Subspace Clustering

Recently, subspace clustering has aroused great interest in researchers in the database community due to the new challenges associated with the high dimensionality of data sets in modern science and technology.

Many clustering algorithms have been developed to identify clusters in the whole data space; we refer to these clustering algorithms as conventional clustering algorithms. Unfortunately, most of these conventional clustering algorithms do not scale well to cluster high-dimensional data sets in terms of effectiveness and efficiency because of their inherent sparsity. In high-dimensional data sets, we encounter several problems. First, the distance between any two data points becomes almost the same (Beyer et al., 1999), so it is difficult to differentiate similar data points from dissimilar ones. Secondly, clusters are embedded in the subspaces of the high-dimensional data space, and different clusters may exist in different subspaces (Agrawal et al., 1998). Because of these problems, almost all conventional clustering algorithms fail to work well for high-dimensional data sets.

One possible solution is to use dimension reduction techniques such as principal component analysis (PCA) and the Karhunen-Loève transformation (Agrawal et al., 1998) or feature selection techniques. In dimension reduction approaches, one first reduces the dimensionality of the original data set by removing less important variables or by transforming the original data set into a low-dimensional space and then applies conventional clustering algorithms to the new data set. In feature selection approaches, one finds the dimensions on which data points are correlated. In both dimension reduction and feature selection approaches, it is necessary to prune off some variables, which may lead to significant loss of information. This can be illustrated by considering a three-dimensional data set that has three clusters: one embedded in the (x, y)-plane, another embedded in the (y, z)-plane, and the third embedded in the (z, x)-plane. For such a data set, application of a dimension reduction or a feature selection method is unable to recover all the clustering structures, because the three clusters are formed in different subspaces. In general, clustering algorithms based on dimension reduction or feature selection techniques generate clusters that may not fully reflect the original structure of a given data set.

This difficulty that conventional clustering algorithms encounter in dealing with high-dimensional data sets motivates the concept of subspace clustering or projected clustering

Table 15.1. *List of some subspace clustering algorithms. Num refers to numerical and Mix refers to mixed-type.*

Algorithms	**Data Type**	**H/P**
CLIQUE (Agrawal et al., 1998)	Num	Other
ENCLUS (Cheng et al., 1999)	Num	Other
MAFIA (Goil et al., 1999)	Num	Other
PROCLUS (Aggarwal et al., 1999)	Num	Partitioning
ORCLUS (Aggarwal and Yu, 2000)	Num	Partitioning
FINDIT (Woo and Lee, 2002)	Num	Partitioning
FLOC (Yang et al., 2002a)	Num	Partitioning
DOC (Procopiuc et al., 2002)	Num	Partitioning
PART (Cao and Wu, 2002)	Num	Hierarchical
CLTree (Liu et al., 2000)	Num	Other
COSA	Mix	Other

(Agrawal et al., 1998), whose goal is to find clusters embedded in subspaces of the original data space with their own associated dimensions. In other words, subspace clustering finds clusters and their relevant attributes from a data set. Table 15.1 lists a few popular subspace clustering algorithms that will be introduced in this chapter.

15.1 CLIQUE

CLIQUE (Agrawal et al., 1998) is a clustering algorithm that is able to identify dense clusters in subspaces of maximum dimensionality. It is also the first subspace clustering algorithm. This algorithm takes two parameters: ξ, which specifies the number of intervals in each dimension, and τ, which is the density threshold. The output is clusters, each of which is represented by a minimal description in the form of a disjunct normal form (DNF) expression. One disadvantage of this algorithm is that it can only find clusters embedded in the same subspace. The clustering model can be described as follows.

Let $\mathcal{A} = \{A_1, A_2, \dots, A_d\}$ be a set of bounded, totally ordered numerical domains (for categorical data, each A_i should be a finite set of categorical values). Let $\mathcal{S} = A_1 \times A_2 \times \cdots \times A_d$ be a d-dimensional numerical space. Let $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ be a database, where $\mathbf{x}_i = (\mathbf{x}_{i1}, \mathbf{x}_{i2}, \dots, \mathbf{x}_{id})$, and the jth component of $\mathbf{x}_i$ is drawn from A_j.

The data space $\mathcal{S}$ is partitioned into nonoverlapping rectangular units that are obtained by partitioning each dimension into ξ (an input parameter) intervals of equal length. Each unit u is the intersection of one interval from each dimension, and it has the form $\{u_1, u_2, \dots, u_d\}$, where $u_i = [l_i, h_i)$ is a right open interval in the partitioning of dimension A_i. A data point $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d)$ is said to be contained in a unit $u = \{u_1, u_2, \dots, u_d\}$ if the following condition is satisfied:

$$l_i \le \mathbf{x}_i < h_i \text{ for all } u_i.$$

A unit u is said to be dense if $selectivity(u) > \tau$, where $selectivity(u)$ is the selectivity of unit u, which is defined to be the fraction of total data points contained in the unit u; the density threshold τ is another input parameter. The units in all subspaces of the original d-dimensional space can be defined similarly. Let $\mathcal{S}' = \{A_{t_1}, A_{t_2}, \ldots, A_{t_r}\}$ be any subspace of $\mathcal{S}$, where $l < d$ and $t_i < t_j$ if $i < j$. The units in $\mathcal{S}'$ are the intersection of an interval from each of the r dimensions of the subspace $\mathcal{S}'$.

In the CLIQUE algorithm, a cluster is defined to be a maximal set of connected dense units in r dimensions. Two r-dimensional units are said to be connected if they have a common face or if there exists another r-dimensional unit such that the unit connects the two units. Two units $u_1 = \{r_{t_1}, r_{t_2}, \ldots, r_{t_r}\}$ and $u_2 = \{r'_{t_1}, r'_{t_2}, \ldots, r'_{t_r}\}$ are said to have a common face if there are $l - 1$ dimensions, assumed to be $A_{t_1}, A_{t_2}, \ldots, A_{t_r}$, such that $r_{t_j} = r'_{t_j}$ $(1 \leq j \leq l - 1)$ and either $h_{t_r} = l'_{t_r}$ or $h'_{t_r} = l_{t_r}$.

A region in r dimensions is an axis-parallel rectangular r-dimensional set that can be expressed as unions of units. A region R is said to be contained in a cluster C if $R \cap C = R$. A region R contained in a cluster C is said to be maximal if no proper superset of R is contained in C. A minimal description of a cluster C is a set $\mathcal{R}$ of maximal regions such that their union equals C but the union of any proper subset of $\mathcal{R}$ does not equal C.

The CLIQUE algorithm consists of three steps. In the first step, the subspaces that contain clusters are identified. In the second step, the clusters embedded in the subspaces identified in step 1 are found. Finally, a minimal description of each cluster is generated.

In the first step, a bottom-up algorithm is employed to find the dense units. The essential observation is that if a set of points S is a cluster in an r-dimensional space, then S is also part of a cluster in any $(r - 1)$-dimensional projections of this space. Based on this fact, the algorithm proceeds level by level. It first determines one-dimensional dense units, and then continues iteratively: when the $(r - 1)$-dimensional dense units are determined, the r-dimensional dense units are determined as follows. Let D_{r-1} be the set of all $(r - 1)$-dimensional dense units, and let C_r be the set of r-dimensional units generated from D_{r-1} as follows. For two units u_1, and u_2 in D_{r-1}, if $u_1.a_j = u_2.a_j$, $u_1.l_j = u_2.l_j$, and $u_1.h_j = u_2.h_j$ for $j = 1, 2, \ldots, r - 2$, and $u_1.a_{r-1} < u_2.a_{r-1}$, then insert $u = u_1.[l_1, h_1) \times u_1.[l_2, h_2) \times \cdots \times u_1.[l_{r-1}, h_{r-1}) \times u_2.[l_{r-1}, h_{r-1})$ into C_l, where $u.a_i$ is the ith dimension of unit u and $u.[l_i, h_j)$ is the interval in the ith dimension of u, and the relation $<$ represents lexicographic ordering on attributes. Then D_r is obtained by discarding those dense units from C_r that have a projection in $(r - 1)$ dimensions that is not included in C_{r-1}.

In order to make the first step faster, the minimal description length (MDL) principle is applied to decide which subspaces and corresponding dense units are interesting. Assume there are $S_1, S_2, \ldots, S_m$ subspaces found in the rth level (then $S_1, S_2, \ldots, S_m$ are r-dimensional subspaces). Let $c(S_j)$ be the coverage of subspace S_j defined as

$$c(S_j) = \sum_{u \in S_j} |u|,$$

where $|u|$ is the number of points that fall inside u. Subspaces with large coverage are selected to form candidate units in the next level of the dense unit generation algorithm, and the rest are pruned. Let the subspaces be sorted in decreasing order of their coverage. The sorted list is divided into the selected set I and the pruned set P. Assume $c(S_1) > c(S_2) >$

$\cdots > c(S_m)$. Then $I = \{S_1, S_2, \ldots, S_{i_0}\}$ and $P = \{S_{i_0+1}, S_{i_0+2}, \ldots, S_m\}$ for some i. In CLIQUE, i_0 is selected such that the total length of encoding $CL(i)$ is minimized, where $CL(i)$ is defined as

$$CL(i) = \log_2(\mu_I(i)) + \sum_{j=1}^{i} \log_2(|c(S_j) - \mu_I(i)|) \\ + \log_2(\mu_P(i)) + \sum_{j=i+1}^{m} \log_2(|c(S_j) - \mu_P(i)|),$$

where $\mu_I(i)$ and $\mu_P(i)$ are defined as

$$\mu_I(i) = \left\lceil \sum_{j=1}^{i} \frac{c(S_j)}{i} \right\rceil,$$

$$\mu_P(i) = \left\lceil \sum_{j=i+1}^{m} \frac{(S_j)}{n-i} \right\rceil,$$

where $\lceil\cdot\rceil$ is the ceiling function, i.e., $\lceil x \rceil$ is the least integer not less than x.

The second step of the CLIQUE algorithm is to find the clusters based on a set of dense units $\mathcal{C}$ found in the first step. All units in $\mathcal{C}$ are in the same r-dimensional space S. In this step, $\mathcal{C}$ will be partitioned into $C_1, C_2, \ldots, C_k$ such that all units in C_i are connected and units in different groups are not connected. This process is converted to finding connected components in a graph defined as follows: graph vertices correspond to dense units, and two vertices have an edge between them if and only if the corresponding dense units have a common face. The depth-first search algorithm (Hopcroft and Tarjan, 1973; Aho et al., 1974) is used to find the connected components of the graph.

The third step is to generate minimal cluster descriptions for each cluster. The input to this step is disjoint sets of connected r-dimensional units in the same subspace. Given a cluster C in an r-dimensional subspace S, a set $\mathcal{R}$ of regions in the same subspace S is said to be a cover of C if every region in $\mathcal{R}$ is contained in C and each unit in C is contained in at least one of the regions in $\mathcal{R}$. In this step, a minimal cover is found for each cluster by first greedily covering the cluster by a number of maximal regions and then discarding the redundant regions.

The time complexity in step 1 is $O(c^k + mk)$ for a constant c. In step 2 and subsequent steps, the dense units are assumed to be stored in memory. The total number of data structure accesses is $2kn$ in step 2.

15.2 PROCLUS

PROCLUS (PROjected CLUSting) (Aggarwal et al., 1999) is a variation of the k-medoid algorithm (Kaufman and Rousseeuw, 1990) in subspace clustering. The input parameters of the algorithm are the number of clusters k and the average number of dimensions l; the output is a partition $\{C_1, C_2, \ldots, C_k, O\}$ together with a possibly different subset of dimensions P_i for each cluster C_i, with O denoting the outlier cluster.

The algorithm consists of three phases: the initialization phase, the iteration phase, and the refinement phase. In the initialization phase, a random set of k medoids is chosen by applying the greedy technique (Gonzalez, 1985; Ibaraki and Katoh, 1988) to samples of the original data set. In the iteration phase, the algorithm progressively improves the quality of medoids by iteratively replacing the bad medoids with new ones. The last phase computes new dimensions associated with each medoid and reassigns the points to the medoids relative to the new sets of dimensions. The PROCLUS algorithm finds out the subspace dimensions of each cluster via a process of evaluating the locality of the space near it.

Algorithm 15.1. The PROCLUS algorithm.

```
Require: $D$-Data set, $k$-Number of clusters, $l$-Average dimensions of cluster;
 1: Let $A, B$ be constant integers;
 2: Draw a sample $S$ of size $A \cdot k$ randomly;
 3: Let medoids set $M \Leftarrow Greedy(S, B \cdot k)$;
 4: Let $BestObjective \Leftarrow \infty$ and $M_{current} \Leftarrow$ random set of medoids $\{m_1, m_2, \ldots, m_k\} \subset M$;
 5: repeat
 6:   Let $\delta_i$ be the distance to the nearest medoid from $m_i$ for $i = 1, 2 \ldots, k$;
 7:   Let $L_i$ be the set of points in a sphere centered at $m_i$ with radius $\delta_i$ for $i = 1, 2, \ldots, k$;
 8:   $(P_1, P_2, \ldots, P_k) \Leftarrow FindDimensions(k, l, L_1, L_2, \ldots, L_k)$;
 9:   $(C_1, C_2, \ldots, C_k) \Leftarrow AssignPoints(P_1, P_2, \ldots, P_k)$;
10:   $Objective \Leftarrow EvaluateClusters(C_1, \ldots, C_k, P_1, \ldots, P_k)$;
11:   if $Objective < BestObjective$ then
12:     Let $BestObjective \Leftarrow Objective$, $M_{best} \Leftarrow M_{current}$;
13:     Compute the bad medoids in $M_{best}$;
14:   end if
15:   Compute $M_{current}$ by replacing the bad medoids in $M_{best}$ with random points from $M$;
16: until Stop criterion
17: $(P_1, P_2, \ldots, P_k) \Leftarrow FindDimensions(k, l, L_1, L_2, \ldots, L_k)$;
18: $(C_1, C_2, \ldots, C_k) \Leftarrow AssignPoints(P_1, P_2, \ldots, P_k)$;
19: Return $M_{best}, P_1, P_2, \ldots, P_k$;
```

The pseudocode of the algorithm is described in Algorithm 15.1. In the initialization phase, a medoid candidate set M is selected using the greedy technique. In order to reduce the running time of the initialization phase, a sample S is drawn randomly from the original data set. The procedure of $Greedy(S, t)$ can be described as follows:

```
1: $M \Leftarrow \{m_1\}$ {$m_1$ is a random point of $S$};
2: Let $dist(x) \Leftarrow d(x, m_1)$ for each $x \in S \backslash M$;
3: for $i = 2$ to $t$ do
4:   Let $m_i \in S \backslash M$ be such that $dist(m_i) = \max\{dist(x) : x \in S \backslash M\}$;
5:   $M \Leftarrow M \cup \{m_i\}$;
6:   Let $dist(x) \Leftarrow \min\{dist(x), d(x, m_i)\}$;
```

```
7: end for
8: Return $M$.
```

In the iteration phase, the quality of medoids is improved progressively by iteratively replacing the bad medoids. This phase consists of three major procedures: $FindDimensions$, $AssignPoints$, and $EvaluateClusters$. In order to find the subspace dimensions, PROCLUS evaluates the locality of the space near the medoids. Let L_i be the set of points in a sphere centered at m_i with radius δ_i, where $\delta_i = \min_{1\le j\le k, j\ne i} d(m_i, m_j)$. Let X_{ij}, Y_i, and σ_i be defined as

$$X_{ij} = \frac{1}{|L_i|}\sum_{\mathbf{x}\in L_i} d(\mathbf{x}_j, m_{ij}),$$

$$Y_i = \sum_{j=1}^{d} \frac{X_{ij}}{d},$$

$$\sigma_i = \left(\frac{1}{d-1}\sum_{j=1}^{d}(X_{ij} - Y_i)^2\right)^{\frac{1}{2}}$$

for $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, where $\mathbf{x}_j$ and m_{ij} are the values of the jth dimension of $\mathbf{x}$ and m_i, respectively, and $d(\cdot, \cdot)$ is the Manhattan segmental distance.

Let $Z_{ij} = \frac{X_{ij}-Y_i}{\sigma_i}$ for $i = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$. The goal of the procedure $FindDimensions$ is to pick up the $k \cdot l$ numbers with the least values of Z_{ij} subject to the constraint that there are at least two dimensions for each cluster and to put dimension j to P_i if Z_{ij} is picked. This can be achieved by the greedy technique.

The $AssignPoints$ procedure assigns each point $\mathbf{x}$ in the data set to the cluster C_i such that $d_{P_i}(\mathbf{x}, m_i)$ has the lowest values among $d_{P_1}(\mathbf{x}, m_1), \ldots, d_{P_k}(\mathbf{x}, m_k)$, where $d_{P_i}(\mathbf{x}, m_i)$ is the Manhattan segmental distance from $\mathbf{x}$ to the medoid m_i relative to dimensions P_i, i.e.,

$$d_{P_i}(\mathbf{x}, m_i) = \frac{1}{|P_i|}\sum_{j\in P_i} |\mathbf{x}_j - m_{ij}|.$$

The $EvaluateClusters$ procedure evaluates the quality of a set of medoids as the average Manhattan segmental distance from the points to the centroids of the clusters to which they belong. The quantity that indicates the goodness of a clustering is defined as

$$\frac{1}{n}\sum_{i=1}^{k} |C_i| \cdot w_i, \tag{15.1}$$

where n is the number of points in the data set and w_i ($i = 1, 2, \ldots, k$) are weights defined as

$$w_i = \frac{1}{|P_i| \cdot |C_i|}\sum_{j\in P_i}\sum_{\mathbf{x}\in C_i} |\mathbf{x}_j - \mathbf{z}_{ij}|,$$

where P_i is the dimension associated with C_i, and z_i is the centroid of C_i, i.e.,

$$\mathbf{z}_i = \frac{1}{|C_i|}\sum_{\mathbf{x}\in C_i} \mathbf{x}.$$

The optimal clustering should have a minimum quantity, which is defined in (15.1). The goal of the iteration phase is to find an approximation of such an optimal clustering. Also in the iteration phase, bad medoids are determined. The medoid of any cluster that has less than $\frac{n}{k} \cdot \sigma_{min}$ points is bad, where σ_{min} is a constant smaller than 1. In PROCLUS, σ_{min} is set to be 0.1. Bad medoids may be outliers.

After the iteration phase, the best set of medoids is found. The refinement phase does one more pass over the data to improve the quality of the clustering. Outliers are also handled in the refinement phase. The running time for computing the segmental distances is $O(nkl)$ for each iteration.

15.3 ORCLUS

ORCLUS (arbitrarily ORiented projected CLUSter generation) (Aggarwal and Yu, 2000) is an extension of PROCLUS. It diagonalizes the covariance matrix of each cluster and finds information about projection subspaces from the diagonalization of the covariance matrix. In order to make the algorithm scale to large databases, ORCLUS also uses extended cluster feature (ECF) vectors (Zhang et al., 1996) and a progressive random sampling approach in the algorithm.

In the algorithm, a generalized projected cluster is defined for a pair (C, P), where C is a set of data points and P is a set of vectors, such that the data points in C are closely clustered in the subspace defined by the vectors in P. Unlike CLIQUE (Agrawal et al., 1998), the subspaces for different clusters found by the ORCLUS algorithm may be different.

The ORCLUS algorithm takes two input parameters: the number of clusters k and the dimensionality l of the subspace in which each cluster is reported. The output of the algorithm consists of two parts: a $(k + 1)$-way partition $\{C_1, C_2, \dots, C_k, O\}$ of the data set and a possibly different orthogonal set P_i of vectors for each cluster C_i $(1 \leq i \leq k)$, where O is the outlier cluster, which can be assumed to be empty. For each cluster C_i, the cardinality of the corresponding set P_i is l, which is the user-specified parameter.

ALGORITHM 15.2. The pseudocode of the ORCLUS algorithm.

```
Require: $D$-Data set, $k$-Number of clusters, and $l$-Number of dimensions;
 1: Pick $k_0 > k$ initial data points from $D$ and denote them by $S = \{s_1, s_2, \dots, s_{k_0}\}$;
 2: Set $k_c \Leftarrow k_0$ and $l_c \Leftarrow d$;
 3: For each $i$, set $P_i$ to be the original axis system;
 4: Set $\alpha \Leftarrow 0.5$ and compute $\beta$ according to equation (15.3);
 5: while $k_c > k$ do
 6:    $(s_1, \dots, s_{k_c}, C_1, \dots, C_{k_c}) = Assign(s_1, \dots, s_{k_c}, P_1, \dots, P_{k_c})$ {find the partitioning
       induced by the seeds};
 7:    Set $k_{new} \Leftarrow \max\{k, k_c \cdot \alpha\}$ and $l_{new} \Leftarrow \max\{l, l_c \cdot \beta\}$;
 8:    $(s_1, \dots, s_{k_{new}}, C_1, \dots, C_{k_{new}}, P_1, \dots, P_{k_{new}}) = Merge(C_1, \dots, C_{k_c}, k_{new}, l_{new})$;
 9:    Set $k_c \Leftarrow k_{new}$ and $l_c \Leftarrow l_{new}$;
10: end while
11: for $i = 1$ to $k$ do
12:    $P_i = FindVectors(C_i, l)$;
```

```
13: end for
14: $(s_1, \dots, s_k, C_1, \dots, C_k) = Assign(s_1, \dots, s_k, P_1, \dots, P_k)$;
15: Output $(C_1, \dots, C_k)$;
```

The ORCLUS algorithm consists of a number of iterations, in each of which a variant of the hierarchical merging method is applied in order to reduce the number of current clusters by the factor $\alpha < 1$. Initially, the dimensionality of each cluster is equal to the full dimensionality d. In each iteration, the dimensionality of the current clusters is reduced by the factor $\beta < 1$. Finally, the dimensionality of the clusters will be reduced gradually to the user-specified dimensionality l. In order to make the algorithm fast, a small number k_0 of initial points are picked as seeds. At each stage of the algorithm, there is a projected cluster (C_i, P_i) associated with each seed s_i. At each stage of the algorithm, k_c and l_c denote the number of current clusters and the dimensionality of the current clusters, respectively.

In order to make the reduction from k_0 to k clusters occur in the same number of iterations as the reduction from $l_0 = d$ to l dimensions, the values of α and β must satisfy the relationship

$$k_0\alpha^N = k \text{ and } l_0\beta^N = l,$$

where N is the number of iterations, which is equivalent to

$$\log_{\frac{1}{\alpha}}\left(\frac{k_0}{k}\right) = \log_{\frac{1}{\beta}}\left(\frac{d}{l}\right). \tag{15.2}$$

In the ORCLUS algorithm, the value of α is set to 0.5 and the value of β is calculated according to equation (15.2), i.e.,

$$\beta = -\exp\left(\frac{(\ln\frac{d}{l})\cdot(\ln\frac{1}{\alpha})}{\ln\frac{k_0}{k}}\right). \tag{15.3}$$

The ORCLUS algorithm is briefly described in Algorithm 15.2. In the algorithm, there are three subprocedures: $Assign(s_1, \dots, s_{k_c}, P_1, \dots, P_{k_c})$, $Merge(s_1, \dots, s_{k_c}, k_{new}, l_{new})$, and $FindVectors(C_i, q)$. Like most partitional clustering algorithms, the $Assign$ procedure assigns each data point to its nearest seed according to the projected distance. Let $\mathbf{x}$ and $\mathbf{y}$ be two data points in the original d-dimensional space, and let $P = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_p\}$ be a set of orthonormal vectors. Then the projected distance between $\mathbf{x}$ and $\mathbf{y}$ in the subspace defined by the vectors in P is defined to be the Euclidean distance between their projections in the p-dimensional space represented by P, i.e.,

$$d_P(\mathbf{x}, \mathbf{y}, P) = \left(\sum_{i=1}^{p}(\mathbf{x}\cdot\mathbf{e}_i - \mathbf{y}\cdot\mathbf{e}_i)^2\right)^{\frac{1}{2}}, \tag{15.4}$$

where $\mathbf{x}\cdot\mathbf{e}_i$ and $\mathbf{y}\cdot\mathbf{e}_i$ denote the dot-products of $\mathbf{x}$ and $\mathbf{e}_i$ and $\mathbf{y}$ and $\mathbf{e}_i$, respectively.

Algorithm 15.3. ***Assign*$(s_1, \dots, s_{k_c}, P_1, \dots, P_{k_c})$.**

```
Require: $D$-Data set;
1: Set $C_i \Leftarrow \Phi$ for $i = 1, 2, \dots, k_c$;
2: for all data points $\mathbf{x} \in D$ do
3:     Let $d_{min} \Leftarrow d_P(\mathbf{x}, s_1, P_1)$ and $i_{min} \Leftarrow 1$;
```

```
4:     for $i = 2$ to $k_c$ do
5:         if $d_{min} > d_P(\mathbf{x}, s_i, P_i)$ then
6:             $d_{min} \Leftarrow d_P(\mathbf{x}, s_i, P_i)$;
7:             $i_{min} \Leftarrow i$;
8:         end if
9:     end for
10:    Assign $\mathbf{x}$ to $C_{i_{min}}$;
11: end for
12: Set $s_i \Leftarrow \bar{X}(C_i)$ for $i = 1, 2, \ldots, k_c$ {$\bar{X}(C_i)$ is the centroid of cluster $C_i$};
13: Return $(s_1, \ldots, s_{k_c}, C_1, \ldots, C_{k_c})$;
```

The *Assign* procedure is described in Algorithm 15.3. In this procedure, the vector sets P_i ($i = 1, 2, \ldots, k_c$) are fixed, and each data point is assigned to the nearest cluster according to the projected distance defined in (15.4). Inside the *Assign* procedure, $\bar{X}(C)$ denotes the centroid of C, which is defined as

$$\bar{X}(C) = \sum_{\mathbf{x} \in C} \frac{\mathbf{x}}{|C|}. \tag{15.5}$$

ALGORITHM 15.4. *Merge*$(C_1, \ldots, C_{k_c}, K_{new}, l_{new})$.

```
1: if $k_{new} \le k$ then
2:     Exit;
3: end if
4: for $i = 1$ to $k_c$ do
5:     for $j = i + 1$ to $k_c$ do
6:         $P_{ij} = FindVectors(C_i \cup C_j, l_{new})$;
7:         $s_{ij} = \bar{X}(C_i \cup C_j)$ {centroid of $C_i \cup C_j$};
8:         $r_{ij} = R(C_i \cup C_j, P_{ij})$;
9:     end for
10: end for
11: while $k_c > k_{new}$ do
12:     Find $i', j'$ such that $r_{i'j'} = \min_{1 \le i < j \le k_c} r_{ij}$;
13:     Set $s_{i'} \Leftarrow s_{i'j'}$, $C_{i'} \Leftarrow C_{i'} \cup C_{j'}$, $P_{i'} \Leftarrow P_{i'j'}${merge $C_{i'}$ and $C_{j'}$};
14:     Discard seed $s_{j'}$ and $C_{j'}$, renumber the seeds and clusters indexed larger than $j'$ by
        subtracting 1, and renumber $s_{ij}, P_{ij}, r_{ij}$ correspondingly for any $i, j \ge j'$;
15:     for $j = 1, j \ne i'$ to $k_c - 1$ do
16:         Update $P_{i'j}, s_{i'j}$ and $r_{i'j}$ if $i' < j$; otherwise update $P_{ji'}, s_{ji'}$, and $r_{ji'}$;
17:     end for
18:     $k_c \Leftarrow k_c - 1$;
19: end while
20: Return $(s_1, \ldots, s_{k_{new}}, C_1, \ldots, C_{k_{new}}, P_1, \ldots, P_{k_{new}})$;
```

The *Merge* procedure is used to reduce the number of clusters from k_c to $k_{new} = \alpha \cdot k_c$ during a given iteration. The pseudocode of the *Merge* procedure is described in

Algorithm 15.4. In this procedure, the two closest pairs of current clusters are merged successively. The closeness between two clusters is measured by projected energy. Let (C, P) be a projected cluster. Then the projected energy of cluster C in subspace P is defined as

$$R(C, P) = \sum_{\mathbf{x}\in C} \frac{1}{|C|} d_P^2(\mathbf{x}, \bar{X}(C), P), \tag{15.6}$$

where $d_P(\cdot, \cdot, \cdot)$ is the projected distance defined in (15.4), and $\bar{X}(C)$ is the centroid of C, which is defined in (15.5).

Algorithm 15.5. *FindVectors*(C, q).

```
1: Compute the $d \times d$ covariance matrix $M$ for $C$;
2: Find the eigenvectors and eigenvalues of matrix $M$;
3: Let $P$ be the set of eigenvectors corresponding to the smallest $q$ eigenvalues;
4: Return $P$;
```

The $FindVectors$ procedure (Algorithm 15.5) is used to determine the optimal subspace associated with each cluster given the dimensionality. This is achieved by finding the eigenvalues and eigenvectors of the covariance matrix for the cluster. For a given cluster C, the subspace associated with C is defined by the orthonormal eigenvectors with the least spread (eigenvalues).

The ORCLUS algorithm can also handle outliers by measuring the projected distance between each data point and the seed of the cluster to which it belongs. More specifically, for each $i \in \{1, \dots, k_c\}$, let δ_i be the projected distance between the seed s_i and its nearest other seed in subspace P_i, let $\mathbf{x}$ be any data point, and let s_r be the seed to which $\mathbf{x}$ is assigned. If the projected distance between $\mathbf{x}$ and s_r in subspace P_r is larger than δ_r, then $\mathbf{x}$ is treated as an outlier.

In order to make the algorithm scalable to very large databases, the concept of ECF-vectors is used, as in BIRCH (Zhang et al., 1996). For each cluster, there is an ECF-vector associated with it. The ECF-vector for each cluster in the ORCLUS algorithm consists of $d^2 + d + 1$ entries. Let C be a cluster. Then the ECF-vector $\mathbf{v}(C) = (\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{d^2+d+1})$ for C is defined as

$$\begin{aligned} \mathbf{v}_{(i-1)d+j} &= \sum_{\mathbf{x}\in C} \mathbf{x}_i \cdot \mathbf{x}_j, \quad i, j = 1, 2, \dots, d, \\ \mathbf{v}_{d^2+i} &= \sum_{\mathbf{x}\in C} \mathbf{x}_i, \quad i = 1, 2, \dots, d, \\ \mathbf{v}_{d^2+d+1} &= |C|, \end{aligned}$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote the ith and jth components of the data point $\mathbf{x}$, respectively.

From the definition of the ECF-vectors, one can see that an ECF-vector can be divided into three parts. The ECF-vectors defined above have some good properties. For example, the ECF-vector satisfies the additive property, i.e., the ECF-vector for the union of two clusters is equal to the sum of the corresponding ECF-vectors. Also, the covariance matrix for a cluster can be derived directly from the ECF-vector of the cluster. Namely, let C be a cluster and $\mathbf{v} = \mathbf{v}(C)$. Then the covariance matrix $M = (m_{ij})_{d\times d}$ for C is

$$m_{ij} = \frac{\mathbf{v}_{(i-1)d+j}}{\mathbf{x}_{d^2+d+1}} - \frac{\mathbf{v}_{d^2+i} \cdot \mathbf{v}_{d^2+j}}{\mathbf{v}^2_{d^2+d+1}}.$$

The running time for the *Merge* procedure is $O(k_0^3 + k_0^2 d)$, and the running time for the *Assign* procedure is $O(\frac{d}{1-\alpha}k_0 n)$. Since the running time for subspace determinations is strictly dominated by the subspace determinations during the *Merge* phase, this running time can be ignored. The total running time of the ORCLUS algorithm is therefore $O(k_0^3 + k_0 nd + k_0^2 d)$.

15.4 ENCLUS

ENCLUS (ENtropy-based CLUStering) (Cheng et al., 1999) is an entropy-based subspace clustering algorithm for clustering numerical data. It can find arbitrarily shaped clusters embedded in the subspaces of the original data space. It follows similar approaches suggested by CLIQUE, but does not make any assumptions about the cluster shapes and hence is capable of finding arbitrarily shaped clusters embedded in subspaces. The ENCLUS algorithm searches for subspace clusters based on the fact that a subspace with clusters typically has lower entropy than a subspace without clusters. In ENCLUS, the entropy of a data set is defined as

$$H(X) = -\sum_{x\in\chi} d(x)\log d(x), \tag{15.7}$$

where X is a set of variables, χ is the set of all units in the space formed by X, and $d(x)$ is the density of a unit x in terms of the percentage of data points contained in unit x. ENCLUS makes use of the relationship between an entropy and a clustering criterion to identify the subspaces with good clustering, and then applies methods developed in CLIQUE or other algorithms to identify all clusters in the subspaces identified. Entropy and clustering criteria are related. For example, the entropy decreases as the density of the dense units increases, and we also have the following correspondence between entropy and dimension correlations:

$$H(V_1, V_2, \dots, V_i) = \sum_{j=1}^{i} H(V_j) \tag{15.8}$$

if and only if $V_1, V_2, \dots, V_i$ are independent;

$$H(V_1, \dots, V_i, Y) = H(V_1, V_2, \dots, V_i)$$

if and only if Y is a function of $V_1, V_2, \dots, V_i$.

To discover the subspaces with good clusters, the algorithm uses the fact of downward closure, i.e., if an l-dimensional subspace $V_1, V_2, \dots, V_l$ has good clustering, so do all $(l-1)$-dimensional projections of this space. To discover the correlations between variables, the algorithm uses the fact of upward closure, i.e., if a set of dimensions S is correlated, so is every superset of S. Upward closure allows us to find the minimally correlated subspaces.

A set of variables $V_1, V_2, \ldots, V_i$ are correlated if equation (15.8) is not satisfied. Define the *interest* of a set of variables by

$$interest(\{V_1, V_2, \ldots, V_i\}) = \sum_{j=1}^{i} H(V_j) - H(V_1, V_2, \ldots, V_i). \tag{15.9}$$

Let S_l be the set of l-dimensional significant subspaces and NS_l be the set of l-dimensional subspaces with good clustering but not minimal correlation. The procedure for mining significant subspaces is described in Algorithm 15.6. In this procedure, one first builds grids in the subspace c and calculates the density of each cell. Then the entropy for this subspace is computed based on the density information using the formula in (15.7). The candidate set C_{l+1} is generated as follows: select $p, q \in NS_l$ that satisfy $p.dim_i = q.dim_i$ for $i = 1, 2, \ldots, l-1$ and $p.dim_l < q.dim_l$ and then insert p and $q.dim_l$ into C_{l+1}.

ALGORITHM 15.6. ENCLUS procedure for mining significant subspaces.

```
Require: $D$-Data set, $\omega$-Entropy threshold, $\epsilon$-Interest threshold;
 1: Let $l \Leftarrow 1$ and $C_1$ be one-dimensional subspaces;
 2: while $C_l \neq \Phi$ do
 3:   for all $c \in C_l$ do
 4:     Calculate the density $f_c(\cdot)$;
 5:     Calculate the entropy $H(c)$ from $f_c(\cdot)$;
 6:     if $H(c) < \omega$ then
 7:       if $interest(c) > \epsilon$ then
 8:         $S_l \Leftarrow S_l \cup c$;
 9:       else
10:         $NS_l \Leftarrow NS_l \cup c$;
11:       end if
12:     end if
13:   end for
14:   Generate $C_{l+1}$ from $NS_l$;
15:   $l \Leftarrow l + 1$;
16: end while
17: Output $\bigcup_l S_l$ as significant subspaces.
```

Since the mining of high-dimensional clusters is often avoided by the above procedure, another procedure used to mine interesting subspaces has been proposed. Define the *interest_gain* of a set of variables by

$$\begin{aligned}&interest\_gain(\{V_1, V_2, \ldots, V_i\})\\ &= interest(\{V_1, V_2, \ldots, V_i\}) - \max_{1 \leq l \leq i}\{interest(\{V_1, V_2, \ldots, V_i\} \backslash \{V_l\})\},\end{aligned}$$

where $interest(\cdot)$ is defined in (15.9). The procedure for mining interesting subspaces is similar to the one for mining significant subspaces. The modified procedure for mining interesting subspaces is described in Algorithm 15.7. The procedure for mining interesting subspaces has more candidates and a longer running time than the one for mining significant subspaces.

Algorithm 15.7. ENCLUS procedure for mining interesting subspaces.

```
Require: $D$-Data set, $\omega$-Entropy threshold, $\epsilon'$-Interest threshold;
 1: Let $l \Leftarrow 1$ and $C_1$ be one-dimensional subspaces;
 2: while $C_l \neq \Phi$ do
 3:    for all $c \in C_l$ do
 4:       Calculate the density $f_c(\cdot)$;
 5:       Calculate the entropy $H(c)$ from $f_c(\cdot)$;
 6:       if $H(c) < \omega$ then
 7:          if $interest\_gain(c) > \epsilon'$ then
 8:             $I_l \Leftarrow I_l \cup c$;
 9:          else
10:             $NI_l \Leftarrow NI_l \cup c$;
11:          end if
12:       end if
13:    end for
14:    Generate $C_{l+1}$ from $I_l \cup NI_l$;
15:    $l \Leftarrow l + 1$;
16: end while
17: Output $\bigcup_l S_l$ as interesting subspaces.
```

15.5 FINDIT

FINDIT (a Fast and INtelligent subspace clustering algorithm using DImension voTing) (Woo and Lee, 2002) is a subspace clustering algorithm that uses a dimension-oriented distance measure and a dimension voting policy to determine the correlated dimensions for each cluster.

The dimension-oriented distance *dod* in FINDIT is defined as

$$dod_\epsilon(\mathbf{x}, \mathbf{y}) = \max\{dod_\epsilon(\mathbf{x} \to \mathbf{y}), dod_\epsilon(\mathbf{y} \to \mathbf{x})\}, \tag{15.10}$$

where $dod_\epsilon(\mathbf{x} \to \mathbf{y})$ is the directed *dod* defined as

$$dod_\epsilon(\mathbf{x} \to \mathbf{y}) = |Q_\mathbf{x}| - |\{j : |\mathbf{x}_j - \mathbf{y}_j| \leq \epsilon, j \in Q_\mathbf{x} \cap Q_\mathbf{y}\}|, \tag{15.11}$$

where $Q_\mathbf{x}$ is the subspace dimension in which the dimensional values of the data point **x** are meaningful.

The FINDIT algorithm consists of three phases: the sampling phase, the cluster-forming phase, and the data-assigning phase. In the sampling phase, two samples S and M are made for the original data set by Chernoff bounds (Motwani and Raghavan, 1995; Guha et al., 1998) according to the data set size and a parameter $C_{minsize}$. Let $C_b(k, n)$ be the minimum size of sample S such that every cluster has more than ξ data points in the sample with probability $1 - \delta$. Then the $C_b(k, n)$ can be computed from the formula (Woo and Lee, 2002; Guha et al., 1998)

$$C_b(k, n) = \xi k\rho + k\rho \log\left(\frac{1}{\delta}\right) + k\rho\sqrt{\left(2\xi + \log\frac{1}{\delta}\right)\log\frac{1}{\delta}}, \tag{15.12}$$

where k is the number of clusters, ρ is a value satisfying the equation

$$\rho = \frac{n}{k \cdot |C_{min}|},$$

and C_{min} is the smallest cluster in the partition. FINDIT takes $\xi = 30$, $\delta = 0.01$, and $C_{min} = C_{minsize}$ in (15.12) for the sample S, and $\xi = 1$, $\delta = 0.01$, and $C_{min} = C_{minsize}$ for the sample M.

In the cluster-forming phase, the subspace dimensions of each cluster are determined by a method called dimension voting and the best medoid cluster set is formed in order to be used in the following data assignment phase. In order to find an appropriate ϵ, the cluster-forming phase is iterated several times with increasing ϵ value in $[\frac{1}{100}R, \frac{25}{100}R]$, where R is the normalized value range. For a fixed ϵ, the V nearest neighbors are sequentially searched from S for each medoid in M in order to determine the correlated dimensions of the medoid. The distance between a point $\mathbf{x} \in S$ and a medoid $\mathbf{z} \in M$ is measured as

$$dod_\epsilon(\mathbf{z}, \mathbf{x}) = d - |\{j : |\mathbf{z}_j - \mathbf{y}_j| \le \epsilon, j \in Q\}|,$$

where $Q = \{1, 2, \ldots, d\}$ and d is the number of dimensions, since no correlated dimensions have been selected for both points yet, i.e., $Q_\mathbf{x} = Q_\mathbf{z} = Q$. After selecting the V nearest neighbors, the algorithm calculates the number of selected neighbors that vote "yes" for each dimension, i.e., the number of selected neighbors from $\mathbf{z}$ is less than or equal to ϵ for each dimension, i.e.,

$$V_j = |\{j : |\mathbf{x}_j - \mathbf{z}_j| \le \epsilon, j \in Q\}|, \quad j = 1, 2, \ldots, d.$$

If $V_j \ge V_s$, then a certain correlation exists in the jth dimension, where V_s is the decision threshold. For example, if $V = 20$, then we could take $V_s = 12$.

ALGORITHM 15.8. The FINDIT algorithm.

```
Require: $D$-The Data set, $C_{minsize}$-Minimum size of clusters, $D_{mindist}$-Minimum difference between two resultant clusters;
 1: Draw two samples $S$ and $M$ from the data set according to the Chernoff bounds {sampling phase};
 2: Let $\epsilon_i \Leftarrow \frac{i}{100}R$ for $i = 1, 2, \ldots, 25$;
 3: Let $MC_i$ be the set of medoid clusters for $\epsilon_i$;
 4: Set the optimal $\epsilon \Leftarrow 0$, $MC \Leftarrow \Phi$;
 5: for $i = 1$ to $25$ do
 6:    Determine correlated dimensions for every medoid in $M$;
 7:    Assign every point in $S$ to the nearest medoid in $M$;
```

```
8:      Merge similar medoids in $M$ to make medoid cluster set $MC_i$;
9:      Refine medoid clusters in $MC_i$ {cluster-forming phase};
10: end for
11: Let $i_0$ be the subscript such that $Soundness(MC_{i_0}) = \max_{1 \le i \le 25} Soundness(MC_i)$,
    and then set $\epsilon \Leftarrow \epsilon_{i_0}$, $MC \Leftarrow MC_{i_0}$;
12: Set $min\_dod \Leftarrow d$, $nearest\_mc \Leftarrow \Phi$;
13: for all $\mathbf{x} \in D$ do
14:     for all $A \in MC_{i_0}$ do
15:         for all $\mathbf{z} \in A$ do
16:             if $dod_{\epsilon_{i_0}}(\mathbf{z} \to \mathbf{x}) = 0$ and $dod_{\epsilon_{i_0}}(\mathbf{z}, \mathbf{x}) < min\_dod$ then
17:                 $min\_dod \Leftarrow dod_{\epsilon_{i_0}}(\mathbf{z}, \mathbf{x})$, $nearest\_mc \Leftarrow A$;
18:             end if
19:         end for
20:     end for
21:     if $nearest\_mc \neq \Phi$ then
22:         Assign $\mathbf{x}$ to $nearest\_mc$;
23:     else
24:         Assign $\mathbf{x}$ to outlier cluster;
25:     end if
26: end for
```

A point $\mathbf{x}$ in S is assigned to the nearest $\mathbf{z}$ in M satisfying

$$dod_\epsilon(\mathbf{z} \to \mathbf{x}) = |Q_\mathbf{z}| - |\{j : |\mathbf{z}_j - \mathbf{x}_j| \le \epsilon, j \in Q\}| = 0,$$

where $Q_\mathbf{z}$ is the set of correlated dimensions of $\mathbf{z}$. If there is more than one such medoid, the point $\mathbf{x}$ will be assigned to the medoid which has the largest number of correlated dimensions.

In FINDIT, a hierarchical clustering algorithm is employed to cluster the medoids. The distance between two medoid clusters A and B is defined to be the weighted average between the medoids belonging to them, i.e.,

$$dod_\epsilon(A, B) = \frac{\sum\limits_{\mathbf{z}_i \in A, \mathbf{z}_j \in B} |\mathbf{z}_i| \cdot |\mathbf{z}_j| \cdot dod_\epsilon(\mathbf{z}_i, \mathbf{z}_j)}{\left(\sum\limits_{\mathbf{z}_i \in A} |\mathbf{z}_i|\right) \cdot \left(\sum\limits_{\mathbf{z}_j \in B} |\mathbf{z}_j|\right)},$$

where $|\mathbf{z}_i|$ is the number of points assigned to $\mathbf{z}_i$, $|\mathbf{z}_j|$ is defined similarly, $dod_\epsilon(\mathbf{z}_i, \mathbf{z}_j)$ is defined as

$$dod_\epsilon(\mathbf{z}_i, \mathbf{z}_j) = \max\{|Q_{\mathbf{z}_i}|, |Q_{\mathbf{z}_j}|\} - |\{l : |\mathbf{z}_{il} - \mathbf{z}_{jl}| \le \epsilon, l \in Q_{\mathbf{z}_i} \cap Q_{\mathbf{z}_j}\}|,$$

and $\mathbf{z}_{il}$ and $\mathbf{z}_{jl}$ are the values of the lth dimension of $\mathbf{z}_i$ and $\mathbf{z}_j$, respectively.

To evaluate a medoid clustering, a criterion called soundness is defined for each MC_i, the set of medoid clusters for ϵ_i, by the formula

$$Soundness(MC_i) = \sum_{A \in MC_i} |A| \cdot |KD_A|,$$

where KD_A is the set of key dimensions or correlated dimensions of the medoid cluster A, which is defined by

$$KD_A = \left\{ j : \frac{\sum\limits_{\mathbf{z}\in A} \delta_j(\mathbf{z}) \cdot |\mathbf{z}|}{\sum\limits_{\mathbf{z}\in A} |\mathbf{z}|} > \delta_0 \right\},$$

where δ_0 is a threshold that could be taken from 0.9 to 1, $|\mathbf{z}|$ is the number of points assigned to the medoid $\mathbf{z}$, and $\delta_j(\mathbf{z}) = 1$ if j is a correlated dimension of $\mathbf{z}$; otherwise it is 0.

In the data-assigning phase, the data points in the original data set are assigned to the nearest medoid cluster in the optimal medoid clustering found in the cluster-forming phase or put into the outlier cluster. The FINDIT algorithm is sketched in Algorithm 15.8.

15.6 MAFIA

MAFIA (Merging of Adaptive Finite Intervals) (Nagesh et al., 2000; Goil et al., 1999) is a parallel subspace clustering algorithm using adaptive computation of the finite intervals in each dimension that are merged to explore clusters embedded in subspaces of a high-dimensional data set. It is also a density- and grid-based clustering algorithm.

The MAFIA algorithm uses adaptive grids to partition the data space into small units and then merges the dense units to form clusters. To compute the adaptive grids, it first constructs a histogram by partitioning the data space into a number of nonoverlapping regions and mapping the data points to each cell. The procedure of computing adaptive grids is listed in Algorithm 15.9.

Algorithm 15.9. Procedure of adaptive grids computation in MAFIA the algorithm.

```
Require: $D_i$ – Domain of $i$th attribute; $N$ – Total number of data points in the data set; $a$
    – Size of the generic interval;
1: for $i = 1$ to $d$ do
2:     Divide $D_i$ into intervals of some small size $x$;
3:     Compute the histogram for each interval in the $i$th dimension, and set the value of
       the window to the maximum in the window;
4:     From left to right, merge two adjacent intervals if they are within a threshold $\beta$;
5:     if number of bins == 1 then
6:         Divide the $i$th dimension into a fixed number of equal intervals and set a threshold
           $\beta'$ for it;
7:     end if
8: end for
```

MAFIA is a parallel clustering algorithm. In its implementation, MAFIA introduces both data parallelism and task parallelism. The main properties of this algorithm are listed in Table 15.2.

Table 15.2. *Description of the MAFIA algorithm, where c is a constant, k is the highest dimensionality of any dense cells in the data set, p is the number of processors, N is the number of data points in the data set, B is the number of records in the memory buffer on each processor, and γ is the I/O access time for a block of B records from the local disk.*

Clustering Data Type	Numerical data
Cluster Shape	Centered cluster
Time Complexity	$O(c^k + \frac{N}{pB}k\gamma + \alpha Spk)$
Algorithm Type	Parallel, hierarchical

15.7 DOC

DOC (Density-based Optimal projective Clustering) (Procopiuc et al., 2002) is a Monte Carlo–based algorithm that computes a good approximation of an optimal projective cluster. The DOC algorithm can identify arbitrarily shaped clusters from the data sets, for example, image data.

In the DOC algorithm, a projective cluster of width ω is defined as a pair (C, P), where C is a subset of data points of the original data set, and P is a subset of dimensions associated with C such that

1. C is α-dense, i.e., $|C| \geq \alpha|S|$, where $\alpha \in [0, 1]$ and S is the original data set;
2. $\forall j \in P$, $\max_{\mathbf{x}\in C} \mathbf{x}_j - \min_{\mathbf{y}\in C} \mathbf{y}_j \leq \omega$, where $\omega \geq 0$;
3. $\forall j \in P^c = \{1, 2, \ldots, d\}\backslash P$, $\max_{\mathbf{x}\in C} \mathbf{x}_j - \min_{\mathbf{y}\in C} \mathbf{y}_j > \omega$, where $\omega \geq 0$.

Let $\mu : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a function such that $\mu(0, 0) = 0$ and μ is monotonically increasing in each argument. Then the quality of a projective cluster (C, P) is defined to be $\mu(|C|, |P|)$. For any $0 \leq \beta < 1$, the function μ is said to be β-balanced if $\mu(x, y) = \mu(\beta x, y + 1)$ for all $x > 0$, $y \geq 0$. One choice of such a function is $\mu(x, y) = x(\frac{1}{\beta})^y (0 < \beta < 1)$.

A projective cluster (C, P) is said to be μ-optimal if it maximizes μ over $\mathcal{P}_\alpha$, where $\mathcal{P}_\alpha$ is the set of all α-dense projective clusters of width at most ω from D. Monotonicity is required since we want the projective cluster (C, P) so that C and P are maximal. The β-balanced condition is used to specify the tradeoff between the number of data points and the number of dimensions in a cluster.

ALGORITHM 15.10. The DOC algorithm for approximating an optimal projective cluster.

```
Require: $D$-Data set; $\alpha$, $\beta$, and $\omega$;
1: Let $r \Leftarrow \frac{\log(2d)}{-\log(2\beta)}$, $m \Leftarrow \left(\frac{2}{\alpha}\right)^r \ln 4$;
2: for $i = 1$ to $\frac{2}{\alpha}$ do
3:    Choose $\mathbf{x} \in D$ uniformly at random;
4:    for $j = 1$ to $m$ do
5:       Choose $X \subseteq D$ of size $r$ uniformly at random;
```

```
6:         $P \Leftarrow \{j : |\mathbf{y}_j - \mathbf{x}_j| \le \omega, \forall \mathbf{y} \in X\}$;
7:         $C \Leftarrow D \cap B_{\mathbf{x},P}$;
8:         if $|C| < \alpha|D|$ then
9:             $(C, P) \Leftarrow (\Phi, \Phi)$;
10:        end if
11:    end for
12: end for
13: Return cluster $(C_0, P_0)$ that maximizes $\mu(|C|, |P|)$ over all computed clusters $(C, P)$.
```

Let (C, P) be a projective cluster. An intuitive geometric objective is an axis-paralleled box given as

$$B_{C,P} = [l_1, h_1] \times [l_2, h_2] \times \cdots \times [l_d, h_d],$$

where l_j and h_j $(j = 1, 2, \ldots, d)$ are defined as

$$l_j = \begin{cases} -\infty & \text{if } j \notin P, \\ \min\limits_{\mathbf{x} \in C} \mathbf{x}_j & \text{if } j \in P, \end{cases}$$

$$h_j = \begin{cases} \infty & \text{if } j \notin P, \\ \max\limits_{\mathbf{x} \in C} \mathbf{x}_j & \text{if } j \in P. \end{cases}$$

From the definition of $B_{C,P}$, a point $\mathbf{x} \in C$ if and only if $\mathbf{x}$ is contained in $B_{C,P}$. Let $\mathbf{x}$ be a data point and let $B_{\mathbf{x},P}$ be defined as

$$B_{\mathbf{x},P} = [l_1, h_1] \times [l_2, h_2] \times \cdots \times [l_d, h_d],$$

where l_j and $h_j (j = 1, 2, \ldots, d)$ are defined as

$$l_j = \begin{cases} -\infty & \text{if } j \notin P, \\ \mathbf{x}_j - \omega & \text{if } j \in P, \end{cases}$$

$$h_j = \begin{cases} \infty & \text{if } j \notin P, \\ \mathbf{x}_j + \omega & \text{if } j \in P. \end{cases}$$

Then for any $\mathbf{x} \in C$, we have $B_{C,P} \subseteq B_{\mathbf{x},P}$. The Monte Carlo algorithm for approximating a projective cluster is described in Algorithm 15.10. It has been shown that the Monte Carlo algorithm works correctly (Procopiuc et al., 2002). The total running time of the Monte Carlo algorithm described above is $O(nd^c)$, where $c = \frac{\log \alpha - \log 2}{\log(2\beta)}$.

There are three parameters in the DOC algorithm, i.e., α, β, and ω. In most experiments reported in (Procopiuc et al., 2002), these parameters are set as $\alpha = 0.1$, $\beta = 0.25$, and $\omega = 15$. Since the parameter ω controls the width of the box, it is hard to choose the right value for ω if we do not have much information about the data set. In order to choose a good value for ω, a heuristic method has been proposed in (Procopiuc et al., 2002) as follows. For a data point $\mathbf{x}_i$, let $\mathbf{y}_i$ be its nearest neighbor. Then let ω be defined as

$$\omega = r \sum_{i=1}^{n} \frac{w_i}{n},$$

where r is a small constant, and w_i $(i = 1, 2, \ldots, n)$ are given by

$$w_i = \sum_{j=1}^{d} \frac{|\mathbf{x}_{ij} - \mathbf{y}_{ij}|}{d}.$$

15.8 CLTree

CLTree (CLustering based on decision Trees) (Liu et al., 2000) is a clustering algorithm for numerical data based on a supervised learning technique called decision tree construction. The CLTree algorithm is able to find clusters in the full dimension space as well as in subspaces. The resulting clusters found by CLTree are described in terms of hyperrectangle regions. The CLTree algorithm is able to separate outliers from real clusters effectively, since it naturally identifies sparse and dense regions.

The basic idea behind CLTree is to regard each data point in the original data set as having a class Y and to assume that the data space is uniformly distributed with another type of points given class N. The main goal of CLTree is to partition the data space into data (dense) regions and empty (sparse) regions. The CLTree algorithm consists of two steps. The first step is to construct a cluster tree using a modified decision tree algorithm; the second step is to prune the cluster tree interactively. The final clusters are described as a list of hyperrectangle regions.

Decision tree construction is a technique for classification. A decision tree has two types of nodes: decision nodes, which specify some test on a single attribute, and leaf nodes, which indicate the class. Since the CLTree algorithm is designed for numerical data, the tree algorithm performs a binary split, i.e., the current space is cut into two parts. For details about the decision tree construction algorithm and its improvements, readers are referred to (Quinlan, 1993), (Shafer et al., 1996), (Gehrke et al., 1998), and (Gehrke et al., 1999).

During the decision tree construction, a different number of N points is added to each node, but not physically. In order to separate cluster regions and noncluster regions, the number of N points is added according to some rules. That is, if the number of N points inherited from the parent node of E is less than the number of Y points in E, then the number of N points for E is increased to the number of Y points in E; otherwise, the number of inherited N points is used for E.

In the CLTree algorithm, the decision tree algorithm is modified, since the N points are not physically added to the data. The first modification is to compute the number of N points in each region based on the area of the region when a cut is made, since the N points are assumed to be distributed uniformly in the current space. For example, a node E has 25 Y points and 25 N points and a cut P cuts E into two regions E_1 and E_2 in the ratio 2:3, i.e., the area of E_1 is 40% of the area of E and the area of E_2 is 60% of the area of E. Then the number of N points in E_1 is $0.4 \times 25 = 10$ and the number of N points in E_2 is $25 - 10 = 15$. The second modification is to evaluate on both sides of data points in order to find the best cuts.

Also, a new criterion is used in order to find the best cuts, since the gain criterion has problems in clustering, such as the cut with best information gain tends to cut into clusters. To do this, the relative density of a region is defined. Let E be a region. Then the relative density of E is $\frac{E.Y}{E.N}$, where $E.Y$ is the number of Y points in E and $E.N$ is defined similarly.

The basic idea of the new criterion is described as follows. For each dimension j of a node E, let d_j_cut1 be the first cut found by the gain criterion, and let the two regions separated by d_j_cut1 along dimension j be denoted by L_j and H_j, where L_j has a lower relative density than H_j; then we have $E = L_j \cup H_j$. Let d_j_cut2 be the cut of L_j found using the gain criterion. Suppose the region L_j is cut into L_j^1 and H_j^1 by d_j_cut2, where L_j^1 has a lower relative density than H_j^1; then we have $L_j = L_j^1 \cup H_j^1$. If H_j^1 is the region between d_j_cut1 and d_j_cut2, then we stop and choose d_j_cut2 as a best cut for dimension j of E. If the region between d_j_cut1 and d_j_cut2 is L_j^1, then we let d_j_cut3 be the cut of L_j^1 found using the gain criterion and choose d_j_cut3 as the best cut for dimension j of E.

After the decision tree has been constructed, it is pruned in order to produce meaningful clusters. CLTree provides two pruning methods: browsing and user-oriented pruning. In the browsing method, a user interface is built in order to let the user simply explore the tree to find meaningful clusters. In the user-oriented pruning method, the decision tree is pruned using two user-specified parameters min_y and min_rd: min_y is the percentage of the total number of data points in the data set that a region must contain, and min_rd specifies whether an N region (node) E should join an adjacent region F to form a bigger cluster region.

15.9 PART

PART (Projective Adaptive Resonance Theory) (Cao and Wu, 2002) is a new neural network architecture that was proposed to find projected clusters for data sets in high-dimensional spaces. The neural network architecture in PART is based on the well known ART (Adaptive Resonance Theory) developed by Carpenter and Grossberg (1987b,a, 1990). In PART, a so-called selective output signaling mechanism is provided in order to deal with the inherent sparsity in the full space of the high-dimensional data points. Under this selective output signaling mechanism, signals generated in a neural node in the input layer can be transmitted to a neural node in the clustering layer only when the signal is similar to the top-down weight between the two neural nodes; hence, with this selective output signaling mechanism, PART is able to find dimensions where subspace clusters can be found.

The basic PART architecture consists of three components: the input layer (comparison layer) F_1, the clustering layer F_2, and a reset mechanism (Cao and Wu, 2002). Let the nodes in the F_1 layer be denoted by $v_i, i = 1, 2, \dots, m$; the nodes in the F_2 layer be denoted by $v_j, j = m+1, \dots, m+n$; the activation of an F_1 node v_i be denoted by x_i; and the activation of an F_2 node v_j be denoted by x_j. Let the bottom-up weight from v_i to v_j be denoted by z_{ij} and the top-down weight from v_j to v_i be denoted by z_{ji}. In PART, the selective output signal of an F_1 node v_i to a committed F_2 node v_j is defined by

$$h_{ij} = h(x_i, z_{ij}, z_{ji}) = h_\sigma(f_1(x_i), z_{ji})l(z_{ij}), \tag{15.13}$$

where f_1 is a signal function; $h_\sigma(\cdot, \cdot)$ is defined as

$$h_\sigma(a, b) = \begin{cases} 1 & \text{if } d(a, b) \le \sigma, \\ 0 & \text{if } d(a, b) > \sigma, \end{cases} \tag{15.14}$$

with $d(a, b)$ being a quasi-distance function; and $l(\cdot)$ is defined as

$$l(z_{ij}) = \begin{cases} 1 & \text{if } z_{ij} > \theta, \\ 0 & \text{if } z_{ij} \leq \theta, \end{cases} \tag{15.15}$$

with θ being 0 or a small number to be specified as a threshold; σ is a distance parameter.

An F_1 node v_i is said to be active to v_j if $h_{ij} = 1$ and inactive to v_j if $h_{ij} = 0$.

In PART, an F_2 node v_j is said to be a winner either if $\Gamma \neq \phi$ and $T_j = \max \Gamma$ or if $\Gamma = \phi$ and node v_j is the next noncommitted node in the F_2 layer, where Γ is a set defined as $\Gamma = \{T_k : F_2$ node v_k is committed and has not been reset on the current trial$\}$, with T_k defined as

$$T_k = \sum_{v_i \in F_1} z_{ik} h_{ik} = \sum_{v_i \in F_1} z_{ik} h(x_i, z_{ik}, z_{ki}). \tag{15.16}$$

A winning F_2 node will become active and all other F_2 nodes will become inactive, since the F_2 layer makes a choice by a winner-take-all paradigm:

$$f_2(x_j) = \begin{cases} 1 & \text{if node } v_j \text{ is a winner,} \\ 0 & \text{otherwise.} \end{cases} \tag{15.17}$$

For the vigilance and reset mechanism of PART, if a winning (active) F_2 node v_j does not satisfy some vigilance conditions, it will be reset so that the node v_j will always be inactive during the rest of the current trial. The vigilance conditions in PART also control the degree of similarity of patterns grouped in the same cluster. A winning F_2 node v_j will be reset if and only if

$$r_j < \rho, \tag{15.18}$$

where $\rho \in \{1, 2, \ldots, m\}$ is a vigilance parameter and r_j is defined as

$$r_j = \sum_i h_{ij}. \tag{15.19}$$

Therefore, the vigilance parameter ρ controls the size of the subspace dimensions, and the distance parameter σ controls the degree of similarity in the specific dimension involved. For real-world data, it is a challenge to choose the distance parameter σ.

In PART, the learning is determined by the following formula. For a committed F_2 node v_j that has passed the vigilance test, the new bottom-up weight is defined as

$$z_{ij}^{new} = \begin{cases} \frac{L}{L-1+|X|} & \text{if } F_1 \text{ node } v_i \text{ is active to } v_j, \\ 0 & \text{if } F_1 \text{ node } v_i \text{ is inactive to } v_j, \end{cases} \tag{15.20}$$

where L is a constant and $|X|$ denotes the number of elements in the set $X = \{i : h_{ij} = 1\}$, and the new top-down weight is defined as

$$z_{ji}^{new} = (1 - \alpha) z_{ji}^{old} + \alpha I_i, \tag{15.21}$$

where $0 \leq \alpha \leq 1$ is the learning rate.

For a noncommitted winner v_j, and for every F_1 node v_i, the new weights are defined as

$$z_{ij}^{new} = \frac{L}{L-1+m}, \tag{15.22}$$

$$z_{ji}^{new} = I_i. \tag{15.23}$$

In PART, each committed F_2 node v_j represents a subspace cluster C_j. Let D_j be the set of subspace dimensions associated with C_j. Then $i \in D_j$ if and only if $l(z_{ij}) = 1$, i.e., the set D_j is determined by $l(z_{ij})$.

PART is very effective at finding the subspaces in which clusters are embedded, but the difficulty of choosing some of its parameters restricts its application. For example, it is very difficult for users to choose an appropriate value for the distance parameter σ. If we choose a relatively small value the algorithm may not capture the similarities of two similar data points; however, if we choose a relatively large value, the algorithm may not differentiate two dissimilar data points.

15.10 SUBCAD

All the subspace clustering algorithms mentioned before are designed for clustering numerical data. SUBCAD (SUBspace clustering for high-dimensional CAtegorical Data) (Gan and Wu, 2004; Gan, 2003) is a subspace clustering algorithm designed for clustering high-dimensional categorical data. In the SUBCAD algorithm, the clustering process is treated as an optimization problem. An objective function is defined in SUBCAD to measure the quality of the clustering. The main goal of SUBCAD is to find an approximation of the optimal solution based on the objective function.

Given a data set D, let Q be the set of dimensions of D, i.e., $Q = \{1, 2, \dots, d\}$, where d is the number of dimensions of D, and let Span(Q) denote the full space of the data set. Then by a subspace cluster we mean a cluster C associated with a set of dimensions P such that

1. the data points in C are "similar" to each other in the subspace Span(P) of Span(Q) (i.e., the data points in C are compact in this subspace);
2. the data points in C are sparse in the subspace Span(R), where $R = Q\backslash P$ (i.e., the data points in C are spread in this subspace).

In SUBCAD, a subspace cluster is denoted by a pair (C, P) $(P \neq \Phi)$, where P is the nonempty set of dimensions associated with C. In particular, if $P = Q$, then this cluster is formed in the whole space of the data set. Therefore, if C is a cluster with the associated set of dimensions P, then C is also a cluster in every subspace of Span(P). Hence, a good subspace clustering algorithm should be able to find clusters and the maximum associated set of dimensions. Consider, for example, the data set with five six-dimensional data points given in Table 2.1. In this data set, it is obvious that $C = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ is a cluster and the maximum set of dimensions should be $P = \{1, 2, 3, 4\}$. A good subspace clustering algorithm should be able to find this cluster and the maximum set of associated dimensions P.

The objective function is defined in terms of the compactness and separation of each subspace cluster. Let (C, P) be a subspace cluster. Then the compactness $\mathrm{Cp}(C, P)$ and the separation $\mathrm{Sp}(C, R)$ of cluster (C, P) are defined as

$$\mathrm{Cp}(C, P) = \frac{2 \sum\limits_{x,y \in C} \|x - y\|_P^2}{|P||C|^2}, \tag{15.24}$$

$$\mathrm{Sp}(C, P) = \begin{cases} \frac{2 \sum\limits_{x,y \in C} \|x-y\|_R^2}{|R||C|^2} & \text{if } R \neq \Phi, \\ 1 & \text{if } R = \Phi, \end{cases} \tag{15.25}$$

where $|P|$ and $|R|$ denote the number of elements in the sets P and R, respectively; $\|x - y\|_P$ denotes the distance between x and y in the subspace spanned by the dimensions in P; and $\|x - y\|_R$ denotes the distance in the subspace spanned by the dimensions in R.

Let $C_1, C_2, \dots, C_k$ be a partition of the data set, where k is the number of clusters. Let P_j be the set of dimensions associated with cluster C_j. The objective function is defined as

$$F_{obj} = \sum_{j=1}^{k} (\mathrm{Cp}(C_j, P_j) + 1 - \mathrm{Sp}(C_j, R_j)). \tag{15.26}$$

Given the number of clusters k, the goal of SUBCAD is to partition the data set into k nonoverlapping groups such that the objective function F_{obj} defined in equation (15.26) is minimized. The algorithm finds an approximation of the optimal partition.

Let C be a cluster associated with the set P of dimensions and $T_f(C)$ be its frequency table (cf. Section 2.1). Since the square of the simple matching distance is equal to itself, we have

$$\begin{aligned} \sum_{\mathbf{x},\mathbf{y} \in C} \|\mathbf{x} - \mathbf{y}\|_P^2 &= \sum_{\mathbf{x},\mathbf{y} \in C} \sum_{j \in P} \delta(\mathbf{x}_j, \mathbf{y}_j)^2 \\ &= \sum_{j \in P} \sum_{\mathbf{x},\mathbf{y} \in C} \delta(\mathbf{x}_j, \mathbf{y}_j) \\ &= \sum_{j \in P} \sum_{1 \le r < s \le n_j} f_{jr}(C) \cdot f_{js}(C) \\ &= \sum_{j \in P} \frac{1}{2} \left[\left(\sum_{r=1}^{n_j} f_{jr}(C) \right)^2 - \sum_{r=1}^{n_j} f_{jr}^2(C) \right] \\ &= \frac{1}{2} |P| \cdot |C|^2 - \frac{1}{2} \sum_{j \in P} \sum_{r=1}^{n_j} f_{jr}^2(C) \\ &= \frac{1}{2} |P| \cdot |C|^2 - \frac{1}{2} \sum_{j \in P} \|\mathbf{f}_j(C)\|^2, \end{aligned} \tag{15.27}$$

where $\delta(\cdot, \cdot)$ is the simple matching distance, and $\mathbf{f}_j(C)$ is defined in equation (2.3).

Thus from equation (15.27), we obtain the following simplified formulas of compactness and separation:

$$\mathrm{Cp}(C, P) = 1 - \frac{\sum\limits_{j \in P} \|\mathbf{f}_j(C)\|^2}{|P||C|^2}, \tag{15.28}$$

$$\mathrm{Sp}(C, R) = \begin{cases} 1 - \frac{\sum\limits_{j \in R} \|\mathbf{f}_j(C)\|^2}{|R||C|^2} & \text{if } R \neq \Phi, \\ 1 & \text{if } R = \Phi. \end{cases} \tag{15.29}$$

The SUBCAD algorithm consists of two phases: the initialization phase and the optimization phase. In the initialization phase, an initial partition is made. Good initial partition leads to fast convergence of the algorithm, while bad initial partition may make the algorithm converge very slowly. Some initialization methods have been proposed in the literature of clustering, such as the cluster-based method (Bradley and Fayyad, 1998) and the kd-tree–based method (Pelleg and Moore, 1999). A sketch of the algorithm is described in Algorithm 15.11.

ALGORITHM 15.11. The SUBCAD algorithm.

```
Require: $D$ - Data Set, $k$ - Number of Clusters;
Ensure: $2 \le k \le |D|$;
 1: if $D$ is a large data set then
 2:    Draw a sample from $D$;
 3: end if
 4: Compute the proximity matrix from the whole data set or the sample;
 5: Pick $k$ most dissimilar data points as seeds;
 6: Assign the remaining data points to the nearest seed;
 7: repeat
 8:    for $i = 1$ to $|D|$ do
 9:       Let $(C_l, P_l)$ be the subspace cluster that contains $\mathbf{x}_i$;
10:       for $m = 1, m \neq l$ to $k$ do
11:          if inequality (15.35) is true then
12:             Move $\mathbf{x}$ from $C_l$ to $C_m$;
13:             Update subspaces $P_l$ and $P_m$;
14:          end if
15:       end for
16:    end for
17: until No further change in the cluster memberships;
18: Output results.
```

Initialization

In the initialization phase, the k most dissimilar data points are picked as seeds, where k is the number of clusters, and then the remaining data points are assigned to the nearest seed. Let D be a data set and k be a positive integer such that $k \leq |D|$. We say that $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \in D$ are the k most dissimilar data points of the data set D if the following condition is satisfied:

$$\min_{1 \leq r < s \leq k} \|\mathbf{x}_r - \mathbf{x}_s\| = \max_{E \in \mathcal{F}} \min_{\mathbf{x},\mathbf{y} \in E} \|\mathbf{x} - \mathbf{y}\|, \tag{15.30}$$

where $\mathcal{F}$ is the class that contains all subsets E of D such that $|E| = k$, i.e.,

$$\mathcal{F} = \{E : E \subseteq D, |E| = k\},$$

and $\|\mathbf{x} - \mathbf{y}\|$ is the distance between $\mathbf{x}$ and $\mathbf{y}$.

Let $X(k, D)$ ($k \leq |D|$) denote the set of the k most dissimilar data points of the data set D. Since there is a total of $\binom{n}{k}$ elements in $\mathcal{F}$, when n is large, it is impractical to enumerate all sets in $\mathcal{F}$ to find the set of the k most dissimilar data points. Thus an approximation algorithm is employed to find a set of k data points that is near the set of the k most dissimilar data points. The basic idea is to choose k initial data points and then continuously replace the bad data points with good ones until no further changes are necessary.

More specifically, let $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a data set or a sample from a data set. First, let $X(k, D)$ be $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$, and let $\mathbf{x}_r$ and $\mathbf{x}_s$ ($1 \leq r < s \leq k$) be such that

$$\|\mathbf{x}_r - \mathbf{x}_s\| = \min_{\mathbf{x},\mathbf{y} \in X(k,D)} \|\mathbf{x} - \mathbf{y}\|. \tag{15.31}$$

Secondly, for each of the data points $\mathbf{x} \in D \backslash X(k, D)$, let

$$S_r = \min_{\mathbf{y} \in (X(k,D) \backslash \{\mathbf{x}_s\})} \|\mathbf{x} - \mathbf{y}\|, \tag{15.32}$$

$$S_s = \min_{\mathbf{y} \in (X(k,D) \backslash \{\mathbf{x}_r\})} \|\mathbf{x} - \mathbf{y}\|. \tag{15.33}$$

Then if $S_r > \|\mathbf{x}_r - \mathbf{x}_s\|$, we let $X(k, D) \cup \{\mathbf{x}\} \backslash \{\mathbf{x}_s\}$ replace $X(k, D)$; if $S_s > \|\mathbf{x}_r - \mathbf{x}_s\|$, we let $X(k, D) \cup \{\mathbf{x}\} \backslash \{\mathbf{x}_r\}$ replace $X(k, D)$.

Optimization

In the optimization phase, the data points are reassigned in order to minimize the objective function (15.26) and update the subspaces associated with those clusters whose membership has changed. Let $(C_1, P_1), (C_2, P_2), \ldots, (C_k, P_k)$ be a partition of the data set D, and let $\mathbf{x}$ be a data point in the subspace cluster (C_l, P_l). To achieve the membership changing rules, the "exact assignment test" (Wishart, 2002) technique is used in the optimization phase. $\mathbf{x}$ will be moved from subspace cluster (C_l, P_l) to another subspace cluster (C_m, P_m) ($m \neq l$) if the resulting cost function decreases, i.e., if the following inequality is true:

$$\begin{aligned}\sum_{i=1}^{k} (\mathrm{Cp}(C_i) + 1 - \mathrm{Sp}(C_i)) > \sum_{i=1, i \neq l,m}^{k} \mathrm{Cp}(C_i) \\ + \mathrm{Cp}(C_l - \mathbf{x}) + 1 - \mathrm{Sp}(C_l - \mathbf{x}) + \mathrm{Cp}(C_m + \mathbf{x}) \\ + 1 - \mathrm{Sp}(C_m + \mathbf{x}),\end{aligned} \tag{15.34}$$

where $C_l - \mathbf{x}$ means $C_l \backslash \{\mathbf{x}\}$ $C_m + \mathbf{x}$ means $C_m \cup \{\mathbf{x}\}$.

The inequality (15.34) is equivalent to

$$\begin{aligned} &\mathrm{Cp}(C_l) - \mathrm{Cp}(C_l - \mathbf{x}) - \mathrm{Sp}(C_l) + \mathrm{Sp}(C_l - \mathbf{x}) \\ &> -\mathrm{Cp}(C_m) + \mathrm{Cp}(C_m + \mathbf{x}) + \mathrm{Sp}(C_m) - \mathrm{Sp}(C_m + \mathbf{x}). \end{aligned} \tag{15.35}$$

Determining Subspace Dimensions

During the optimization phase of the algorithm, if a data point is moved from its current cluster to another one, then the subspace dimensions associated with the two clusters should be updated. In the SUBCAD algorithm, the set of subspace dimensions associated with each cluster is determined by an objective function.

Let (C, E) be a subspace cluster of a d-dimensional data set D. In order to determine the set P of subspace dimensions associated with C, an objective function whose domain is all the subsets of $Q = \{1, 2, \ldots, d\}$ is defined as

$$F(C, E) = \mathrm{Cp}(C, E) + 1 - \mathrm{Sp}(C, Q \backslash E), \quad \Phi \neq E \subseteq Q, \tag{15.36}$$

where $\mathrm{Cp}(C, E)$ and $\mathrm{Sp}(C, Q \backslash E)$ are the compactness and separation of cluster C under the subspace dimensions set E.

Our general idea to determine the set P associated with the cluster C is to find a P such that the objective function defined in equation (15.36) is minimized. Also, from equation (15.27), if $P \neq \Phi$ or $P \neq Q$, the objective function in equation (15.36) can be written as

$$F(C, E) = 1 - \frac{\sum\limits_{j \in E} \|\mathbf{f}_j(C)\|^2}{|E||C|^2} + \frac{\sum\limits_{j \in Q \backslash E} \|\mathbf{f}_j(C)\|^2}{|Q \backslash E||C|^2}, \quad E \subseteq Q, \tag{15.37}$$

where $R = Q \backslash P$.

The objective function defined in (15.37) has the following properties.

Theorem 15.1 (Condition of constancy). *The objective function defined in equation (15.36) is constant for any subset E of Q if and only if*

$$\|\mathbf{f}_1(C)\| = \|\mathbf{f}_2(C)\| = \cdots = \|\mathbf{f}_d(C)\|. \tag{15.38}$$

In addition, if the objective function is a constant, then it is equal to 1.

From the definition of the objective function $F(C, E)$, the proof of the above theorem is straightforward and is thus omitted. Thus, from Theorem 15.1, if the objective function is constant, then it is minimized at any subset of Q. In this case, the set of subspace dimensions associated with C is defined to be Q. If the objective function is not constant, then the set of subspace dimensions associated with C is defined to be the set $P \subseteq Q$ that minimizes the objective function. In fact, it can be shown that such a set P is unique if the objective is not constant (Gan, 2003). Hence, the following definition of subspace associated with each cluster is well defined.

Definition 15.2. *Let C be a cluster. Then the set P of subspace dimensions associated with C is defined as follows:*

1. *If the objective function $F(C, E)$ is constant for any $E \subseteq Q$, then let $P = Q$.*
2. *If the objective function $F(C, E)$ is not constant, then P is defined as*

$$P = \arg\max_{E \in \mathcal{E}} |E|, \tag{15.39}$$

where $\mathcal{E}$ is defined as

$$\mathcal{E} = \{O : F(C, O) = \min_{E \in \aleph} F(C, E),\ O \in \aleph\} \tag{15.40}$$

and is defined as

$$\aleph = \{E : E \subset Q, E \neq \Phi, E \neq Q\}. \tag{15.41}$$

From Definition 15.2, the set P defined in equation (15.39) is nonempty. Moreover, if the objective function $F(C, E)$ is not constant, then the set P is a true subset of Q, i.e., $P \subsetneqq Q$. The objective function in (15.37) has many good properties, which will be given as theorems or corollaries as follows. Readers may refer to Gan (2003) for detailed proofs.

Theorem 15.3. *Let (C, P) be a subspace cluster of a d-dimensional data set D $(d > 2)$, and let P be defined in equation (15.39). Then we have the following:*

1. *Let $r \in P$. If there exists an s $(1 \le s \le d)$ such that $\|\mathbf{f}_s(C)\| > \|\mathbf{f}_r(C)\|$, then $s \in P$.*
2. *Let $r \in R$. If there exists an s $(1 \le s \le d)$ such that $\|\mathbf{f}_s(C)\| < \|\mathbf{f}_r(C)\|$, then $s \in R$, where $R = Q \backslash P$.*

Corollary 15.4 (Monotonicity). *Let (C, P) be a subspace cluster of D, where P is the set of subspace dimensions defined in equation (15.39) and let $T_f(C)$ be the frequency table of C. Then for any $r \in P$ and $s \in R (R = Q \backslash P)$, we have*

$$\|\mathbf{f}_r(C)\| \ge \|\mathbf{f}_s(C)\|. \tag{15.42}$$

Theorem 15.5. *Let (C, P) be a subspace cluster of a d-dimensional data set D $(d > 2)$, where P is defined in equation (15.39). Let $T_f(C)$ be the frequency table and let r, s $(1 \le r, s \le d)$ be given so that $\|\mathbf{f}_s(C)\| = \|\mathbf{f}_r(C)\|$. Then either $r, s \in P$ or $r, s \in R$, i.e., r, s must be in the same set P or R, where $R = Q \backslash P$.*

Corollary 15.6. *Let (C, P) be a subspace cluster of D, where P is the set of subspace dimensions defined in equation (15.39), and let $T_f(C)$ be the frequency table of C. If the objective function $F(C, E)$ is not constant, then for any $r \in P$ and $s \in R$ $(R = Q \backslash P)$, we have*

$$\|\mathbf{f}_r(C)\| > \|\mathbf{f}_s(C)\|. \tag{15.43}$$

Theorem 15.7 (Uniqueness of subspace). *Let $F(C, E)$ be the objective function defined in equation (15.36) and let P be the set defined in equation (15.39). If the objective function $F(C, E)$ is not constant, then the set P is unique.*

Theorem 15.8 (Contiguity). *Let (C, P) be a subspace cluster of D, where P is the set of subspace dimensions defined in equation (15.39). Let $T_f(C)$ be the frequency table of C, and let $i_1, i_2, \ldots, i_d$ be a combination of $1, 2, \ldots, d$ such that*

$$\|\mathbf{f}_{i_1}(C)\| \geq \|\mathbf{f}_{i_2}(C)\| \geq \cdots \geq \|\mathbf{f}_{i_d}(C)\|.$$

Finally, let be the set of subscripts defined as

$$G_s = \left\{t : \|\mathbf{f}_{i_t}(C)\| \neq \|\mathbf{f}_{i_{t+1}}(C)\|, 1 \leq t \leq d-1\right\}. \tag{15.44}$$

If the objective function defined in equation (15.36) is not constant, then the set of subspace dimensions P defined in equation (15.39) must be one of the P_k's ($k = 1, 2, \ldots, |G_s|$) defined as

$$P_k = \{i_t : t = 1, 2, \ldots, g_k\}, \quad k = 1, 2, \ldots, |G_s|, \tag{15.45}$$

where $g_1 < g_2 < \cdots < g_{|G_s|}$ are elements of G_s.

Based on Theorem 15.8, the set of subspace dimensions P for a cluster C can be found very quickly. There are $2^d - 1$ nonempty subsets of Q, so it is impractical to find an optimal P by enumerating these $2^d - 1$ subsets. Based on Theorem 15.8, a fast algorithm is possible for determining the set of subspace dimensions.

15.11 Fuzzy Subspace Clustering

In fuzzy clustering algorithms, each object has a fuzzy membership associated with each cluster indicating the degree of association of the object to the cluster. In this section we present a fuzzy subspace clustering (FSC) algorithm (Gan et al., 2006; Gan, 2006) for clustering high-dimensional data sets. In FSC each dimension has a fuzzy membership associated with each cluster indicating the degree of importance of the dimension to the cluster (see Definition 15.9). Using fuzzy techniques for subspace clustering, FSC avoids the difficulty of choosing appropriate subspace dimensions for each cluster during the iterations.

Definition 15.9 (Fuzzy dimension weight matrix). *A $k \times d$ matrix $W = (w_{jh})$ is said to be a fuzzy dimension weight matrix if W satisfies the following conditions:*

$$w_{jh} \in [0, 1], \quad 1 \leq j \leq k, \quad 1 \leq h \leq d, \tag{15.46a}$$

$$\sum_{h=1}^{d} w_{jh} = 1, \quad 1 \leq j \leq k. \tag{15.46b}$$

The set of all such fuzzy dimension weight matrices is denoted by M_{fk}, i.e.,

$$M_{fk} = \{W \in \mathbb{R}^{kd} | W \text{ satisfies equation } (15.46)\}. \tag{15.47}$$

The main idea behind the FSC algorithm is to impose weights on the distance measure of the k-means algorithm (Hartigan, 1975; Hartigan and Wong, 1979) in order to capture appropriate subspace information. To describe FSC, we first briefly describe the k-means algorithm. Let the set of k centers be denoted by $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$. Then the objective function of the k-means algorithm is

$$E = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mathbf{z}_j\|^2, \tag{15.48}$$

where $\|\cdot\|$ is the Euclidean norm and C_j is the jth cluster. Alternatively, the objective function of k-means in equation (15.48) can be formulated as

$$E = \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ji} \|\mathbf{x} - \mathbf{z}_j\|^2,$$

where $U = (u_{ji})$ is a hard k-partition (see Subsection 1.2.4) of D.

Like locally adapative clustering (LAC) (Domeniconi et al., 2004), FSC associates with each cluster a weight vector in order to capture the subspace information of that cluster. For example, the hth dimension is associated with the jth cluster to a degree of w_{jh} or the jth cluster has fuzzy dimension weights specified by $\mathbf{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jd})^T$. The fuzzy dimension weights of the k clusters are represented by a fuzzy dimension weight matrix $W = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k)^T$.

Mathematically, the objective function of the algorithm is formated by imposing weights on the distance measure in equation (15.48) as

$$E_{\alpha,\epsilon}(W, Z, U) = \sum_{j=1}^{k} \sum_{i=1}^{n} u_{ji} \sum_{h=1}^{d} w_{jh}^{\alpha} (x_{ih} - z_{jh})^2 + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w_{jh}^{\alpha}, \tag{15.49}$$

where W, Z, and U are the fuzzy dimension weight matrix, the centers, and the hard k-partition of D, respectively; $\alpha \in (1, \infty)$ is a weight component or fuzzifier; and ϵ is a very small positive real number. Note that any one of W, Z, and U can be determined from the other two. The following three theorems (Theorems 15.10–15.12) show how to estimate one of W, Z, and U from the other two such that the objective function $E_{\alpha,\epsilon}(W, Z, U)$ is minimized.

From the definition of the objective function in equation (15.49), we see that there is a term $\epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w_{jh}^{\alpha}$. As you will see while analyzing the FSC algorithm, this term is introduced in the objective function in order to avoid the divide-by-zero error when calculating W given the estimates of Z and U. The parameter ϵ is specified to be a very small positive real number so that the term has very little impact on the objective function.

The following theorem is used to find a hard k-partition U given the estimates of W and Z such that the objective function $E_{\alpha,\epsilon}(W, Z, U)$ defined in equation (15.49) is minimized.

Theorem 15.10. *Given an estimate W^* of W and an estimate Z^* of Z, then a hard k-partition $U = (u_{ji})$ minimizes the objective function $E_{\alpha,\epsilon}(W^*, Z^*, U)$ if it satisfies the* *condition*

$$u_{ji} = 1 \text{ implies } j \in \left\{ r \in \{1, 2, \ldots, k\} | r = \arg\min_{1 \le l \le k} d_{li} \right\}, \tag{15.50}$$

where $d_{li} = \sum_{h=1}^{d} (w_{lh}^*)^\alpha (x_{ih} - z_{lh}^*)^2$. *The corresponding clusters* C_j $(j = 1, 2, \ldots, k)$ *are formulated as*

$$C_j = \left\{ \mathbf{x}_i \in D | u_{ij} = 1,\ 1 \le i \le n \right\}. \tag{15.51}$$

Proof. We rearrange the objective function $E_{\alpha,\epsilon}(W^*, Z^*, U)$ as

$$\begin{aligned} E_{\alpha,\epsilon}(W^*, Z^*, U) &= \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ji} \sum_{h=1}^{d} (w_{jh}^*)^\alpha (x_{ih} - z_{jh}^*)^2 + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} (w_{jh}^*)^\alpha \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ji} d_{ji} + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} (w_{jh}^*)^\alpha, \end{aligned}$$

where $d_{ji} = \sum_{h=1}^{d} (w_{jh}^*)^\alpha (x_{ih} - z_{jh}^*)^2$. Since $\sum_{j=1}^{k} u_{ji} d_{ji}$ $(i = 1, 2, \ldots, n)$ are mutually uncorrelated and $\epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} (w_{jh}^*)^\alpha$ is fixed, $E_{\alpha,\epsilon}(W^*, Z^*, U)$ is minimized if each term $\sum_{j=1}^{k} u_{ji} d_{ji}$ is minimized for $i = 1, 2, \ldots, n$. Note that only one of $u_{1i}, u_{2i}, \ldots, u_{ki}$ is 1 and all others are zero. From this it follows immediately that $\sum_{j=1}^{k} u_{ji} d_{ji}$ is minimized if u_{ji} satisfies equation (15.50).

Equation (15.51) follows from the fact that every point is assigned to its nearest center in terms of the weighted distance. This proves the theorem. □

From Theorem 15.10, we see that the objective function $E_{\alpha,\epsilon}(W, Z, U)$ of the FSC algorithm can be formulated as

$$\begin{aligned} &E_{\alpha,\epsilon}(W, Z, U) \\ &= \sum_{i=1}^{n} \min \left\{ \sum_{h=1}^{d} w_{jh}^\alpha (x_{ih} - z_{jh})^2 | j = 1, 2, \ldots, k \right\} + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w_{jh}^\alpha, \end{aligned} \tag{15.52}$$

since only one of $u_{1i}, u_{2i}, \ldots, u_{ki}$ is equal to 1 and the others are equal to zero.

Note that the hard k-partition U calculated from equation (15.50) is not unique, since a data point can be assigned to any center to which it has the nearest distance. A particular choice of U must be made when implementing the FSC algorithm.

The following theorem is used to find the cluster centers Z given the estimates of W and U such that the objective function $E_{\alpha,\epsilon}(W, Z, U)$ defined in equation (15.49) is minimized.

Theorem 15.11. *Given an estimate* W^* *of* W *and an estimate* U^* *of* U*, then the objective function* $E_{\alpha,\epsilon}(W^*, Z, U^*)$ *is minimized if* $Z = (z_{jh})$ *is calculated as*

$$z_{jh} = \frac{\sum\limits_{i=1}^{n} u_{ji}^* x_{ih}}{\sum\limits_{i=1}^{n} u_{ji}^*}, \quad 1 \le j \le k,\ 1 \le h \le d. \tag{15.53}$$

Proof. To prove this theorem, we take partial derivatives of $E_{\alpha,\epsilon}(W^*, Z, U^*)$ with respect to z_{jh}s, set them to zero, and solve the resulting system. That is,

$$\frac{\partial E_{\alpha,\epsilon}(W^*, Z, U^*)}{\partial z_{jh}} = \sum_{i=1}^{n} -2u_{ji}^*(w_{jh}^*)^\alpha (x_{ih} - z_{jh}) = 0, \quad 1 \le j \le k,\ 1 \le h \le d.$$

This gives

$$z_{jh} = \frac{\sum\limits_{i=1}^{n} u_{ji}^*(w_{jh}^*)^\alpha x_{ih}}{\sum\limits_{i=1}^{n} u_{ji}(w_{jh}^*)^\alpha} = \frac{\sum\limits_{i=1}^{n} u_{ji}^* x_{ih}}{\sum\limits_{i=1}^{n} u_{ji}^*}, \quad 1 \le j \le k,\ 1 \le h \le d. \tag{15.54}$$

This proves the theorem. □

The following theorem is used to find the fuzzy dimension weight matrix W given the estimates of Z and U such that the objective function $E_{\alpha,\epsilon}(W, Z, U)$ is minimized.

Theorem 15.12. *Given an estimate Z^* of Z and an estimate U^* of U, then the objective function $E_{\alpha,\epsilon}(W, Z^*, U^*)$ is minimized if $W = (w_{jh})$ is calculated as*

$$w_{jh} = \frac{1}{\sum\limits_{l=1}^{d} \left[\frac{\sum_{i=1}^{n} u_{ji}^*(x_{ih}-z_{jh}^*)^2+\epsilon}{\sum_{i=1}^{n} u_{ji}^*(x_{il}-z_{jl}^*)^2+\epsilon} \right]^{\frac{1}{\alpha-1}}}, \quad 1 \le j \le k,\ 1 \le h \le d. \tag{15.55}$$

Proof. To prove this theorem, we use the method of Lagrange multipliers. To do this, let us first write the Lagrangian function as

$$F(W, Z^*, U^*, \Lambda) = E_{\alpha,\epsilon}(W, Z^*, U^*) - \sum_{j=1}^{k} \lambda_j \left(\sum_{h=1}^{d} w_{jh} - 1 \right).$$

By taking partial derivatives with respect to W_{jh}, we have

$$\begin{aligned} &\frac{\partial F(W, Z^*, U^*, \Lambda)}{\partial w_{jh}} \\ &= \sum_{i=1}^{n} \alpha u_{ji}^* w_{jh}^{\alpha-1}(x_{ih} - z_{jh}^*)^2 + \epsilon\alpha w_{jh}^{\alpha-1} - \lambda_j \\ &= \alpha w_{jh}^{\alpha-1} \left(\sum_{i=1}^{n} u_{ji}^*(x_{ih} - z_{jh}^*)^2 + \epsilon \right) - \lambda_j \\ &= 0 \end{aligned} \tag{15.56}$$

for $1 \le j \le k,\ 1 \le h \le d$ and

$$\frac{\partial F(W, Z^*, U^*, \Lambda)}{\partial \lambda_j} = \sum_{h=1}^{d} w_{jh} - 1 = 0 \tag{15.57}$$

for $1 \le j \le k$.

From equation (15.56), we have

$$w_{jh} = \left[\frac{\lambda_j}{\alpha \left(\sum_{i=1}^{n} u_{ji}^* (x_{ih} - z_{jh}^*)^2 + \epsilon \right)} \right]^{\frac{1}{\alpha-1}}, \quad 1 \le j \le k,\ 1 \le h \le d. \tag{15.58}$$

Combining equation (15.57) and equation (15.58) gives

$$\sum_{h=1}^{d} \left[\frac{\lambda_j}{\alpha \left(\sum_{i=1}^{n} u_{ji}^* (x_{ih} - z_{jh}^*)^2 + \epsilon \right)} \right]^{\frac{1}{\alpha-1}} = 1, \quad 1 \le j \le k,$$

or

$$\lambda_j^{\frac{1}{\alpha-1}} = \frac{1}{\sum_{h=1}^{d} \left[\frac{1}{\alpha \left(\sum_{i=1}^{n} u_{ji}^* (x_{ih} - z_{jh}^*)^2 + \epsilon \right)} \right]^{\frac{1}{\alpha-1}}}, \quad 1 \le j \le k. \tag{15.59}$$

Plugging λ_j from equation (15.59) into equation (15.58), we have

$$w_{jh} = \frac{1}{\sum_{l=1}^{d} \left[\frac{\sum_{i=1}^{n} u_{ji}^* (x_{ih} - z_{jh}^*)^2 + \epsilon}{\sum_{i=1}^{n} u_{ji}^* (x_{il} - z_{jl}^*)^2 + \epsilon} \right]^{\frac{1}{\alpha-1}}}, \quad 1 \le j \le k,\ 1 \le h \le d.$$

This proves the theorem. □

Algorithm 15.12. The pseudocode of the FSC algorithm.

```
Require: $D$ - the data set, $k$ - the number of clusters, and $\alpha$ - the fuzzifier;
 1: Initialize $Z$ by choosing $k$ points from $D$ randomly;
 2: Initialize $W$ with $w_{jh} = \frac{1}{d}$ $(1 \le j \le k,\ 1 \le h \le d)$;
 3: Estimate $U$ from initial values of $W$ and $Z$ according to equation (15.50);
 4: Let $error = 1$ and $Obj = E_{\alpha,\epsilon}(W, Z)$;
 5: while $error > 0$ do
 6:     Update $Z$ according to Theorem 15.11;
```

```
7:      Update $W$ according to Theorem 15.12;
8:      Update $U$ according to Theorem 15.10;
9:      Calculate $NewObj = E_{\alpha,\epsilon}(W, Z)$;
10:     Let $error = |NewObj - Obj|$, and then $Obj \Leftarrow NewObj$;
11: end while
12: Output $W$, $Z$, and $U$.
```

We see from equation (15.53) and equation (15.55) that FSC is very similar to the fuzzy k-means algorithm (Huang and Ng, 1999) in terms of the way they update centers and fuzzy memberships. Like the fuzzy k-means algorithm, FSC is implemented recursively (see Algorithm 15.12). FSC starts with initial estimates of Z, W, and the partition U calculated from Z and W, and then repeats estimating the centers Z given the estimates of W and U, estimating the fuzzy dimension weight matrix W given the estimates of Z and U, and estimating the partition U given the estimates of W and Z, until it converges.

15.12 Mean Shift for Subspace Clustering

This section presents a novel approach, the mean shift for subspace clustering (MSSC) algorithm (Gan, 2006), to identifying subspace clusters. First, we introduce the relationship among several clustering algorithms: the mean shift algorithm, the maximum-entropy clustering (MEC) algorithm, the FSC algorithm, the k-means algorithm, and the MSSC algorithm. Then we briefly introduce the MSSC algorithm. Finally, we examine some special cases of the MSSC algorithm and compute the critical value for β, a key parameter required by the algorithm, when the first phase transition occurs.

Figure 15.1 shows the relationship between the mean shift algorithm and its derivatives. The k-means algorithm is a limit case of the MEC algorithm (Rose et al., 1990) which, in turn, is a special case of the mean shift algorithm (Cheng, 1995). The MSSC algorithm is an extension of the MEC algorithm for subspace clustering.

The main idea behind the MSSC algorithm is to impose weights on the distance measure of the MEC algorithm (Rose et al., 1990) in order to capture appropriate subspace information. Readers are referred to Section 9.7 for a brief review of the MEC algorithm. Given a data set $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and a set of centers $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$, recall that the free energy defined in the MEC algorithm is defined as

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

where β is a parameter called the Lagrange multiplier.

Like the FSC algorithm (see Section 15.11), the MSSC algorithm associates with each cluster a weight vector in order to capture the subspace information of that cluster.In particular, the hth dimension is associated with the jth cluster C_j to a degree of w_{jh}

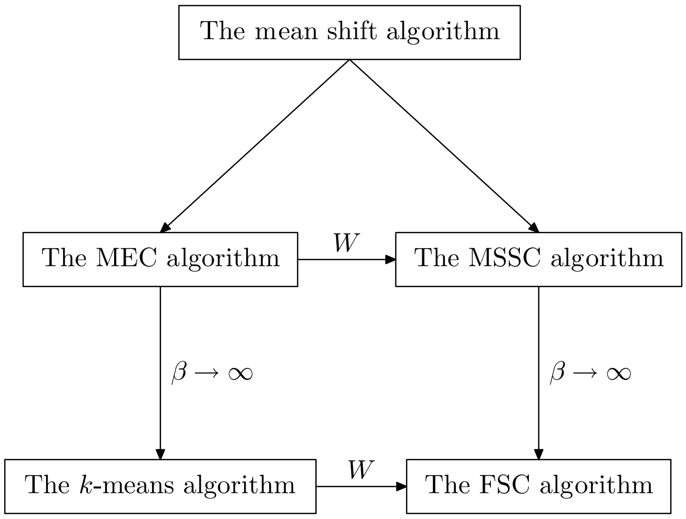

Figure 15.1. *The relationship between the mean shift algorithm and its derivatives.*

or the jth cluster has fuzzy dimension weights specified by $\mathbf{w}_j = (w_{j1}, w_{j2}, \dots, w_{jd})^T$. Mathematically, the objective function of the MSSC algorithm is formulated by imposing weights on the distance measure in the free energy as

$$\begin{aligned} &F_{\alpha,\beta,\epsilon}(W, Z) \\ &= -\frac{1}{\beta}\sum_{i=1}^{n} \ln \left[\sum_{j=1}^{k} \exp\left(-\beta \sum_{h=1}^{d} w_{jh}^{\alpha}(x_{ih} - z_{jh})^2 \right) \right] + \epsilon \sum_{j=1}^{k}\sum_{h=1}^{d} w_{jh}^{\alpha} \\ &= -\frac{1}{\beta}\sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} e^{-\beta d_{ji}(W,Z)} \right) + \epsilon \sum_{j=1}^{k}\sum_{h=1}^{d} w_{jh}^{\alpha}, \end{aligned} \tag{15.60}$$

where $\alpha \in (1, \infty)$ is the fuzzifier controlling the fuzzy dimension weights; β is a parameter ranging from zero to infinity, i.e., $\beta \in (0, \infty)$; ϵ is a very small positive number; $W = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_k)^T$ is the fuzzy dimension weight matrix; $Z = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$ is the set of centers; and $d_{ji}(W, Z)$ is defined as

$$d_{ji}(W, Z) = \sum_{h=1}^{d} w_{jh}^{\alpha}(x_{ih} - z_{jh})^2, \quad 1 \le i \le n,\ 1 \le j \le k. \tag{15.61}$$

The following theorem is used to find the set of centers Z given the estimate of W such that the objective function $F_{\alpha,\beta,\epsilon}(W, Z)$ is minimized.

Theorem 15.13. *Let $\gamma : \mathbb{R}^{kd} \to \mathbb{R}$ be defined as*

$$\gamma(Z) = F_{\alpha,\beta,\epsilon}(W, Z), \tag{15.62}$$

where $W \in M_{fk}$ is fixed. Assume $w_{jh} > 0$ for all $1 \le j \le k$ and $1 \le h \le d$. Then the set of centers Z^* optimizes the function γ if Z^* satisfies the implicit equation

$$z^*_{jh} = \frac{\sum\limits_{i=1}^{n} x_{ih} P(\mathbf{x}_i \in C_j)}{\sum\limits_{i=1}^{n} P(\mathbf{x}_i \in C_j)}, \quad 1 \le j \le k,\ 1 \le h \le d, \tag{15.63a}$$

or

$$\mathbf{z}^*_j = \frac{\sum\limits_{i=1}^{n} \mathbf{x}_i P(\mathbf{x}_i \in C_j)}{\sum\limits_{i=1}^{n} P(\mathbf{x}_i \in C_j)} = \frac{\sum\limits_{i=1}^{n} \mathbf{x}_i u_{ji}}{\sum\limits_{i=1}^{n} u_{ji}}, \quad 1 \le j \le k, \tag{15.63b}$$

where $P(\mathbf{x}_i \in C_j) = u_{ji}$ is the fuzzy membership of $\mathbf{x}_i$ associated with cluster C_j and is defined as

$$P(\mathbf{x}_i \in C_j) = u_{ji} = \frac{e^{-\beta d_{ji}(W, Z^*)}}{\sum\limits_{l=1}^{k} e^{-\beta d_{li}(W, Z^*)}}, \quad 1 \le i \le n,\ 1 \le j \le k, \tag{15.64}$$

with $d_{ji}(\cdot,\cdot)$ defined as in equation (15.61). *In addition, for $\beta = 0$, the set of centers Z^* satisfying equation* (15.63) *is the global minimum for $F_{\alpha,0,\epsilon}$.*

Proof. Since minimization of γ over $\mathbb{R}^{kd}$ is an unconstrained problem, the necessity of equation (15.63) follows by requiring $\nabla_Z \gamma$ to vanish. Equivalently, the directional derivative $\frac{\partial \gamma(Z^* + tZ)}{\partial Z}$ vanishes at Z^* in arbitrary directions $Z \in \mathbb{R}^{kd}$, $Z \ne 0$. Let $t \in \mathbb{R}$ and define

$$h(t) = \gamma(Z^* + tZ) = -\frac{1}{\beta} \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{k} e^{-\beta d_{ji}(W, Z^* + tZ)} \right) + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w^{\alpha}_{jh},$$

where $d_{ji}(\cdot,\cdot)$ is defined in equation (15.61). Rearranging the terms in d_{ji} in the above equation leads to

$$h(t) = -\frac{1}{\beta} \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{k} e^{-\beta(a_{ji}t^2 + b_{ji}t + c_{ji})} \right) + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w^{\alpha}_{jh},$$

where a_{ji}, b_{ji}, and c_{ji} are defined as

$$\begin{aligned} a_{ji} &= \sum_{h=1}^{d} w_{jh}^{\alpha} z_{jh}^{2}, \\ b_{ji} &= -2\sum_{h=1}^{d} w_{jh}^{\alpha}(x_{ih} - z_{jh}^{*}) z_{jh}, \\ c_{ji} &= \sum_{h=1}^{d} w_{jh}^{\alpha}(x_{ih} - z_{jh}^{*})^{2}. \end{aligned}$$

Taking the derivative of $h(t)$, we have

$$h'(t) = \frac{\mathrm{d}h(t)}{\mathrm{d}t} = \sum_{i=1}^{n} \frac{\sum\limits_{j=1}^{k}(2a_{ji}t + b_{ji})e^{-\beta(a_{ji}t^2 + b_{ji}t + c_{ji})}}{\sum\limits_{j=1}^{k} e^{-\beta(a_{ji}t^2 + b_{ji}t + c_{ji})}}, \tag{15.65}$$

and

$$\begin{aligned} h'(0) &= \sum_{i=1}^{n} \frac{\sum\limits_{j=1}^{k} b_{ji}e^{-\beta c_{ji}}}{\sum\limits_{j=1}^{k} e^{-\beta c_{ji}}} = -2\sum_{i=1}^{n} \frac{\sum\limits_{j=1}^{k}\sum\limits_{h=1}^{d} w_{jh}^{\alpha}(x_{ih} - z_{jh}^{*}) z_{jh} e^{-\beta c_{ji}}}{\sum\limits_{j=1}^{k} e^{-\beta c_{ji}}} \\ &= -2\sum_{j=1}^{k}\sum_{h=1}^{d} w_{jh}^{\alpha} z_{jh} \sum_{i=1}^{n} \frac{(x_{ih} - z_{jh}^{*})e^{-\beta c_{ji}}}{\sum\limits_{l=1}^{k} e^{-\beta c_{li}}} \\ &= 0. \end{aligned}$$

Since $h'(0) = 0$ holds for arbitrary directions $Z = (z_{jh})$, noting that $w_{jh} > 0$ for all j, h, it is necessary for every j, h that

$$\sum_{i=1}^{n} \frac{(x_{ih} - z_{jh}^{*})e^{-\beta c_{ji}}}{\sum\limits_{l=1}^{k} e^{-\beta c_{li}}} = 0,$$

from which equation (15.63) follows and the necessity is established.

To prove that Z^* is the global minimum of $F_{\alpha,\beta,\epsilon}$ for $\beta = 0$, we examine the $kd \times kd$ Hessian matrix $H_\gamma(Z^*)$ of γ evaluated at Z^*. In fact, from equation (15.65), we have

$$h''(t) = \sum_{i=1}^{n} \frac{\sum_{j=1}^{k} \left[2a_{ji} - \beta(2a_{ji}t + b_{ji})^2\right] e^{-\beta(a_{ji}t^2+b_{ji}t+c_{ji})} \sum_{j=1}^{k} e^{-\beta(a_{ji}t^2+b_{ji}t+c_{ji})}}{\left(\sum_{j=1}^{k} e^{-\beta(a_{ji}t^2+b_{ji}t+c_{ji})}\right)^2}$$
$$- \sum_{i=1}^{n} \frac{-\beta\left(\sum_{j=1}^{k}(2a_{ji}t + b_{ji})e^{-\beta(a_{ji}t^2+b_{ji}t+c_{ji})}\right)^2}{\left(\sum_{j=1}^{k} e^{-\beta(a_{ji}t^2+b_{ji}t+c_{ji})}\right)^2}.$$

Thus, at $t = 0$, noting that $\beta = 0$ we have

$$\begin{aligned}
&h''(0) \\
&= (z_{11}, z_{12}, \dots, z_{kd})H_\gamma(Z^*)(z_{11}, z_{12}, \dots, z_{kd})^T \\
&= \sum_{i=1}^{n} \frac{\sum_{j=1}^{k}\left[2a_{ji} - \beta b_{ji}^2\right] e^{-\beta c_{ji}} \sum_{j=1}^{k} e^{-\beta c_{ji}} + \beta\left(\sum_{j=1}^{k} b_{ji}e^{-\beta c_{ji}}\right)^2}{\left(\sum_{l=1}^{k} e^{-\beta c_{li}}\right)^2} \\
&= \sum_{i=1}^{n} \frac{\sum_{j=1}^{k} 2a_{ji}e^{-\beta c_{ji}} \sum_{j=1}^{k} e^{-\beta c_{ji}} + \beta\left[\left(\sum_{j=1}^{k} b_{ji}e^{-\beta c_{ji}}\right)^2 - \sum_{j=1}^{k} b_{ji}^2 e^{-\beta c_{ji}} \sum_{j=1}^{k} e^{-\beta c_{ji}}\right]}{\left(\sum_{l=1}^{k} e^{-\beta c_{li}}\right)^2} \\
&= \frac{2}{k}\sum_{i=1}^{n}\sum_{j=1}^{k} a_{ji} \\
&> 0.
\end{aligned}$$

Hence, the Hessian matrix is positive definite, so Z^* is the global minimum of $F_{\alpha,0,\epsilon}$. This proves the theorem. □

From Theorem 15.13 we see that at $\beta = 0$ there is only one cluster, since $u_{ji} = \frac{1}{k}$ for all j, i and $\mathbf{z}_j$, $j = 1, 2, \dots, k$, are identical to the center of the data set. At higher β, the objective function $F_{\alpha,\beta,\epsilon}$ may have many local minima and the cluster will split into smaller clusters. Once we obtain a new set of centers, we need to update the fuzzy dimension weight for the clusters. The following theorem tells how to do this.

Theorem 15.14. *Let $\eta : M_{fk} \to \mathbb{R}$ be defined as*

$$\eta(W) = F_{\alpha,\beta,\epsilon}(W, Z), \tag{15.66}$$

where $Z \in \mathbb{R}^{kd}$ is fixed. Then the fuzzy dimension weight W^* optimizes the function η if W^* satisfies the implicit function

$$w^*_{jh} = \frac{1}{\sum\limits_{l=1}^{d} \left[\frac{\sum\limits_{i=1}^{n} u_{ji}(x_{ih}-z_{jh})^2+\epsilon}{\sum\limits_{i=1}^{n} u_{ji}(x_{il}-z_{jl})^2+\epsilon} \right]^{\frac{1}{\alpha-1}}}, \quad 1 \le j \le k,\ 1 \le h \le d, \tag{15.67}$$

where the fuzzy membership u_{ji} is defined as

$$u_{ji} = \frac{\exp\left(-\beta \sum\limits_{h=1}^{d} (w^*_{jh})^\alpha (x_{ih} - z_{jh})^2 \right)}{\sum\limits_{l=1}^{k} \exp\left(-\beta \sum\limits_{h=1}^{d} (w^*_{lh})^\alpha (x_{ih} - z_{lh})^2 \right)}, \quad 1 \le j \le k,\ 1 \le i \le n. \tag{15.68}$$

Proof. Minimization of η over M_{fk} is a Kuhn-Tucker problem carrying the kd inequality constraints (15.46a) and k equality constraints (15.46b) (Bezdek, 1980). This can be done via the method of Lagrange multipliers. Let $\Lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$ be the multipliers and $\Gamma(W, \Lambda)$ be the Lagrangian

$$\Gamma(W, \Lambda) = -\frac{1}{\beta} \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} e^{-\beta d_{ji}(W,Z)} \right) + \epsilon \sum_{j=1}^{k} \sum_{h=1}^{d} w^\alpha_{jh} - \sum_{j=1}^{k} \lambda_j \left(\sum_{h=1}^{d} w_{jh} - 1 \right).$$

If (W^*, Λ^*) is to minimize Γ, its gradient in both sets of variables must vanish, i.e.,

$$\begin{aligned}
\frac{\partial \Gamma(W^*, \Lambda^*)}{\partial w_{jh}} &= \sum_{i=1}^{n} \frac{\alpha (w^*_{jh})^{\alpha-1} (x_{ih} - z_{jh})^2 e^{-\beta d_{ji}(W^*,Z^*)}}{\sum\limits_{j=1}^{k} e^{-\beta d_{ji}(W^*,Z^*)}} + \epsilon\alpha (w^*_{jh})^{\alpha-1} - \lambda_j \\
&= \alpha (w^*_{jh})^{\alpha-1} \left(\sum_{i=1}^{n} u_{ji}(x_{ih} - z_{jh})^2 + \epsilon \right) - \lambda_j \\
&= 0, \quad 1 \le j \le k,\ 1 \le h \le d,
\end{aligned} \tag{15.69a}$$

where u_{ji} is defined in Equation (15.68), and

$$\frac{\partial \Gamma(W^*, \Lambda^*)}{\partial \lambda_j} = \sum_{h=1}^{d} w^*_{jh} - 1 = 0, \quad 1 \le j \le k. \tag{15.69b}$$

Equation (15.67) following immediately by solving the $k(d+1)$ equations in (15.69a) and (15.69b). This proves the theorem. $\square$

From equation (15.68), we see that $u_{ji} > 0$ for all $1 \le j \le k$ and $1 \le i \le n$. If the data set D is such that for each dimension h there exist two distinct values, then $\sum_{i=1}^{n} u_{li}(x_{ih} - z_{lh})^2 > 0$ for all $1 \le l \le k$ and $1 \le h \le d$. In this case, we set the parameter

$\epsilon = 0$, since the purpose of ϵ is to avoid divide-by-zero errors when implementing the MSSC algorithm. In all analysis below, we assume $\epsilon = 0$.

The FSC algorithm presented in Section 15.11 is a limit case of the MSSC algorithm presented above when $\beta \to \infty$ (see Figure 15.1). In fact, letting β approach infinity, we have

$$\begin{aligned}
&\lim_{\beta\to\infty} F_{\alpha,\beta,\epsilon}(W, Z)\\
&= \lim_{\beta\to\infty} \frac{1}{-1} \sum_{i=1}^{n} \frac{\sum\limits_{j=1}^{k} -d_{ji}(W, Z)e^{-\beta d_{ji}(W,Z)}}{\sum\limits_{j=1}^{k} e^{-\beta d_{ji}(W,Z)}} + \epsilon \sum_{j=1}^{k}\sum_{h=1}^{d} w_{jh}^{\alpha}\\
&= \sum_{i=1}^{n} \min\left\{d_{ji}(W, Z) | j = 1, 2, \ldots, k\right\} + \epsilon \sum_{j=1}^{k}\sum_{h=1}^{d} w_{jh}^{\alpha}\\
&= \sum_{i=1}^{n} \min\left\{\sum_{h=1}^{d} w_{jh}^{\alpha}(x_{ih} - z_{jh})^2 | j = 1, 2, \ldots, k\right\} + \epsilon \sum_{j=1}^{k}\sum_{h=1}^{d} w_{jh}^{\alpha},
\end{aligned}$$

which is exactly the objective function of the FSC algorithm defined in equation (15.52).

Clearly equation (15.63b) is an implicit equation in $\mathbf{z}_j$ through equation (15.64). It is natural to propose the following iterative algorithm.

Definition 15.15 (The MSSC algorithm). *Let $D = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a finite data set. Let $G_\alpha^\beta(\mathbf{x}, \mathbf{w})$ be the Gaussian kernel defined as*

$$G_\alpha^\beta(\mathbf{x}, \mathbf{w}) = e^{-\beta \sum\limits_{h=1}^{d} w_h^\alpha x_h^2}. \tag{15.70}$$

Let $v : D \to (0, \infty)$ be a weight function defined as

$$v(\mathbf{x}) = \frac{1}{\sum\limits_{j=1}^{k} G_\alpha^\beta(\mathbf{x} - \mathbf{z}_j, \mathbf{w}_j)}. \tag{15.71}$$

The sample mean with the kernel G_α^β at $\mathbf{z}_j \in D$ is defined as

$$m(\mathbf{z}_j) = \frac{\sum\limits_{\mathbf{x}\in D} G_\alpha^\beta(\mathbf{x} - \mathbf{z}_j, \mathbf{w}_j)v(\mathbf{x})\mathbf{x}}{\sum\limits_{\mathbf{x}\in D} G_\alpha^\beta(\mathbf{x} - \mathbf{z}_j, \mathbf{w}_j)v(\mathbf{x})}. \tag{15.72}$$

Let $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\} \subset D$ be a finite set of cluster centers. The evolution of Z in the form of iterations

$$Z^{(r)} = \left\{\mathbf{z}_j^{(r)} = m(\mathbf{z}_j^{(r-1)}) | j = 1, 2, \ldots, k\right\}, \quad r = 1, 2, \ldots, \tag{15.73}$$

is called the MSSC algorithm, where $\mathbf{z}_j^{(0)} = \mathbf{z}_j$, $j = 1, 2, \ldots, k$. *The weights* $v(\mathbf{x})$ *and the fuzzy memberships* $U = (u_{ji})$ *are reevaluated after each iteration. The sequence* $\mathbf{z}, m(\mathbf{z}), m^2(\mathbf{z}), \ldots$ *is called a trajectory of* $\mathbf{z}$.

From Definition 9.5, we know that the above fixed-point iteration is the mean shift process with $v(\mathbf{x})$ being the weight of the point $\mathbf{x}$. If we choose $Z = D$, then this algorithm is also called a blurring process (Cheng, 1995).

ALGORITHM 15.13. The pseudocode of the MSSC algorithm.

```
Require: $D$ - the data set, $k$ - the number of starting points, $\alpha$ - the fuzzifier, and $\beta$ - the
    Lagrange multiplier;
 1: Initialize $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ by choosing $k$ points from $D$ randomly;
 2: Initialize $W = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_k\}$ with $w_{jh} = \frac{1}{d}$ $(1 \le j \le k,\ 1 \le h \le d)$;
 3: Calculate fuzzy memberships $U$ according to equation (15.64) or (15.68);
 4: $Z^{new} \Leftarrow m(Z) = \{m(\mathbf{z}_1), m(\mathbf{z}_2), \ldots, m(\mathbf{z}_k)\}$ according to equation (15.63);
 5: Update $W$ according to equation (15.67);
 6: while $\|Z^{new} - Z\| > 0$ do
 7:    $Z \Leftarrow Z^{new}$;
 8:    $Z^{new} \Leftarrow m(Z)$ according to equation (15.63);
 9:    Update $W$ according to equation (15.67);
10:    Calculate fuzzy memberships $U$ according to Equation (15.64) or (15.68);
11: end while
12: Output $W$, $Z$, and $U$.
```

Let $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ be an initial set of centers. Then the procedure defined in equation (15.73) determines a set of equilibrium points $Z^* = \{\mathbf{z}_1^*, \mathbf{z}_2^*, \ldots, \mathbf{z}_k^*\}$ for fixed β, α, and ϵ, where $\mathbf{z}_j^* = \lim_{r\to\infty} m^r(\mathbf{z}_j)$. In principle, changing k will modify the resulting set of equilibrium points. However, there exists some k_c such that for all $k > k_c$, one gets only k_c distinct limit points for the simple reason that some points converge to the same equilibrium point.

The pseudocode of the MSSC algorithm is described in Algorithm 15.13. Once we obtain the fuzzy dimension weight matrix W, the equilibrium set Z, and the fuzzy memberships U, we can use the procedure in Algorithm 15.14 to get the final subspace clusters.

ALGORITHM 15.14. The postprocessing procedure to get the final subspace clusters.

```
Require: $\eta$, $W$, $Z$, and $U$;
 1: Let $Q = \{1, 2, \ldots, k\}$;
 2: for $j = 2$ to $k$ do
 3:    for $i = 1$ to $j - 1$ do
 4:       if $\|\mathbf{z}_j - \mathbf{z}_i\|_\infty < \eta$ and $i \in Q$ then
```

```
5:         $Q = Q - \{j\}$ $\{\mathbf{z}_j$ is identical to $\mathbf{z}_i\}$;
6:         Break;
7:       **end if**
8:     **end for**
9:  **end for**
10: Let $Q = \{j_1, j_2, \ldots, j_{k_c}\}$ $\{k_c$ is the number of distinct equilibrium points$\}$;
11: **for** $i = 1$ to $n$ **do**
12:   **if** $l_0 = \arg\max_{1 \le l \le k_c} u_{j_l i}$ **then**
13:     The point $\mathbf{x}_i$ is assigned to the $l_0$th cluster $C_{l_0}$;
14:   **end if**
15: **end for**
16: Get set of cluster dimensions $Q_l$ for $C_l$ by applying $k$-means to $\mathbf{w}_{j_l}$;
17: Output $C_l$ and $Q_l$ for $l = 1, 2, \ldots, k_c$.
```

If $\beta = 0$, each data point is uniformly associated with all clusters, and thus all the centers $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k$ are identical (Gan, 2006). Let k_c be the number of distinct centers in $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$. Then, at $\beta = 0$ we have $k_c = 1$, but at some positive β we shall have $k_c > 1$. In other words, the single cluster will split into smaller clusters. The new clusters will split at higher β. At $\beta = \infty$, $k_c = k$, i.e., we shall get k clusters, where k $(< n)$ is the number of initial centers.

Clearly, when $k = 1$ the objective function is

$$\begin{aligned} F_{\alpha,\beta,0}(W, Z) &= -\frac{1}{\beta}\sum_{i=1}^{n} \ln\left(e^{-\beta d_{1i}(W,Z)}\right) = \sum_{i=1}^{n} d_{1i}(W, Z) \\ &= \sum_{i=1}^{n}\sum_{h=1}^{d} w_{1h}^{\alpha}(x_{ih} - z_{1h})^2, \end{aligned}$$

which has a single minimum or the global minimum; when $k \geq 2$ and $\beta = 0$ the objective function is

$$F_{\alpha,0,0}(W, Z) = -\frac{1}{\beta}\sum_{i=1}^{n} \ln(k) = -\infty.$$

Thus, at $\beta = 0$ the objective function has a single minimum or the global minimum. At higher β, the objective function may have many local minima.

Since the objective function has only one minimum when $k = 1$, to derive the critical value β_α at which the first phase transition occurs, we assume that $k > 1$ in the following discussion. We know that at $\beta = \beta_\alpha$ there is one cluster centered at the mean of the data set. Without loss of generality, we move the data set such that the mean of the new data set is located at the origin. Then the first phase transition occurs when the objective function has nonzero minimum, i.e., nonzero solution to the equations

$$\begin{aligned}\frac{\partial F_{\alpha,\beta,0}(W,Z)}{\partial z_{jh}} &= -\frac{1}{\beta}\sum_{i=1}^{n}\frac{2\beta w_{jh}^{\alpha}(x_{ih}-z_{jh})e^{-\beta d_{ji}(W,Z)}}{\sum\limits_{l=1}^{k}e^{-\beta d_{li}(W,Z)}}\\ &= -2w_{jh}^{\alpha}\sum_{i=1}^{n}\frac{(x_{ih}-z_{jh})e^{-\beta d_{ji}(W,Z)}}{\sum\limits_{l=1}^{k}e^{-\beta d_{li}(W,Z)}}\\ &= 0, \quad 1\le j\le k,\ 1\le h\le d,\end{aligned}$$

and

$$\sum_{i=1}^{n}\frac{(\mathbf{x}_i-\mathbf{z}_j)e^{-\beta d_{ji}(W,Z)}}{\sum\limits_{l=1}^{k}e^{-\beta d_{li}(W,Z)}} = \mathbf{0}, \quad 1\le j\le k, \tag{15.74}$$

where $\mathbf{0}$ is the zero vector and $d_{ji}(W,Z)$ is defined as

$$d_{ji}(W,Z)=\sum_{h=1}^{d}w_{jh}^{\alpha}(x_{ih}-z_{jh})^2, \quad 1\le i\le n,\ 1\le j\le k.$$

Note that in a small neighborhood of the origin, we have

$$w_{jh}=w_h\approx\frac{1}{\sum\limits_{l=1}^{d}\left[\frac{\sum\limits_{i=1}^{n}x_{ih}^2}{\sum\limits_{i=1}^{n}x_{il}^2}\right]^{\frac{1}{\alpha-1}}}, \quad 1\le j\le k,\ 1\le h\le d,$$

and

$$z_{jh}=z_h\approx 0, \quad 1\le j\le k,\ 1\le h\le d.$$

Now expanding equation (15.74) on a small neighborhood of the origin by Taylor's series in $\mathbf{z}_j=(z_{j1},z_{j2},\ldots,z_{jd})^T$ and ignoring higher-order terms of z_{jh}, we have

$$\begin{aligned}\mathbf{0} &= \sum_{i=1}^{n}\frac{(\mathbf{x}_i-\mathbf{z}_j)e^{-\beta d_{ji}(W,Z)}}{\sum\limits_{l=1}^{k}e^{-\beta d_{li}(W,Z)}}\\ &\approx \sum_{i=1}^{n}\frac{(\mathbf{x}_i-\mathbf{z}_j)\exp\left(-\beta\sum\limits_{t=1}^{d}w_t^{\alpha}(x_{it}-z_{jt})^2\right)}{k\exp\left(-\beta\sum\limits_{t=1}^{d}w_t^{\alpha}x_{it}^2\right)}\\ &\approx \frac{1}{k}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{z}_j)\exp\left(2\beta\sum_{t=1}^{d}w_t^{\alpha}x_{it}z_{jt}-\beta\sum_{t=1}^{d}w_t^{\alpha}z_{jt}^2\right)\\ &\approx \frac{1}{k}\sum_{i=1}^{n}(\mathbf{x}_i-\mathbf{z}_j)\left(1+2\beta\sum_{t=1}^{d}w_t^{\alpha}x_{it}z_{jt}\right).\end{aligned} \tag{15.75}$$

Since $\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}$, rearranging equation (15.75) and ignoring higher-order terms of z_{jh}s give

$$\sum_{i=1}^{n} \left(\mathbf{x}_i \cdot 2\beta \sum_{t=1}^{d} w_t^\alpha x_{it} z_{jt} - \mathbf{z}_j \right) = \mathbf{0},$$

or

$$\sum_{i=1}^{n} \left[2\beta \begin{pmatrix} w_1^\alpha x_{i1}^2 & w_2^\alpha x_{i1} x_{i2} & \cdots & w_d^\alpha x_{i1} x_{id} \\ w_1^\alpha x_{i2} x_{i1} & w_2^\alpha x_{i2}^2 & \cdots & w_d^\alpha x_{i2} x_{id} \\ \vdots & \vdots & \ddots & \vdots \\ w_1^\alpha x_{id} x_{i1} & w_2^\alpha x_{id} x_{i2} & \cdots & w_d^\alpha x_{id}^2 \end{pmatrix} \begin{pmatrix} z_{j1} \\ z_{j2} \\ \vdots \\ z_{jd} \end{pmatrix} - \begin{pmatrix} z_{j1} \\ z_{j2} \\ \vdots \\ z_{jd} \end{pmatrix} \right] = \mathbf{0}. \tag{15.76}$$

Let $\mathbf{v}_j = (x_{1j}, x_{2j}, \ldots, x_{nj})$ for $j = 1, 2, \ldots, k$, and let the variance and covariance be defined as

$$\mathrm{var}(\mathbf{v}_j) = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \mu_j)^2 = \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2,$$

$$\mathrm{Cor}(\mathbf{v}_j, \mathbf{v}_l) = \frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \mu_j)(x_{il} - \mu_l),$$

where

$$\mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}, \quad 1 \le j \le k.$$

From equation (15.76), we have

$$(I - 2\beta C_\alpha) \mathbf{z}_j = \mathbf{0}, \tag{15.77}$$

where I is the $d \times d$ identity matrix and C_α is the $d \times d$ matrix defined as

$$C_\alpha = \begin{pmatrix} w_1^\alpha \mathrm{var}(\mathbf{v}_1) & w_2^\alpha \mathrm{Cor}(\mathbf{v}_1, \mathbf{v}_2) & \cdots & w_d^\alpha \mathrm{Cor}(\mathbf{v}_1, \mathbf{v}_d) \\ w_1^\alpha \mathrm{Cor}(\mathbf{v}_2, \mathbf{v}_1) & w_2^\alpha \mathrm{var}(\mathbf{v}_2) & \cdots & w_d^\alpha \mathrm{Cor}(\mathbf{v}_2, \mathbf{v}_d) \\ \vdots & \vdots & \ddots & \vdots \\ w_1^\alpha \mathrm{Cor}(\mathbf{v}_d, \mathbf{v}_1) & w_2^\alpha \mathrm{Cor}(\mathbf{v}_d, \mathbf{v}_2) & \cdots & w_d^\alpha \mathrm{var}(\mathbf{v}_d) \end{pmatrix}.$$

Thus the critical value for β is

$$\beta_\alpha = \frac{1}{2\lambda_{max}}, \tag{15.78}$$

where λ_{max} is the largest eigenvalue of C_α. Moreover, the center of the new cluster is the eigenvector of C_α.

15.13 Summary

The subspace clustering introduced in this chapter is an extension of traditional clustering. In general, subspace clustering algorithms can be classified into two major categories (Parsons et al., 2004b): top-down algorithms and bottom-up algorithms. A top-down algorithm finds an initial clustering in the full set of the dimensions and evaluates the subspaces of each cluster, whereas a bottom-up algorithm finds dense regions in low-dimensional spaces and then combines them to form clusters. CLIQUE, ENCLUS, MAFIA, CLTree, DOC, and CBF (Cell-Based Clustering method) (Chang and Jin, 2002) are examples of bottom-up subspace clustering algorithms, PART, PROCLUS, ORCLUS, FINDIT, and δ-cluster (Yang et al., 2002a) are examples of top-down subspace clustering algorithms. A comparison of these subspace clustering algorithms can be found in Parsons et al. (2004a). Other discussions related to subspace clustering can be found in Aggarwal and Yu (2002), Amir et al. (2003), Agarwal and Mustafa (2004), Domeniconi et al. (2004), Har-Peled and Varadarajan (2002), Ke and Kanade (2004), Krishnapuram and Freg (1991), Kroeger et al. (2004), Sarafis et al. (2003), Wang et al. (2004), and Narahashi and Suzuki (2002).