Chapter 12

Representative-based Clustering

Given a dataset with n points in a d-dimensional space, and k, the number of clusters to discover, our goal is to partition the points into k groups. Each such partitioning of the points into k groups is called a *clustering*. We denote a clustering with k clusters as $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$. We now need some scoring function that evaluates the goodness of a clustering.

In representative-based clustering we designate one point as the representative of a cluster. One common representative is the mean point of a cluster, which is also called the *centroid* of the cluster. A common scoring function is based on the squared deviations of the points within a cluster from their centroid. This *sum of squared errors* based scoring function is defined as:

$$SSE(\mathcal{C}) = \sum_{i=1}^{k} \sum_{j=1}^{|C_i|} (\mathbf{x}_j - \boldsymbol{\mu}_i)^2 \tag{12.1}$$

where $\boldsymbol{\mu}_i$ is the centroid of cluster C_i, given as: $\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \mathbf{x}_j$.

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

$$\mathcal{C}^* = \arg\min\{SSE(\mathcal{C}_i)\} \tag{12.2}$$

Naive Solution: One naive algorithm for finding a good clustering is as follows:

1. Generate all the possible partitions of n points into k clusters.
2. Rank the clusterings based on the SSE score.
3. Output the clustering with the best (least) score.

Let us consider how many clusterings there are. The simplest way to derive this number is to observe that each point can be assigned to any one of the k clusters. Thus there are $O(k^n)$ possible clusterings. Thus the brute-force naive algorithm is clearly infeasible from a computational viewpoint.

12.1 K-means Clustering Algorithm

K-means is a greedy search algorithm for good clusterings. As it is a heuristic method, it can converge to a locally optimum score instead of the globally optimum clustering guaranteed by the brute-force algorithm.

The basic idea behind K-means is to randomly initialize cluster representatives, by selecting a random set of k points as the centroids. The method then refines these initial centroids by assigning the points to the closest centroid. New centroids can be computed once we know which points belong to a cluster. The cluster assignment and new centroid computation steps are carried out iteratively until we reach a fixed point. The complete K-means algorithm is described as follows:

1. Randomly initialize k centroids: $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k$
2. Cluster Assignment Step:
   For all $\mathbf{x}_j \in \mathcal{D}$
   a. For all $i \in [1,k]$, compute $\delta(\mathbf{x}_j, \boldsymbol{\mu}_i)$
   b. Assign $\mathbf{x}_j$ to closest centroid, $C_i^* = \arg\min\{\delta(\mathbf{x}_j, \boldsymbol{\mu}_i)\}$
3. Recompute the means/centroids
   For all $i \in [1,k]$, $\boldsymbol{\mu}_i' = \frac{1}{|C_i|}\sum_{j=1}^{|C_i|} \mathbf{x}_j$
4. Convergence Condition:
   If $\sum_i |\boldsymbol{\mu}_i' - \boldsymbol{\mu}_i| < \varepsilon$ then stop, else go to (2).

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

Example: As an example of how K-means works, assume we have the one-dimensional data as show in the Figure 12.1.

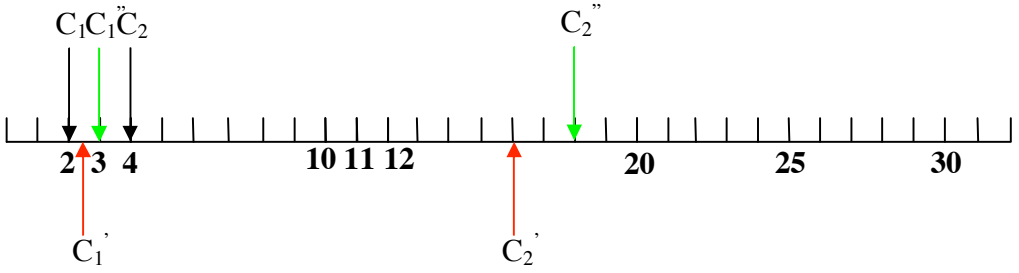

Figure 12.1: Example Data for K-means

(a)

centroids	clusters	new centroids
$\boldsymbol{\mu}_1 = 2$ $\boldsymbol{\mu}_2 = 4$	2,3 4,10,11,12,20,25,30	$\boldsymbol{\mu}_1' = \frac{5}{2} = 2.5$ $\boldsymbol{\mu}_1' = \frac{112}{7} = 16$

(b)

centroids	clusters	new centroids
$\boldsymbol{\mu}_1 = 2.5$ $\boldsymbol{\mu}_2 = 16$	2,3,4 10,11,12,20,25,30	$\boldsymbol{\mu}_1' = \frac{9}{3} = 3$ $\boldsymbol{\mu}_1' = \frac{108}{6} = 18$

Table 12.1: Cluster Assignment and Centroids Recomputation: 2 Iterations

Suppose that the data is to be clustered into two groups ($k = 2$), and suppose that the initial centroids are chosen to be $\boldsymbol{\mu}_1 = 2$ and $\boldsymbol{\mu}_2 = 4$. The cluster assignment is shown in Table 12.1(a). With the new centroids

$\boldsymbol{\mu}_1' = 2.5$ and $\boldsymbol{\mu}_2' = 16$, we repeat the cluster assignment and centroids recomputation steps, as shown in Table 12.1(b). We repeat this process until the convergence criteria is met, i.e., when the centroids change very little from one iteration to the next.

12.2 Expectation Maximization (EM) Clustering

In statistical computing, Expectation-Maximization (EM) is a common approach for finding the maximum likelihood estimates of parameters of a probabilistic model. The EM method for clustering is a natural extension of the K-Means method. Whereas in K-means a point can belong to only one cluster, i.e., a *hard assignment*, in EM we allow each point to belong to a cluster with some probability, i.e., a *soft assignment*. In other words, for K-means the probability of a point belong to a cluster $P(\mathbf{x}|C)$ is either 0 or 1, whereas in EM, it can take any value in $[0,1]$.

Bayes Theorem Given some probability model for the clusters, and given a dataset of n point in d-dimensions, along with the number of clusters to mine, k, the goal in EM method is to compute the probability of each cluster C_i based on the points $\mathbf{x}_j \in \mathcal{D}$:

$$P(C_i|\mathbf{x}_j) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(\mathbf{x}_j)} \tag{12.3}$$

Note also that

$$P(\mathbf{x}_j|C_i) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(C_i)} \implies P(C_i \text{ and } \mathbf{x}_j) = P(\mathbf{x}_j|C_i)P(C_i) \tag{12.4}$$

Plugging this into (12.3) we get

$$P(C_i|\mathbf{x}_j) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(\mathbf{x}_j)} = \frac{P(\mathbf{x}_j|C_i)P(C_i)}{P(\mathbf{x}_j)} \tag{12.5}$$

Now, since a given point $\mathbf{x}_i$ may belong to any one of the k clusters, which are mutually exclusive events, we get:

$$P(\mathbf{x}_j) = \sum_{i=1}^{k} P(C_i \text{ and } \mathbf{x}_j) = \sum_{i=1}^{k} P(\mathbf{x}_j|C_i)P(C_i) \tag{12.6}$$

Putting it all together, we have the classic statement of the Bayes Theorem:

$$\boxed{P(C_i|\mathbf{x}_j) = \frac{P(C_i \text{ and } \mathbf{x}_j)}{P(\mathbf{x}_j)} = \frac{P(\mathbf{x}_j|C_i)P(C_i)}{\sum_{a=1}^{k} P(\mathbf{x}_j|C_a)P(C_a)}} \tag{12.7}$$

The Bayes theorem can also be stated as follows:

$$\boxed{\textbf{Posterior Probability:}P(C_i|\mathbf{x}_j) = \frac{\textbf{Likelihood:}P(\mathbf{x}_j|C_i) \times \textbf{Prior Probability:}P(C_i)}{\textbf{Normalization Term:}P(\mathbf{x}_j)}} \tag{12.8}$$

In other words, the Bayes theorem states that we can find the posterior probability of a cluster, given a point, which is something we do not know, in terms of the likelihood of the point given the cluster and the prior probability of the cluster, which is something we can compute, given the underlying probability model on the clusters. For example, a common assumption is that each cluster follows a normal distribution, and that the data points have been generated from a *mixture model* of k normal distributions. Below we will see how the EM method works, first in one dimension, and then in d-dimensions.

12.2.1 EM in 1-Dimension

Let us assume that the dataset has been generated from a mixture of $k = 2$ normal distributions, $f(x|\mu_1, \sigma_1^2)$ and $f(x|\mu_2, \sigma_2^2)$. These two distributions represent the two clusters we want to find. However, all we see is the set of data points $\mathcal{D}$ generated from the mixture, and we do not know the values of the two cluster parameters: μ_1, σ_1^2 and μ_2, σ_2^2. Our goal is to estimate these parameters via the EM approach, which consist of three steps: initialization, expectation and maximization.

Initialization: For each cluster C_i, with $i = 1, 2, \cdots, k$, randomly initialize μ_i and σ_i^2. Also initialize $P(C_i) = \frac{1}{k}$.

Expectation Step: Given the parameters of k clusters, namely $\mu_i, \sigma_i^2, P(C_i)$, compute the likelihood $P(x_j|C_i)$ for all points $x_j \in \mathcal{D}$ where $j = 1, 2, \cdots, n$ and all clusters $C_i, i = 1, 2, \cdots, k$. Under the normal distribution for each cluster, we have

$$P(x_j|C_i) = f(x_j|\mu_i, \sigma_i^2) = \frac{1}{\sqrt{2\pi}\,\sigma_i} e^{-\frac{(x_j-\mu_i)^2}{2\sigma_i^2}} \tag{12.9}$$

Next we use the likelihood $P(x_j|C_i)$ to compute $P(C_i|x_j)$ using the Bayes theorem:

$$P(C_i|x_j) = \frac{f(x_j|\mu_i, \sigma_i^2) \cdot P(C_i)}{\Sigma_{a=1}^{k} f(x_j|\mu_a, \sigma_a^2) \cdot P(C_a)} \tag{12.10}$$

Maximization Step: Given all the values posterior probabilities $P(C_i|x_j)$, maximize the cluster parameters by re-estimating $\mu_i', \sigma_i'^2, P'(C_i)$ as follows. For convenience, let $w_i(x_j) = P(C_i|x_j)$ denote the *weight* of a point for the cluster C_i.

We compute the new μ_i' as the weighted mean of all the points x_j:

$$\mu_i' = \frac{\sum_{j=1}^{n} w_i(x_j) \cdot x_j}{\sum_{j=1}^{n} w_i(x_j)} \tag{12.11}$$

and $(\sigma_i'^2)$ as the weighted variance across all the points:

$$\sigma_i'^2 = \frac{\Sigma_{j=1}^{n} w_i(x_j)(x_j - \mu_i)^2}{\Sigma_{j=1}^{n} w_i(x_j)} \tag{12.12}$$

and finally we compute $P'(C_i)$ as the fraction of the total weight belonging to a given cluster:

$$P'(C_i) = \frac{\Sigma_{j=1}^{n} w_i(x_j)}{\Sigma_{a=1}^{k} \Sigma_{j=1}^{n} w_a(x_j)} \tag{12.13}$$

Now that we have updated estimates for the cluster parameters and prior probabilities, we can repeat the expectation and maximization steps until convergence, for example, until the means change very little from one iteration to the next. Note also that although we assumed a normal mixture model for the clusters, the EM approach can be applied for other models as well, by replacing the way we compute $P(x_j|C_i)$.

K-Mean as specialization of EM: We can obtain the K-means algorithm by replacing the likelihood function as follows:

$$P(x_j|C_i) = \begin{cases} 1 & C_i = \arg\min_a\{\delta(x_j,\mu_a)\} \\ 0 & \text{otherwise} \end{cases} \tag{12.14}$$

12.3 EM in d-Dimensions

If we have a d-dimensional dataset, we have to generalize the EM method to the d-dimensional case. If we assume that the data is produced from a mixture of d-dimensional normal distributions/clusters, we need to estimate for each cluster C_i, the d-dimensional mean vector:

$$\boldsymbol{\mu}_i = (\mu_i^1, \mu_i^2, \cdots, \mu_i^d) \tag{12.15}$$

and the $d \times d$ covariance matrix:

$$\boldsymbol{\Sigma}_i = \begin{pmatrix} \sigma^i_{\mathbf{x}^1\mathbf{x}^1} & \sigma^i_{\mathbf{x}^1\mathbf{x}^2} & \cdots & \sigma^i_{\mathbf{x}^1\mathbf{x}^d} \\ \sigma^i_{\mathbf{x}^2\mathbf{x}^1} & \sigma^i_{\mathbf{x}^2\mathbf{x}^2} & \cdots & \sigma^i_{\mathbf{x}^2\mathbf{x}^d} \\ \vdots & \vdots & \ddots & \\ \sigma^i_{\mathbf{x}^d\mathbf{x}^1} & \sigma^i_{\mathbf{x}^d\mathbf{x}^2} & \cdots & \sigma^i_{\mathbf{x}^d\mathbf{x}^d} \end{pmatrix} \tag{12.16}$$

The total number of unknown variables per cluster is d for the mean and $\frac{d(d-1)}{2}$ for the covariance matrix $\boldsymbol{\Sigma}_i$ (since it is symmetric). Thus altogether there are $\frac{d(d+1)}{2} = O(d^2)$ parameters to estimate, which is too large for practical purposes. For example, if $d = 100$, then we have to estimate $100 * 101/2 = 5050$ parameters! This is a very difficult task, and we may not have enough data to estimate all of these reliably.

In order to apply EM approach in practice, we can assume that all dimensions are independent, and thus all covariances can be set to zero:

$$\boldsymbol{\Sigma}_i = \begin{pmatrix} \sigma^i_{\mathbf{x}^1\mathbf{x}^1} & 0 & \cdots & 0 \\ 0 & \sigma^i_{\mathbf{x}^2\mathbf{x}^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \cdots & \sigma^i_{\mathbf{x}^d\mathbf{x}^d} \end{pmatrix} \tag{12.17}$$

We now only have d variances to estimate for a total of only $2d$ parameters to estimate per cluster.

The main EM approach is essentially the same as that given for the one dimensional case above, with the following differences. First, instead of a univariate normal distribution in (12.9), we compute the likelihood using:

$$P(\mathbf{x}_j|C_i) = f(\mathbf{x}_j|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}} e^{-\frac{(\mathbf{x}_j - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_j - \boldsymbol{\mu}_i)}{2}} \tag{12.18}$$

Second, instead of computing the variance in (12.12), we compute the covariances:

$$\sigma'^i_{\mathbf{x}^a\mathbf{x}^b} = \frac{\sum_{j=1}^n w_i(\mathbf{x}_j)(\mathbf{x}_j^a - \boldsymbol{\mu}_i^a) \times (\mathbf{x}_j^b - \boldsymbol{\mu}_i^b)}{\sum_{j=1}^n w_i(\mathbf{x}_j)} \tag{12.19}$$

where $\mathbf{x}_j^a$ is the value of the data point, and $\boldsymbol{\mu}_i^a$ is the value of the cluster mean, for the a-th attribute. Once we compute $\sigma'^i_{\mathbf{x}^a\mathbf{x}^b}$ for all pairs of attributes, we obtain the new covariance matrix $\boldsymbol{\Sigma}'_i$.

The computational complexity of the EM method is also $O(tnkd)$, and the I/O complexity is $O(t)$ where t is the number of iterations.