers," *Journal of Franklin Institute*, vol. 313, pp. 373–384, June 1982.
22. Tufts, D. W. and Li, Q., "Principal feature classification," in *Neural Networks for Signal Processing V, Proceedings of the 1995 IEEE Workshop*, (Cambridge MA), August 1995.
23. Viterbi, A. J., "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260–269, April 1967.
24. Wilson, S. G., "Magnitude/phase quantization of independent gaussian variates," *IEEE Transactions on Communications*, vol. COM-28, pp. 1924–1929, November 1980.

http://www.springer.com/978-3-642-23730-0

Speaker Authentication

Li, Q.P.

2012, XXVI, 238 p., Hardcover
ISBN: 978-3-642-23730-0