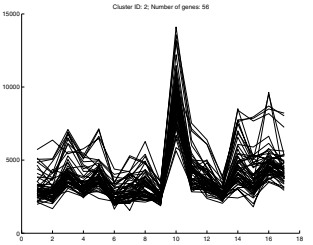

Chapter 9
Center-based Clustering Algorithms

Compared to other types of clustering algorithms, center-based algorithms are very efficient for clustering large databases and high-dimensional databases. Usually, center-based algorithms have their own objective functions, which define how good a clustering solution is. The goal of a center-based algorithm is to minimize its objective function. Clusters found by center-based algorithms have convex shapes and each cluster is represented by a center. Therefore, center-based algorithms are not good choices for finding clusters of arbitrary shapes. In this chapter, we shall present and discuss some center-based clustering algorithms and their advantages and disadvantages. We should mention that the expectation-maximization (EM) algorithm can be treated as a center-based algorithm, but we will defer the introduction of the EM algorithm to the chapter on model-based algorithms (Chapter 14).

9.1 The *k*-means Algorithm

The conventional k-means algorithm described in Algorithm 9.1, one of the most used clustering algorithms, was first described by Macqueen (1967). It was designed to cluster numerical data in which each cluster has a center called the mean. The k-means algorithm is classified as a partitional or nonhierarchical clustering method (Jain and Dubes, 1988). In this algorithm, the number of clusters k is assumed to be fixed. There is an error function in this algorithm. It proceeds, for a given initial k clusters, by allocating the remaining data to the nearest clusters and then repeatedly changing the membership of the clusters according to the error function until the error function does not change significantly or the membership of the clusters no longer changes. The conventional k-means algorithm (Hartigan, 1975; Hartigan and Wong, 1979) is briefly described below.

Let D be a data set with n instances, and let $C_1, C_2, \ldots, C_k$ be the k disjoint clusters of D. Then the error function is defined as

$$E = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mu(C_i)), \tag{9.1}$$

where $\mu(C_i)$ is the centroid of cluster C_i. $d(\mathbf{x}, \mu(C_i))$ denotes the distance between $\mathbf{x}$ and $\mu(C_i)$, and it can be one of the many distance measures discussed before, a typical choice of which is the Euclidean distance $d_{euc}(\cdot, \cdot)$ defined in (6.11).

ALGORITHM 9.1. The conventional *k*-means algorithm.

Require: Data set D, Number of Clusters k, Dimensions d:
{C_i is the ith cluster}
{1. Initialization Phase}
1: $(C_1, C_2, \ldots, C_k)$ = Initial partition of D.
{2. Iteration Phase}
2: **repeat**
3: d_{ij} = distance between case i and cluster j;
4: $n_i = \arg\min_{1 \le j \le k} d_{ij}$;
5: Assign case i to cluster n_i;
6: Recompute the cluster means of any changed clusters above;
7: **until** no further changes of cluster membership occur in a complete iteration
8: Output results.

The k-means algorithm can be divided into two phases: the initialization phase and the iteration phase. In the initialization phase, the algorithm randomly assigns the cases into k clusters. In the iteration phase, the algorithm computes the distance between each case and each cluster and assigns the case to the nearest cluster.

We can treat the k-means algorithm as an optimization problem. In this sense, the goal of the algorithm is to minimize a given objective function under certain conditions. Let $D = \{\mathbf{x}_i, i = 1, 2, \ldots, n\}$ be a data set with n instances and k be a given integer. The objective function can be defined as

$$P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{il} d_{euc}(\mathbf{x}_i, \mathbf{q}_l), \tag{9.2}$$

where $Q = \{\mathbf{q}_l, l = 1, 2, \ldots, k\}$ is a set of objects, $d_{euc}(\cdot, \cdot)$ is the Euclidean distance defined in (6.11), and W is an $n \times k$ matrix that satisfies the following conditions:

1. $w_{il} \in \{0, 1\}$ for $i = 1, 2, \ldots, n, l = 1, 2, \ldots, k$,
2. $\sum_{l=1}^{k} w_{il} = 1$ for $i = 1, 2, \ldots, n$.

The k-means algorithm can be formatted as the following optimization problem P (Selim and Ismail, 1984; Bobrowski and Bezdek, 1991): Minimize $P(W, Q)$ in (9.2) subject to conditions (1) and (2).

The optimization problem P can be solved by iteratively solving the following two subproblems (Huang, 1998):

- Subproblem P_1: Fix $Q = \hat{Q}$ and solve the reduced problem $P(W, \hat{Q})$.
- Subproblem P_2: Fix $W = \hat{W}$ and solve the reduced problem $P(\hat{W}, Q)$.

As to how to solve the subproblems P_1 and P_2, we have the following theorems (Huang, 1998).

Theorem 9.1. *In subproblem P_1, let $\hat{Q} = \{\hat{\mathbf{1}}_l, l = 1, 2 \ldots, k\}$ be fixed. Then the function $P(W, \hat{Q})$ is minimized if and only if*

$$w_{il} = \begin{cases} 1 & \text{if } d_{euc}(\mathbf{x}_i, \hat{\mathbf{q}}_l) = \min\limits_{1 \le t \le k} d_{euc}(\mathbf{x}_i, \hat{\mathbf{q}}_t), \\ 0 & \text{otherwise} \end{cases} \tag{9.3}$$

for $i = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, k$.

Theorem 9.2. *In subproblem P_2, let $\hat{W} = (\hat{w}_{il})$ be fixed. Then the function $P(\hat{W}, Q)$ is minimized if and only if*

$$\mathbf{q}_{lj} = \frac{\sum_{i=1}^{n} \hat{w}_{il} \mathbf{x}_{ij}}{\sum_{i=1}^{n} \hat{w}_{il}} \tag{9.4}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$.

Algorithm 9.2. The k-means algorithm treated as an optimization problem.

Require: Data set D, Number of Clusters k, Dimensions d:
1: Choose an initial Q^0 and solve $P(W, Q^0)$ to obtain W^0;
2: Let T be the number of iterations;
3: **for** $t = 0$ to T **do**
4: Let $\hat{W} \Leftarrow W^t$ and solve $P(\hat{W}, Q)$ to obtain Q^{t+1};
5: **if** $P(\hat{W}, Q^t) = P(\hat{W}, Q^{t+1})$ **then**
6: Output $\hat{W}$, Q^t;
7: Break;
8: **end if**
9: Let $\hat{Q} \Leftarrow Q^{t+1}$ and solve $P(W^t, \hat{Q})$ to obtain W^{t+1};
10: **if** $P(W^t, \hat{Q}) = P(W^{t+1}, \hat{Q})$ **then**
11: Output W^t, $\hat{Q}$;
12: Break;
13: **end if**
14: **end for**
15: Output W^{T+1}, Q^{T+1}.

The pseudocode of the optimization algorithm is described in Algorithm 9.2. The computational complexity of the algorithm is $O(nkd)$ per iteration (Phillips, 2002), where d is the dimension, k is the number of clusters, and n is the number of data points in the data set.

Since the sequence $P(\cdot, \cdot)$ generated by the algorithm is strictly decreasing, the algorithm will converge to a local minimum point after a finite number of iterations (Selim and Ismail, 1984). Convergence and some probability properties regarding the k-means

algorithm are also discussed in (Pollard, 1981), (Pollard, 1982), and (Serinko and Babu, 1992). García-Escudero and Gordaliza (1999) discussed the robustness properties of the k-means algorithm.

As one of the most often used clustering algorithms, the k-means algorithm has some important properties:

- It is efficient in clustering large data sets, since its computational complexity is linearly proportional to the size of the data sets.
- It often terminates at a local optimum (Anderberg, 1973; Selim and Ismail, 1984).
- The clusters have convex shapes, such as a ball in three-dimensional space (Anderberg, 1973).
- It works on numerical data.
- The performance is dependent on the initialization of the centers.

The k-means algorithm has some drawbacks (Peña et al., 1999). In particular, the performance is dependent on the initialization of the centers, as mentioned above. As a result, some methods for selecting good initial centers are proposed, for example, in (Babu and Murty, 1993) and (Bradley and Fayyad, 1998). Peña et al. (1999) provide a comparison of four initialization methods: a random method, Forgy's approach (Anderberg, 1973), Macqueen's approach (Macqueen, 1967), and Kaufman's approach (Kaufman and Rousseeuw, 1990). Other initialization methods are presented in (Khan and Ahmad, 2004).

In the iteration phase of the algorithm, the objects will be moved from one cluster to another in order to minimize the objective function. Tarsitano (2003) presents a computational study of the shortcomings and relative merits of 17 reallocation methods for the k-means algorithm.

Another drawback of the k-means algorithm is that it does not work effectively on high-dimensional data (Keim and Hinneburg, 1999). Also, working only on numerical data restricts some applications of the k-means algorithm.

The algorithm presented above is usually called the standard k-means algorithm. The standard k-means algorithm has several variations, such as the k-harmonic algorithm, the fuzzy k-means algorithm, and the Gaussian EM algorithm. Hamerly and Elkan (2002) investigated the properties of the standard k-means algorithm and its variations and alternatives.

9.2 Variations of the k-means Algorithm

Many clustering algorithms originating from the k-means algorithm are presented in (Faber, 1994), (Bradley and Fayyad, 1998), (Alsabti et al., 1998), and (Bottou and Bengio, 1995). These clustering algorithms were developed to improve the performance of the standard k-means algorithm. We will address some of these algorithms in subsequent sections.

9.2.1 The Continuous k-means Algorithm

The continuous k-means algorithm, proposed by Faber (1994), is faster than the standard k-means algorithm. It is different from the standard k-means algorithm in the following aspects. Firstly, in the continuous k-means algorithm, the prototypes (or reference points) are chosen as a random sample from the whole database, while in the standard k-means algorithm the initial points are chosen arbitrarily. Secondly, the data points are treated differently. During each complete iteration, the continuous k-means algorithm examines only a sample of the data points, while the standard k-means algorithm examines all the data points in sequence.

Theoretically, random sampling represents a return to Macqueen's original concept of the algorithm as a method of clustering data over a continuous space. In Macqueen's formulation, the error measure E_i for each region R_i is given by

$$E_i = \int_{R_i} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{z}_i\|^2 d\mathbf{x},$$

where $\rho(\cdot)$ is the probability distribution function, which is a continuous function defined over the space, and $\mathbf{z}_i$ is the centroid of the region R_i. The sum of all the E_i's is the total error measure.

A random sample of the data set can be a good estimate of the probability distribution function $\rho(\mathbf{x})$. Such a sample yields a representative set of centroids and a good estimate of the error measure without using all the data points in the original data set. Since both the reference points and the data points for updates are chosen by random sampling, the continuous k-means algorithm is generally faster than the standard k-means algorithm, and ten times faster than Lloyd's algorithm (Lloyd, 1982). Ways of further reducing the computer time are discussed in (Faber, 1994).

9.2.2 The Compare-means Algorithm

In order to accelerate the k-means algorithm, the algorithm compare-means (Phillips, 2002) uses a simple approach to avoid many unnecessary comparisons.

Let $\mathbf{x}$ be a point in D and μ_i and μ_j be two means. By the triangle inequality, we have $d(\mathbf{x}, \mu_i) + d(\mathbf{x}, \mu_j) \geq d(\mu_i, \mu_j)$, so $d(\mathbf{x}, \mu_j) \geq d(\mu_i, \mu_j) - d(\mathbf{x}, \mu_i)$. Therefore, we have $d(\mathbf{x}, \mu_j) \geq d(\mathbf{x}, \mu_i)$ if $d(\mu_i, \mu_j) \geq 2d(\mathbf{x}, \mu_i)$. In this case, computing $d(\mathbf{x}, \mu_j)$ is unnecessary.

Since the number of clusters k is usually small, distances of all pairs of means are precomputed before each iteration. Then, before comparing a point $\mathbf{x}$ to a mean μ_j, the above test is performed using the closest known mean to $\mathbf{x}$. The compare-means algorithm is described in Algorithm 9.3.

ALGORITHM 9.3. The compare-means algorithm.

Require: Data set D, Number of Clusters k, Dimensions d:
{C_i is the ith cluster}
{1. Initialization Phase}

1: $(C_1, C_2, \ldots, C_k)$ = Initial partition of D.
{2. Iteration Phase}
2: **repeat**
3: Calculate $D_{ij} = d(\mu_i, \mu_j)$ for all $i, j = 1, 2, \ldots, k$ {μ_i is the mean of the ith cluster in the previous iteration};
4: Let n_i be the subscript such that $\mathbf{x}_i \in C_{n_i}$;
5: $D_{min} \Leftarrow d(\mathbf{x}_i, \mu_{n_i})$;
6: **for** $j = 1$ to k **do**
7: **if** $D_{jn_i} < 2 * D_{min}$ and $j \neq n_i$ **then**
8: $dist = d(\mathbf{x}_i, \mu_j)$;
9: **if** $dist < D_{min}$ **then**
10: $D_{min} \Leftarrow dist$;
11: $n_i \Leftarrow j$;
12: **end if**
13: **end if**
14: **end for**
15: Assign case i to cluster n_i;
16: Recompute the cluster means of any changed clusters above;
17: **until** no further changes in cluster membership occur in a complete iteration
18: Output results;

The number of comparisons made by compare-means is harder to determine, but the overhead of compare-means is $\Theta(k^2 d + nkd)$ (Phillips, 2002), where n is the number of records, k is the number of clusters, and d is the dimension.

9.2.3 The Sort-means Algorithm

The algorithm sort-means (Phillips, 2002) is an extension of compare-means. In this algorithm, the means are sorted in order of increasing distance from each mean in order to obtain a further speedup.

Let $D_{ij} = d(\mu_i, \mu_j)$ for $i, j = 1, 2, \ldots, k$, where μ_i is the mean of the ith cluster. Let M be a $k \times k$ array in which row i $(m_{i1}, m_{i1}, \ldots, m_{ik})$ is a permutation of $1, 2, \ldots, k$ such that $d(\mu_i, \mu_{m_{i1}}) \leq d(\mu_i, \mu_{m_{i2}}) \leq \cdots \leq d(\mu_i, \mu_{m_{ik}})$. An iteration of sort-means is described in Algorithm 9.4.

ALGORITHM 9.4. An iteration of the sort-means algorithm.

1: Calculate $D_{ij} = d(\mu_i, \mu_j)$ for all $i, j = 1, 2, \ldots, k$ {μ_i is the mean of the ith cluster in the previous iteration};
2: Construct the array M;
3: Let n_i be the subscript such that $\mathbf{x}_i \in C_{n_i}$;
4: $D_{inmin} \Leftarrow d(\mathbf{x}_i, \mu_{n_i})$;
5: $D_{min} \Leftarrow D_{inmin}$;
6: **for** $j = 2$ to k **do**
7: $l \Leftarrow M_{n_i j}$;

```
    if D_{n_i l} >= 2 * D_{inmin} then
      break;
    end if
    dist = d(x, μ_l);
    if dist < D_{inmin} then
      D_{min} ⇐ dist;
      n_i ⇐ l;
    end if
  end for
  Assign case i to cluster n_i;
  Recompute the cluster means of any changed clusters above;
```

For the sort-means algorithm, the running time of an iteration is $O(nd\gamma + k^2 d + k^2 \log k)$ (Phillips, 2002), where n is the number of records, k is the number of clusters, d is the dimension, and γ is the average over all points $\mathbf{x}$ of the number of means that are no more than twice as far as $\mathbf{x}$ is from the mean $\mathbf{x}$ was assigned to in the previous iteration.

9.2.4 Acceleration of the *k*-means Algorithm with the *kd*-tree

Pelleg and Moore (1999) proposed an algorithm for the k-means clustering problem using the kd-tree data structure. The kd-tree data structure, described in Appendix B, can be used to reduce the large number of nearest-neighbor queries issued by the traditional k-means algorithm. Hence, an analysis of the geometry of the current cluster centers can lead to a great reduction in the work needed to update the cluster centers. In addition, the initial centers of the k-means algorithm can be chosen by the kd-tree efficiently.

One way to use the kd-tree in the inner loop of the k-means algorithm is to store the centers in the tree; another way is to store the whole data set in the tree. The latter method is used in (Pelleg and Moore, 1999). To describe the application of the kd-tree in the k-means algorithm, let us start with an iteration of the k-means algorithm.

Let $C^{(i)}$ denote the set of centroids after the ith iteration. Before the first iteration, $C^{(0)}$ is initialized to a set of random values. The stop criterion of the algorithm is that $C^{(i)}$ and $C^{(i-1)}$ are identical. In each iteration of the algorithm, the following two steps are performed:

1. For each data point $\mathbf{x}$, find the center in $C^{(i)}$ that is closest to $\mathbf{x}$ and associate $\mathbf{x}$ with this center.

2. Update $C^{(i)}$ to $C^{(i+1)}$ by taking, for each center, the center of mass of all the data points associated with this center.

Pelleg's algorithm involves modifying the second step in the iteration. The procedure to update the centroids in $C^{(i)}$ is recursive and has a parameter, a hyperrectangle h. The procedure starts with the initial value of h being the hyperrectangle containing all the input points. If the procedure can find $owner_{C^{(i)}}(h)$, it updates its counters using the center of mass and number of points that are stored in the kd-tree node corresponding to h; otherwise, it splits h by recursively calling itself with the children of h. Hence, given a set of centroids

C and a hyperrectangle h, $owner_C(h)$ is defined to be a center $\mathbf{z}$ in C such that any point in h is closer to $\mathbf{z}$ than to any other center in C, if such a center exists.

Performance comparison with BIRCH (Zhang et al., 1996) was presented in (Pelleg and Moore, 1999). Pelleg's method performs badly for high-dimensional (e.g., > 8) data but scales very well with the number of centers. Interested readers are referred to (Alsabti et al., 1998) and (Kanungo et al., 2002) for other examples of applying kd-tree in the k-means algorithm.

9.2.5 Other Acceleration Methods

We presented some methods previously for improving the performance of the k-means algorithm. Besides the above-mentioned methods, several other extensions of the standard k-means are proposed in order to improve the speed and quality of the k-means algorithm.

In order to improve the performance of the k-means algorithm in terms of solution quality and robustness, Chen et al. (2004) proposed a clustering algorithm that integrates the concepts of hierarchical approaches and the k-means algorithm. The initialization phase of this algorithm is similar to that of the k-means algorithm except that the number of initial centers m is larger than the number of clusters k. The iteration phase is the same as that of the k-means algorithm. The last phase is to merge clusters until k clusters are formed. The pair of clusters with the smallest score values will be merged into one cluster. The score between clusters C_i and C_j is defined as

$$Score(C_i, C_j) = \frac{|C_i||C_j|}{|C_i| + |C_j|} d^2_{euc}(\mu(C_i), \mu(C_j)),$$

where $\mu(C_i)$ and $\mu(C_j)$ are the centers of clusters C_i and C_j, respectively, and $d_{euc}(\cdot, \cdot)$ is the Euclidean distance.

Matoušek (2000) proposed a $(1 + \epsilon)$-approximate (i.e., the objective function value of the approximation is no worse than $(1 + \epsilon)$ times the minimum value of the objective function) k-clustering algorithm whose complexity is

$$O(n \log^k n \epsilon^{-2k^2 d})$$

for $k \geq 3$, where $\epsilon > 0$.

Har-Peled and Mazumdar (2004) proposed a similar approximation algorithm for the k-means by applying the k-means algorithm to, instead of the original data set D, a small weighted set $S \subset D$, of size $O(k\epsilon^{-d} \log n)$, where ϵ is a positive number, n is the number of objects in D, d is the dimensionality, and k is the number of clusters. It has been shown that the complexity of the approximation algorithm is

$$O\left(n + k^{k+2} \epsilon^{-(2d+1)k} \log^{k+1} n \log^k \frac{1}{\epsilon}\right),$$

which is linear to n for fixed k and ϵ. Details of this algorithm are omitted, but interested readers are referred to (Har-Peled and Mazumdar, 2004) for how the core set is computed.

Su and Chou (2001) proposed a modified version of the k-means algorithm that adopts a nonmetric distance measure based on the idea of "point symmetry." Precisely, for a given

data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a center $\mathbf{z}$, the point symmetry distance between an object $\mathbf{x}_i$ and the center $\mathbf{z}$ is defined as

$$d(\mathbf{x}_i, \mathbf{z}) = \min_{1 \le j \le n,\ j \ne i} \frac{\|(\mathbf{x}_i - \mathbf{z}) + (\mathbf{x}_j - \mathbf{z})\|}{\|\mathbf{x}_i - \mathbf{z}\| + \|\mathbf{x}_j - \mathbf{z}\|}.$$

An application of this algorithm for human face detection is presented in Su and Chou (2001).

In order to handle high-dimensional data, Stute and Zhu (1995) proposed a modified version of the k-means algorithm based on the projection pursuit, and Agarwal and Mustafa (2004) proposed an extension of the k-means algorithm for projective clustering in arbitrary subspaces with techniques to avoid local minima. Kantabutra and Couch (2000) implemented a parallel k-means algorithm to handle large databases.

9.3 The Trimmed k-means Algorithm

The trimmed k-means algorithm (Cuesta-Albertos et al., 1997), based on "impartial trimming," is a procedure that is more robust than the standard k-means algorithm. The main idea of the trimmed k-means algorithm is presented in this section.

The k-means algorithm can be viewed as a procedure based on the minimization of the expected value of a "penalty function" Φ of the distance to k-sets (sets of k points) through the following problem: Given an $\mathbb{R}^d$-valued random vector X, find the k-set $M = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$ in $\mathbb{R}^d$ such that

$$V_\Phi(M) = \int \Phi\left(\inf_{i=1,2,\ldots,k} \|X - \mathbf{m}_i\|\right) dP$$

is minimized.

The trimmed k-means procedure based on the methodology of "impartial trimming," which is a way to obtain a trimmed set with the lowest possible variation at some given level α, is formulated as follows (Cuesta-Albertos et al., 1997).

Let $\alpha \in (0, 1)$, the number of clusters k, and the penalty function Φ be given. For every set A such that $P(A) \ge 1 - \alpha$ and every k-set $M = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_k\}$ in $\mathbb{R}^d$, the variation of M given A is defined as

$$V_\Phi^A(M) = \frac{1}{P(A)} \int_A \Phi\left(\inf_{i=1,2,\ldots,k} \|X - \mathbf{m}_i\|\right) dP.$$

$V_\Phi^A(M)$ measures how well the set M represents the probability mass of P living on A. To find the best representation of the "more adequate" set containing a given amount of probability mass, we can minimize $V_\Phi^A(M)$ on A and M in the following way:

1. Obtain the k-variation given A, $V_{k,\Phi}^A$, by minimizing with respect to M:

$$V_{k,\Phi}^A = \inf_{M \subset \mathbb{R}^d, |M| = k} V_\Phi^A(M).$$

2. Obtain the trimmed k-variation $V_{k,\Phi,\alpha}$ by minimizing with respect to A:

$$V_{k,\Phi,\alpha} = V_{k,\Phi,\alpha}(X) = V_{k,\Phi,\alpha}(P_X) = \inf_{A\in\beta^d, P(A)\geq 1-\alpha} V_{k,\Phi}^A.$$

The goal of the algorithm is to obtain a trimmed set A_0 and a k-set M_0, if both of them exist, through the condition

$$V_\Phi^{A_0}(M_0) = V_{k,\Phi,\alpha}.$$

The trimmed k-means algorithm described above can be generalized as follows. Let $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a sample of independently identically distributed random variables in $\mathbb{R}^d$ with common distribution F. Let $\Phi : \mathbb{R}^+ \to \mathbb{R}$ be a suitable nondecreasing penalty function and $1-\gamma \in (0,1)$ be a trimming level. Then the generalized trimmed k-means of D is a k-set $\{\mathbf{m}_1^*, \mathbf{m}_2^*, \ldots, \mathbf{m}_k^*\} \subset \mathbb{R}^d$ solving the double optimization problem

$$\min_Y \min_{\{\mathbf{m}_1,\mathbf{m}_2,\ldots,\mathbf{m}_k\}\subset\mathbb{R}^d} \frac{1}{\lfloor n\gamma \rfloor} \sum_{\mathbf{x}\in Y} \Phi\left(\inf_{1\leq j\leq k} \|\mathbf{x}-\mathbf{m}_j\|\right),$$

where Y ranges in the class of the subsets of D with $\lfloor n\gamma \rfloor$ data points, and $\lfloor x \rfloor$ denotes the smallest integer greater than or equal to x.

The properties of existence and consistency of the trimmed k-means are shown to hold under certain conditions. Details of the theorems are omitted here; interested readers are referred to (Cuesta-Albertos et al., 1997). A central limit theory for the generalized trimmed k-means algorithm is given in (García-Escudero et al., 1999b). García-Escudero and Gordaliza (1999) investigated the performance of the generalized k-means algorithm and the generalized trimmed k-means algorithm from the viewpoint of Hampel's robustness criteria (Hampel, 1971). Further discussions of the trimmed k-means algorithm are given in (García-Escudero et al., 1999a).

9.4 The x-means Algorithm

In the k-means algorithm, the number of clusters k is an input parameter specified by the user. In order to reveal the true number of clusters underlying the distribution, Pelleg and Moore (2000) proposed an algorithm, called x-means, by optimizing the Bayesian information criterion (BIC) or the Akaike information criterion (AIC) measure (Bozdogan, 1987).

In the x-means algorithm, the BIC or Schwarz criterion (Kass and Raftery, 1995; Schwarz, 1978) is used globally and locally in order to find the best number of clusters k. Given a data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ containing n objects in a d-dimensional space and a family of alternative models $M_j = \{C_1, C_2, \ldots, C_k\}$, (e.g., different models correspond to solutions with different values of k), the posterior probabilities $P(M_j|D)$ are used to score the models. The Schwarz criterion can be used to approximate the posteriors.

The Schwarz criterion is defined as

$$BIC(M_j) = \hat{l}_j(D) - \frac{p_j}{2}\log n,$$

where $\hat{l}_j(D)$ is the loglikelihood of D according to the jth model and taken at the maximum likelihood point, and p_j is the number of parameters in M_j. The model with the largest score is selected.

Under the identical spherical Gaussian distribution, the maximum likelihood estimate for variance is

$$\hat{\sigma}^2 = \frac{1}{n-k}\sum_{i=1}^{n}(\mathbf{x}_i - \mu_{(i)})^2,$$

where $\mu_{(i)}$ is the centroid associated with the object $\mathbf{x}_i$, i.e., (i) denotes the index of the centroid that is closest to $\mathbf{x}_i$. The point probabilities are

$$\hat{P}(\mathbf{x}_i) = \frac{|C_{(i)}|}{n} \cdot \frac{1}{\sqrt{2\pi}\hat{\sigma}^d} \exp\left(-\frac{1}{2\hat{\sigma}^2}\|\mathbf{x}_i - \mu_{(i)}\|^2\right).$$

Thus, the loglikelihood of the data is

$$l(D) = \log\prod_{i=1}^{n} P(\mathbf{x}_i) = \sum_{i=1}^{n}\left(\log\frac{1}{\sqrt{2\pi}\hat{\sigma}^d} - \frac{1}{2\hat{\sigma}^2}\|\mathbf{x}_i - \mu_{(i)}\|^2 + \log\frac{|C_{(i)}|}{n}\right).$$

The number of free parameters p_j is $k - 1 + dk + 1 = (d+1)k$.

The Schwarz criterion is used in x-means globally to choose the best model it encounters and locally to guide all centroid splits. The algorithm can be briefly described as follows.

Given a range for k, $[k_{min}, k_{max}]$, the x-means algorithm starts with $k = k_{min}$ and continues to add centroids when they are needed until the upper bound is reached. New centroids are added by splitting some centroids into two according to the Schwarz criterion. During the process, the centroid set with the best score is recorded as the one that is the final output. The algorithm can be implemented efficiently using ideas of "blacklisting"(Pelleg and Moore, 1999) and kd-trees.

9.5 The k-harmonic Means Algorithm

k-harmonic means (Zhang et al., 2000a, 1999) is developed from the k-means algorithm and it is essentially insensitive to the initialization of centers.

We know that the error function (or performance function) of the k-means algorithm can be written as

$$E = \sum_{i=1}^{n}\min\{d(\mathbf{x}_i, \mu_j),\ j = 1, 2, \ldots, k\}, \tag{9.5}$$

where μ_j is the mean of the jth cluster.

Then the error function of the k-harmonic means algorithm is obtained by replacing the minimum function $\min(\cdot)$ by the harmonic average (or harmonic mean) function $\mathrm{HA}(\cdot)$

and using the squared Euclidean distance, i.e.,

$$E = \sum_{i=1}^{n} \mathrm{HA}(\{d_{seuc}(\mathbf{x}_i, \mu_j),\ j = 1, 2, \ldots, k\}) = \sum_{i=1}^{n} \frac{k}{\sum_{j=1}^{k} \frac{1}{(\mathbf{x}_i - \mu_j)^T (\mathbf{x}_i - \mu_j)}}, \tag{9.6}$$

where μ_j is the mean of the jth cluster, $d_{seuc}(\cdot, \cdot)$ is the squared Euclidean distance, and $\mathrm{HA}(\cdot)$ is the harmonic average defined as

$$\mathrm{HA}(\{a_i : i = 1, 2, \ldots, m\}) = \frac{m}{\sum_{i=1}^{m} a_i^{-1}}. \tag{9.7}$$

The recursive formula for the k-harmonic means algorithm can be obtained by taking partial derivatives of the error function (9.6) with respect to the means $\mu_l, l = 1, 2, \ldots, k$, and setting them to zero. That is,

$$\frac{\partial E}{\partial \mu_l} = -k \sum_{i=1}^{n} \frac{2(\mathbf{x}_i - \mu_l)}{d_{il}^4 \left(\sum_{j=1}^{k} d_{ij}^{-2} \right)^2} = \mathbf{0}, \tag{9.8}$$

where $d_{ij} = d_{euc}(\mathbf{x}_i, \mu_j) = \left[(\mathbf{x}_i - \mu_j)^T (\mathbf{x}_i - \mu_j) \right]^{\frac{1}{2}}$.

By solving equation (9.8), we obtain new centers $\mu_l^*, l = 1, 2, \ldots, k$ as follows:

$$\mu_l^* = \frac{\sum_{i=1}^{n} d_{il}^{-4} \left(\sum_{j=1}^{k} d_{ij}^{-2} \right)^{-2} \mathbf{x}_i}{\sum_{i=1}^{n} d_{il}^{-4} \left(\sum_{j=1}^{k} d_{ij}^{-2} \right)^{-2}}. \tag{9.9}$$

Then given a set of initial centers, we can obtain new centers by (9.9). This recursion is continued until the centers stabilize.

In order to reduce the sensitivity of the convergence quality to the initial centers, Zhang et al. (2000) proposed a generalized k-harmonic means algorithm as follows:

$$\mu_l^* = \frac{\sum_{i=1}^{n} d_{il}^{-s} \left(\sum_{j=1}^{k} d_{ij}^{-2} \right)^{-2} \mathbf{x}_i}{\sum_{i=1}^{n} d_{il}^{-s} \left(\sum_{j=1}^{k} d_{ij}^{-2} \right)^{-2}} \tag{9.10}$$

for $l = 1, 2, \ldots, k$, where s is a parameter. Unfortunately, no method has been developed to choose the parameter s.

9.6 The Mean Shift Algorithm

The mean shift algorithm (Fukunaga and Hostetler, 1975; Cheng, 1995; Comaniciu and Meer, 2002, 1999) is a simple iterative procedure that shifts each data point to the average of data points in its neighborhood. To introduce the mean shift algorithm, let us start with some definitions and notation.

Definition 9.3 (Profile). *A profile k is a function $k : [0, \infty] \to [0, \infty]$ satisfying the following conditions:*

1. *k is nonincreasing,*
2. *k is piecewise continuous, and*
3. *$\int_0^\infty k(r)\mathrm{d}r < \infty$.*

Definition 9.4 (Kernel). *A function $K : \mathbb{R}^d \to \mathbb{R}$ is said to be a kernel if there exists a profile k such that*

$$K(\mathbf{x}) = k\left(\|\mathbf{x}\|^2\right),$$

where $\|\cdot\|$ denotes the Euclidean norm.

Let $\alpha > 0$. If K is a kernel, then

$$\begin{aligned}(\alpha K)(\mathbf{x}) &= \alpha K(\mathbf{x}),\\ K_\alpha(\mathbf{x}) &= K\left(\frac{\mathbf{x}}{\alpha}\right),\\ (K^\alpha)(\mathbf{x}) &= (K(\mathbf{x}))^\alpha\end{aligned}$$

are all kernels.

Definition 9.5 (The mean shift algorithm). *Let $D \subset \mathbb{R}^d$ be a finite data set, K a kernel, and $w : D \to (0, \infty)$ a weight function. The sample mean with kernel K at $\mathbf{x} \in \mathbb{R}^d$ is defined as*

$$m(\mathbf{x}) = \frac{\sum_{\mathbf{y}\in D} K(\mathbf{y}-\mathbf{x})w(\mathbf{y})\mathbf{y}}{\sum_{\mathbf{y}\in D} K(\mathbf{y}-\mathbf{x})w(\mathbf{y})}.$$

Let $T \subset \mathbb{R}^d$ be a finite set of cluster centers. The evolution of T in the form of iterations $T \leftarrow m(T)$ with $m(T) = \{m(\mathbf{y}) : \mathbf{y} \in T\}$ is called the mean shift algorithm.

The mean shift algorithm is a very general iterative procedure to the extent that some well-known clustering algorithms are its special cases. The maximum-entropy clustering (MEC) algorithm (Rose et al., 1990), for example, is a mean shift algorithm when T and D are separate sets, $G^\beta(\mathbf{x}) = e^{-\beta\|\mathbf{x}\|^2}$ is the kernel, and

$$w(\mathbf{y}) = \frac{1}{\sum_{\mathbf{t}\in T} G^\beta(\mathbf{y}-\mathbf{t})}, \quad \mathbf{y} \in D.$$

In addition, the well-known k-means algorithm is a limiting case of the mean shift algorithm (Cheng, 1995).

We now introduce some definitions in order to describe the convergence properties of the mean shift algorithm.

Definition 9.6 (Direction). *A direction in $\mathbb{R}^d$ is a point on the unit sphere, i.e.,* $\mathbf{a}$ *is a direction if and only if* $|\mathbf{a}| = 1$.

Definition 9.7 (Projection). *The projection in the direction* $\mathbf{a}$ *is defined as the mapping* $\pi_{\mathbf{a}} : \mathbb{R}^d \to \mathbb{R}$ *with* $\pi_{\mathbf{a}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{a} \rangle$, *where* $\langle \cdots, \cdots \rangle$ *denotes the inner product.*

Definition 9.8 (Convex hull). *The convex hull $h(Y)$ of a set $Y \subset \mathbb{R}^d$ is defined as*

$$\bigcap_{\|\mathbf{a}\|=1} \{\mathbf{x} \in \mathbb{R}^d : \min \pi_{\mathbf{a}}(Y) \leq \pi_{\mathbf{a}}(\mathbf{x}) \leq \max \pi_{\mathbf{a}}(Y)\}.$$

Definition 9.9 (Translation). $h(D) \supseteq h(m(D)) \supseteq h(m(m(D))) \supseteq \cdots$, *i.e., a translation is a transformation of the data so that the origin is in all the convex hulls of data.*

Definition 9.10 (Radius). *Suppose after a translation, the origin is in all the convex hulls of data. Then the radius of data is*

$$\rho(D) = \max\{\|\mathbf{x}\| : \mathbf{x} \in D\}.$$

Definition 9.11 (Diameter). *The diameter of data is defined as*

$$d(D) = \sup_{\|\mathbf{a}\|}(\max \pi_{\mathbf{a}}(D) - \min \pi_{\mathbf{a}}(D)).$$

Regarding the convergence of the mean shift algorithm, Cheng (1995) proved the following two theorems.

Theorem 9.12 (Convergence with broad kernels). *Let k be the profile of the kernel used in a blurring process and S_0 be the initial data. If $k(d^2(S_0)) \geq \kappa$ for some $\kappa > 0$, then the diameter of the data approaches zero. The convergence rate is at least as fast as*

$$\frac{d(m(D))}{d(D)} \leq 1 - \frac{\kappa}{4k(0)}.$$

Theorem 9.13 (Convergence with truncated kernels). *If data points cannot move arbitrarily close to each other and $K(\mathbf{x})$ is either zero or larger than a fixed positive constant, then the blurring process reaches a fixed point in finitely many iterations.*

The mean shift algorithm is not only an intuitive and basic procedure but also a deterministic process. It is more efficient than gradient descent or ascent methods in terms

of adapting to the right step size (Cheng, 1995). There are also some factors that make the mean shift algorithm not popular. For example, the computational cost of an iteration of the mean shift algorithm is $O(n^2)$ (Cheng, 1995), where n is the number of data points in the data set. The mean shift algorithm is also not suitable for high-dimensional data sets and large data sets. Other discussions of the mean shift algorithm can be found in Fashing and Tomasi (2005), Yang et al. (2003a), Chen and Meer (2005), and Georgescu et al. (2003).

9.7 MEC

The MEC algorithm, based on statistical physics, was introduced by Rose et al. (1990). The MEC algorithm is a fuzzy clustering algorithm and the fuzzy membership is obtained by maximizing the entropy at a given average variance. A deterministic annealing process is derived from the relationship between the corresponding Lagrange multiplier and the "temperature" so that the free energy is minimized at each temperature.

The energy or cost contributed to the cluster C_j by a data point $\mathbf{x}$ is denoted by $E_j(\mathbf{x})$. In MEC, the energy $E_j(\mathbf{x})$ is defined as

$$E_j(\mathbf{x}) = \|\mathbf{x} - \mathbf{z}_j\|^2,$$

where $\|\cdot\|$ is the Euclidean norm and $\mathbf{z}_j$ is the centroid of C_j. The average total energy for a given partition is defined as

$$E = \sum_{\mathbf{x}\in D} \sum_{j=1}^{k} P(\mathbf{x} \in C_j) E_j(\mathbf{x}), \tag{9.11}$$

where $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is the data set under consideration, k is the number of clusters, and $P(\mathbf{x} \in C_j),\ j = 1, 2, \ldots, k$, are the association probabilities or fuzzy memberships. The association probabilities that maximize the entropy under the constraint (9.11) are Gibbs distributions defined as

$$P(\mathbf{x} \in C_j) = \frac{e^{-\beta E_j(\mathbf{x})}}{Z_{\mathbf{x}}}, \tag{9.12}$$

where $Z_{\mathbf{x}}$ is the partition function defined as

$$Z_{\mathbf{x}} = \sum_{j=1}^{k} e^{-\beta E_j(\mathbf{x})}.$$

The parameter β is the Lagrange multiplier determined by the given value of E in equation (9.11). The total partition function is defined as

$$Z = \prod_{\mathbf{x}\in D} Z_{\mathbf{x}}.$$

Based on the partition function, the free energy is defined as

$$F = -\frac{1}{\beta} \ln Z = -\frac{1}{\beta} \sum_{\mathbf{x}\in D} \ln \left(\sum_{j=1}^{k} e^{-\beta \|\mathbf{x}-\mathbf{z}_j\|^2} \right). \tag{9.13}$$

The set of centroids $\mathbf{z}_j$ that optimizes the free energy satisfies

$$\frac{\partial F}{\partial \mathbf{z}_j} = 0 \quad \forall j$$

or

$$\sum_{\mathbf{x}\in D} \frac{(\mathbf{x}-\mathbf{z}_j)e^{-\beta\|\mathbf{x}-\mathbf{z}_j\|^2}}{\sum_{l=1}^{k} e^{-\beta\|\mathbf{x}-\mathbf{z}_l\|^2}} = 0 \quad \forall j,$$

which leads to

$$\mathbf{z}_j = \frac{\sum_{\mathbf{x}\in D} \mathbf{x}P(\mathbf{x}\in C_j)}{\sum_{\mathbf{x}\in D} P(\mathbf{x}\in C_j)}.$$

The MEC algorithm is an iterative process $\mathbf{z}^{(r)} \to \mathbf{z}^{(r+1)}$, $r = 1, 2, \ldots$. Note that the MEC algorithm is a special case of the mean shift algorithm (Cheng, 1995) and the k-means algorithm is the limiting case of the MEC algorithm when β approaches infinity (Cheng, 1995).

9.8 The *k*-modes Algorithm (Huang)

The k-modes algorithm (Huang, 1997b, 1998) comes from the k-means algorithm (see Section 9.1), and it was designed to cluster categorical data sets. The main idea of the k-modes algorithm is to specify the number of clusters (say, k) and then to select k initial modes, followed by allocating every object to the nearest mode.

The k-modes algorithm uses the simple match dissimilarity measure (see Section 6.3.1) to measure the distance of categorical objects. The mode of a cluster is defined as follows.

Let D be a set of categorical objects described by d categorical attributes, $A_1, A_2, \ldots, A_d$. Let $X \subseteq D$. Then the mode of X is defined to be a vector $\mathbf{q} = (q_1, q_2, \ldots, q_d)$ such that the function

$$D(X, \mathbf{q}) = \sum_{\mathbf{x}\in X} d_{sim}(\mathbf{x}, \mathbf{q}) \tag{9.14}$$

is minimized, where $d_{sim}(\cdot,\cdot)$ is defined in (6.23).

Hence, according to this definition, the mode is not necessarily an element of that data set. The following theorem (Huang, 1998) shows how to minimize the function given in (9.14).

Theorem 9.14. *Let the domain of A_j be $\mathrm{DOM}(A_j) = \{A_{j1}, A_{j2}, \ldots, A_{jn_j}\}$ for $j = 1, 2, \ldots, d$, and let $X \subseteq D$. Let $f_{jr}(X)(1 \le j \le d, 1 \le r \le n_j)$ be the number of objects in X that take value A_{jr} at the jth attribute, i.e.,*

$$f_{jr}(X) = |\{\mathbf{x} \in X : \mathbf{x}_j = A_{jr}\}|. \tag{9.15}$$

Then the function given in (9.14) is minimized if and only if $q_j \in \mathrm{DOM}(A_j)$ for $j = 1, 2, \ldots, d$, and

$$f_{jr_j}(X) \ge f_{jl}(X) \forall l \ne r_j,\ j = 1, 2, \ldots, d,$$

where r_j is the subscript defined as $q_j = A_{jr_j}$ for $j = 1, 2, \ldots, d$.

Theorem 9.14 provides us with a way to find $\mathbf{q}$ for a given data set X. This theorem also implies that the mode of a data set is not necessarily unique.

Since the k-modes algorithm comes from the k-means algorithm, it can also be treated as an optimization problem. The objective function for the k-modes algorithm can be defined as in (9.2) by changing the Euclidean distance to the simple matching distance, i.e.,

$$P(W, Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} w_{il} d_{sim}(\mathbf{x}_i, \mathbf{q}_l), \tag{9.16}$$

where $Q = \{\mathbf{q}_l, l = 1, 2, \ldots, k\}$ is a set of objects, $d_{sim}(\cdot, \cdot)$ is the simple matching distance defined in (6.23), and W is an $n \times k$ matrix that satisfies the following conditions:

1. $w_{il} \in \{0, 1\}$ for $i = 1, 2, \ldots, n, l = 1, 2, \ldots, k$,

2. $\sum_{l=1}^{k} w_{il} = 1$ for $i = 1, 2, \ldots, n$.

Thus, Algorithm 9.5 can be used for the k-modes algorithm by using the objective function defined in (9.16). But this algorithm is not efficient, since we need to calculate the total cost P of the whole data set each time a new Q or W is obtained. To make the computation more efficient, we use the algorithm described in Algorithm 9.5 (Huang, 1998).

Algorithm 9.5. The k-modes algorithm.

Require: Data set D, Number of Clusters k, Dimensions d:
1: Select k initial modes $Q = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k\}$, and $\mathbf{q}_l$ for cluster l;
2: **for** $i = 1$ to n **do**
3: Find an l such that $d_{sim}(\mathbf{x}_i, \mathbf{q}_l) = \min_{1 \le t \le k} d_{sim}(\mathbf{x}_i, \mathbf{q}_t)$;
4: Allocate $\mathbf{x}_i$ to cluster l;
5: Update the mode $\mathbf{q}_l$ for cluster l;
6: **end for**
7: **repeat**
8: **for** $i = 1$ to n **do**
9: Let l_0 be the index of the cluster to which $\mathbf{x}_i$ belongs;
10: Find an l_1 such that $d_{sim}(\mathbf{x}_i, \mathbf{q}_{l_1}) = \min_{1 \le t \le k, t \ne l_0} d_{sim}(\mathbf{x}_i, \mathbf{q}_t)$;
11: **if** $d_{sim}(\mathbf{x}_i, \mathbf{q}_{l_1}) < d_{sim}(\mathbf{x}_i, \mathbf{q}_{l_0})$ **then**
12: Reallocate $\mathbf{x}_i$ to cluster l_1;
13: Update $\mathbf{q}_{l_0}$ and $\mathbf{q}_{l_1}$;
14: **end if**
15: **end for**
16: **until** No changes in cluster membership
17: Output results.

The proof of convergence for this algorithm is not available (Anderberg, 1973), but its practical use has shown that it always converges (Huang, 1998).

The k-modes algorithm is very popular for clustering categorical data. It has some important properties:

- It is efficient for clustering large data sets.
- It also produces locally optimal solutions that are dependent on initial modes and the order of objects in the data set (Huang, 1998).
- It works only on categorical data.

9.8.1 Initial Modes Selection

Since the clustering results and convergence speed of the k-modes algorithm are dependent on the initial modes, the selection of initial modes is an important issue in the k-modes algorithm. Good initial modes lead to fast convergence and good results, while bad initial modes lead to slow convergence. Many initial modes selection methods have been discussed in the literature. A commonly used approach, called the direct method, is to choose the first k distinct objects as initial modes. For example, for a given data set $D = \{\mathbf{x}_i, i = 1, 2, \ldots, n\}$, we choose $\mathbf{q}_l = \mathbf{x}_l$ for $l = 1, 2, \ldots, k$ as modes if $\mathbf{x}_l \neq \mathbf{x}_t$ for all $1 \leq l < t \leq k$. Another approach, called the diverse modes method, is to spread the initial modes over the whole data set by assigning the most frequent categories equally to the initial modes (Huang, 1998). Given a data set D, we first sort each column of its symbol table T_s such that each column of its corresponding frequency table of D is in decreasing order. In other words for each j, we sort $A_{j1}, A_{j2}, \ldots, A_{jn_j}$ such that $f_{j1}(D) \geq f_{j2} \geq \cdots \geq f_{jn_j}(D)$. Secondly, the most frequent categories are equally assigned to the initial modes $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_k$. For example, $A_{11}, A_{21}, \ldots, A_{d1}$ are in different initial modes. Finally, we start with $\mathbf{q}_1$, select the record most similar to $\mathbf{q}_1$, and replace $\mathbf{q}_1$ as the first initial mode. After $\mathbf{q}_i$ $(i = 1, 2, \ldots, l)$ are replaced, we select the record in D most similar to $\mathbf{q}_{l+1}$ and replace $\mathbf{q}_{l+1}$ with that record as the $(l + 1)$th initial mode. We keep doing this until $\mathbf{q}_k$ is replaced.

The last step is taken to avoid the occurrence of an empty cluster. The initial modes found by this method are diverse in the data set. These initial modes can lead to better clustering results, but this costs time.

9.9 The k-modes Algorithm (Chaturvedi et al.)

Chaturvedi et al. (2001) proposed a nonparametric bilinear model to derive clusters from categorical data. The clustering procedure is analogous to the traditional k-means algorithm (Macqueen, 1967). To describe the algorithm, let us begin with the bilinear clustering model.

Let $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a data set with n objects, each of which is described by d categorical attributes. Let k be the number of clusters. Then the bilinear clustering model is (Chaturvedi et al., 2001)

$$C = SW + error, \tag{9.17}$$

where C is an $n \times d$ data matrix; S is an $n \times k$ binary indicator matrix for membership of the n objects in k mutually exclusive, nonoverlapping clusters (i.e., the (i, j)th entry of S is 1 if $\mathbf{x}_i$ belongs to the jth clusters, and 0 otherwise); and W is the matrix of centroids.

The data matrix C in equation (9.17) is known, whereas both S and W are unknown and must be estimated. The algorithm iterates as follows: estimate S given estimates of W, and then revise the estimates of W given the new estimates of S. This process will be repeated until the quality of clustering is not improved. In this algorithm, the quality of clustering is indicated by an L_0 loss function.

Let $\hat{C} = SW$. Then the L_p-norm–based loss function is defined as

$$L_p = \sum_{i=1}^{n} \sum_{j=1}^{d} |c_{ij} - \hat{c}_{ij}|^p$$

for positive values of $p \to 0$, where c_{ij} and $\hat{c}_{ij}$ are the (i, j)th entries of C and $\hat{C}$, respectively. L_0 is the limiting case as $p \to 0$ and simply counts the number of mismatches in the matrices C and $\hat{C}$, i.e.,

$$L_0 = \sum_{i=1}^{n} \sum_{j=1}^{d} \delta(c_{ij}, \hat{c}_{ij}),$$

where $\delta(\cdot, \cdot)$ is defined in equation (6.22).

The goal of the algorithm is to minimize L_0. The matrices S and W are estimated iteratively until the value of the L_0 loss function is not improved. The detailed estimation procedure is described as follows.

To estimate $S = (s_{il})$ given the estimates of $W = (w_{lj})$, we consider the functions

$$f_i = \sum_{j=1}^{d} \left(c_{ij} - \sum_{l=1}^{k} s_{il} w_{lj} \right)^0 = \sum_{l=1}^{k} s_{il} \left(\sum_{j=1}^{d} \delta(c_{ij}, w_{lj}) \right)$$

for $i = 1, 2, \ldots, n$. Then $L_0 = \sum_{i=1}^{n} f_i$. To minimize L_0, we can separately minimize f_i. To minimize f_i, we try all the d patterns $s_{il} = 1$ for $l = 1, 2, \ldots, d$ and choose $\hat{s}_{il_0} = 1$ if

$$\sum_{j=1}^{d} \delta(c_{ij}, w_{l_0 j}) = \min_{1 \le l \le k} \sum_{j=1}^{d} \delta(c_{ij}, w_{lj}).$$

To estimate $W = (w_{lj})$ given the estimates of $S = (s_{il})$, we can estimate w_{lj} separately. Precisely, let C_l be the lth cluster, i.e., $C_l = \{\mathbf{x}_i : s_{il} = 1,\ 1 \le i \le n\}$, and consider the mode of $\{x_j : \mathbf{x} \in C_l\}$, where x_j is the jth attribute value of $\mathbf{x}$. Let $\hat{w}_{lj}$ be the mode of $\{x_j : \mathbf{x} \in C_l\}$.

Although the above k-modes algorithm is faster than other procedures in some cases, such as the latent class procedure (Goodman, 1974), it has some disadvantages. Firstly, it can only guarantee a locally optimal solution. Secondly, the number of clusters k is required. In order to achieve a globally optimal solution, Gan et al. (2005) proposed a genetic k-modes algorithm based on the k-modes algorithm and the genetic algorithm.

9.10 The k-probabilities Algorithm

The k-probabilities algorithm (Wishart, 2002) is an extension of the k-modes algorithm. It was designed for clustering mixed-type data sets. The k-probabilities algorithm uses the

general distance coefficient (Gower, 1971) measure between two records, and it uses the squared distance to compute the distance between a case and a cluster; for instance, the distance d_{ip} between any case i and a cluster p is defined as

$$d_{ip}^2 = \sum_k \frac{w_{ipk}(x_{ik} - \mu_{pk})^2}{\sum_k w_{ipk}}, \tag{9.18}$$

where x_{ik} is the value of the kth variable for case i, μ_{pk} is the mean of the kth variable for cluster p, and w_{ipk} is a weight of 1 or 0 depending on whether or not the comparison between case i and cluster p is valid for the kth variable, i.e., $w_{ipk} = 1$ if we can compare the kth variable between case i and cluster p; otherwise $w_{ipk} = 0$. Notice that for nominal variables, the mean μ_{pk} is a vector φ_{pks} of probabilities for each state s of the kth variable within cluster p.

The object function of this algorithm is

$$E = \sum_p E_p, \tag{9.19}$$

where E_p is the Euclidean sum of squares defined as

$$E_p = \sum_{i \in p} n_i \sum_k \frac{w_k(x_{ik} - \mu_{pk})^2}{\sum_k w_k}, \tag{9.20}$$

where x_{ik} and μ_{pk} are the same as in equation (9.18), n_i is a differential weight for case i (normally 1), and w_k is a differential weight for the kth variable, where $w_k = 0$ if x_{ik} or μ_{pk} has a missing value at the kth variable.

The object of the k-probabilities algorithm is to minimize the total Euclidean sum of squares E in equation (9.19). The algorithm starts with an initial partition of the data set into k clusters, and then reassigns the cases to another cluster such that the total Euclidean sum of squares E is minimized. To minimize E, a case i should only be reassigned from cluster p to cluster q if (Wishart, 1978)

$$E_p + E_q > E_{p-i} + E_{q+i},$$

which is equivalent to

$$I_{p-i,i} > I_{q,i}.$$

ALGORITHM 9.6. The k-probabilities algorithm.

Require: Data set D, No. of Clusters:k, Dimensions: d:
{C_i is the ith cluster}
{1. Initialization Phase}
1: $(C_1, C_2, \ldots, C_k)$ = Initial partition of D.
{2. Reallocation Phase}
2: **repeat**
3: **for** $i = 1$ to $|D|$ **do**

```
    Compute I_{p-i,i} {p is the current cluster number of case i};
    for q = 1 to k, q ≠ p do
      compute I_{q+i,i};
      if I_{p-i,i} > I_{q+i,i} then
        Assign case i to cluster n_i;
        break;
      end if
    end for
  end for
  Recompute the cluster means of any changed clusters above;
until no further changes in cluster membership occur in a complete iteration
```

The pseudocode of the k-probabilities algorithm is given in Algorithm 9.6. Gupta et al. (1999) also proposed an algorithm that extends the k-means algorithm to cluster categorical data through defining the objective function based on the new Condorcet criterion (Michaud, 1997).

9.11 The k-prototypes Algorithm

The k-prototypes algorithm (Huang, 1998) comes from the k-means and k-modes algorithm; it was designed to cluster mixed-type data sets. A related work is (Huang, 1997a). In the k-prototypes algorithm, the prototype is the center of a cluster, just as the mean and mode are the centers of a cluster in the k-means and k-modes algorithms, respectively.

Let two mixed-type objects be described by attributes $A_1^r, A_2^r, \ldots, A_p^r, A_{p+1}^c, \ldots, A_m^c$, where the first p attributes are numerical while the remaining $m-p$ attributes are categorical. Let $X = [x_1, x_2, \ldots, x_m]$, and $Y = [y_1, y_2, \ldots, y_m]$, where x_i, and y_i $(1 \leq i \leq p)$ take numerical values while the rest take categorical values. Then the dissimilarity measure between X and Y can be

$$d(X, Y) = \sum_{j=1}^{p} (x_j - y_j)^2 + \gamma \sum_{j=p+1}^{m} \delta(x_j, y_j),$$

where γ is a balance weight used to avoid favoring either type of attribute. In the definition of the dissimilarity measure, the squared Euclidean distance is used to measure the numerical attributes and the simple matching dissimilarity (Kaufman and Rousseeuw, 1990) measure is used to measure the categorical attributes.

The goal of the k-prototypes algorithm is to minimize the cost function

$$P(W, Q) = \sum_{l=1}^{k} (P_l^r + \gamma P_l^c),$$

where

$$P_l^r = \sum_{i=1}^{n} w_{i,l} \sum_{j=1}^{p} (x_{i,j} - q_{l,j})^2,$$

$$P_l^c = \sum_{i=1}^{n} w_{i,l} \sum_{j=p+1}^{m} \delta(x_{i,j}, q_{l,j}).$$

ALGORITHM 9.7. The *k*-prototypes algorithm.

Require: k: the number of clusters;
1: Select k initial prototypes from the database, one for each cluster;
2: Allocate each object in the database to a cluster whose prototype is the nearest to it according to the dissimilarity measure, and update the prototype of the cluster after each allocation;
3: **repeat**
4: Retest the similarity between each object and the prototype; if an object is found that is nearest to another prototype rather than the current one, reallocate the object to the nearest cluster;
5: Update the prototypes of both clusters;
6: **until** no further changes in the cluster membership

The k-prototypes algorithm (Algorithm 9.7) is the same as the k-probabilities algorithm (see Algorithm 9.6) except for the reallocation phase. The complexity of the k-prototypes algorithm is $O((t+1)kn)$, where n is the number of data points in the data set, k is the number of clusters, and t is the number of iterations of the reallocation process.

9.12 Summary

A popular center-based clustering algorithm, the k-means algorithm, and its variations, has been presented in this chapter. To handle categorical data, two versions of the k-modes algorithm are also presented. Center-based algorithms are easy to implement and the results are easy to interpret. In addition, center-based algorithms are faster than hierarchical algorithms in general. Therefore, they are popular for clustering large databases. Zhang and Hsu (2000) discussed accelerating center-based clustering algorithms by parallelism.

In center-based clustering algorithms, each cluster has one center. Instead of generating a cluster center as a point, Bradley and Mangasarian (2000) proposed a clustering algorithm, called the k-plane algorithm, in which the entity of the center is changed from a point to a plane. In some cases, the k-plane algorithm outperforms the k-means algorithm (Bradley and Mangasarian, 2000).