Lecture Outline

Reading: Chapter 9

This lecture covers

- K-means Clustering
- Relation to Gaussian mixture model
- Application: nonuniform, scalar quantization
- Application: vector quantization

9.1 K-means Clustering

We begin by considering the problem of identifying groups, or clusters, of data points in a multidimensional space. Suppose we have a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ consisting of N observations of a random D-dimensional variable $\mathbf{x}$. This data set has no associated class labels. Our goal is to partition the data set into K clusters. Intuitively, we might think of a cluster as comprising a group of data points whose inter-point Euclidean distances are small compared with the distances to points outside of the cluster.

We can formalize this notion by first introducing a set of D-dimensional vectors $\boldsymbol{\mu}_k$ where $k = 1, \ldots, K$, in which $\boldsymbol{\mu}_k$ is a prototype associated with the kth cluster. We can think of the $\boldsymbol{\mu}_k$ as representing the centers of the clusters. Our goal is then to find an assignment of data points to the clusters as well as a set of vectors $\{\boldsymbol{\mu}_k\}$, such that the sum of the squares of the distances of each data point to its closest vector $\boldsymbol{\mu}_k$ is a minimum.

For each data point $\mathbf{x}_n$, we introduce a corresponding set of indicator variables $r_{nk} \in \{0, 1\}$ describing which cluster the data point $\mathbf{x}_n$ is assigned to. That is, if data point $\mathbf{x}_n$ is assigned to cluster k then $r_{nk} = 1$, and $r_{nj} = 0$ for $j \neq k$.

We then define an objective function, sometimes called a *distortion measure*, given by

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2 \tag{1}$$

which represents the total squared distance from each data point to its cluster center. Our goal is to find the values for $\{r_{nk}\}$ and the $\{\boldsymbol{\mu}_k\}$ so as to minimize J.

Iterative Solution

We can solve this problem through an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the r_{nk} and the $\boldsymbol{\mu}_k$. First, we choose some initial values for the $\boldsymbol{\mu}_k$. Then in the first phase we minimize J with respect to the r_{nk}, keeping $\boldsymbol{\mu}_k$ fixed. In the second phase we minimize J with respect to the $\boldsymbol{\mu}_k$, keeping r_{nk} fixed. This two-stage optimization is then repeated until convergence. We shall see that these two stages of updating r_{nk} and updating $\boldsymbol{\mu}_k$ correspond respectively to the E (expectation) step and M (maximization) steps of the EM algorithm.

Consider first the determination of the r_{nk}. Because J is a linear function of r_{nk}, the optimal solution can be expressed in closed form:

$$r_{nk} = \begin{cases} 1, & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Now consider the optimization of the $\boldsymbol{\mu}_k$ with the r_{nk} held fixed. Because J is a quadratic function of $\boldsymbol{\mu}_k$, the optimal solution can be also expressed in closed form:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}. \tag{3}$$

The numerator in this expression is equal to the sum of the data points assigned to the cluster k and the denominator in this expression is equal to the number of points assigned to the cluster k. This result has a simple interpretation: namely set $\boldsymbol{\mu}_k$ equal to the *mean* of all the data points $\mathbf{x}_n$ assigned to cluster k, i.e. $\boldsymbol{\mu}_k$ is the centroid of cluster k. For this reason, the procedure is known as the K-means algorithm.

The two phases of re-assigning data points to clusters and re-computing the cluster means are repeated until there is no further change in assignments. The cluster assignment step is analogous to the "E" step and re-computing of the cluster centers is analogous to the "M" step in the EM algorithm for GMM parameter estimation. In practice, we often initialize the $\{\boldsymbol{\mu}_k\}$ to be equal to a random subset of K data points.

Example

The K-means algorithm is illustrated using the Old Faithful dataset in Figure 4. Note that the dataset has been *standardized*, so that the data points have a zero mean and unit variance. A plot of the cost function J is given in Figure 5.

Demo (MATLAB)

Remarks

1. A direct implementation of the K-means algorithm can be relatively slow, because in each E step it is necessary to compute the Euclidean distance between every pair of $\boldsymbol{\mu}_k$ and $\mathbf{x}_n$. There is an interesting "fast" K-means algorithm based on bounding these distance measures via the triangle inequality in order to reduce the number which must be computed.

2. The K-means algorithm is based on the use of squared Euclidean distance as the measure of dissimilarity between a data point and a prototype vector. We can generalize the K-means algorithm by introducing a more general dissimilarity measure $\mathcal{V}(\mathbf{x}, \mathbf{x}')$ between two vectors $\mathbf{x}$ and $\mathbf{x}'$ and then minimizing the following distortion measure

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\mathcal{V}(\mathbf{x}, \boldsymbol{\mu}_k) \tag{4}$$

   which gives the K-medoids algorithm.

Silhouette Index for Determining K

These subsection taken from https://en.wikipedia.org/wiki/Silhouette_(clustering)

The silhouette index (or value) is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters, hence the silhouette index can be used to assist in choosing K.

Assume the data have been clustered into K clusters. For each datum i, let $a(i)$ be the average dissimilarity (distance) of i with all other data within the same cluster. We can interpret $a(i)$ as how well i is assigned to its cluster (the smaller the value, the better the assignment). We then define the average dissimilarity of point i to a cluster c as the average of the distance from i to all points in c.

Let $b(i)$ be the lowest average dissimilarity of i to any other cluster, of which i is not a member. The cluster with this lowest average dissimilarity is said to be the "neighbouring cluster" of i because it is the next best fit cluster for point i. We now define a silhouette:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{5}$$

Which can be also written as:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & a(i) < b(i) \\ 0, & a(i) = b(i) \\ b(i)/a(i) - 1 & a(i) > b(i) \end{cases} \tag{6}$$

From the above definition it is clear that

$$-1 \leq s(i) \leq 1 \tag{7}$$

For $s(i)$ to be close to 1 we require $a(i) \ll b(i)$. As $a(i)$ is a measure of how dissimilar i is to its own cluster, a small value means it is well matched. Furthermore, a large $b(i)$ implies that i is badly matched to its neighbouring cluster. Thus an $s(i)$ close to one means that the datum is appropriately clustered. If $s(i)$ is close to -1, then by the same logic we see that i would be more appropriate if it was clustered in its neighbouring cluster. An $s(i)$ near zero means that the datum is on the border of two natural clusters.

The average $s(i)$ over all data of a cluster is a measure of how tightly grouped all the data in the cluster are. Thus the average $s(i)$ over all data of the entire dataset is a measure of how appropriately the data have been clustered. If there are too many or too few clusters, as may occur when a poor choice of K is used, some of the clusters will typically display much narrower silhouettes than the rest. Thus silhouette plots and averages may be used to determine the natural number of clusters within a dataset.

Information Criterion Approaches for Determining K

Another set of methods for determining the number of clusters are information criteria, such as the Akaike information criterion (AIC) or Bayesian information criterion (BIC). For more information, please software library documentation.

9.3.2 Relation of Gaussian Mixture Models to K-means

One notable feature of the K-means algorithm is that at each iteration, every data point is assigned to one, and only one, of the clusters. Whereas some data points will be much closer to a particular center $\boldsymbol{\mu}_k$ than to any other center, there may be other data points that lie roughly midway between cluster centers. In the latter case, it is not clear that the hard assignment to the nearest cluster is the most appropriate.

Comparison of the K-means algorithm with the EM algorithm for Gaussian mixtures shows there is a close similarity. Whereas the K-means algorithm performs a *hard* assignment of data points to clusters, in which each data point is associated uniquely with one cluster, the EM algorithm makes a *soft* assignment based on posterior probabilities. See text for more details.

Application: Nonuniform, Scalar Quantization

Uniform, Scalar Quantization

In the conversion of a discrete-time signal, $x[n]$ to a digital signal, $\hat{x}[n]$ we quantize the amplitude of $x[n]$ by uniformly partitioning the full-scale range, R into 2^B levels where B is the number of bits on the code-word. A typical uniform quantizer is shown in Figure 1 and a sinusoid with 16 and 3 bit code words is shown in Figure 2.

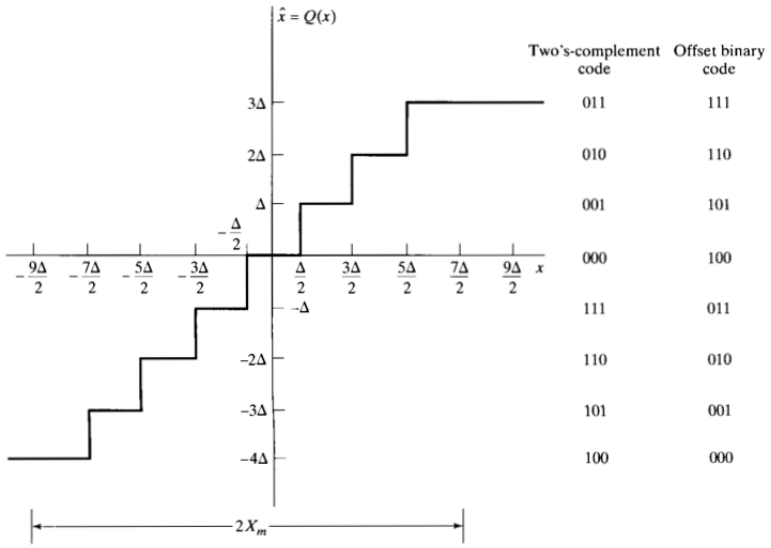
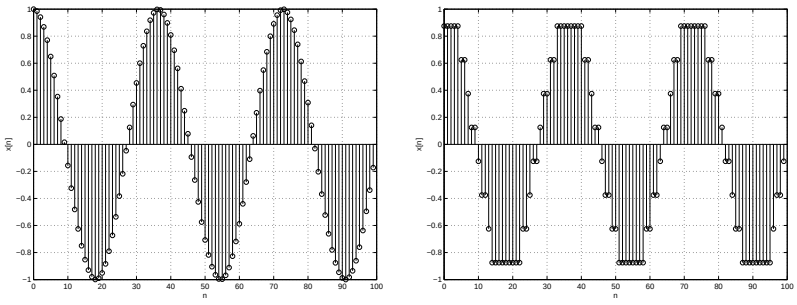


Figure 4.48 Typical quantizer for A/D conversion.

Figure 1:



(a) (b)

Figure 2: Sinusoid quantized with (a) 16 bits and (b) 3 bits.

Assuming the discrete-time sample values are *uniformly* distributed (and making other, simplifying assumptions) it can be shown that the distortion introduced by quantization can be quantified at roughly 6 dB per bit. In other words, for each bit in the code word, the SNR increases by about 6 dB, i.e. 6 dB-per-bit-rule. For uniform sample distributions, the optimal (in the sense of maximizing SNR) quantizer is uniform.

Nonuniform, Scalar Quantization

Speech signals, for example, do not have uniform distributions (see Figure 3) and therefore the uniform quantizer is not optimal. To understand this limitation, suppose small sample values are more likely to occur than large sample values (either positive or negative values). Intuition, tells us to use more resolution (smaller quantization steps) for the small-valued samples (where the probability of occurrence is high) and less resolution (larger quantization steps) for the large-valued samples (where the probability of occurrence is low). In theory, this would give us a lot of small quantization errors and a few large quantization errors with the hope that, overall, the quantization error would be less than that with a uniform quantizer.

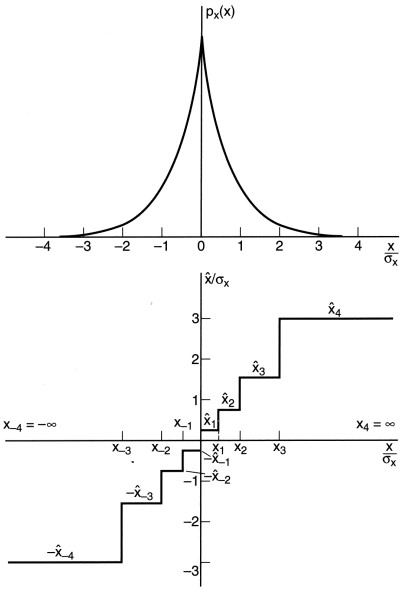


Figure 12.6 3-bit nonuniform quantizer: (a) Laplacian pdf; (b) decision and reconstruction levels.

Figure 3:

Quantization in which reconstruction and decision levels do not have equal spacing is called a *nonuniform quantization* (see Figure 3) . The K-means algorithm can be used to design a nonuniform quantizer. We typically begin with training samples and set K equal to 2^B where B is the number of codeword bits. After iterating, the decision levels are taken as the resulting cluster centers, $\boldsymbol{\mu}_k$.

Demo: See Lloyd demo in EE589 Lecture 12.

9.1.1 Application: vector quantization

We can use the result of a clustering algorithm to perform data compression. It is important to distinguish between *lossless data compression*, in which the goal is to be able to reconstruct the original data exactly from the compressed representation, and *lossy data compression*, in which we accept some errors in the reconstruction in return for higher levels of compression than can be achieved in the lossless case.

We can apply K-means algorithm to the problem of lossy data compression as follows. For each of the N data points, we store only the identity k of the cluster to which it is assigned. We also store the values of the K cluster centers $\boldsymbol{\mu}_k$ which typically requires significantly less data, provided we choose $K \ll N$. Each data point is then approximated by its nearest center $\boldsymbol{\mu}_k$, New data points can similarly be compressed by first finding the nearest $\boldsymbol{\mu}_k$ and then storing the label k instead of the original data vector. This framework is often called *vector quantization*, and the vectors $\boldsymbol{\mu}_k$ are called *code-book vectors*.

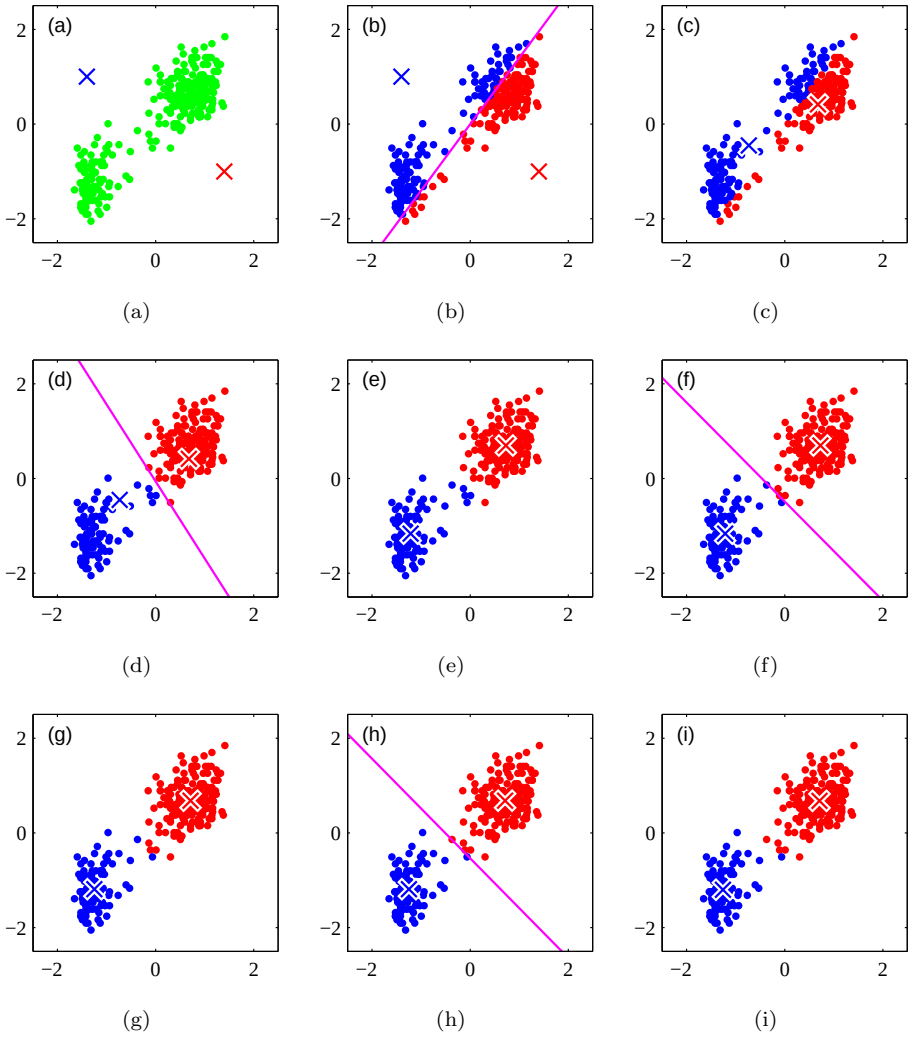


Figure 4: Illustration of the K means algorithm using the Old Faithful data set.

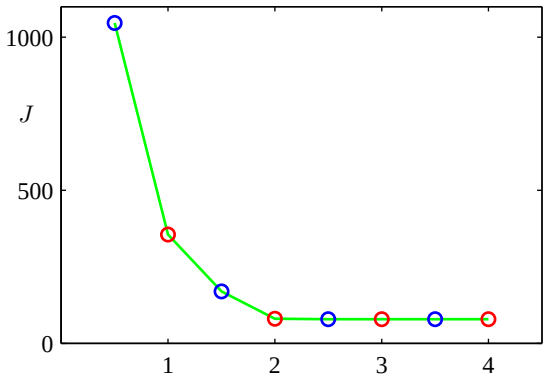


Figure 5: