CHAPTER 13 Representative-based Clustering

Given a dataset with n points in a d-dimensional space, $\mathbf{D} = \{\mathbf{x}_i\}_{i=1}^n$, and given the number of desired clusters k, the goal of representative-based clustering is to partition the dataset into k groups or clusters, which is called a *clustering* and is denoted as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$. Further, for each cluster C_i there exists a representative point that summarizes the cluster, a common choice being the mean (also called the *centroid*) $\boldsymbol{\mu}_i$ of all points in the cluster, that is,

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{x_j \in C_i} \mathbf{x}_j$$

where $n_i = |C_i|$ is the number of points in cluster C_i.

A brute-force or exhaustive algorithm for finding a good clustering is simply to generate all possible partitions of n points into k clusters, evaluate some optimization score for each of them, and retain the clustering that yields the best score. The *exact* number of ways of partitioning n points into k nonempty and disjoint parts is given by the *Stirling numbers of the second kind*, given as

$$S(n,k) = \frac{1}{k!} \sum_{t=0}^{k} (-1)^t \binom{k}{t} (k-t)^n$$

Informally, each point can be assigned to any one of the k clusters, so there are at most k^n possible clusterings. However, any permutation of the k clusters within a given clustering yields an equivalent clustering; therefore, there are $O(k^n/k!)$ clusterings of n points into k groups. It is clear that exhaustive enumeration and scoring of all possible clusterings is not practically feasible. In this chapter we describe two approaches for representative-based clustering, namely the K-means and expectation-maximization algorithms.

13.1 K-MEANS ALGORITHM

Given a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ we need some scoring function that evaluates its quality or goodness. This *sum of squared errors* scoring function is defined as

$$SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2 \tag{13.1}$$

The goal is to find the clustering that minimizes the SSE score:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \{SSE(\mathcal{C})\}$$

K-means employs a greedy iterative approach to find a clustering that minimizes the SSE objective [Eq. (13.1)]. As such it can converge to a local optima instead of a globally optimal clustering.

K-means initializes the cluster means by randomly generating k points in the data space. This is typically done by generating a value uniformly at random within the range for each dimension. Each iteration of K-means consists of two steps: (1) cluster assignment, and (2) centroid update. Given the k cluster means, in the cluster assignment step, each point $\mathbf{x}_j \in \mathbf{D}$ is assigned to the closest mean, which induces a clustering, with each cluster C_i comprising points that are closer to $\boldsymbol{\mu}_i$ than any other cluster mean. That is, each point $\mathbf{x}_j$ is assigned to cluster C_{j^*}, where

$$j^* = \arg\min_{i=1}^{k} \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2 \right\} \tag{13.2}$$

Given a set of clusters C_i, $i = 1, \ldots, k$, in the centroid update step, new mean values are computed for each cluster from the points in C_i. The cluster assignment and centroid update steps are carried out iteratively until we reach a fixed point or local minima. Practically speaking, one can assume that K-means has converged if the centroids do not change from one iteration to the next. For instance, we can stop if $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$, where $\epsilon > 0$ is the convergence threshold, t denotes the current iteration, and $\boldsymbol{\mu}_i^t$ denotes the mean for cluster C_i in iteration t.

The pseudo-code for K-means is given in Algorithm 13.1. Because the method starts with a random guess for the initial centroids, K-means is typically run several times, and the run with the lowest SSE value is chosen to report the final clustering. It is also worth noting that K-means generates convex-shaped clusters because the region in the data space corresponding to each cluster can be obtained as the intersection of half-spaces resulting from hyperplanes that bisect and are normal to the line segments that join pairs of cluster centroids.

In terms of the computational complexity of K-means, we can see that the cluster assignment step take $O(nkd)$ time because for each of the n points we have to compute its distance to each of the k clusters, which takes d operations in d dimensions. The centroid re-computation step takes $O(nd)$ time because we have to add at total of n d-dimensional points. Assuming that there are t iterations, the total time for K-means is given as $O(tnkd)$. In terms of the I/O cost it requires $O(t)$ full database scans, because we have to read the entire database in each iteration.

Example 13.1. Consider the one-dimensional data shown in Figure 13.1a. Assume that we want to cluster the data into $k = 2$ groups. Let the initial centroids be $\mu_1 = 2$ and $\mu_2 = 4$. In the first iteration, we first compute the clusters, assigning each point

ALGORITHM 13.1. K-means Algorithm

K-MEANS ($\mathbf{D}, k, \epsilon$):

```
1  t = 0
2  Randomly initialize k centroids: μ_1^t, μ_2^t, ..., μ_k^t ∈ R^d
3  repeat
4      t ← t + 1
5      C_j ← ∅ for all j = 1, ..., k
       // Cluster Assignment Step
6      foreach x_j ∈ D do
7          j* ← arg min_i { ||x_j − μ_i^t||^2 } // Assign x_j to closest centroid
8          C_j* ← C_j* ∪ {x_j}
       // Centroid Update Step
9      foreach i = 1 to k do
10         μ_i^t ← (1/|C_i|) Σ_{x_j ∈ C_i} x_j
11 until Σ_{i=1}^{k} ||μ_i^t − μ_i^{t−1}||^2 ≤ ε
```

to the closest mean, to obtain

$$C_1 = \{2, 3\} \qquad C_2 = \{4, 10, 11, 12, 20, 25, 30\}$$

We next update the means as follows:

$$\mu_1 = \frac{2+3}{2} = \frac{5}{2} = 2.5$$

$$\mu_2 = \frac{4+10+11+12+20+25+30}{7} = \frac{112}{7} = 16$$

The new centroids and clusters after the first iteration are shown in Figure 13.1b. For the second step, we repeat the cluster assignment and centroid update steps, as shown in Figure 13.1c, to obtain the new clusters:

$$C_1 = \{2, 3, 4\} \qquad C_2 = \{10, 11, 12, 20, 25, 30\}$$

and the new means:

$$\mu_1 = \frac{2+3+4}{4} = \frac{9}{3} = 3$$

$$\mu_2 = \frac{10+11+12+20+25+30}{6} = \frac{108}{6} = 18$$

The complete process until convergence is illustrated in Figure 13.1. The final clusters are given as

$$C_1 = \{2, 3, 4, 10, 11, 12\} \qquad C_2 = \{20, 25, 30\}$$

with representatives $\mu_1 = 7$ and $\mu_2 = 25$.

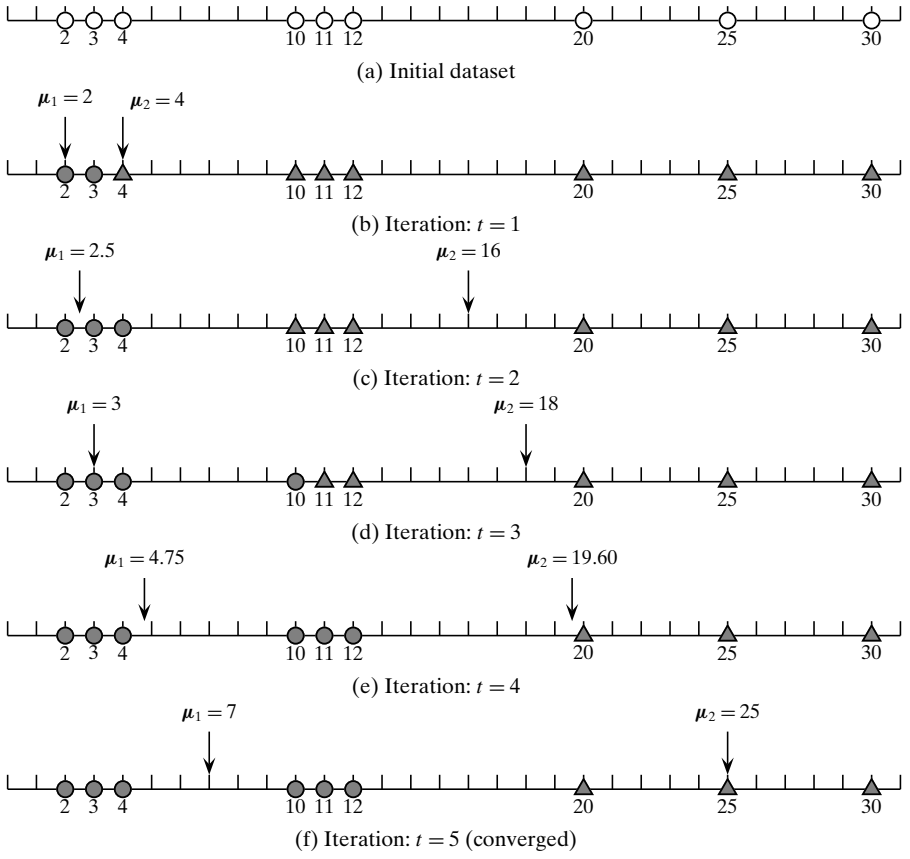


Figure 13.1. K-means in one dimension.

Example 13.2 (K-means in Two Dimensions). In Figure 13.2 we illustrate the K-means algorithm on the Iris dataset, using the first two principal components as the two dimensions. Iris has $n = 150$ points, and we want to find $k = 3$ clusters, corresponding to the three types of Irises. A random initialization of the cluster means yields

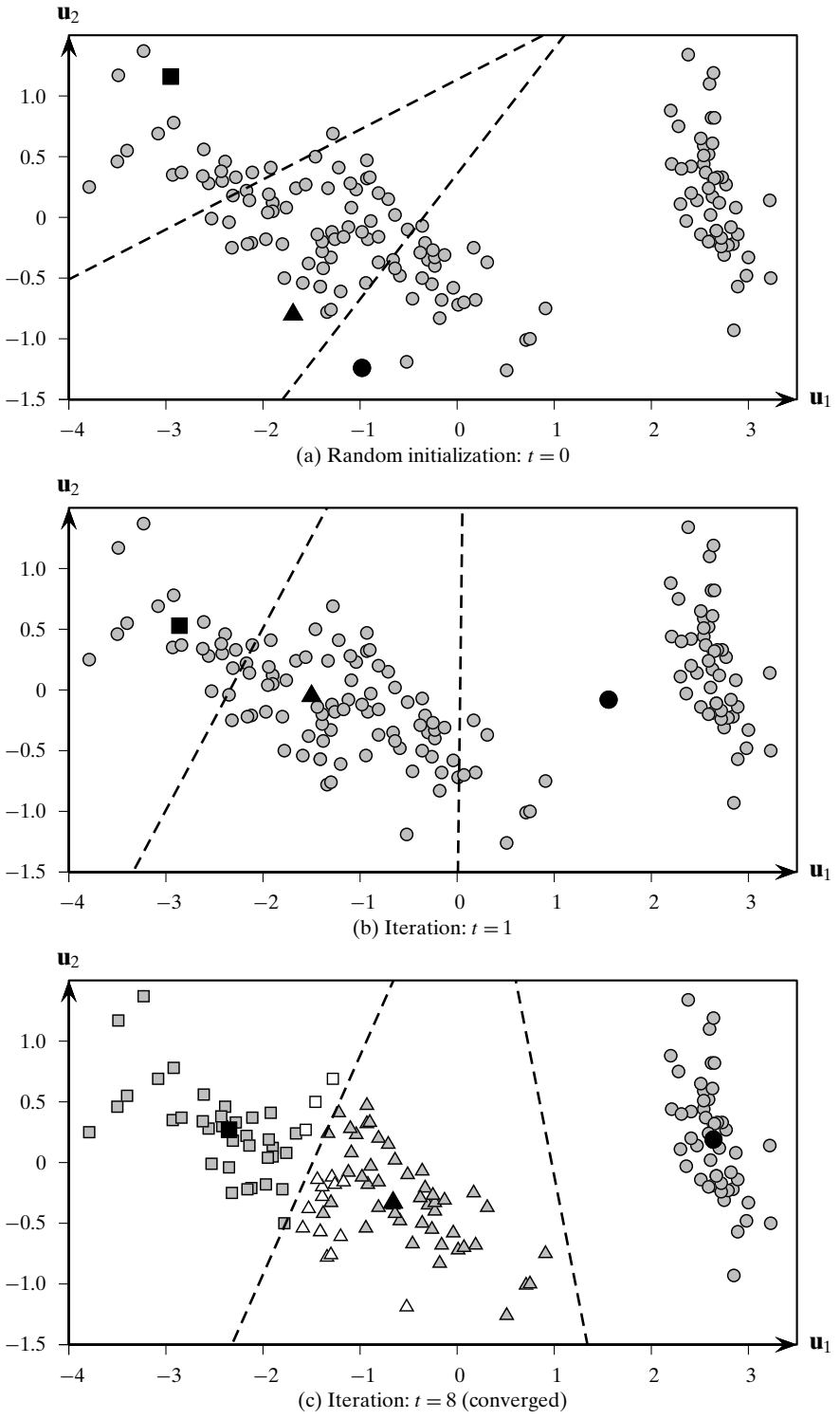
$$\boldsymbol{\mu}_1 = (-0.98, -1.24)^T \qquad \boldsymbol{\mu}_2 = (-2.96, 1.16)^T \qquad \boldsymbol{\mu}_3 = (-1.69, -0.80)^T$$

as shown in Figure 13.2a. With these initial clusters, K-means takes eight iterations to converge. Figure 13.2b shows the clusters and their means after one iteration:

$$\boldsymbol{\mu}_1 = (1.56, -0.08)^T \qquad \boldsymbol{\mu}_2 = (-2.86, 0.53)^T \qquad \boldsymbol{\mu}_3 = (-1.50, -0.05)^T$$

Finally, Figure 13.2c shows the clusters on convergence. The final means are as follows:

$$\boldsymbol{\mu}_1 = (2.64, 0.19)^T \qquad \boldsymbol{\mu}_2 = (-2.35, 0.27)^T \qquad \boldsymbol{\mu}_3 = (-0.66, -0.33)^T$$



(a) Random initialization: $t = 0$

(b) Iteration: $t = 1$

(c) Iteration: $t = 8$ (converged)

Figure 13.2. K-means in two dimensions: Iris principal components dataset.

Figure 13.2 shows the cluster means as black points, and shows the convex regions of data space that correspond to each of the three clusters. The dashed lines (hyperplanes) are the perpendicular bisectors of the line segments joining two cluster centers. The resulting convex partition of the points comprises the clustering.

Figure 13.2c shows the final three clusters: C_1 as circles, C_2 as squares, and C_3 as triangles. White points indicate a wrong grouping when compared to the known Iris types. Thus, we can see that C_1 perfectly corresponds to `iris-setosa`, and the majority of the points in C_2 correspond to `iris-virginica`, and in C_3 to `iris-versicolor`. For example, three points (white squares) of type `iris-versicolor` are wrongly clustered in C_2, and 14 points from `iris-virginica` are wrongly clustered in C_3 (white triangles). Of course, because the Iris class label is not used in clustering, it is reasonable to expect that we will not obtain a perfect clustering.

13.2 KERNEL K-MEANS

In K-means, the separating boundary between clusters is linear. Kernel K-means allows one to extract nonlinear boundaries between clusters via the use of the kernel trick outlined in Chapter 5. This way the method can be used to detect nonconvex clusters.

In kernel K-means, the main idea is to conceptually map a data point $\mathbf{x}_i$ in input space to a point $\phi(\mathbf{x}_i)$ in some high-dimensional feature space, via an appropriate nonlinear mapping ϕ. However, the kernel trick allows us to carry out the clustering in feature space purely in terms of the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, which can be computed in input space, but corresponds to a dot (or inner) product $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ in feature space.

Assume for the moment that all points $\mathbf{x}_i \in \mathbf{D}$ have been mapped to their corresponding images $\phi(\mathbf{x}_i)$ in feature space. Let $\mathbf{K} = \{K(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1,\ldots,n}$ denote the $n \times n$ symmetric kernel matrix, where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Let $\{C_1, \ldots, C_k\}$ specify the partitioning of the n points into k clusters, and let the corresponding cluster means in feature space be given as $\{\boldsymbol{\mu}_1^\phi, \ldots, \boldsymbol{\mu}_k^\phi\}$, where

$$\boldsymbol{\mu}_i^\phi = \frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i} \phi(\mathbf{x}_j)$$

is the mean of cluster C_i in feature space, with $n_i = |C_i|$.

In feature space, the kernel K-means sum of squared errors objective can be written as

$$\min_{\mathcal{C}} SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^\phi \right\|^2$$

Expanding the kernel SSE objective in terms of the kernel function, we get

$$\begin{aligned} SSE(\mathcal{C}) &= \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^\phi \right\|^2 \\ &= \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in C_i} \left\| \phi(\mathbf{x}_j) \right\|^2 - 2\phi(\mathbf{x}_j)^T \boldsymbol{\mu}_i^\phi + \left\| \boldsymbol{\mu}_i^\phi \right\|^2 \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{k}\Big(\Big(\sum_{\mathbf{x}_j \in C_i} \big\|\phi(\mathbf{x}_j)\big\|^2\Big) - 2n_i\Big(\frac{1}{n_i}\sum_{\mathbf{x}_j \in C_i}\phi(\mathbf{x}_j)\Big)^T \boldsymbol{\mu}_i^{\phi} + n_i \left\|\boldsymbol{\mu}_i^{\phi}\right\|^2\Big) \\
&= \Big(\sum_{i=1}^{k}\sum_{\mathbf{x}_j \in C_i}\phi(\mathbf{x}_j)^T\phi(\mathbf{x}_j)\Big) - \Big(\sum_{i=1}^{k} n_i \left\|\boldsymbol{\mu}_i^{\phi}\right\|^2\Big) \\
&= \sum_{i=1}^{k}\sum_{\mathbf{x}_j \in C_i} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k}\frac{1}{n_i}\sum_{\mathbf{x}_a \in C_i}\sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b) \\
&= \sum_{j=1}^{n} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{i=1}^{k}\frac{1}{n_i}\sum_{\mathbf{x}_a \in C_i}\sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)
\end{aligned} \tag{13.3}
$$

Thus, the kernel K-means SSE objective function can be expressed purely in terms of the kernel function. Like K-means, to minimize the SSE objective we adopt a greedy iterative approach. The basic idea is to assign each point to the closest mean in feature space, resulting in a new clustering, which in turn can be used obtain new estimates for the cluster means. However, the main difficulty is that we cannot explicitly compute the mean of each cluster in feature space. Fortunately, explicitly obtaining the cluster means is not required; all operations can be carried out in terms of the kernel function $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

Consider the distance of a point $\phi(\mathbf{x}_j)$ to the mean $\boldsymbol{\mu}_i^{\phi}$ in feature space, which can be computed as

$$
\begin{aligned}
\left\|\phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^{\phi}\right\|^2 &= \big\|\phi(\mathbf{x}_j)\big\|^2 - 2\phi(\mathbf{x}_j)^T\boldsymbol{\mu}_i^{\phi} + \left\|\boldsymbol{\mu}_i^{\phi}\right\|^2 \\
&= \phi(\mathbf{x}_j)^T\phi(\mathbf{x}_j) - \frac{2}{n_i}\sum_{\mathbf{x}_a \in C_i}\phi(\mathbf{x}_j)^T\phi(\mathbf{x}_a) + \frac{1}{n_i^2}\sum_{\mathbf{x}_a \in C_i}\sum_{\mathbf{x}_b \in C_i}\phi(\mathbf{x}_a)^T\phi(\mathbf{x}_b) \\
&= K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{n_i}\sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j) + \frac{1}{n_i^2}\sum_{\mathbf{x}_a \in C_i}\sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)
\end{aligned} \tag{13.4}
$$

Thus, the distance of a point to a cluster mean in feature space can be computed using only kernel operations. In the cluster assignment step of kernel K-means, we assign a point to the closest cluster mean as follows:

$$
\begin{aligned}
C^*(\mathbf{x}_j) &= \arg\min_i \left\{\left\|\phi(\mathbf{x}_j) - \boldsymbol{\mu}_i^{\phi}\right\|^2\right\} \\
&= \arg\min_i \left\{K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{n_i}\sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j) + \frac{1}{n_i^2}\sum_{\mathbf{x}_a \in C_i}\sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)\right\} \\
&= \arg\min_i \left\{\frac{1}{n_i^2}\sum_{\mathbf{x}_a \in C_i}\sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b) - \frac{2}{n_i}\sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j)\right\}
\end{aligned} \tag{13.5}
$$

where we drop the $K(\mathbf{x}_j, \mathbf{x}_j)$ term because it remains the same for all k clusters and does not impact the cluster assignment decision. Also note that the first term is simply the average pairwise kernel value for cluster C_i and is independent of the point $\mathbf{x}_j$. It is in fact the squared norm of the cluster mean in feature space. The second term is twice the average kernel value for points in C_i with respect to $\mathbf{x}_j$.

Algorithm 13.2 shows the pseudo-code for the kernel K-means method. It starts from an initial random partitioning of the points into k clusters. It then iteratively updates the cluster assignments by reassigning each point to the closest mean in feature space via Eq. (13.5). To facilitate the distance computation, it first computes the average kernel value, that is, the squared norm of the cluster mean, for each cluster (for loop in line 5). Next, it computes the average kernel value for each point $\mathbf{x}_j$ with points in cluster C_i (for loop in line 7). The main cluster assignment step uses these values to compute the distance of $\mathbf{x}_j$ from each of the clusters C_i and assigns $\mathbf{x}_j$ to the closest mean. This reassignment information is used to re-partition the points into a new set of clusters. That is, all points $\mathbf{x}_j$ that are closer to the mean for C_i make up the new cluster for the next iteration. This iterative process is repeated until convergence.

For convergence testing, we check if there is any change in the cluster assignments of the points. The number of points that do not change clusters is given as the sum $\sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}|$, where t specifies the current iteration. The fraction of points

ALGORITHM 13.2. Kernel K-means Algorithm

KERNEL-KMEANS($\mathbf{K}, k, \epsilon$):

1 $t \leftarrow 0$
2 $\mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$ // Randomly partition points into k clusters
3 **repeat**
4 $\quad t \leftarrow t + 1$
5 $\quad$ **foreach** $C_i \in \mathcal{C}^{t-1}$ **do** // Compute squared norm of cluster means
6 $\quad\quad \text{sqnorm}_i \leftarrow \frac{1}{n_i^2} \sum_{\mathbf{x}_a \in C_i} \sum_{\mathbf{x}_b \in C_i} K(\mathbf{x}_a, \mathbf{x}_b)$
7 $\quad$ **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do** // Average kernel value for $\mathbf{x}_j$ and C_i
8 $\quad\quad$ **foreach** $C_i \in \mathcal{C}^{t-1}$ **do**
9 $\quad\quad\quad \text{avg}_{ji} \leftarrow \frac{1}{n_i} \sum_{\mathbf{x}_a \in C_i} K(\mathbf{x}_a, \mathbf{x}_j)$
$\quad$ // Find closest cluster for each point
10 $\quad$ **foreach** $\mathbf{x}_j \in \mathbf{D}$ **do**
11 $\quad\quad$ **foreach** $C_i \in \mathcal{C}^{t-1}$ **do**
12 $\quad\quad\quad d(\mathbf{x}_j, C_i) \leftarrow \text{sqnorm}_i - 2 \cdot \text{avg}_{ji}$
13 $\quad\quad j^* \leftarrow \arg\min_i \{d(\mathbf{x}_j, C_i)\}$
14 $\quad\quad C_{j^*}^t \leftarrow C_{j^*}^t \cup \{\mathbf{x}_j\}$ // Cluster reassignment
15 $\quad \mathcal{C}^t \leftarrow \{C_1^t, \ldots, C_k^t\}$
16 **until** $1 - \frac{1}{n} \sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}| \leq \epsilon$

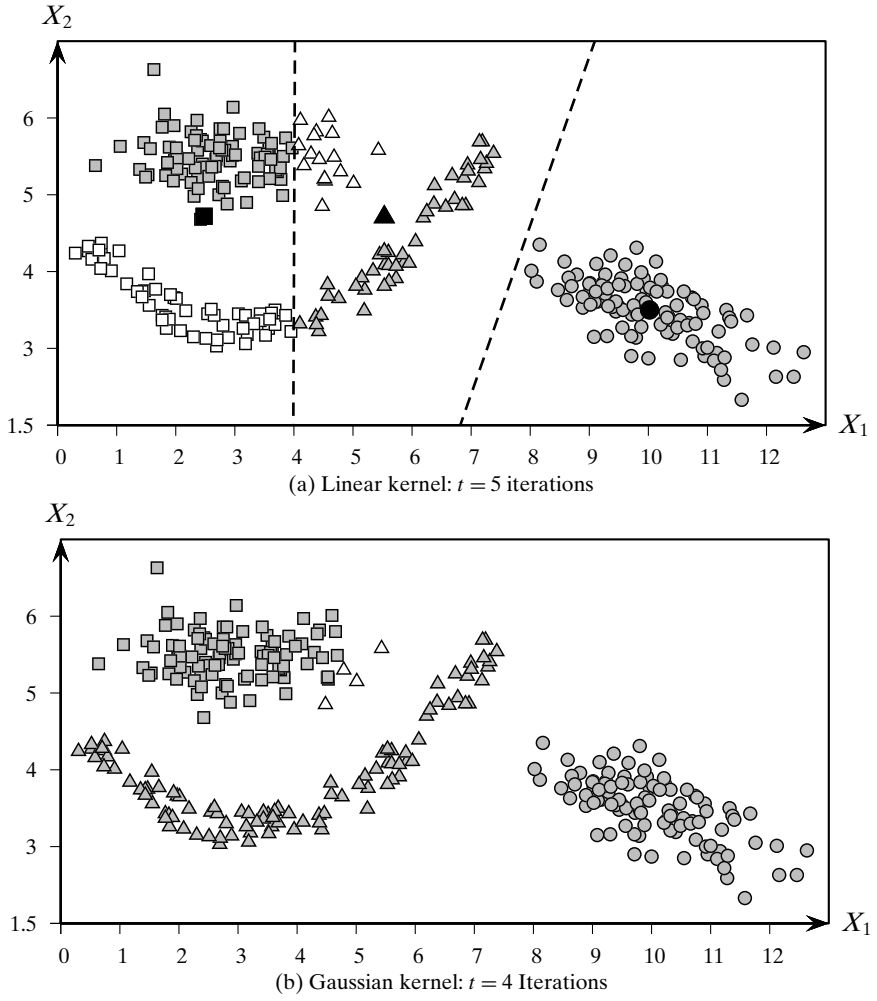


Figure 13.3. Kernel K-means: linear versus Gaussian kernel.

reassigned to a different cluster in the current iteration is given as

$$\frac{n - \sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}|}{n} = 1 - \frac{1}{n} \sum_{i=1}^{k} |C_i^t \cap C_i^{t-1}|$$

Kernel K-means stops when the fraction of points with new cluster assignments falls below some threshold $\epsilon \geq 0$. For example, one can iterate until no points change clusters.

Computational Complexity

Computing the average kernel value for each cluster C_i takes time $O(n^2)$ across all clusters. Computing the average kernel value of each point with respect to each of the k clusters also takes $O(n^2)$ time. Finally, computing the closest mean for each point and cluster reassignment takes $O(kn)$ time. The total computational complexity of kernel

K-means is thus $O(tn^2)$, where t is the number of iterations until convergence. The I/O complexity is $O(t)$ scans of the kernel matrix $\mathbf{K}$.

Example 13.3. Figure 13.3 shows an application of the kernel K-means approach on a synthetic dataset with three embedded clusters. Each cluster has 100 points, for a total of $n = 300$ points in the dataset.

Using the linear kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$ is equivalent to the K-means algorithm because in this case Eq. (13.5) is the same as Eq. (13.2). Figure 13.3a shows the resulting clusters; points in C_1 are shown as squares, in C_2 as triangles, and in C_3 as circles. We can see that K-means is not able to separate the three clusters due to the presence of the parabolic shaped cluster. The white points are those that are wrongly clustered, comparing with the ground truth in terms of the generated cluster labels.

Using the Gaussian kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left\{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right\}$ from Eq. (5.10), with $\sigma = 1.5$, results in a near-perfect clustering, as shown in Figure 13.3b. Only four points (white triangles) are grouped incorrectly with cluster C_2, whereas they should belong to cluster C_1. We can see from this example that kernel K-means is able to handle nonlinear cluster boundaries. One caveat is that the value of the spread parameter σ has to be set by trial and error.

13.3 EXPECTATION-MAXIMIZATION CLUSTERING

The K-means approach is an example of a *hard assignment* clustering, where each point can belong to only one cluster. We now generalize the approach to consider *soft assignment* of points to clusters, so that each point has a probability of belonging to each cluster.

Let $\mathbf{D}$ consist of n points $\mathbf{x}_j$ in d-dimensional space $\mathbb{R}^d$. Let X_a denote the random variable corresponding to the ath attribute. We also use X_a to denote the ath column vector, corresponding to the n data samples from X_a. Let $\mathbf{X} = (X_1, X_2, \ldots, X_d)$ denote the vector random variable across the d-attributes, with $\mathbf{x}_j$ being a data sample from $\mathbf{X}$.

Gaussian Mixture Model

We assume that each cluster C_i is characterized by a multivariate normal distribution, that is,

$$f_i(\mathbf{x}) = f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} \exp\left\{-\frac{(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)}{2}\right\} \tag{13.6}$$

where the cluster mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d \times d}$ are both unknown parameters. $f_i(\mathbf{x})$ is the probability density at $\mathbf{x}$ attributable to cluster C_i. We assume that the probability density function of $\mathbf{X}$ is given as a *Gaussian mixture model* over all

the k cluster normals, defined as

$$f(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x}) P(C_i) = \sum_{i=1}^{k} f(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i) \tag{13.7}$$

where the prior probabilities $P(C_i)$ are called the *mixture parameters*, which must satisfy the condition

$$\sum_{i=1}^{k} P(C_i) = 1$$

The Gaussian mixture model is thus characterized by the mean $\boldsymbol{\mu}_i$, the covariance matrix $\boldsymbol{\Sigma}_i$, and the mixture probability $P(C_i)$ for each of the k normal distributions. We write the set of all the model parameters compactly as

$$\boldsymbol{\theta} = \{\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1, P(C_i) \ldots, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, P(C_k)\}$$

Maximum Likelihood Estimation

Given the dataset $\mathbf{D}$, we define the *likelihood* of $\boldsymbol{\theta}$ as the conditional probability of the data $\mathbf{D}$ given the model parameters $\boldsymbol{\theta}$, denoted as $P(\mathbf{D}|\boldsymbol{\theta})$. Because each of the n points $\mathbf{x}_j$ is considered to be a random sample from $\mathbf{X}$ (i.e., independent and identically distributed as $\mathbf{X}$), the likelihood of $\boldsymbol{\theta}$ is given as

$$P(\mathbf{D}|\boldsymbol{\theta}) = \prod_{j=1}^{n} f(\mathbf{x}_j)$$

The goal of maximum likelihood estimation (MLE) is to choose the parameters $\boldsymbol{\theta}$ that maximize the likelihood, that is,

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{P(\mathbf{D}|\boldsymbol{\theta})\}$$

It is typical to maximize the log of the likelihood function because it turns the product over the points into a summation and the maximum value of the likelihood and log-likelihood coincide. That is, MLE maximizes

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \{\ln P(\mathbf{D}|\boldsymbol{\theta})\}$$

where the *log-likelihood* function is given as

$$\ln P(\mathbf{D}|\boldsymbol{\theta}) = \sum_{j=1}^{n} \ln f(\mathbf{x}_j) = \sum_{j=1}^{n} \ln\left(\sum_{i=1}^{k} f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)\right) \tag{13.8}$$

Directly maximizing the log-likelihood over $\boldsymbol{\theta}$ is hard. Instead, we can use the expectation-maximization (EM) approach for finding the maximum likelihood estimates for the parameters $\boldsymbol{\theta}$. EM is a two-step iterative approach that starts from an initial guess for the parameters $\boldsymbol{\theta}$. Given the current estimates for $\boldsymbol{\theta}$, in the *expectation step* EM computes the cluster posterior probabilities $P(C_i|\mathbf{x}_j)$ via the Bayes theorem:

$$P(C_i|\mathbf{x}_j) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(\mathbf{x}_j)} = \frac{P(\mathbf{x}_j|C_i)P(C_i)}{\sum_{a=1}^{k} P(\mathbf{x}_j|C_a)P(C_a)}$$

Because each cluster is modeled as a multivariate normal distribution [Eq. (13.6)], the probability of $\mathbf{x}_j$ given cluster C_i can be obtained by considering a small interval $\epsilon > 0$ centered at $\mathbf{x}_j$, as follows:

$$P(\mathbf{x}_j|C_i) \simeq 2\epsilon \cdot f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = 2\epsilon \cdot f_i(\mathbf{x}_j)$$

The posterior probability of C_i given $\mathbf{x}_j$ is thus given as

$$P(C_i|\mathbf{x}_j) = \frac{f_i(\mathbf{x}_j) \cdot P(C_i)}{\sum_{a=1}^{k} f_a(\mathbf{x}_j) \cdot P(C_a)} \tag{13.9}$$

and $P(C_i|\mathbf{x}_j)$ can be considered as the weight or contribution of the point $\mathbf{x}_j$ to cluster C_i. Next, in the *maximization step*, using the weights $P(C_i|\mathbf{x}_j)$ EM re-estimates $\boldsymbol{\theta}$, that is, it re-estimates the parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(C_i)$ for each cluster C_i. The re-estimated mean is given as the weighted average of all the points, the re-estimated covariance matrix is given as the weighted covariance over all pairs of dimensions, and the re-estimated prior probability for each cluster is given as the fraction of weights that contribute to that cluster. In Section 13.3.3 we formally derive the expressions for the MLE estimates for the cluster parameters, and in Section 13.3.4 we describe the generic EM approach in more detail. We begin with the application of the EM clustering algorithm for the one-dimensional and general d-dimensional cases.

13.3.1 EM in One Dimension

Consider a dataset $\mathbf{D}$ consisting of a single attribute X, where each point $x_j \in \mathbb{R}$ ($j = 1, \ldots, n$) is a random sample from X. For the mixture model [Eq. (13.7)], we use univariate normals for each cluster:

$$f_i(x) = f(x|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left\{-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right\}$$

with the cluster parameters μ_i, σ_i^2, and $P(C_i)$. The EM approach consists of three steps: initialization, expectation step, and maximization step.

Initialization

For each cluster C_i, with $i = 1, 2, \ldots, k$, we can randomly initialize the cluster parameters μ_i, σ_i^2, and $P(C_i)$. The mean μ_i is selected uniformly at random from the range of possible values for X. It is typical to assume that the initial variance is given as $\sigma_i^2 = 1$. Finally, the cluster prior probabilities are initialized to $P(C_i) = \frac{1}{k}$, so that each cluster has an equal probability.

Expectation Step

Assume that for each of the k clusters, we have an estimate for the parameters, namely the mean μ_i, variance σ_i^2, and prior probability $P(C_i)$. Given these values the clusters posterior probabilities are computed using Eq. (13.9):

$$P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2) \cdot P(C_i)}{\sum_{a=1}^{k} f(x_j|\mu_a, \sigma_a^2) \cdot P(C_a)}$$

For convenience, we use the notation $w_{ij} = P(C_i|x_j)$, treating the posterior probability as the weight or contribution of the point x_j to cluster C_i. Further, let

$$\mathbf{w}_i = (w_{i1}, \ldots, w_{in})^T$$

denote the weight vector for cluster C_i across all the n points.

Maximization Step

Assuming that all the posterior probability values or weights $w_{ij} = P(C_i|x_j)$ are known, the maximization step, as the name implies, computes the maximum likelihood estimates of the cluster parameters by re-estimating μ_i, σ_i^2, and $P(C_i)$.

The re-estimated value for the cluster mean, μ_i, is computed as the weighted mean of all the points:

$$\mu_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot x_j}{\sum_{j=1}^{n} w_{ij}}$$

In terms of the weight vector $\mathbf{w}_i$ and the attribute vector $X = (x_1, x_2, \ldots, x_n)^T$, we can rewrite the above as

$$\mu_i = \frac{\mathbf{w}_i^T X}{\mathbf{w}_i^T \mathbf{1}}$$

The re-estimated value of the cluster variance is computed as the weighted variance across all the points:

$$\sigma_i^2 = \frac{\sum_{j=1}^{n} w_{ij}(x_j - \mu_i)^2}{\sum_{j=1}^{n} w_{ij}}$$

Let $Z_i = X - \mu_i \mathbf{1} = (x_1 - \mu_i, x_2 - \mu_i, \ldots, x_n - \mu_i)^T = (z_{i1}, z_{i2}, \ldots, z_{in})^T$ be the centered attribute vector for cluster C_i, and let Z_i^s be the squared vector given as $Z_i^s = (z_{i1}^2, \ldots, z_{in}^2)^T$. The variance can be expressed compactly in terms of the dot product between the weight vector and the squared centered vector:

$$\sigma_i^2 = \frac{\mathbf{w}_i^T Z_i^s}{\mathbf{w}_i^T \mathbf{1}}$$

Finally, the prior probability of cluster C_i is re-estimated as the fraction of the total weight belonging to C_i, computed as

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{a=1}^{k} \sum_{j=1}^{n} w_{aj}} = \frac{\sum_{j=1}^{n} w_{ij}}{\sum_{j=1}^{n} 1} = \frac{\sum_{j=1}^{n} w_{ij}}{n} \tag{13.10}$$

where we made use of the fact that

$$\sum_{i=1}^{k} w_{ij} = \sum_{i=1}^{k} P(C_i|x_j) = 1$$

In vector notation the prior probability can be written as

$$P(C_i) = \frac{\mathbf{w}_i^T \mathbf{1}}{n}$$

Iteration

Starting from an initial set of values for the cluster parameters μ_i, σ_i^2 and $P(C_i)$ for all $i = 1, \ldots, k$, the EM algorithm applies the expectation step to compute the weights $w_{ij} = P(C_i|x_j)$. These values are then used in the maximization step to compute the updated cluster parameters μ_i, σ_i^2 and $P(C_i)$. Both the expectation and maximization steps are iteratively applied until convergence, for example, until the means change very little from one iteration to the next.

Example 13.4 (EM in 1D). Figure 13.4 illustrates the EM algorithm on the one-dimensional dataset:

$$x_1 = 1.0 \quad x_2 = 1.3 \quad x_3 = 2.2 \quad x_4 = 2.6 \quad x_5 = 2.8$$
$$x_6 = 5.0 \quad x_7 = 7.3 \quad x_8 = 7.4 \quad x_9 = 7.5 \quad x_{10} = 7.7 \quad x_{11} = 7.9$$

We assume that $k = 2$. The initial random means are shown in Figure 13.4a, with the initial parameters given as

$$\mu_1 = 6.63 \qquad \sigma_1^2 = 1 \qquad P(C_2) = 0.5$$
$$\mu_2 = 7.57 \qquad \sigma_2^2 = 1 \qquad P(C_2) = 0.5$$

After repeated expectation and maximization steps, the EM method converges after five iterations. After $t = 1$ (see Figure 13.4b) we have

$$\mu_1 = 3.72 \qquad \sigma_1^2 = 6.13 \qquad P(C_1) = 0.71$$
$$\mu_2 = 7.4 \qquad \sigma_2^2 = 0.69 \qquad P(C_2) = 0.29$$

After the final iteration ($t = 5$), as shown in Figure 13.4c, we have

$$\mu_1 = 2.48 \qquad \sigma_1^2 = 1.69 \qquad P(C_1) = 0.55$$
$$\mu_2 = 7.56 \qquad \sigma_2^2 = 0.05 \qquad P(C_2) = 0.45$$

One of the main advantages of the EM algorithm over K-means is that it returns the probability $P(C_i|\mathbf{x}_j)$ of each cluster C_i for each point $\mathbf{x}_j$. However, in this 1-dimensional example, these values are essentially binary; assigning each point to the cluster with the highest posterior probability, we obtain the hard clustering

$$C_1 = \{x_1, x_2, x_3, x_4, x_5, x_6\} \text{ (white points)}$$
$$C_2 = \{x_7, x_8, x_9, x_{10}, x_{11}\} \text{ (gray points)}$$

as illustrated in Figure 13.4c.

13.3.2 EM in d Dimensions

We now consider the EM method in d dimensions, where each cluster is characterized by a multivariate normal distribution [Eq. (13.6)], with parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(C_i)$. For each cluster C_i, we thus need to estimate the d-dimensional mean vector:

$$\boldsymbol{\mu}_i = (\mu_{i1}, \mu_{i2}, \ldots, \mu_{id})^T$$

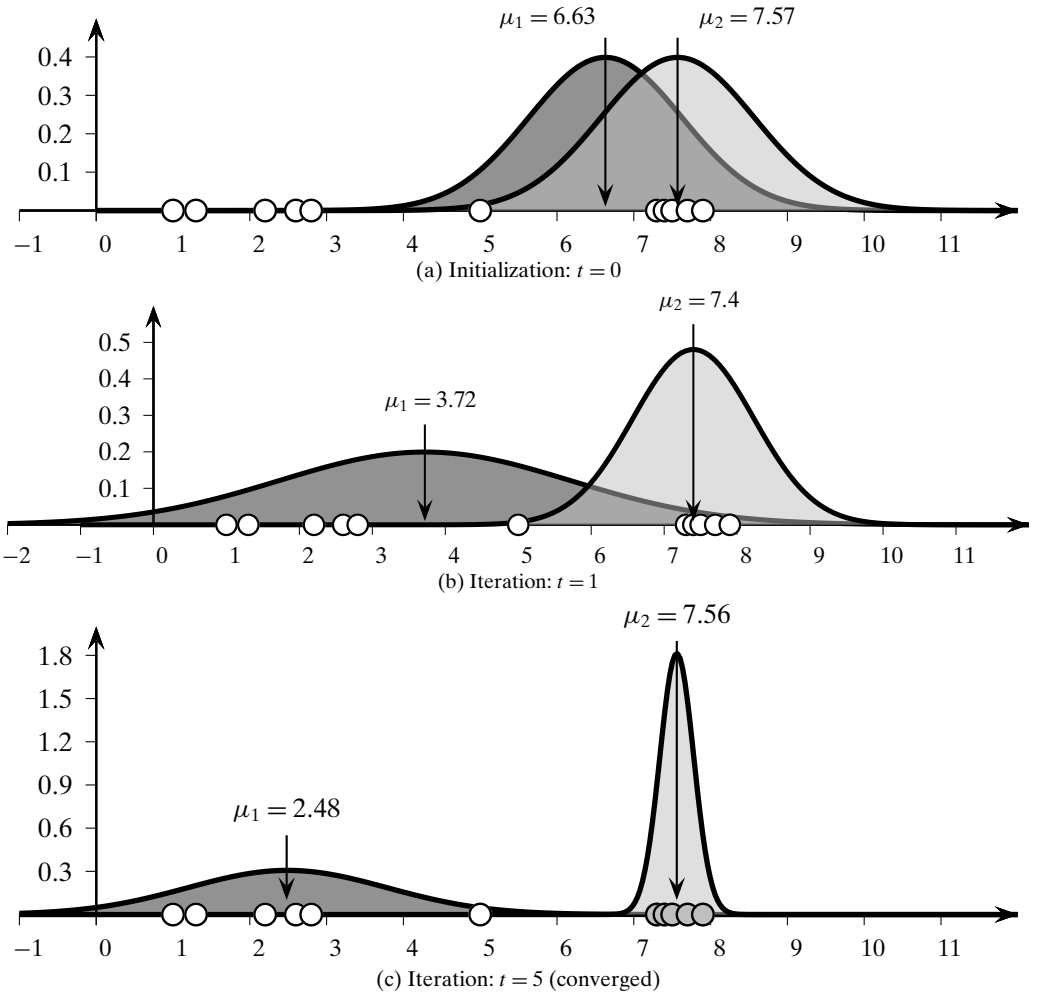


Figure 13.4. EM in one dimension.

and the $d \times d$ covariance matrix:

$$\boldsymbol{\Sigma}_i = \begin{pmatrix} (\sigma_1^i)^2 & \sigma_{12}^i & \ldots & \sigma_{1d}^i \\ \sigma_{21}^i & (\sigma_2^i)^2 & \ldots & \sigma_{2d}^i \\ \vdots & \vdots & \ddots & \\ \sigma_{d1}^i & \sigma_{d2}^i & \ldots & (\sigma_d^i)^2 \end{pmatrix}$$

Because the covariance matrix is symmetric, we have to estimate $\binom{d}{2} = \frac{d(d-1)}{2}$ pairwise covariances and d variances, for a total of $\frac{d(d+1)}{2}$ parameters for $\boldsymbol{\Sigma}_i$. This may be too many parameters for practical purposes because we may not have enough data to estimate all of them reliably. For example, if $d = 100$, then we have to estimate $100 \cdot 101/2 = 5050$ parameters! One simplification is to assume that all dimensions are

independent, which leads to a diagonal covariance matrix:

$$\boldsymbol{\Sigma}_i = \begin{pmatrix} (\sigma_1^i)^2 & 0 & \ldots & 0 \\ 0 & (\sigma_2^i)^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \ldots & (\sigma_d^i)^2 \end{pmatrix}$$

Under the independence assumption we have only d parameters to estimate for the diagonal covariance matrix.

Initialization

For each cluster C_i, with $i = 1, 2, \ldots, k$, we randomly initialize the mean $\boldsymbol{\mu}_i$ by selecting a value μ_{ia} for each dimension X_a uniformly at random from the range of X_a. The covariance matrix is initialized as the $d \times d$ identity matrix, $\boldsymbol{\Sigma}_i = \mathbf{I}$. Finally, the cluster prior probabilities are initialized to $P(C_i) = \frac{1}{k}$, so that each cluster has an equal probability.

Expectation Step

In the expectation step, we compute the posterior probability of cluster C_i given point $\mathbf{x}_j$ using Eq. (13.9), with $i = 1, \ldots, k$ and $j = 1, \ldots, n$. As before, we use the shorthand notation $w_{ij} = P(C_i|\mathbf{x}_j)$ to denote the fact that $P(C_i|\mathbf{x}_j)$ can be considered as the weight or contribution of point $\mathbf{x}_j$ to cluster C_i, and we use the notation $\mathbf{w}_i = (w_{i1}, w_{i2}, \ldots, w_{in})^T$ to denote the weight vector for cluster C_i, across all the n points.

Maximization Step

Given the weights w_{ij}, in the maximization step, we re-estimate $\boldsymbol{\Sigma}_i$, $\boldsymbol{\mu}_i$ and $P(C_i)$. The mean $\boldsymbol{\mu}_i$ for cluster C_i can be estimated as

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^n w_{ij} \cdot \mathbf{x}_j}{\sum_{j=1}^n w_{ij}} \tag{13.11}$$

which can be expressed compactly in matrix form as

$$\boldsymbol{\mu}_i = \frac{\mathbf{D}^T \mathbf{w}_i}{\mathbf{w}_i^T \mathbf{1}}$$

Let $\mathbf{Z}_i = \mathbf{D} - \mathbf{1} \cdot \boldsymbol{\mu}_i^T$ be the centered data matrix for cluster C_i. Let $\mathbf{z}_{ji} = \mathbf{x}_j - \boldsymbol{\mu}_i \in \mathbb{R}^d$ denote the jth centered point in $\mathbf{Z}_i$. We can express $\boldsymbol{\Sigma}_i$ compactly using the outer-product form

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^n w_{ij} \mathbf{z}_{ji} \mathbf{z}_{ji}^T}{\mathbf{w}_i^T \mathbf{1}} \tag{13.12}$$

Considering the pairwise attribute view, the covariance between dimensions X_a and X_b is estimated as

$$\sigma_{ab}^i = \frac{\sum_{j=1}^n w_{ij}(x_{ja} - \mu_{ia})(x_{jb} - \mu_{ib})}{\sum_{j=1}^n w_{ij}}$$

where x_{ja} and μ_{ia} denote the values of the ath dimension for $\mathbf{x}_j$ and $\boldsymbol{\mu}_i$, respectively.

Finally, the prior probability $P(C_i)$ for each cluster is the same as in the one-dimensional case [Eq. (13.10)], given as

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n} = \frac{\mathbf{w}_i^T \mathbf{1}}{n} \tag{13.13}$$

A formal derivation of these re-estimates for $\boldsymbol{\mu}_i$ [Eq. (13.11)], $\boldsymbol{\Sigma}_i$ [Eq. (13.12)], and $P(C_i)$ [Eq. (13.13)] is given in Section 13.3.3.

EM Clustering Algorithm

The pseudo-code for the multivariate EM clustering algorithm is given in Algorithm 13.3. After initialization of $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(C_i)$ for all $i = 1, \ldots, k$, the expectation and maximization steps are repeated until convergence. For the convergence test, we check whether $\sum_i \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$, where $\epsilon > 0$ is the convergence threshold, and t denotes the iteration. In words, the iterative process continues until the change in the cluster means becomes very small.

ALGORITHM 13.3. Expectation-Maximization (EM) Algorithm

EXPECTATION-MAXIMIZATION ($\mathbf{D}, k, \epsilon$):

1 $t \leftarrow 0$
// Initialization
2 Randomly initialize $\boldsymbol{\mu}_1^t, \ldots, \boldsymbol{\mu}_k^t$
3 $\boldsymbol{\Sigma}_i^t \leftarrow \mathbf{I}$, $\forall i = 1, \ldots, k$
4 $P^t(C_i) \leftarrow \frac{1}{k}$, $\forall i = 1, \ldots, k$
5 **repeat**
6 $t \leftarrow t + 1$
// Expectation Step
7 **for** $i = 1, \ldots, k$ *and* $j = 1, \ldots, n$ **do**
8 $w_{ij} \leftarrow \frac{f(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \cdot P(C_i)}{\sum_{a=1}^{k} f(\mathbf{x}_j | \boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a) \cdot P(C_a)}$ // posterior probability $P^t(C_i | \mathbf{x}_j)$
// Maximization Step
9 **for** $i = 1, \ldots, k$ **do**
10 $\boldsymbol{\mu}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} \cdot \mathbf{x}_j}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate mean
11 $\boldsymbol{\Sigma}_i^t \leftarrow \frac{\sum_{j=1}^{n} w_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} w_{ij}}$ // re-estimate covariance matrix
12 $P^t(C_i) \leftarrow \frac{\sum_{j=1}^{n} w_{ij}}{n}$ // re-estimate priors
13 **until** $\sum_{i=1}^{k} \left\| \boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1} \right\|^2 \leq \epsilon$

Example 13.5 (EM in 2D). Figure 13.5 illustrates the EM algorithm for the two-dimensional Iris dataset, where the two attributes are its first two principal components. The dataset consists of $n = 150$ points, and EM was run using $k = 3$, with full covariance matrix for each cluster. The initial cluster parameters are $\mathbf{\Sigma}_i = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $P(C_i) = 1/3$, with the means chosen as

$$\boldsymbol{\mu}_1 = (-3.59, 0.25)^T \qquad \boldsymbol{\mu}_2 = (-1.09, -0.46)^T \qquad \boldsymbol{\mu}_3 = (0.75, 1.07)^T$$

The cluster means (shown in black) and the joint probability density function are shown in Figure 13.5a.

The EM algorithm took 36 iterations to converge (using $\epsilon = 0.001$). An intermediate stage of the clustering is shown in Figure 13.5b, for $t = 1$. Finally at iteration $t = 36$, shown in Figure 13.5c, the three clusters have been correctly identified, with the following parameters:

$$\boldsymbol{\mu}_1 = (-2.02, 0.017)^T \qquad \boldsymbol{\mu}_2 = (-0.51, -0.23)^T \qquad \boldsymbol{\mu}_3 = (2.64, 0.19)^T$$

$$\mathbf{\Sigma}_1 = \begin{pmatrix} 0.56 & -0.29 \\ -0.29 & 0.23 \end{pmatrix} \qquad \mathbf{\Sigma}_2 = \begin{pmatrix} 0.36 & -0.22 \\ -0.22 & 0.19 \end{pmatrix} \qquad \mathbf{\Sigma}_3 = \begin{pmatrix} 0.05 & -0.06 \\ -0.06 & 0.21 \end{pmatrix}$$

$$P(C_1) = 0.36 \qquad P(C_2) = 0.31 \qquad P(C_3) = 0.33$$

To see the effect of a full versus diagonal covariance matrix, we ran the EM algorithm on the Iris principal components dataset under the independence assumption, which took $t = 29$ iterations to converge. The final cluster parameters were

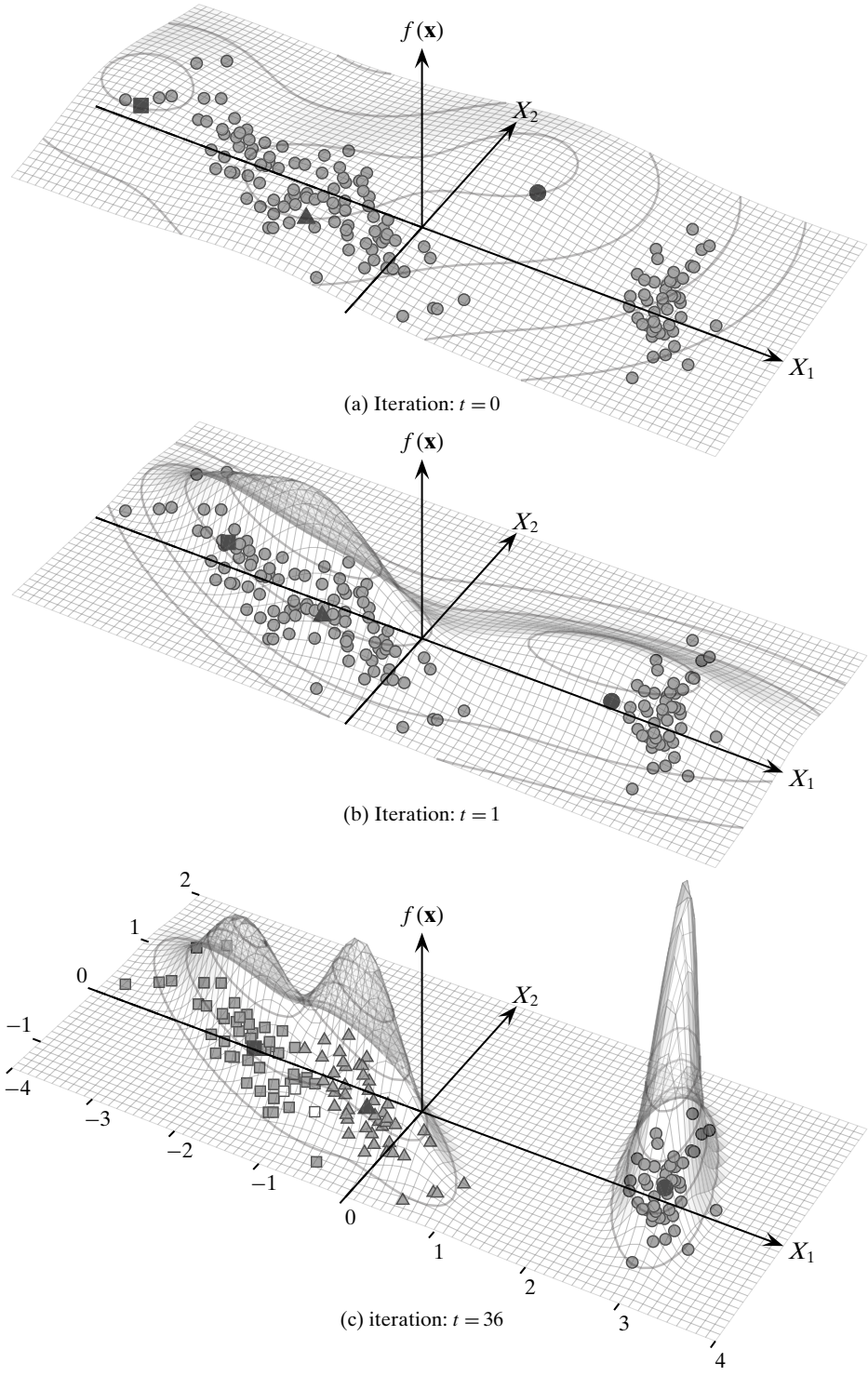
$$\boldsymbol{\mu}_1 = (-2.1, 0.28)^T \qquad \boldsymbol{\mu}_2 = (-0.67, -0.40)^T \qquad \boldsymbol{\mu}_3 = (2.64, 0.19)^T$$

$$\mathbf{\Sigma}_1 = \begin{pmatrix} 0.59 & 0 \\ 0 & 0.11 \end{pmatrix} \qquad \mathbf{\Sigma}_2 = \begin{pmatrix} 0.49 & 0 \\ 0 & 0.11 \end{pmatrix} \qquad \mathbf{\Sigma}_3 = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.21 \end{pmatrix}$$

$$P(C_1) = 0.30 \qquad P(C_2) = 0.37 \qquad P(C_3) = 0.33$$

Figure 13.6b shows the clustering results. Also shown are the contours of the normal density function for each cluster (plotted so that the contours do not intersect). The results for the full covariance matrix are shown in Figure 13.6a, which is a projection of Figure 13.5c onto the 2D plane. Points in C_1 are shown as squares, in C_2 as triangles, and in C_3 as circles.

One can observe that the diagonal assumption leads to axis parallel contours for the normal density, contrasted with the rotated contours for the full covariance matrix. The full matrix yields much better clustering, which can be observed by considering the number of points grouped with the wrong Iris type (the white points). For the full covariance matrix only three points are in the wrong group, whereas for the diagonal covariance matrix 25 points are in the wrong cluster, 15 from `iris-virginica` (white triangles) and 10 from `iris-versicolor` (white squares). The points corresponding to `iris-setosa` are correctly clustered as C_3 in both approaches.

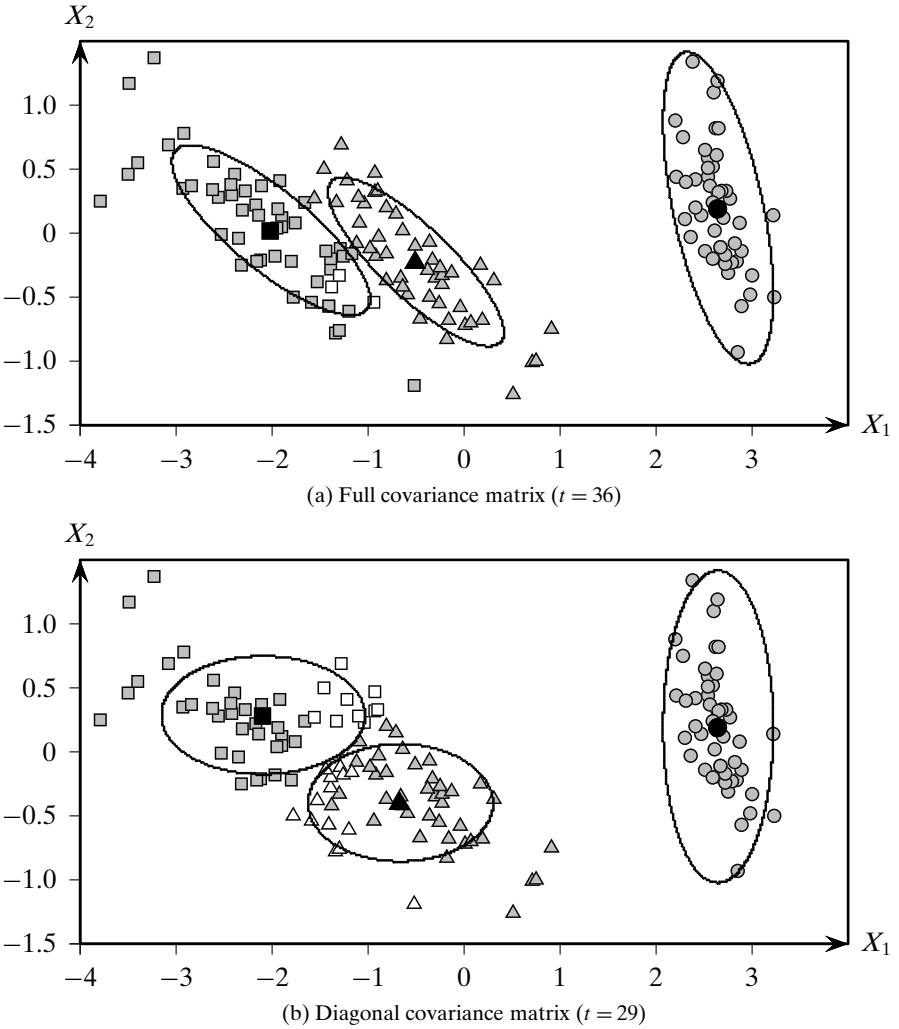


(a) Iteration: $t = 0$

(b) Iteration: $t = 1$

(c) iteration: $t = 36$

Figure 13.5. EM algorithm in two dimensions: mixture of $k = 3$ Gaussians.



(a) Full covariance matrix ($t = 36$)

(b) Diagonal covariance matrix ($t = 29$)

Figure 13.6. Iris principal components dataset: full versus diagonal covariance matrix.

Computational Complexity

For the expectation step, to compute the cluster posterior probabilities, we need to invert $\mathbf{\Sigma}_i$ and compute its determinant $|\mathbf{\Sigma}_i|$, which takes $O(d^3)$ time. Across the k clusters the time is $O(kd^3)$. For the expectation step, evaluating the density $f(\mathbf{x}_j|\boldsymbol{\mu}_i, \mathbf{\Sigma}_i)$ takes $O(d^2)$ time, for a total time of $O(knd^2)$ over the n points and k clusters. For the maximization step, the time is dominated by the update for $\mathbf{\Sigma}_i$, which takes $O(knd^2)$ time over all k clusters. The computational complexity of the EM method is thus $O(t(kd^3 + nkd^2))$, where t is the number of iterations. If we use a diagonal covariance matrix, then inverse and determinant of $\mathbf{\Sigma}_i$ can be computed in $O(d)$ time. Density computation per point takes $O(d)$ time, so that the time for the expectation step is $O(knd)$. The maximization step also takes $O(knd)$ time to re-estimate $\mathbf{\Sigma}_i$. The total time for a diagonal covariance matrix is therefore $O(tnkd)$. The I/O complexity for the

EM algorithm is $O(t)$ complete database scans because we read the entire set of points in each iteration.

K-means as Specialization of EM

Although we assumed a normal mixture model for the clusters, the EM approach can be applied with other models for the cluster density distribution $P(\mathbf{x}_j|C_i)$. For instance, K-means can be considered as a special case of the EM algorithm, obtained as follows:

$$P(\mathbf{x}_j|C_i) = \begin{cases} 1 & \text{if } C_i = \arg\min_{C_a} \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_a \right\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

Using Eq. (13.9), the posterior probability $P(C_i|\mathbf{x}_j)$ is given as

$$P(C_i|\mathbf{x}_j) = \frac{P(\mathbf{x}_j|C_i)P(C_i)}{\sum_{a=1}^{k} P(\mathbf{x}_j|C_a)P(C_a)}$$

One can see that if $P(\mathbf{x}_j|C_i) = 0$, then $P(C_i|\mathbf{x}_j) = 0$. Otherwise, if $P(\mathbf{x}_j|C_i) = 1$, then $P(\mathbf{x}_j|C_a) = 0$ for all $a \neq i$, and thus $P(C_i|\mathbf{x}_j) = \frac{1 \cdot P(C_i)}{1 \cdot P(C_i)} = 1$. Putting it all together, the posterior probability is given as

$$P(C_i|\mathbf{x}_j) = \begin{cases} 1 & \text{if } \mathbf{x}_j \in C_i, \text{i.e., if } C_i = \arg\min_{C_a} \left\{ \left\| \mathbf{x}_j - \boldsymbol{\mu}_a \right\|^2 \right\} \\ 0 & \text{otherwise} \end{cases}$$

It is clear that for K-means the cluster parameters are $\boldsymbol{\mu}_i$ and $P(C_i)$; we can ignore the covariance matrix.

13.3.3 Maximum Likelihood Estimation

In this section, we derive the maximum likelihood estimates for the cluster parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ and $P(C_i)$. We do this by taking the derivative of the log-likelihood function with respect to each of these parameters and setting the derivative to zero.

The partial derivative of the log-likelihood function [Eq. (13.8)] with respect to some parameter $\boldsymbol{\theta}_i$ for cluster C_i is given as

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_i} \ln\Big(P(\mathbf{D}|\boldsymbol{\theta})\Big) &= \frac{\partial}{\partial \boldsymbol{\theta}_i}\left(\sum_{j=1}^{n} \ln f(\mathbf{x}_j)\right) \\ &= \sum_{j=1}^{n}\left(\frac{1}{f(\mathbf{x}_j)} \cdot \frac{\partial f(\mathbf{x}_j)}{\partial \boldsymbol{\theta}_i}\right) \\ &= \sum_{j=1}^{n}\left(\frac{1}{f(\mathbf{x}_j)} \sum_{a=1}^{k} \frac{\partial}{\partial \boldsymbol{\theta}_i}\Big(f(\mathbf{x}_j|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)P(C_a)\Big)\right) \\ &= \sum_{j=1}^{n}\left(\frac{1}{f(\mathbf{x}_j)} \cdot \frac{\partial}{\partial \boldsymbol{\theta}_i}\Big(f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)\Big)\right) \end{aligned}$$

The last step follows from the fact that because $\boldsymbol{\theta}_i$ is a parameter for the ith cluster the mixture components for the other clusters are constants with respect to $\boldsymbol{\theta}_i$. Using the

fact that $|\boldsymbol{\Sigma}_i| = \frac{1}{|\boldsymbol{\Sigma}_i^{-1}|}$ the multivariate normal density in Eq. (13.6) can be written as

$$f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = (2\pi)^{-\frac{d}{2}} \, |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \, \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} \tag{13.14}$$

where

$$g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = -\frac{1}{2}(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_i) \tag{13.15}$$

Thus, the derivative of the log-likelihood function can be written as

$$\frac{\partial}{\partial \boldsymbol{\theta}_i} \ln\Big(P(\mathbf{D}|\boldsymbol{\theta})\Big) = \sum_{j=1}^{n} \left(\frac{1}{f(\mathbf{x}_j)} \cdot \frac{\partial}{\partial \boldsymbol{\theta}_i} \Big((2\pi)^{-\frac{d}{2}} \, |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} P(C_i) \Big) \right) \tag{13.16}$$

Below, we make use of the fact that

$$\frac{\partial}{\partial \boldsymbol{\theta}_i} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} = \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} \cdot \frac{\partial}{\partial \boldsymbol{\theta}_i} g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{13.17}$$

Estimation of Mean

To derive the maximum likelihood estimate for the mean $\boldsymbol{\mu}_i$, we have to take the derivative of the log-likelihood with respect to $\boldsymbol{\theta}_i = \boldsymbol{\mu}_i$. As per Eq. (13.16), the only term involving $\boldsymbol{\mu}_i$ is $\exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\}$. Using the fact that

$$\frac{\partial}{\partial \boldsymbol{\mu}_i} g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_i) \tag{13.18}$$

and making use of Eq. (13.17), the partial derivative of the log-likelihood [Eq. (13.16)] with respect to $\boldsymbol{\mu}_i$ is

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\mu}_i} \ln(P(\mathbf{D}|\boldsymbol{\theta})) &= \sum_{j=1}^{n} \left(\frac{1}{f(\mathbf{x}_j)} (2\pi)^{-\frac{d}{2}} \, |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} P(C_i) \; \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_i) \right) \\ &= \sum_{j=1}^{n} \left(\frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)}{f(\mathbf{x}_j)} \cdot \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_i) \right) \\ &= \sum_{j=1}^{n} w_{ij} \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_j - \boldsymbol{\mu}_i) \end{aligned}$$

where we made use of Eqs. (13.14) and (13.9), and the fact that

$$w_{ij} = P(C_i|\mathbf{x}_j) = \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(C_i)}{f(\mathbf{x}_j)}$$

Setting the partial derivative of the log-likelihood to the zero vector, and multiplying both sides by $\boldsymbol{\Sigma}_i$, we get

$$\sum_{j=1}^{n} w_{ij}(\mathbf{x}_j - \boldsymbol{\mu}_i) = \mathbf{0}, \text{ which implies that}$$

$$\sum_{j=1}^{n} w_{ij}\mathbf{x}_j = \boldsymbol{\mu}_i \sum_{j=1} w_{ij}, \text{ and therefore}$$

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} w_{ij}\mathbf{x}_j}{\sum_{j=1}^{n} w_{ij}} \tag{13.19}$$

which is precisely the re-estimation formula we used in Eq. (13.11).

Estimation of Covariance Matrix

To re-estimate the covariance matrix $\boldsymbol{\Sigma}_i$, we take the partial derivative of Eq. (13.16) with respect to $\boldsymbol{\Sigma}_i^{-1}$ using the product rule for the differentiation of the term $|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\}$.

Using the fact that for any square matrix $\mathbf{A}$, we have $\frac{\partial |\mathbf{A}|}{\partial \mathbf{A}} = |\mathbf{A}| \cdot (\mathbf{A}^{-1})^T$ the derivative of $|\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}}$ with respect to $\boldsymbol{\Sigma}_i^{-1}$ is

$$\frac{\partial |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}}}{\partial \boldsymbol{\Sigma}_i^{-1}} = \frac{1}{2} \cdot |\boldsymbol{\Sigma}_i^{-1}|^{-\frac{1}{2}} \cdot |\boldsymbol{\Sigma}_i^{-1}| \cdot \boldsymbol{\Sigma}_i = \frac{1}{2} \cdot |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \cdot \boldsymbol{\Sigma}_i \tag{13.20}$$

Next, using the fact that for the square matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ and vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, we have $\frac{\partial}{\partial \mathbf{A}} \mathbf{a}^T \mathbf{A} \mathbf{b} = \mathbf{a}\mathbf{b}^T$ the derivative of $\exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\}$ with respect to $\boldsymbol{\Sigma}_i^{-1}$ is obtained from Eq. (13.17) as follows:

$$\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} = -\frac{1}{2} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\}(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \tag{13.21}$$

Using the product rule on Eqs. (13.20) and (13.21), we get

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} \\
&\quad = \frac{1}{2} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \boldsymbol{\Sigma}_i \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} - \frac{1}{2} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\}(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \\
&\quad = \frac{1}{2} \cdot |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \cdot \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\}\Big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\Big)
\end{aligned} \tag{13.22}$$

Plugging Eq. (13.22) into Eq. (13.16) the derivative of the log-likelihood function with respect to $\boldsymbol{\Sigma}_i^{-1}$ is given as

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} \ln(P(\mathbf{D}|\boldsymbol{\theta})) \\
&\quad = \frac{1}{2} \sum_{j=1}^{n} \frac{(2\pi)^{-\frac{d}{2}} |\boldsymbol{\Sigma}_i^{-1}|^{\frac{1}{2}} \exp\big\{g(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big\} P(C_i)}{f(\mathbf{x}_j)} \Big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\Big)
\end{aligned}$$

$$= \frac{1}{2}\sum_{j=1}^{n} \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)}{f(\mathbf{x}_j)} \cdot \Big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\Big)$$

$$= \frac{1}{2}\sum_{j=1}^{n} w_{ij}\Big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\Big)$$

Setting the derivative to the $d \times d$ zero matrix $\mathbf{0}_{d\times d}$, we can solve for $\boldsymbol{\Sigma}_i$:

$$\sum_{j=1}^{n} w_{ij}\Big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\Big) = \mathbf{0}_{d\times d}, \text{which implies that}$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{n} w_{ij}(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} w_{ij}} \tag{13.23}$$

Thus, we can see that the maximum likelihood estimate for the covariance matrix is given as the weighted outer-product form in Eq. (13.12).

Estimating the Prior Probability: Mixture Parameters

To obtain a maximum likelihood estimate for the mixture parameters or the prior probabilities $P(C_i)$, we have to take the partial derivative of the log-likelihood [Eq. (13.16)] with respect to $P(C_i)$. However, we have to introduce a Lagrange multiplier α for the constraint that $\sum_{a=1}^{k} P(C_a) = 1$. We thus take the following derivative:

$$\frac{\partial}{\partial P(C_i)}\left(\ln(P(\mathbf{D}|\boldsymbol{\theta})) + \alpha\Big(\sum_{a=1}^{k} P(C_a) - 1\Big)\right) \tag{13.24}$$

The partial derivative of the log-likelihood in Eq. (13.16) with respect to $P(C_i)$ gives

$$\frac{\partial}{\partial P(C_i)} \ln(P(\mathbf{D}|\boldsymbol{\theta})) = \sum_{j=1}^{n} \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{f(\mathbf{x}_j)}$$

The derivative in Eq. (13.24) thus evaluates to

$$\left(\sum_{j=1}^{n} \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{f(\mathbf{x}_j)}\right) + \alpha$$

Setting the derivative to zero, and multiplying on both sides by $P(C_i)$, we get

$$\sum_{j=1}^{n} \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)}{f(\mathbf{x}_j)} = -\alpha P(C_i)$$

$$\sum_{j=1}^{n} w_{ij} = -\alpha P(C_i) \tag{13.25}$$

Taking the summation of Eq. (13.25) over all clusters yields

$$\sum_{i=1}^{k}\sum_{j=1}^{n} w_{ij} = -\alpha \sum_{i=1}^{k} P(C_i)$$

$$\text{or } n = -\alpha \tag{13.26}$$

The last step follows from the fact that $\sum_{i=1}^{k} w_{ij} = 1$. Plugging Eq. (13.26) into Eq. (13.25), gives us the maximum likelihood estimate for $P(C_i)$ as follows:

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n} \tag{13.27}$$

which matches the formula in Eq. (13.13).

We can see that all three parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, and $P(C_i)$ for cluster C_i depend on the weights w_{ij}, which correspond to the cluster posterior probabilities $P(C_i|\mathbf{x}_j)$. Equations (13.19), (13.23), and (13.27) thus do not represent a closed-form solution for maximizing the log-likelihood function. Instead, we use the iterative EM approach to compute the w_{ij} in the expectation step, and we then re-estimate $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ and $P(C_i)$ in the maximization step. Next, we describe the EM framework in some more detail.

13.3.4 EM Approach

Maximizing the log-likelihood function [Eq. (13.8)] directly is hard because the mixture term appears inside the logarithm. The problem is that for any point $\mathbf{x}_j$ we do not know which normal, or mixture component, it comes from. Suppose that we knew this information, that is, suppose each point $\mathbf{x}_j$ had an associated value indicating the cluster that generated the point. As we shall see, it is much easier to maximize the log-likelihood given this information.

The categorical attribute corresponding to the cluster label can be modeled as a vector random variable $\mathbf{C} = (C_1, C_2, \ldots, C_k)$, where C_i is a Bernoulli random variable (see Section 3.1.2 for details on how to model a categorical variable). If a given point is generated from cluster C_i, then $C_i = 1$, otherwise $C_i = 0$. The parameter $P(C_i)$ gives the probability $P(C_i = 1)$. Because each point can be generated from only one cluster, if $C_a = 1$ for a given point, then $C_i = 0$ for all $i \neq a$. It follows that $\sum_{i=1}^{k} P(C_i) = 1$.

For each point $\mathbf{x}_j$, let its cluster vector be $\mathbf{c}_j = (c_{j1}, \ldots, c_{jk})^T$. Only one component of $\mathbf{c}_j$ has value 1. If $c_{ji} = 1$, it means that $C_i = 1$, that is, the cluster C_i generates the point $\mathbf{x}_j$. The probability mass function of $\mathbf{C}$ is given as

$$P(\mathbf{C} = \mathbf{c}_j) = \prod_{i=1}^{k} P(C_i)^{c_{ji}}$$

Given the cluster information $\mathbf{c}_j$ for each point $\mathbf{x}_j$, the conditional probability density function for $\mathbf{X}$ is given as

$$f(\mathbf{x}_j|\mathbf{c}_j) = \prod_{i=1}^{k} f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)^{c_{ji}}$$

Only one cluster can generate $\mathbf{x}_j$, say C_a, in which case $c_{ja} = 1$, and the above expression would simplify to $f(\mathbf{x}_j|\mathbf{c}_j) = f(\mathbf{x}_j|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$.

The pair $(\mathbf{x}_j, \mathbf{c}_j)$ is a random sample drawn from the joint distribution of vector random variables $\mathbf{X} = (X_1, \ldots, X_d)$ and $\mathbf{C} = (C_1, \ldots, C_k)$, corresponding to the d data attributes and k cluster attributes. The joint density function of $\mathbf{X}$ and $\mathbf{C}$ is given as

$$f(\mathbf{x}_j \text{ and } \mathbf{c}_j) = f(\mathbf{x}_j|\mathbf{c}_j)P(\mathbf{c}_j) = \prod_{i=1}^{k}\Big(f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)\Big)^{c_{ji}}$$

The log-likelihood for the data given the cluster information is as follows:

$$\begin{aligned}
\ln P(\mathbf{D}|\boldsymbol{\theta}) &= \ln \prod_{j=1}^{n} f(\mathbf{x}_j \text{ and } \mathbf{c}_j|\boldsymbol{\theta}) \\
&= \sum_{j=1}^{n} \ln f(\mathbf{x}_j \text{ and } \mathbf{c}_j|\boldsymbol{\theta}) \\
&= \sum_{j=1}^{n} \ln\Big(\prod_{i=1}^{k}\Big(f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)\Big)^{c_{ji}}\Big) \\
&= \sum_{j=1}^{n}\sum_{i=1}^{k} c_{ji}\Big(\ln f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) + \ln P(C_i)\Big) \qquad (13.28)
\end{aligned}$$

Expectation Step

In the expectation step, we compute the expected value of the log-likelihood for the labeled data given in Eq. (13.28). The expectation is over the missing cluster information $\mathbf{c}_j$ treating $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$, $P(C_i)$, and $\mathbf{x}_j$ as fixed. Owing to the linearity of expectation, the expected value of the log-likelihood is given as

$$E[\ln P(\mathbf{D}|\boldsymbol{\theta})] = \sum_{j=1}^{n}\sum_{i=1}^{k} E[c_{ji}]\Big(\ln f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) + \ln P(C_i)\Big)$$

The expected value $E[c_{ji}]$ can be computed as

$$\begin{aligned}
E[c_{ji}] &= 1 \cdot P(c_{ji} = 1|\mathbf{x}_j) + 0 \cdot P(c_{ji} = 0|\mathbf{x}_j) = P(c_{ji} = 1|\mathbf{x}_j) = P(C_i|\mathbf{x}_j) \\
&= \frac{P(\mathbf{x}_j|C_i)P(C_i)}{P(\mathbf{x}_j)} = \frac{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)P(C_i)}{f(\mathbf{x}_j)} \\
&= w_{ij} \qquad (13.29)
\end{aligned}$$

Thus, in the expectation step we use the values of $\boldsymbol{\theta} = \{\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, P(C_i)\}_{i=1}^{k}$ to estimate the posterior probabilities or weights w_{ij} for each point for each cluster. Using $E[c_{ji}] = w_{ij}$, the expected value of the log-likelihood function can be rewritten as

$$E[\ln P(\mathbf{D}|\boldsymbol{\theta})] = \sum_{j=1}^{n}\sum_{i=1}^{k} w_{ij}\Big(\ln f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) + \ln P(C_i)\Big) \qquad (13.30)$$

Maximization Step

In the maximization step, we maximize the expected value of the log-likelihood [Eq. (13.30)]. Taking the derivative with respect to $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ or $P(C_i)$ we can ignore the terms for all the other clusters.

The derivative of Eq. (13.30) with respect to $\boldsymbol{\mu}_i$ is given as

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}_i} \ln E[P(\mathbf{D}|\boldsymbol{\theta})] &= \frac{\partial}{\partial \boldsymbol{\mu}_i} \sum_{j=1}^{n} w_{ij} \ln f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \\
&= \sum_{j=1}^{n} w_{ij} \cdot \frac{1}{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \frac{\partial}{\partial \boldsymbol{\mu}_i} f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \\
&= \sum_{j=1}^{n} w_{ij} \cdot \frac{1}{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \cdot f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\ \boldsymbol{\Sigma}_i^{-1}\ (\mathbf{x}_j - \boldsymbol{\mu}_i) \\
&= \sum_{j=1}^{n} w_{ij}\ \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_i)
\end{aligned}$$

where we used the observation that

$$\frac{\partial}{\partial \boldsymbol{\mu}_i} f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\ \boldsymbol{\Sigma}_i^{-1}\ (\mathbf{x}_j - \boldsymbol{\mu}_i)$$

which follows from Eqs. (13.14), (13.17), and (13.18). Setting the derivative of the expected value of the log-likelihood to the zero vector, and multiplying on both sides by $\boldsymbol{\Sigma}_i$, we get

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{n} w_{ij} \mathbf{x}_j}{\sum_{j=1}^{n} w_{ij}}$$

matching the formula in Eq. (13.11).

Making use of Eqs. (13.22) and (13.14), we obtain the derivative of Eq. (13.30) with respect to $\boldsymbol{\Sigma}_i^{-1}$ as follows:

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\Sigma}_i^{-1}} \ln E[P(\mathbf{D}|\boldsymbol{\theta})] \\
&\quad = \sum_{j=1}^{n} w_{ij} \cdot \frac{1}{f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \cdot \frac{1}{2}\ f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\big) \\
&\quad = \frac{1}{2} \sum_{j=1}^{n} w_{ij} \cdot \big(\boldsymbol{\Sigma}_i - (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T\big)
\end{aligned}$$

Setting the derivative to the $d \times d$ zero matrix and solving for $\boldsymbol{\Sigma}_i$ yields

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{n} w_{ij} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T}{\sum_{j=1}^{n} w_{ij}}$$

which is the same as that in Eq. (13.12).

Using the Lagrange multiplier α for the constraint $\sum_{i=1}^{k} P(C_i) = 1$, and noting that in the log-likelihood function [Eq. (13.30)], the term $\ln f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is a constant with respect to $P(C_i)$, we obtain the following:

$$\frac{\partial}{\partial P(C_i)}\left(\ln E[P(\mathbf{D}|\boldsymbol{\theta})] + \alpha\left(\sum_{i=1}^{k} P(C_i) - 1\right)\right) = \frac{\partial}{\partial P(C_i)}\Big(w_{ij} \ln P(C_i) + \alpha P(C_i)\Big)$$

$$= \left(\sum_{j=1}^{n} w_{ij} \cdot \frac{1}{P(C_i)}\right) + \alpha$$

Setting the derivative to zero, we get

$$\sum_{j=1}^{n} w_{ij} = -\alpha \cdot P(C_i)$$

Using the same derivation as in Eq. (13.26) we obtain

$$P(C_i) = \frac{\sum_{j=1}^{n} w_{ij}}{n}$$

which is identical to the re-estimation formula in Eq. (13.13).

13.4 FURTHER READING

The K-means algorithm was proposed in several contexts during the 1950s and 1960s; among the first works to develop the method include MacQueen (1967); Lloyd (1982) and Hartigan (1975). Kernel k-means was first proposed in Schölkopf, Smola, and Müller (1996). The EM algorithm was proposed in Dempster, Laird, and Rubin (1977). A good review on EM method can be found in McLachlan and Krishnan (2008). For a scalable and incremental representative-based clustering method that can also generate hierarchical clusterings see Zhang, Ramakrishnan, and Livny (1996).

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society, Series B*, 39 (1): 1–38.

Hartigan, J. A. (1975). *Clustering Algorithms*. New York: John Wiley & Sons.

Lloyd, S. (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, 28 (2): 129–137.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations." *In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, University of California Press, Berkeley.

McLachlan, G. and Krishnan, T. (2008). *The EM Algorithm and Extensions,* 2nd ed. Hoboken, NJ: John Wiley & Sons.

Schölkopf, B., Smola, A., and Müller, K.-R. (1996). *Nonlinear component analysis as a kernel eigenvalue problem*. Technical Report No. 44. Tübingen, Germany: Max-Planck-Institut für biologische Kybernetik.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). "BIRCH: an efficient data clustering method for very large databases." *ACM SIGMOD Record*, 25 (2): 103–114.

13.5 EXERCISES

Q1. Given the following points: $2, 4, 10, 12, 3, 20, 30, 11, 25$. Assume $k = 3$, and that we randomly pick the initial means $\mu_1 = 2$, $\mu_2 = 4$ and $\mu_3 = 6$. Show the clusters obtained using K-means algorithm after one iteration, and show the new means for the next iteration.

Table 13.1. Dataset for Q2

x	$P(C_1\|x)$	$P(C_2\|x)$
2	0.9	0.1
3	0.8	0.1
7	0.3	0.7
9	0.1	0.9
2	0.9	0.1
1	0.8	0.2

Q2. Given the data points in Table 13.1, and their probability of belonging to two clusters. Assume that these points were produced by a mixture of two univariate normal distributions. Answer the following questions:

(a) Find the maximum likelihood estimate of the means μ_1 and μ_2.

(b) Assume that $\mu_1 = 2$, $\mu_2 = 7$, and $\sigma_1 = \sigma_2 = 1$. Find the probability that the point $x = 5$ belongs to cluster C_1 and to cluster C_2. You may assume that the prior probability of each cluster is equal (i.e., $P(C_1) = P(C_2) = 0.5$), and the prior probability $P(x = 5) = 0.029$.

Table 13.2. Dataset for Q3

	X_1	X_2
$\mathbf{x}_1$	0	2
$\mathbf{x}_2$	0	0
$\mathbf{x}_3$	1.5	0
$\mathbf{x}_4$	5	0
$\mathbf{x}_5$	5	2

Q3. Given the two-dimensional points in Table 13.2, assume that $k = 2$, and that initially the points are assigned to clusters as follows: $C_1 = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4\}$ and $C_2 = \{\mathbf{x}_3, \mathbf{x}_5\}$. Answer the following questions:

(a) Apply the K-means algorithm until convergence, that is, the clusters do not change, assuming (1) the usual Euclidean distance or the L_2-*norm* as the distance

between points, defined as $\|\mathbf{x}_i - \mathbf{x}_j\|_2 = \left(\sum_{a=1}^{d}(x_{ia} - x_{ja})^2\right)^{1/2}$, and (2) the Manhattan distance or the L_1-*norm* defined as $\|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{a=1}^{d}|x_{ia} - x_{ja}|$.

(b) Apply the EM algorithm with $k = 2$ assuming that the dimensions are independent. Show one complete execution of the expectation and the maximization steps. Start with the assumption that $P(C_i|x_{ja}) = 0.5$ for $a = 1, 2$ and $j = 1, \ldots, 5$.

Q4. Given the categorical database in Table 13.3. Find $k = 2$ clusters in this data using the EM method. Assume that each attribute is independent, and that the domain of each attribute is $\{A, C, T\}$. Initially assume that the points are partitioned as follows: $C_1 = \{\mathbf{x}_1, \mathbf{x}_4\}$, and $C_2 = \{\mathbf{x}_2, \mathbf{x}_3\}$. Assume that $P(C_1) = P(C_2) = 0.5$.

Table 13.3. Dataset for Q4

	X_1	X_2
$\mathbf{x}_1$	A	T
$\mathbf{x}_2$	A	A
$\mathbf{x}_3$	C	C
$\mathbf{x}_4$	A	C

The probability of an attribute value given a cluster is given as

$$P(x_{ja}|C_i) = \frac{\text{No. of times the symbol } x_{ja} \text{ occurs in cluster } C_i}{\text{No. of objects in cluster } C_i}$$

for $a = 1, 2$. The probability of a point given a cluster is then given as

$$P(\mathbf{x}_j|C_i) = \prod_{a=1}^{2} P(x_{ja}|C_i)$$

Instead of computing the mean for each cluster, generate a partition of the objects by doing a hard assignment. That is, in the expectation step compute $P(C_i|\mathbf{x}_j)$, and in the maximization step assign the point $\mathbf{x}_j$ to the cluster with the largest $P(C_i|\mathbf{x}_j)$ value, which gives a new partitioning of the points. Show one full iteration of the EM algorithm and show the resulting clusters.

Table 13.4. Dataset for Q5

	X_1	X_2	X_3
$\mathbf{x}_1$	0.5	4.5	2.5
$\mathbf{x}_2$	2.2	1.5	0.1
$\mathbf{x}_3$	3.9	3.5	1.1
$\mathbf{x}_4$	2.1	1.9	4.9
$\mathbf{x}_5$	0.5	3.2	1.2
$\mathbf{x}_6$	0.8	4.3	2.6
$\mathbf{x}_7$	2.7	1.1	3.1
$\mathbf{x}_8$	2.5	3.5	2.8
$\mathbf{x}_9$	2.8	3.9	1.5
$\mathbf{x}_{10}$	0.1	4.1	2.9

Q5. Given the points in Table 13.4, assume that there are two clusters: C_1 and C_2, with $\boldsymbol{\mu}_1 = (0.5, 4.5, 2.5)^T$ and $\boldsymbol{\mu}_2 = (2.5, 2, 1.5)^T$. Initially assign each point to the closest mean, and compute the covariance matrices $\boldsymbol{\Sigma}_i$ and the prior probabilities $P(C_i)$ for $i = 1, 2$. Next, answer which cluster is more likely to have produced $\mathbf{x}_8$?

Q6. Consider the data in Table 13.5. Answer the following questions:

(a) Compute the kernel matrix $\mathbf{K}$ between the points assuming the following kernel:

$$K(\mathbf{x}_i, \mathbf{x}_j) = 1 + \mathbf{x}_i^T \mathbf{x}_j$$

(b) Assume initial cluster assignments of $C_1 = \{\mathbf{x}_1, \mathbf{x}_2\}$ and $C_2 = \{\mathbf{x}_3, \mathbf{x}_4\}$. Using kernel K-means, which cluster should $\mathbf{x}_1$ belong to in the next step?

Table 13.5. Data for Q6

	X_1	X_2	X_3
$\mathbf{x}_1$	0.4	0.9	0.6
$\mathbf{x}_2$	0.5	0.1	0.6
$\mathbf{x}_3$	0.6	0.3	0.6
$\mathbf{x}_4$	0.4	0.8	0.5

Q7. Prove the following equivalence for the multivariate normal density function:

$$\frac{\partial}{\partial \boldsymbol{\mu}_i} f(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = f(\mathbf{x}_j | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\, \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_i)$$