聚类算法
Clustering Algorithms

引言

- 聚类是一种典型的无监督学习任务
  - 没有给出数据实例的类别值来说明其先验分组。
- 目的是发现数据中的相似群组（称为聚簇），使得：
  - 彼此相似（距离近）的数据实例被分到同一个聚簇中
  - 彼此不同（距离远）的数据实例被分到不同的聚簇中。.

聚类效果的判断标准

- 聚类结果应当满足
  - 聚簇内部的高相似度（同一聚簇内部的数据对象相互类似）
  - 聚簇之间的低相似度（不同聚簇之间的数据对象彼此不同）

- 聚类结果的质量取决于：
  - 距离函数
  - 聚类算法
  - 具体应用

数据对象间的距离/相异性度量

- 个体对象间的相似性/相异性度量至关重要
  - 它对聚类效果的影响要远远大于聚类算法的选择。
- 大多数算法都假定有一个相异度矩阵，该矩阵的元素都是非负的，且对角元素为0
  - 如果最初收集的数据是相似度，那么可以通过一个合适的单调递减函数将其转换为相异度。
- 大多数算法都假定相异度矩阵D是对称的，如果不对称，可以用$\frac{D+D^T}{2}$来替代D 。
- 此外，相异度并不等同于距离（距离需要满足三角不等式）

连续型变量
定量属性

- 连续型变量的取值通常为连续的实数。
  - 一种常用的“差值”定义为绝对差异的单调递增函数$\mathrm{d}(x_i, x_{i'}) = l(|x_i - x_{i'}|)$。
  - 例如：$(x_i - x_{i'})^2$或者$|x_i - x_{i'}|$。

序数型变量(Ordinal Variables)

- 序数型变量的值域被认为是一个有序的集合。
- 例如：
  - 学生的课程成绩{A, B, C, D, F}，
  - 偏爱度{can’t stand, dislike, OK, like, terrific}

- 序数变量可以通过如下公式转换为连续的定量属性:

$$\frac{i-0.5}{M}, i = 1, \ldots, M$$

离散型变量

- 如果该变量有M种不同取值，不同取值之间的差异度可以表示为一个$M \times M$的对称非负矩阵，对角线上的元素都为0。
  - 对于任意两个不同取值，其差异度最常用的度量是1。

基本聚类方法的类型

- 划分方法(Partitioning Methods)*
- 层次方法(Hierarchical Methods)*
- 基于模型的方法(Model-based Methods)*
- 基于密度的方法(Density-based Methods)
- 基于网格的方法(Grid-based Methods)

划分方法(Partitioning Methods)

- 给定N个对象，划分方法将数据划分成k个群组，每个群组表示一个聚簇
  - 划分方法通常是基于距离的
  - 可能使用均值(mean)或中心点(medoid)等等来表示聚簇中心
  - 对于小型或中型数据集有效

层次方法(Hierarchical Methods)

- 层次聚类构造了给定数据对象集合的一种层次分解（即多个层次）
  - 根据如何形成这种层次分解，可以分为凝聚型和分裂型两大类。
  - 一旦完成了一次合并或分类，就再无法取消它，因此无法纠正错误的合并和分裂

基于密度的方法(Density-based)

- 可以发现任意形状的聚簇
- 聚簇是空间中被低密度区域所隔离的那些对象密集区域
- 聚簇密度：
- 可以过滤掉离群数据点(outliers)

- 常见方法：DBSCAN、OPTICS、DENCLUE

基于网格的方法(Grid-based)

- 使用多分辨率（或成为多解析度）的网格数据结构
- 快速处理时间（通常独立于数据对象的数目，但是依赖于网格的大小）

- 常见方法：STING、CLIQUE

划分聚类的组合算法
Combinatorial Algorithms

- 预先指定聚簇数目为K，每个聚簇都通过一个整数$k \in \{1,2,\ldots,K\}$来标记。
- 每条观测数据$\mathbf{x}$都分配到一个聚簇k，记为$cl(\mathbf{x}) = k$
- 簇内平方和：

$$W(cl) = \frac{1}{2}\sum_{k=1}^{K}\sum_{cl(\mathbf{x}_i)=k}\sum_{cl(\mathbf{x}_j)=k} d(\mathbf{x},\mathbf{x}')$$

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d(x_i,x_{i'}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left(\sum_{C(i')=k} d(x_i,x_{i'}) + \sum_{C(i')\neq k} d(x_i,x_{i'})\right)$$

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d(x_i,x_{i'})$$

组合优化

- 组合优化在原理上非常简单：在将N个数据点分配到K个聚簇的所有可能组合中，寻求能够最大化B(C)或者最小化W(C)的那种分配方案
  - 不同分配方案的数目为$S(N,K)=\frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}k^N$。
  - 例如，$S(10,4)=34{,}105$ 而 $S(19,4)\cong 10^{10}$。大多数聚类问题的数据对象的数量是远远大于19的。

k-均值算法
k-Means Algorithm

- k-Means算法是一种基于划分的聚类算法

- 它以一种迭代的方式将N条观测数据划分到k个不同的聚簇中
  - 不同聚簇间的对象之间的距离最大化
  - 同一聚簇内的对象之间的距离最小化

- 迭代下降算法

适用范围

- 它适用于所有的属性都是连续型属性（即定量的），并且使用平方欧式距离
- 相异度度量：

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{D} (x_{id} - x_{jd})^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2$$

K-均值算法
主要步骤

- 第一步：随机地初始化k个重心(centroids)
- 第二步：数据的隶属关系计算（分配数据点到距离它最近的那个重心）
- 第三步：聚簇的重心更新（根据当前的聚簇隶属关系）
- 第四步：如果当前没有收敛，则重复执行第二步和第三步

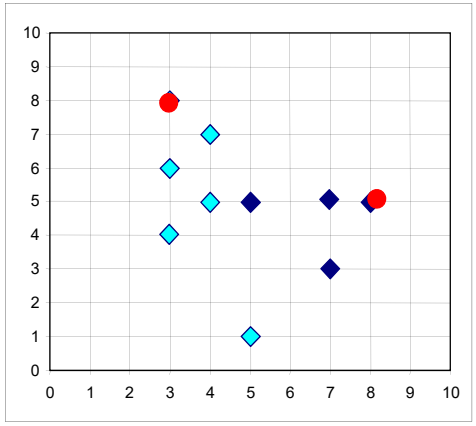
K-均值算法
分配(Assignment)与更新(Update)

- 分配（保持聚簇重心不变）

$$S_i^{(t)} = \left\{x_p : \forall 1 \le j \le K \quad dist\left(x_p - m_i^{(t)}\right) \le dist\left(x_p - m_j^{(t)}\right)\right\}$$

- 更新（保持聚簇隶属关系不变）

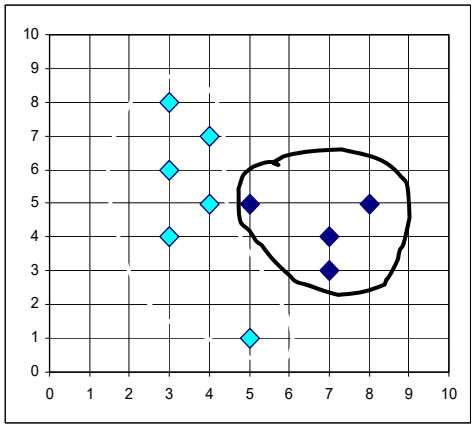
$$m_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_j \in S_i^{(t)}} x_j$$



K=2

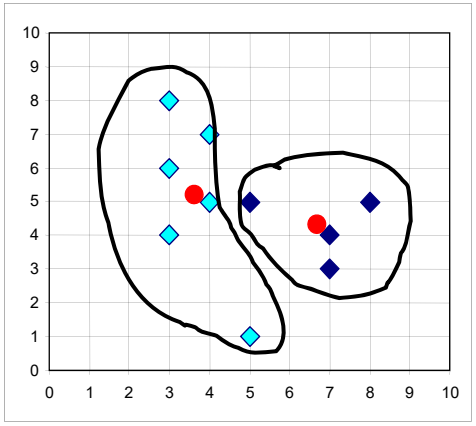
任意选择k个对象作为初始的聚簇中心

将每个对象都分配给最近的中心

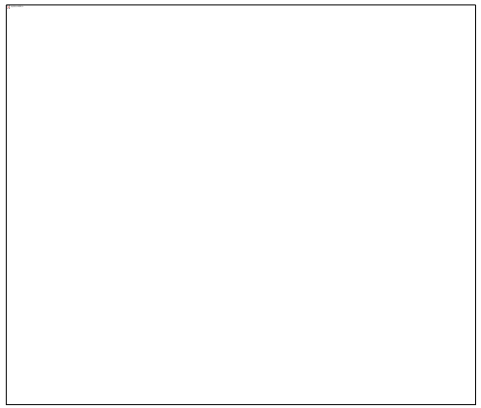


重新分配

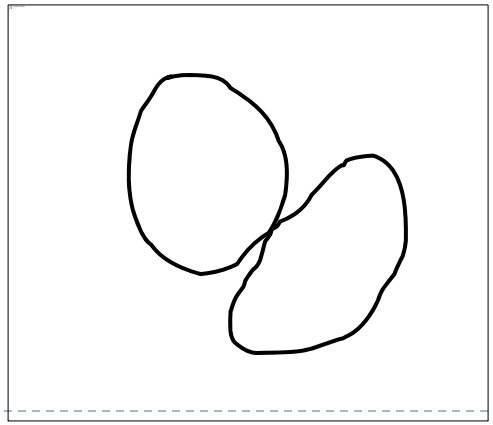
更新聚簇均值



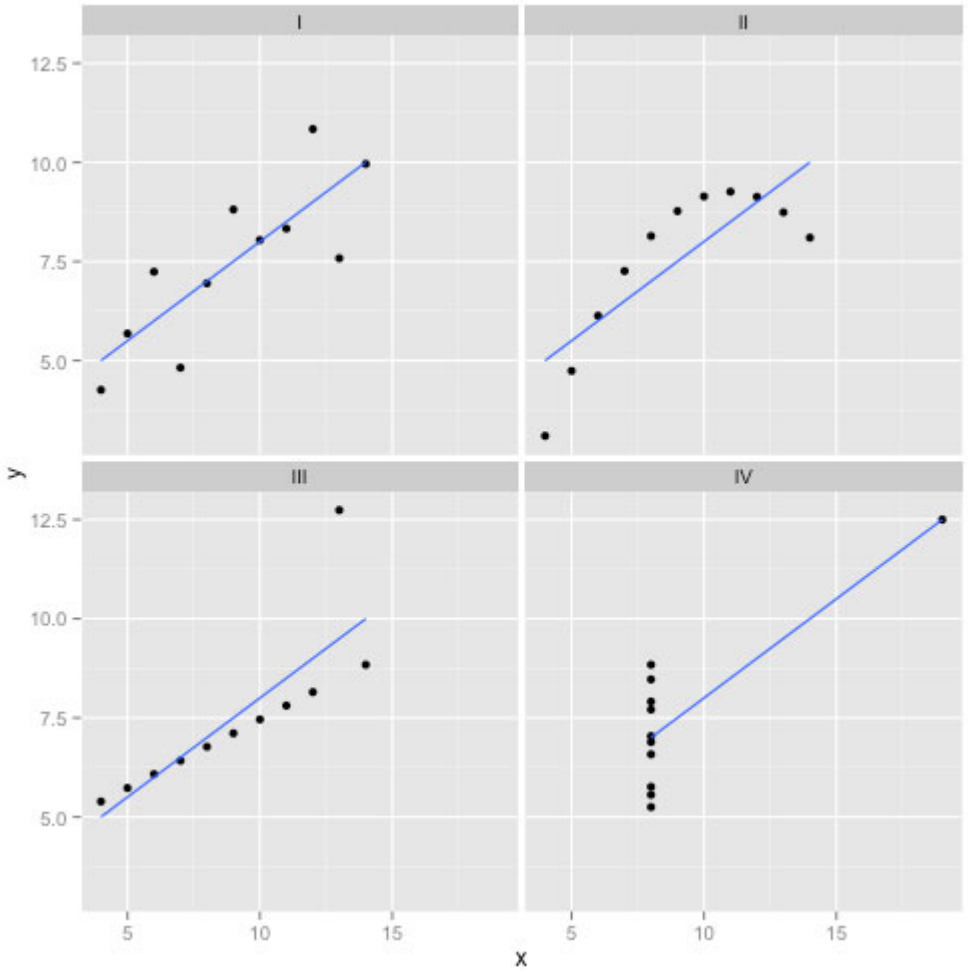
重新分配



更新聚簇均值



方法与假设
Methods and Assumptions



K-Means算法的优点

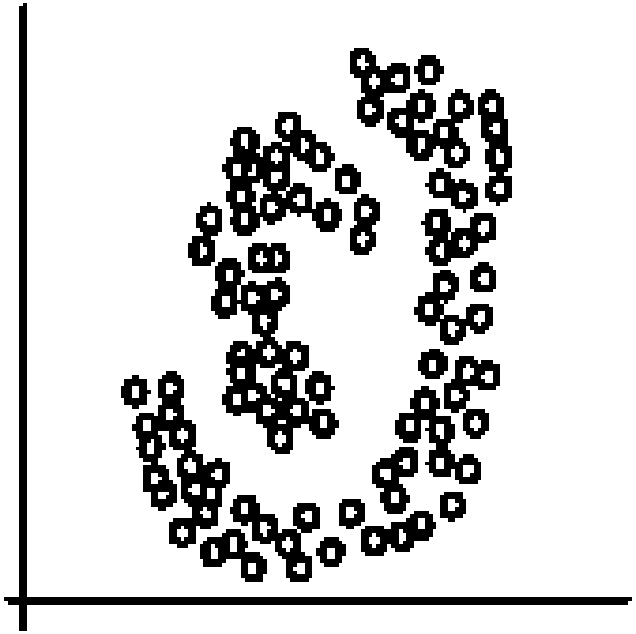
- 简单：易于理解和实现
- 高效：其时间复杂度为$O(nkdt)$，其中n是观测数据的数目、k是指定的聚簇数目、d是观测数据的维度、而t是迭代的次数

k-Means算法所存在的问题-I

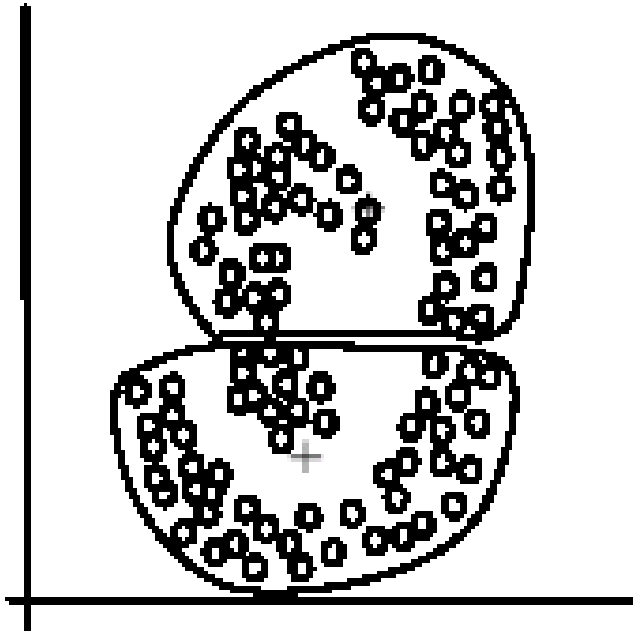
- k-means算法的结果依赖于最初对于聚簇中心的随机选择
  - 实践中为了获得好的聚类结果，通常可以使用不同的初始聚簇中心来多次运行k-means算法
- 该算法只有在可以定义均值的情况下才能使用
  - k-Modes算法是k-Means算法的一个变体，可以处理离散型数据的聚类问题
  - k-means算法和k-modes算法可以结合起来处理混合有连续型属性和离散型属性的数据

k-Means算法所存在的问题-II

- k-means算法不适用于发现非凸(nonconvex)形状的聚簇，也不适于发现大小差异很大的聚簇。



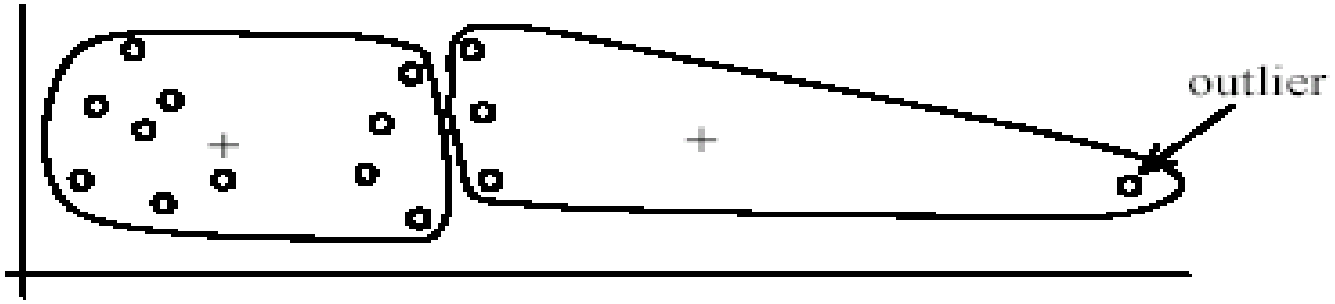
(A): Two natural clusters



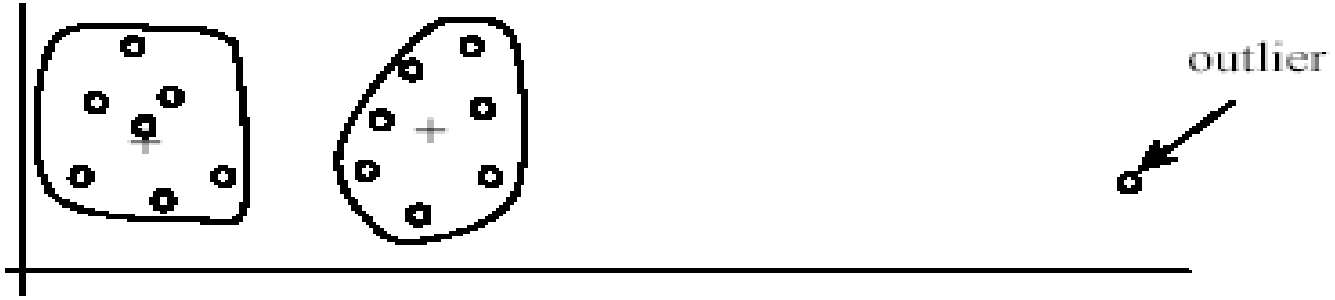
(B): *k*-means clusters

k-Means算法所存在的问题-III

- 用户需要指定k的取值
- 该算法对噪音和离群数据(outliers)敏感



(A): Undesirable clusters



(B): Ideal clusters

均值、中位数与众数
Mean, Median, and Mode

- 均值的含义是平均值
- 中位数则是将一系列的数值按照按照升序排列而位于中间位置的那个数
- 众数则是出现次数最多的那个数值

- 例子：求出数值列表
  13, 18, 13, 14, 13, 16, 14, 21, 13
  的均值、中位数以及众数各是多少？

k-中心点方法
k-Medoids Method

- k-Means算法对离群数据敏感。
- k-Medoids算法中，每个聚簇由该聚簇中的一个对象来表示。
  - 该对象到该聚簇内所有对象的平均相异度（或平均静距离）最小

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(p, o_i)$$

k-Medoids算法

- k-Medoids聚类的主要步骤：
  - 初始化：从N个数据点中选择k个来作为中心点(medoids)
  - 分配阶段：将每个数据点分配给与之最近的那个中心点(medoid)。
  - 更新阶段：对于每个聚簇，在该聚簇内找出一个对象o，使得o到聚簇内其它对象的相异度总和最小；并以o作为该聚簇的中心点。

k-Medoids算法
优缺点

- 在有噪音和离群数据的情况下，k-medoids方法比k-means方法更加强健（鲁棒）-robust。
  - 因为k-medoids算法优化的是相异度的和，而k-means优化的则是欧式距离的平方和。
- K-medoids算法中每次迭代的复杂度是$O(k(n-k)^2)$，计算代价比k-means高很多。

K-Modes: See J. X. Huang's paper online

(Data Mining and Knowledge Discovery Journal, Springer)

4.1. Dissimilarity measure

Let X, Y be two categorical objects described by m categorical attributes. The dissimilarity measure between X and Y can be defined by the total mismatches of the corresponding attribute categories of the two objects. The smaller the number of mismatches is, the more similar the two objects. This measure is often referred to as simple matching (Kaufman and Rousseeuw, 1990). Formally,

$$d_1(X, Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \tag{5}$$

where

$$\delta(x_j, y_j) = \begin{cases} 0 & (x_j = y_j) \\ 1 & (x_j \neq y_j) \end{cases} \tag{6}$$

K-Modes (Cont.)

4.2. Mode of a set

Let $\boldsymbol{X}$ be a set of categorical objects described by categorical attributes, $A_1, A_2, \ldots, A_m$.

Definition 1. A mode of $\boldsymbol{X} = \{X_1, X_2, \ldots, X_n\}$ is a vector $Q = [q_1, q_2, \ldots, q_m]$ that minimises

$$D(\boldsymbol{X}, Q) = \sum_{i=1}^{n} d_1(X_i, Q) \tag{7}$$

Here, Q is not necessarily an element of $\boldsymbol{X}$.

K-Modes

4.3. Find a mode for a set

Let $n_{c_{k,j}}$ be the number of objects having the kth category $c_{k,j}$ in attribute A_j and $f_r(A_j = c_{k,j} | X) = \frac{n_{c_{k,j}}}{n}$ the relative frequency of category $c_{k,j}$ in X.

Theorem 1. *The function $D(X, Q)$ is minimised iff $f_r(A_j = q_j | X) \geq f_r(A_j = c_{k,j} | X)$ for $q_j \neq c_{k,j}$ for all $j = 1, \ldots, m$.*

The proof of Theorem 1 is given in Appendix.

Theorem 1 defines a way to find Q from a given X, and therefore is important because it allows the k-means paradigm to be used to cluster categorical data. The theorem implies that the mode of a data set X is not unique. For example, the mode of set {[a, b], [a, c], [c, b], [b, c]} can be either [a, b] or [a, c].

K-Modes: Cost Function

4.4. The k-modes algorithm

When (5) is used as the dissimilarity measure for categorical objects, the cost function (1) becomes

$$P(W, \boldsymbol{Q}) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} w_{i,l} \delta(x_{i,j}, q_{l,j}) \qquad (8)$$

where $w_{i,l} \in W$ and $Q_l = [q_{l,1}, q_{l,2}, \ldots, q_{l,m}] \in \boldsymbol{Q}$.

Finding K-Modes

In the basic algorithm we need to calculate the total cost P against the whole data set each time when a new $\boldsymbol{Q}$ or W is obtained. To make the computation more efficient we use the following algorithm instead in practice.

1. Select k initial modes, one for each cluster.
2. Allocate an object to the cluster whose mode is the nearest to it according to (5). Update the mode of the cluster after each allocation according to Theorem 1.
3. After all objects have been allocated to clusters, retest the dissimilarity of objects against the current modes. If an object is found such that its nearest mode belongs to another cluster rather than its current one, reallocate the object to that cluster and update the modes of both clusters.
4. Repeat 3 until no object has changed clusters after a full cycle test of the whole data set.

Mixed Types: K-Prototypes

5. The *k*-prototypes algorithm

It is straightforward to integrate the *k*-means and *k*-modes algorithms into the *k*-prototypes algorithm that is used to cluster the mixed-type objects. The *k*-prototypes algorithm is practically more useful because frequently encountered objects in real world databases are mixed-type objects.

The dissimilarity between two mixed-type objects X and Y, which are described by attributes $A^r_1, A^r_2, \ldots, A^r_p, A^c_{p+1}, \ldots, A^c_m$, can be measured by

$$d_2(X, Y) = \sum_{j=1}^{p} (x_j - y_j)^2 + \gamma \sum_{j=p+1}^{m} \delta(x_j, y_j) \qquad (9)$$

K-Modes: Evaluation Data

6.1.2. k-Modes clustering results. We used the *k*-modes algorithm to cluster each test data set of the soybean disease data into four clusters with the two initial mode selection methods and produced 200 clustering results. For each clustering result we used a misclassification matrix to analyse the correspondence between clusters and the disease classes of the instances. Two misclassification matrices for the test data sets 1 and 9 are shown in figure 2. The capital letters D, C, R, P in the first column of the matrices represent the four disease classes. In figure 2(a) there is one to one correspondence between clusters and disease classes, which means the instances in the same disease classes were clustered into the same clusters. This represents a complete recovery of the four disease classes from the test data set.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
D			10	
C				10
R	10			
P		17		

(a)

K-Modes: Evaluation

In figure 2(b) two instances of the disease class P were misclassified into cluster 1 which was dominated by the instances of the disease type R. However, the instances in the other two disease classes were correctly clustered into clusters 3 and 4. This clustering result can also be considered good.

We have used the clustering accuracy as a measure of a clustering result. Clustering accuracy r is defined as

$$r = \frac{\sum_{i=1}^{4} a_i}{n}$$

where a_i is the number of instances occurring in both cluster i and its corresponding class and n is the number of instances in the data set. The clustering error is defined as $e = 1 - r$. For instance in figure 2(b), $r = \frac{10+15+10+10}{47} = 0.9574$ and $e = 0.0426$.

The 200 clustering results are summarised in Table 1. The first column in the table gives the clustering accuracy. The second and third columns show the numbers of clustering results, 100 in each column.

Some Experiments

Table 1. Summary of the 200 clustering results.

Accuracy	Select method 1	Select method 2
1.0	13	14
0.98	7	8
0.96	12	26
0.94	4	9
0.92	7	6
0.89	2	1
≤0.87	55	36

If we consider the accuracy $r > 0.87$ as a "good" clustering result, then 45 good results were produced with the first selection method and 64 good results with the second selection method. Both selection methods produced more than 10 complete recovery results (0 misclassification). These results indicate that if we randomly choose one test data set, we have a 45% chance to obtain a good clustering result with the first selection method and a 64% chance with the second selection method.

层次聚类
Hierarchical Clustering

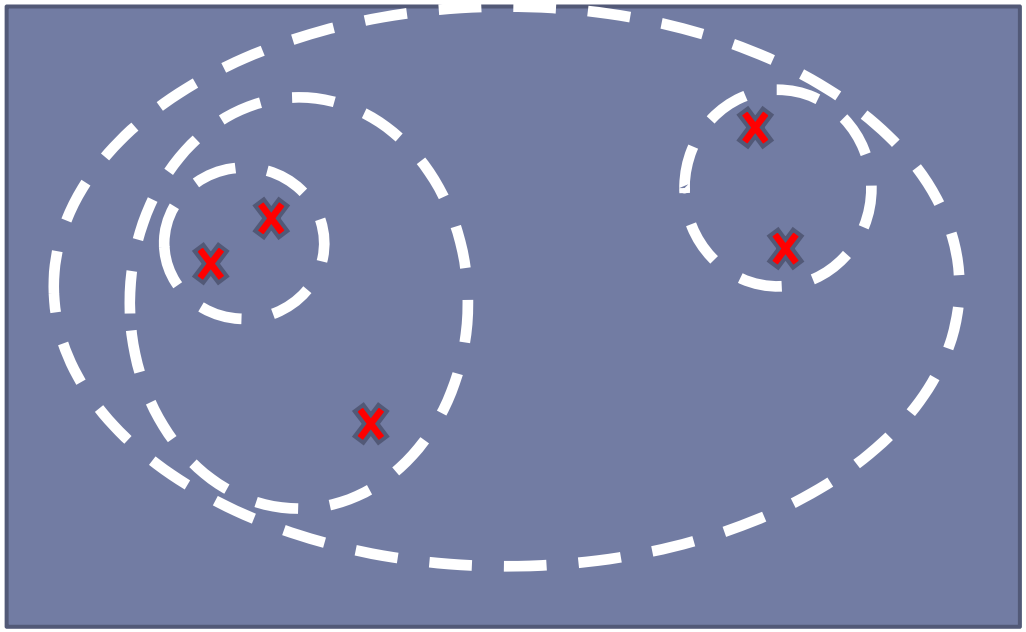
- 凝聚(Agglomerative)聚类：自底向上
  - 它从底层开始建立树状图(dendrogram)，它不断地合并最相似（或距离最近）的一组聚簇，直到所有的数据点都合并到一个单一的聚簇（根聚簇）才停止。
- 分裂(Divisive)聚类：自顶向下
  - 它从最顶部的根节点开始，根节点中含有所有的数据点；它将根节点分割成一系列子聚簇的集合，又进一步递归地分裂每一个子聚簇，直到每一个聚簇只含有单个的数据点为止。

凝聚型层次聚类
算法步骤

- 凝聚型层次聚类比分裂型层次聚类应用得更加广泛。
  - （初始化）每个数据点都形成一个单点(singleton)聚簇
  - 合并两个具有最小距离的聚簇（核心问题：如何计算两个聚簇之间的距离？）
  - 不断合并直到所有的结点都隶属于同一个聚簇

总共需要合并$N-1$次（N是数据点的数量）

凝聚型层次聚类



聚簇之间的距离度量

- 如何衡量两个聚簇之间的距离？
  - 最小距离(Minimum Distance)：→Single Linkage 单连接

$$dist_{min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} dist(p, q)$$

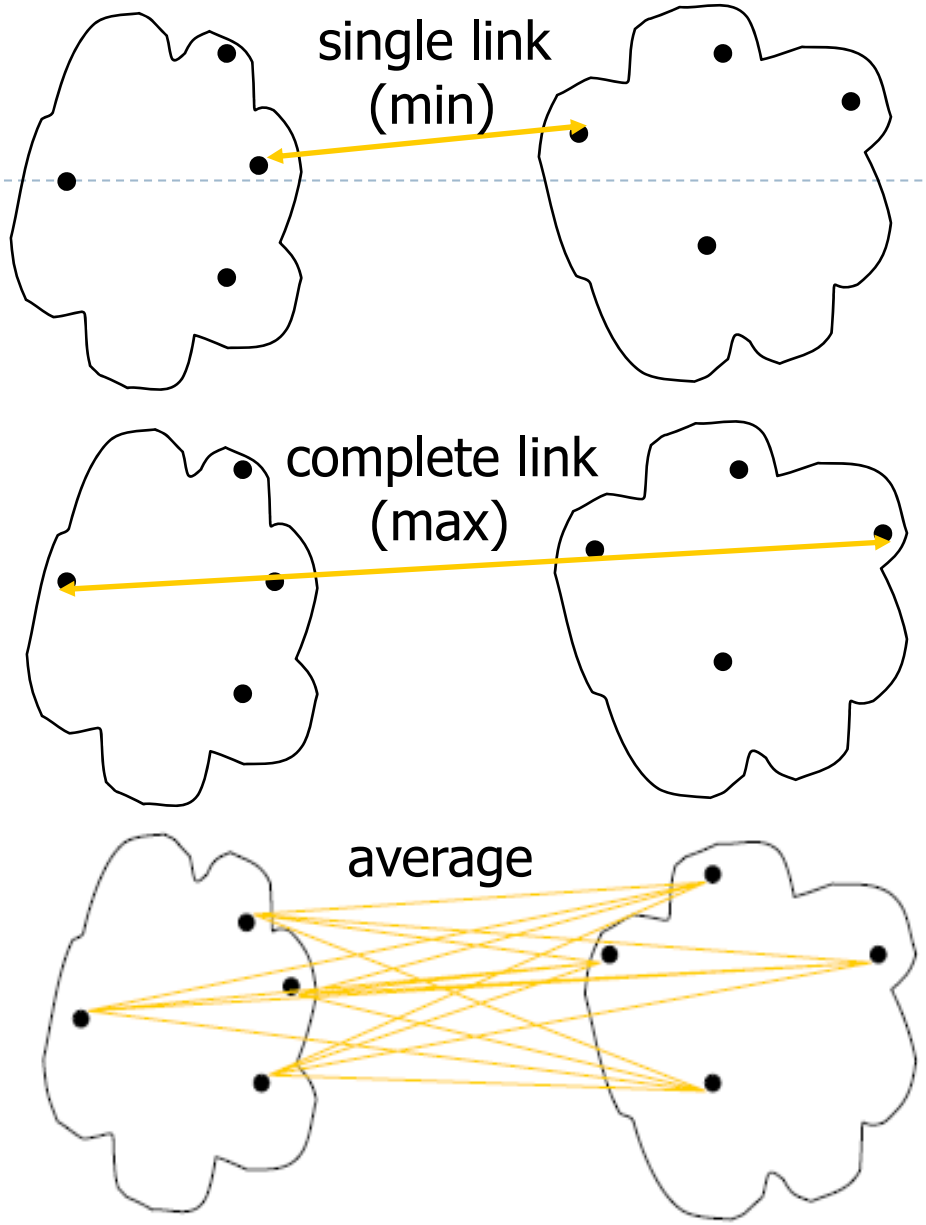
  - 最大距离(Maximum Distance)：→Complete Linkage全连接

$$dist_{max}(C_i, C_j) = \max_{p \in C_i, q \in C_j} dist(p, q)$$

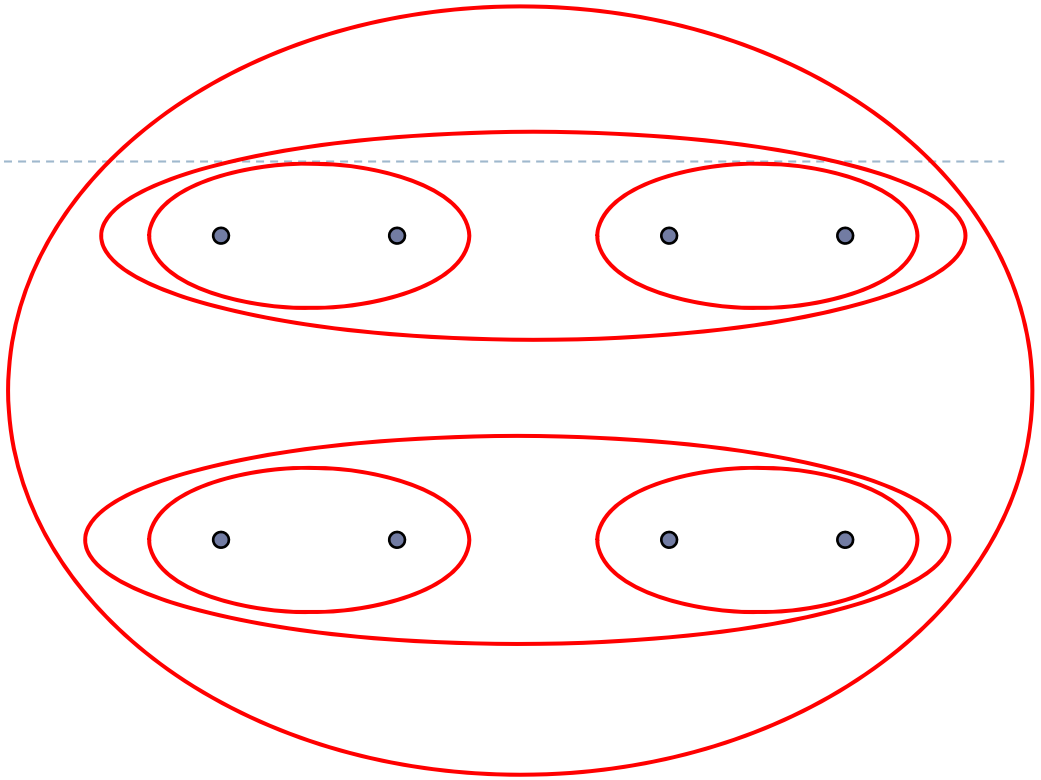
  - 平均距离(Average Distance)：→Average Linkage平均连接

$$dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, q \in C_j} dist(p, q)$$

聚簇之间三种不同的距离度量



单连接
Single Linkage

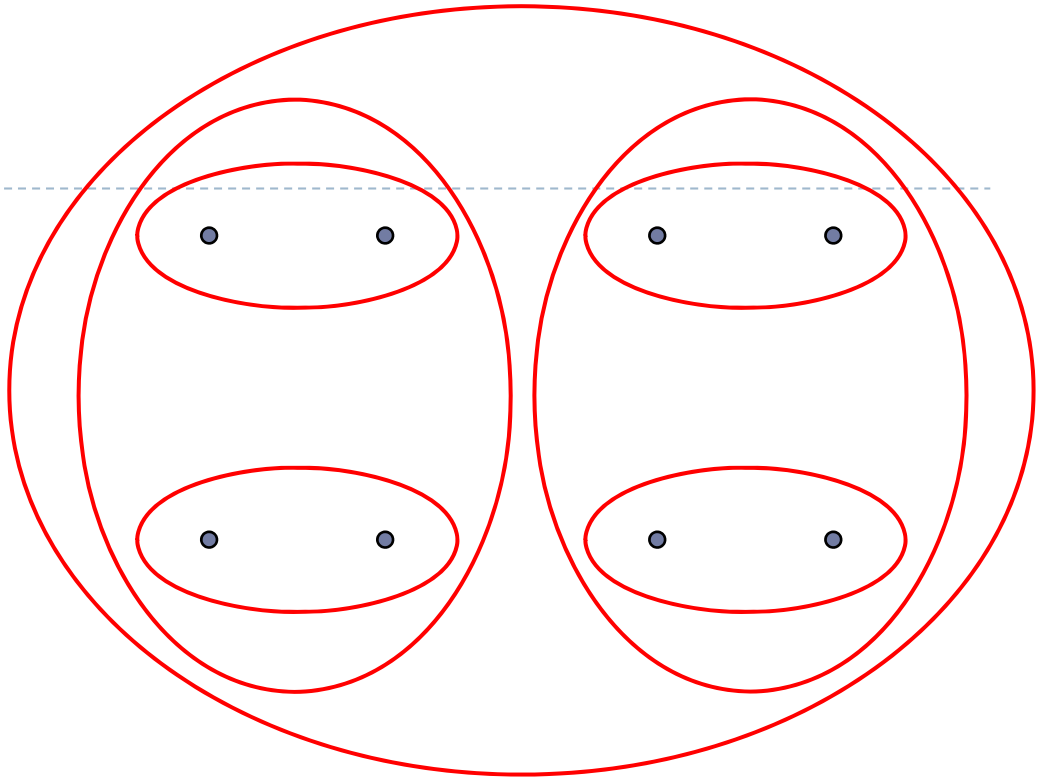


单连接可以产生“长瘦型”的聚簇

在合并了两个聚簇c_i和c_j之后，所产生的聚簇$c_i \cup c_j$与另一个聚簇c_k之间的距离为：

$$dist\left(\left(c_i \cup c_j\right), c_k\right) = \min\left(dist(c_i, c_k), dist(c_j, c_k)\right)$$

全连接
Complete Linkage



全连接会偏向于生成“比较紧凑的”球形聚簇。

在合并了两个聚簇c_i和c_j之后，所产生的聚簇$c_j \cup c_j$与另一个聚簇c_k之间的距离为：

$$dist\left(\left(c_j \cup c_j\right), c_k\right) = \max\left(dist(c_i, c_k), dist(c_j, c_k)\right)$$

平均连接
Average Linkage

$$dist_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i, q \in C_j} dist(p, q)$$

例子

	a	b	c	d	e
Feature	1	2	4	5	6

计算出两个聚簇之间的距离：$C1 = \{a, b\}, C2 = \{c, d, e\}$

1. 计算出距离矩阵

	a	b	c	d	e
a	0	1	3	4	5
b	1	0	2	3	4
c	3	2	0	1	2
e	4	3	1	0	1
e	5	4	2	1	0

单连接

$$dist(C1, C2) = \min\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \min\{3,4,5,2,3,4\} = 2$$

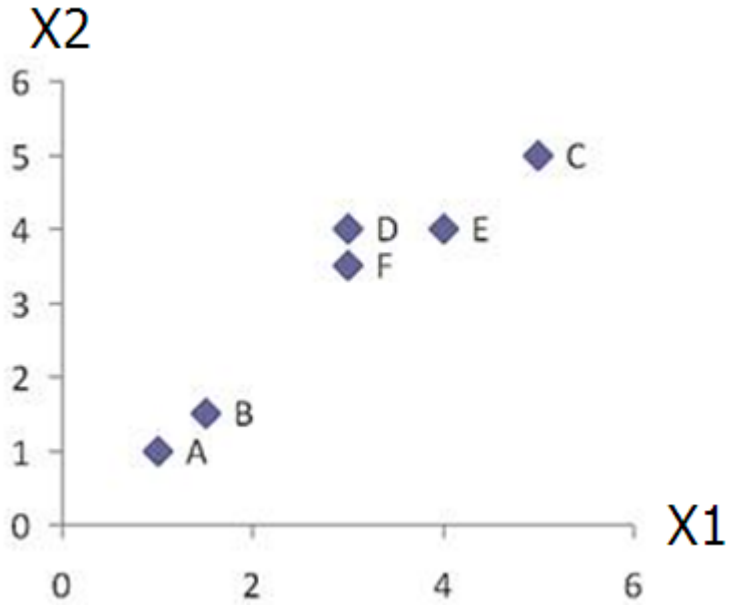
全连接

$$dist(C1, C2) = \max\{d(a,c), d(a,d), d(a,e), d(b,c), d(b,d), d(b,e)\}$$
$$= \max\{3,4,5,2,3,4\} = 5$$

平均连接

$$dist(C1, C2) = \frac{d(a,c) + d(a,d) + d(a,e) + d(b,c) + d(b,d), + d(b,e)}{6}$$
$$= \frac{3+4+5+2+3+4}{6} = \frac{21}{6} = 3.5$$

聚类过程示例



	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

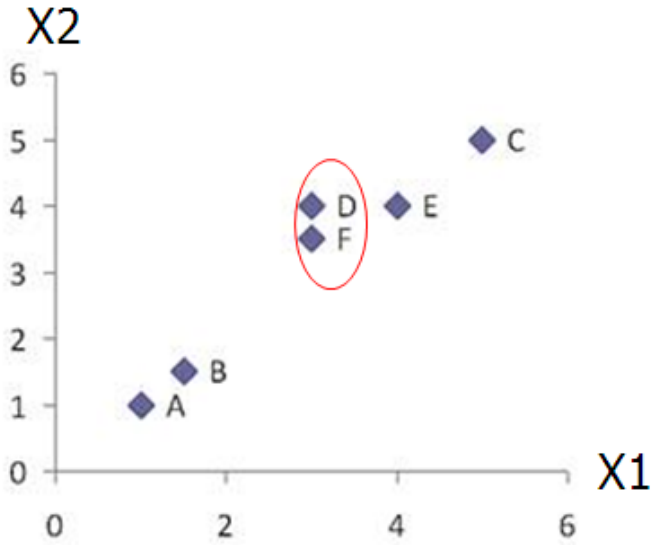
$$d_{AB} = \left((1-1.5)^2 + (1-1.5)^2\right)^{\frac{1}{2}} = \sqrt{\tfrac{1}{2}} = 0.7071$$

$$d_{DF} = \left((3-3)^2 + (4-3.5)^2\right)^{\frac{1}{2}} = 0.5$$

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

聚类过程示例

合并两个最近的聚簇



Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

聚类过程示例

更新距离矩阵

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

$$d_{(D,F)\to A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

$$d_{(D,F)\to B} = \min\left(d_{DB}, d_{FB}\right) = \min\left(2.92, 2.50\right) = 2.50$$

$$d_{(D,F)\to C} = \min\left(d_{DC}, d_{FC}\right) = \min\left(2.24, 2.50\right) = 2.24$$

$$d_{E\to(D,F)} = \min\left(d_{ED}, d_{EF}\right) = \min\left(1.00, 1.12\right) = 1.00$$

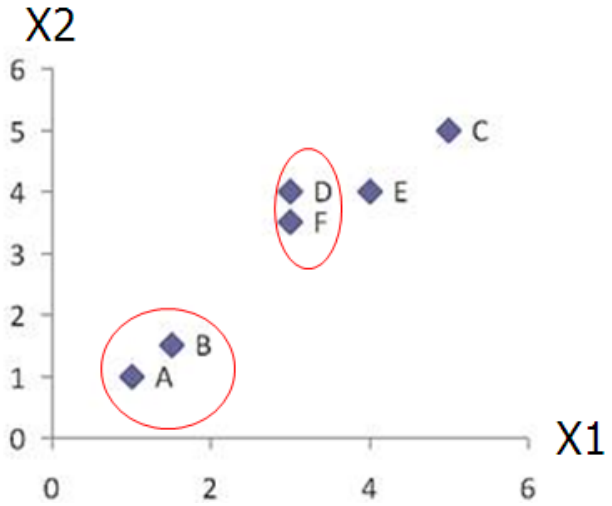
Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

聚类过程示例

合并两个最近的聚簇（第二次循环）



Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

4.95

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

聚类过程示例

更新距离矩阵（第二次循环）

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

4.95

$$d_{C\to(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

$$d_{(D,F)\to(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}\right) = \min\left(3.61, 2.92, 3.20, 2.50\right) = 2.50$$

$$d_{E\to(A,B)} = \min\left(d_{EA}, d_{EB}\right) = \min\left(4.24, 3.54\right) = 3.54$$

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	**4.95**	0	2.24	1.41
(D, F)	**2.50**	2.24	0	1.00
E	**3.54**	1.41	1.00	0