Lecture 5
Cluster Analysis

CA4010: Data Warehousing and Data Mining
2016/2017 Semester 1

Dr. Mark Roantree
Dublin City University

Agenda

1. **Data Types in Cluster Analysis**
2. **Partitioning Methods**
   - k-means Example
3. **k-Means Clustering**
   - k-Medoids Method
4. **Hierarchical Clustering**
   - Recording the Distance between Clusters
5. **Outlier Analysis**
   - Statistical Distribution-Based Outlier Detection

Extracting Information from Unlabelled Data

- **Clustering** is concerned with grouping together objects that are similar to each other and dissimilar to the objects belonging to other clusters.
  - In economics, finding countries whose economies are similar.
  - In finance, find clusters of companies that have similar financial performance.
  - In marketing, find clusters of customers with similar buying behaviour.
  - In medicine, find clusters of patients with similar symptoms.
  - In document retrieval, find clusters of documents with related content.
  - In crime analysis look for clusters of high volume crimes such as burglaries.

Data Matrix

- This *object-by-variable* structure represents *n* objects, such as persons, with *p* variables (also called measurements or attributes), such as age, height, weight, gender, and so on.
- The structure is in the form of a relational table, or *n-by-p* matrix (*n* objects $\times$ *p* variables):

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

Figure 1: Data Matrix

Dissimilarity Matrix

- This *object-by-object* structure stores a collection of proximities that are available for all pairs of n objects.
- It is often represented by an *n-by-n* table:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Figure 2: Dissimilarity Matrix

Dissimilarity Matrix

- Here *d(i,j)* is the measured **difference** or **dissimilarity** between objects *i* and *j*.
- In general, *d(i,j)* is a non-negative number that is close to 0 when objects *i* and *j* are highly similar or *near* each other, and becomes larger the more they differ.
- Since *d(i,j)=d(j,i)* and *d(i,i)=0*, we have the matrix in figure 2.

Dissimilarity Matrix Usage

- The rows and columns of the data matrix represent *different* entities, while those of the dissimilarity matrix represent the *same* entity.
- Many clustering algorithms operate on a **dissimilarity matrix**.
- If the data are presented in the form of a data matrix, first transform into a dissimilarity matrix before applying clustering algorithms.

Interval-Scaled Variables

- **Interval-scaled** variables are continuous measurements of a roughly linear scale.
- Typical examples include weight and height, latitude and longitude coordinates (e.g., when clustering houses), and weather temperature.
- The *measurement unit* used can affect the clustering analysis.
- For example, changing measurement units from metres to feet, or from kilograms to pounds, may lead to a very different clustering structure.
- In general, expressing a variable in smaller units will lead to a larger range for that variable and thus, a larger effect on the resulting clustering structure.

Standardizing Measurements

- To help avoid dependence on the choice of measurement units, data should be standardized.
- Standardizing measurements attempts to give all variables an equal weight.
- This is particularly useful when given no prior knowledge of the data.
- However, in some applications, users may intentionally want to give more weight to a certain set of variables than to others.
- For example, when clustering basketball player candidates, one may prefer to give more weight to the variable height.

How can data be standardized?

- One method is to convert the original measurements to *unitless* variables.
- Given measurements for a variable *f*, this can be performed in 2 steps.
- **Step 1**. Calculate the **mean absolute deviation**, s_f:

$$s_f = \frac{1}{n}(| x_{1f} - m_f | + | x_{2f} - m_f | + \cdots + | x_{nf} - m_f |) \quad (1)$$

- where x_{1f},..,x_{nf} are *n* measurements of *f*, and m_f is the mean value of *f*.

Step 2

- **Step 2.** Calculate the standardised measurement or z-score:

$$Z_{if} = \frac{x_{if} - m_f}{s_f} \tag{2}$$

Summary

- The mean absolute deviation s_f, is more robust to outliers than the *standard deviation* σf.
- When computing the mean absolute deviation, the deviations from the mean ($| x_{if} - m_f |$) are not squared.
- Thus, the effect of outliers is somewhat reduced.
- There are more robust measures of dispersion, such as the **median absolute deviation**.
- However, the advantage of using the mean absolute deviation is that the *z*-scores of outliers do not become too small and thus, outliers remain detectable.

Euclidean Distance

- After standardization, the dissimilarity (or similarity) between the objects described by interval-scaled variables is typically computed based on the distance between each pair of objects.
- The most popular distance measure is **Euclidean Distance** is defined as:

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{in} - x_{jn})^2} \tag{3}$$

- where $i = (x_{i1}, x_{i2}, \ldots x_{in})$ and $j = (x_{j1}, x_{j2}, \ldots x_{jn})$ are 2 n-dimensional data objects.

Manhattan Distance

- Another well-known metric is **Manhattan Distance**, defined as:

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \cdots + |x_{in} - x_{jn}| \quad (4)$$

Requirements

Both the Euclidean distance and Manhattan distance satisfy the following mathematic requirements of a distance function:

1. d(i,j) $\geq$ 0: Distance is a non-negative number.
2. d(i,i) = 0: The distance of an object to itself is 0.
3. d(i,j) = d(j,i): Distance is a symmetric function.
4. d(i,j) $\leq$ d(i,h)+d(h,j): Going directly from object *i* to object *j* in space is no more than making a detour over any other object *h* (triangular inequality).

Examples

Let x_1 = (1,2) and x_2 = (3,5) represent two objects.

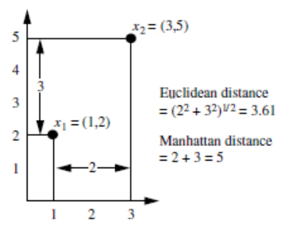

Figure 3: Euclidean and Manhatten distances between 2 objects

Binary Variables

- A binary variable has only two states: 0 or 1, where 0 means that the variable is absent, and 1 means that it is present.
- Given the variable *smoker* describing a patient: 1 indicates that the patient smokes, while 0 indicates that the patient does not.
- Treating binary variables as if they are interval-scaled can lead to misleading clustering results.
- Therefore, methods specific to binary data are necessary for computing dissimilarities.

Compute Dissimilarity between two binary variables

- One approach is to compute a dissimilarity matrix from the given binary data.
- If all binary variables are thought of as having the same weight, we have the 2-by-2 contingency table of figure 4, where:
  - q is the number of variables that equal 1 for both objects i and j;
  - r is the number of variables that equal 1 for object i but that are 0 for object j;
  - s is the number of variables that equal 0 for object i but equal 1 for object j;
  - and t is the number of variables that equal 0 for both objects i and j.
- The total number of variables is p, where $p = q+r+s+t$.

Contingency Table

		object j		
		1	0	sum
	1	q	r	$q+r$
object i	0	s	t	$s+t$
	sum	$q+s$	$r+t$	p

Figure 4: Contingency Table for Binary Variables

Symmetric binary variables

- A binary variable is symmetric if both of its states are equally valuable and carry the same weight: there is no preference on which outcome should be coded as 0 or 1.
- One such example could be the attribute *gender* having the states *male* and *female*.
- Dissimilarity that is based on symmetric binary variables is called **symmetric binary dissimilarity**.
- Its dissimilarity (or distance) measure defined in Equation 5, can be used to assess the dissimilarity between objects i and j.

$$d(i,j) = \frac{r+s}{q+r+s+t} \tag{5}$$

Asymmetric binary variables

- A binary variable is asymmetric if the outcomes of the states are not equally important, such as the *positive* and *negative* outcomes of a disease test.
- By convention, we shall code the most important outcome which is usually the rarest one, by 1 (e.g., HIV positive) and the other by 0 (e.g., HIV negative).
- Given two asymmetric binary variables, the agreement of two 1s (a positive match) is then considered more significant than that of two 0s (a negative match).
- Therefore, such binary variables are often considered *monary* (as if having one state).
- The dissimilarity based on such variables is called **asymmetric binary dissimilarity**, where the number of negative matches t, is considered unimportant and thus, ignored in computation, as shown in Equation 6.

$$d(i,j) = \frac{r+s}{q+r+s} \tag{6}$$

Jaccard coefficient

- Alternatively, one can measure the distance between two binary variables based on the notion of *similarity* instead of dissimilarity.
- For example, the **asymmetric binary similarity** between the objects *i* and *j*, or *sim(i,j)* is shown below.
- The coefficient *sim(i,j)* is called the Jaccard coefficient.

$$sim(i,j) = \frac{q}{q+r+s} = 1 - d(i,j) \tag{7}$$

Binary Attributes Example

name	*gender*	*fever*	*cough*	*test-1*	*test-2*	*test-3*	*test-4*
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	Y	N	N	N	N
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 5: Table with Patients described by binary attributes

Dissimilarity between binary variables

- Suppose that a patient record table (Figure 5) contains the attributes name, gender, fever, cough, test-1, test-2, test-3, and test-4, where *name is an object identifier*, **gender is a symmetric attribute**, and the remaining attributes are *asymmetric binary*.
- For asymmetric attribute values, let the values Y (yes) and P (positive) be set to 1, and the value N (no or negative) be set to 0.
- Suppose that the distance between objects (patients) is computed based only on the asymmetric variables.
- The distance between each pair of the three patients, Jack, Mary, and Jim, is calculated using equation 6.

Dissimilarity Values

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33$$
$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67$$
$$d(Mary, Jim) = \frac{1+2}{1+1+2} = 0.75$$

- These measurements suggest that Mary and Jim are unlikely to have a similar disease because they have the *highest dissimilarity value* among the three pairs.
- Of the three patients, Jack and Mary are the most likely to have a similar disease.

Categorical Variables

- A **categorical variable** is a generalization of the binary variable in that it can take on more than two states.
- For example, map color is a categorical variable that may have five states: red, yellow, green, pink, and blue.
- Let the number of states of a categorical variable be M.
- The states can be denoted by letters, symbols, or a set of integers, such as $1, 2, \ldots, M$.

Dissimilarity by categorical variables

The dissimilarity between two objects *i* and *j* can be computed based on the ratio of mismatches.

$$d(i,j) = \frac{p - m}{p} \tag{8}$$

where *m* is the number of matches (the number of variables for which *i* and *j* have the same state) and *p* is the total number of variables.
Weights can be assigned to increase the effect of *m* or to assign greater weight to the matches in variables having a larger number of states.

Categorical Example

object identifier	test-1 (categorical)
1	code-A
2	code-B
3	code-C
4	code-A

Figure 6: Categorical Data

Assume we have the sample data of Table 6, with only the object-identifier and the variable *test-1* which is categorical.

Categorical Example

$$\begin{bmatrix} 0 & & & \\ d(2,1) & 0 & & \\ d(3,1) & d(3,2) & 0 & \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{bmatrix}$$

a)

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ 1 & 1 & 0 & \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

b)

Figure 7: a) Dissimilarity Matrix . . . b) Binary Variable Encoding

Since here we have one categorical variable *test-1*, we set *p = 1* in Equation 8 so that *d(i,j)* evaluates to 0 if objects *i* and *j* match, and 1 if the objects differ.

Partitioning Methods

- Given D, a data set of n objects and k, the number of clusters to form, a partitioning algorithm organizes the objects into k partitions ($k \leq n$), where each partition represents a cluster.
- The clusters are formed to optimize an *objective partitioning criterion*, such as a dissimilarity function based on distance, so that:
  the objects within a cluster are *similar*,
  whereas the objects of different clusters are *dissimilar*
  in terms of the data set attributes.
- Cluster similarity is measured in regard to the *mean value* of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity.

The *k*-means algorithm

- First, it randomly selects *k* of the objects, each of which initially represents a *cluster mean* or *center*.
- Each remaining object is then assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean.
- It then computes the new mean for each cluster.
- This process iterates until the criterion function converges.

Square-error Criterion

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} | p - m_i |^2 \quad (9)$$

- where E is the sum of the square error for all objects in the data set;
  p is the point in space representing a given object; and m_i is the mean of cluster C_i (both p and m_i are multi-dimensional).
- In other words, for every object in each cluster, the distance from the object to its cluster center is squared, and the distances are summed.
- This criterion tries to make the resulting k clusters as compact and as separate as possible.

Similarity between objects

- There are many algorithms for clustering.
- We focus on two methods for which the *similarity* between objects is based on *a measure of the distance between them*.
- In the restricted case where each object is described by the values of just two attributes, we can represent them as points in a two-dimensional space as in Figure 8.

Objects for Clustering

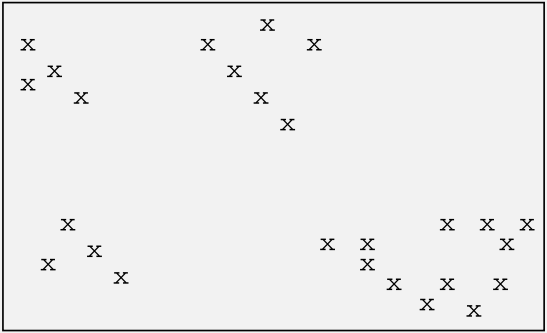

Figure 8: Can you see the obvious Clusters?

Clusters: One Possibility

- It is usually easy to visualise clusters in two dimensions.
- The points in Figure 8 seem to fall naturally into four groups as shown in Figure 9.

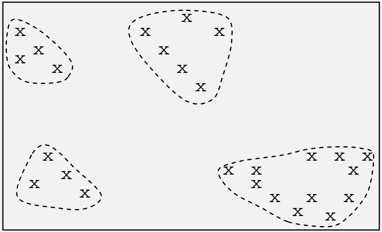

Figure 9: Four Clusters

Clusters: frequently more than one possibility

- Are the points in the lower-right corner of Figure 8 one cluster (Figure 9) or two (Figure 10)?

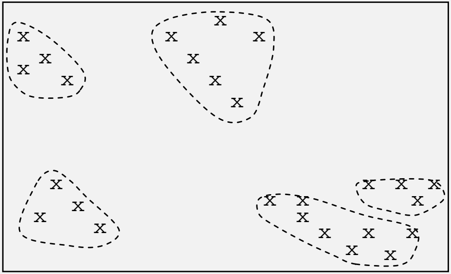

Figure 10: Five Clusters

Clustering and Multi-dimensions

- For 3 attributes, we can think of the objects as being points in a 3-D space (such as a room) and visualising clusters is fairly easy.
- For larger dimensions, we cannot!
- For simplicity, we will use only 2 dimensions although in practice, the number of attributes will usually be more than 2 and can often be large.
- Before using a distance-based clustering algorithm to cluster objects, it is first necessary to decide on a way of measuring the distance between two points.

Euclidean Distance

- As for *nearest neighbour classification*, we again use the *Euclidean distance*.
- To avoid complications, we assume that all attribute values are continuous.
- First, we introduce the notion of the *centre* of a cluster, generally called its **centroid**.
- Assuming that we are using Euclidean distance or something similar as a measure, we can define the **centroid of a cluster** to be *the point for which each attribute value is the average of the values of the corresponding attribute* for all the points in the cluster.

Calculating the Centroid

Centroid of the four points (with 6 attributes)

Table 1: Centroid calculated at the bottom of each column

8.0	7.2	0.3	23.1	11.1	-6.1
2.0	-3.4	0.8	24.2	18.3	-5.2
-3.5	8.1	0.9	20.6	10.2	-7.3
-6.0	6.7	0.5	12.5	9.2	-8.4
0.125	4.65	0.625	20.1	12.2	-6.75

Centroid Approach

- The centroid of a cluster will sometimes be one of the points in the cluster.
- Frequently, as in the previous example, it will be an *imaginary* point, not part of the cluster itself, which we can take as marking its centre.
- There are many methods of clustering.
- We will examine two of the most commonly used: *k-means clustering* and *hierarchical clustering*.

k-Means Clustering

- *k*-means clustering is an *exclusive* clustering algorithm.
- Each object is assigned to precisely *one* of a set of clusters. (There are other methods that allow objects to be in more than one cluster.)
- Begin by deciding how many clusters one would like to form from the data.
- We call this value *k*.
- The value of *k* is generally a small integer, such as 2, 3, 4 or 5, but may be larger.

k-means Approach (1)

- There are many ways in which k clusters might potentially be formed.
- We can measure the *quality* of a set of clusters using the value of an objective function which we will take to be the *sum of the squares of the distances of each point from the centroid* of the cluster to which it is assigned.
- We would like the value of this function to be as small as possible.

k-means Approach (2)

- We next select *k* points (generally corresponding to the location of *k* of the objects).
- These are treated as the centroids of *k* clusters, or to be more precise as the centroids of *k* potential clusters, which at present have no members.
- We can select any points initially, but the method should work better if we pick *k* initial points that are far apart.
- We now assign each of the points one by one to the cluster which has the nearest centroid.

k-means Approach (3)

- When all the objects have been assigned, we will have k clusters based on the original k centroids but the 'centroids' will no longer be the *true centroids* of the clusters.
- Thus, we recalculate the centroids of the clusters, and then repeat the previous steps, assigning each object to the cluster with the nearest centroid etc.
- The algorithm is summarised on the following slide.

The *k*-Means Clustering Algorithm

1. Choose a value of *k*.
2. Select *k* objects in an arbitrary fashion. There are the initial set of *k* centroids.
3. Assign each of the objects to the cluster for which it is nearest to the centroid.
4. Recalculate the centroids of the *k* clusters.
5. Repeat steps 3 and 4 until the centroids no longer move.

- We can illustrate the *k*-means algorithm by using it to cluster the 16 objects with two attributes *x* and *y*.

x	y
6.8	12.6
0.8	9.8
1.2	11.6
2.8	9.6
3.8	9.9
4.4	6.5
4.8	1.1
6.0	19.9
6.2	18.5
7.6	17.4
7.8	12.2
6.6	7.7
8.2	4.5
8.4	6.9
9.0	3.4
9.6	11.1

Figure 11: Objects for Clustering

- The 16 points from figure 11 are shown diagrammatically in Figure 12.

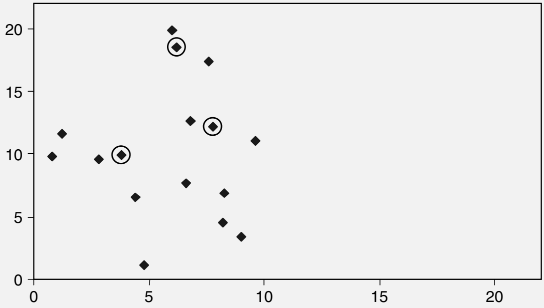

Figure 12: Horizontal (x) and vertical (y) axes.

Initial Centroids

- Three of the points shown in Figure 12 are highlighted by small circles.
- Assume *k = 3* and that these three points have been selected to be the locations of the initial three centroids.

	Initial	
	x	y
Centroid 1	3.8	9.9
Centroid 2	7.8	12.2
Centroid 3	6.2	18.5

Figure 13: Initial Choice of Centroids

Euclidean Distance

- The columns headed *d1*, *d2* and *d3* in Figure 14 show the Euclidean distance of each of the 16 points from the three centroids.
- For the purposes of this example, we will not normalise or weight either of the attributes.
- Thus, the distance of the first point (6.8, 12.6) from the first centroid (3.8, 9.9) is:

$$\sqrt{(6.8-3.8)^2+(12.6-9.9)^2}=4.0$$

Column *cluster* indicates the centroid closest to each point.

x	y	$d1$	$d2$	$d3$	cluster
6.8	12.6	4.0	1.1	5.9	2
0.8	9.8	3.0	7.4	10.2	1
1.2	11.6	3.1	6.6	8.5	1
2.8	9.6	1.0	5.6	9.5	1
3.8	9.9	0.0	4.6	8.9	1
4.4	6.5	3.5	6.6	12.1	1
4.8	1.1	8.9	11.5	17.5	1
6.0	19.9	10.2	7.9	1.4	3
6.2	18.5	8.9	6.5	0.0	3
7.6	17.4	8.4	5.2	1.8	3
7.8	12.2	4.6	0.0	6.5	2
6.6	7.7	3.6	4.7	10.8	1
8.2	4.5	7.0	7.7	14.1	1
8.4	6.9	5.5	5.3	11.8	2
9.0	3.4	8.3	8.9	15.4	1
9.6	11.1	5.9	2.1	8.1	2

Figure 14: Objects for Clustering (Augmented)

Initial Clusters

The resulting centroids for Figure 14

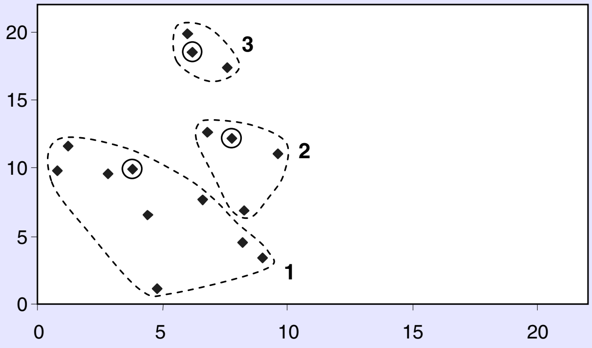

Figure 15: Initial Clusters and Allocations

Next Iteration

- The centroids are indicated by small circles.
- For this first iteration, they are also actual points within the clusters.
- The centroids are those that were used to construct the three clusters but are not the true centroids of the clusters once they have been created.
- We next calculate the centroids of the three clusters using the x and y values of the objects currently assigned to each centroid.

Centroids after First Iteration

- The results are shown in Figure 16.

	Initial		After first iteration	
	x	y	x	y
Centroid 1	3.8	9.9	4.6	7.1
Centroid 2	7.8	12.2	8.2	10.7
Centroid 3	6.2	18.5	6.6	18.6

Figure 16: Centroids: Initial and First Iteration

Revised Clusters (Figure 17)

- The three centroids have all been moved by the assignment process (the third by much less).
- Next, reassign the 16 objects to one of three clusters.

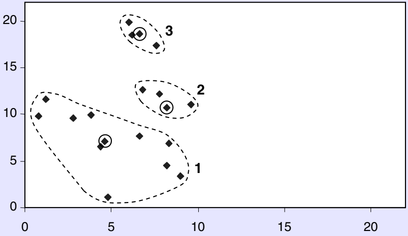

Figure 17: Revised Clusters

After first Reassignment

- The centroids are again indicated by small circles.
- However, from now on the centroids are *imaginary points* corresponding to the *centre* of each cluster, not actual points within the clusters.
- These clusters are very similar to the previous three, shown in Figure 15.
- In fact, only one point has moved clusters: the object at (8.3, 6.9) has moved from cluster 2 to cluster 1.
- Next, recalculate the positions of the three centroids (see figure 18).

After 2 Iterations

- The first two centroids have moved a little, but the third has not moved at all.
- Now reassign the 16 objects to clusters (Figure 19).

	Initial		After first iteration		After second iteration	
	x	y	x	y	x	y
Centroid 1	3.8	9.9	4.6	7.1	5.0	7.1
Centroid 2	7.8	12.2	8.2	10.7	8.1	12.0
Centroid 3	6.2	18.5	6.6	18.6	6.6	18.6

Figure 18: Centroids after First 2 Iterations

Clusters: Third Iteration

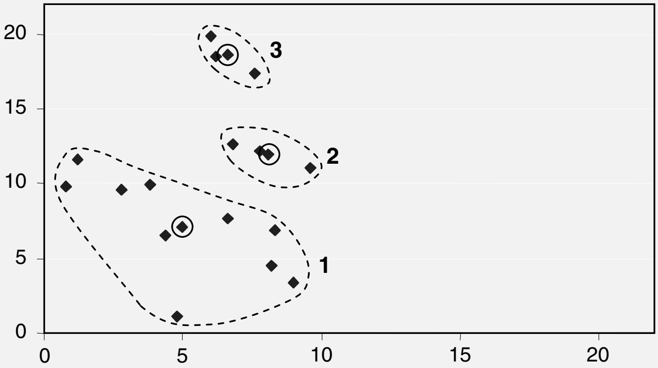

Figure 19: Third set of Clusters

Terminating the Process

- These are the same clusters as before.
- Their centroids will be the same as those from which the clusters were generated.
- Thus, the termination condition of the *k*-means algorithm *repeat ... until the centroids no longer move* has been met.
- These are the final clusters produced by the algorithm for the *initial choice of centroids made*.

Finding the Best Set of Clusters

- The basic clustering problem is simple to state.
- Given a set of n distinguishable objects, we wish to distribute the objects into groups or clusters in such a way that **the objects within each group are similar** whereas **the groups themselves are different**.
- While the k-means algorithm will always terminate, it does not necessarily find the best set of clusters, corresponding to minimising the value of the objective function.
- The initial selection of centroids can significantly affect the result.

Finding a value for *k*

- To overcome this, the algorithm can be run several times for a given value of *k*, each time with a different choice of the initial *k* centroids, the set of clusters with the *smallest value* of the objective function then being taken.
- The obvious drawback of this approach is that there is no way to know what the value of *k* ought to be.
- Looking at the final set of clusters in the above example (Figure 19), it is not clear that *k = 3* is the most appropriate choice.
- Cluster 1 might well be broken into several separate clusters.

Systematic values for *k*

- We can choose a value of k pragmatically as follows.
- Assume choosing $k = 1$, i.e. all the objects are in a single cluster, with the initial centroid selected in a random way (a very poor idea): the value of the objective function is likely to be large.
- We can then try $k = 2$, $k = 3$ and $k = 4$, each time experimenting with a different choice of the initial centroids and choosing the set of clusters with the smallest value.
- Figure 20 shows the (imaginary) results of such a series of experiments.

Analysis

- Results suggest that the best value of k is probably 3.
- The value of the function for $k = 3$ is much less than for
  $k = 2$, but only a little better when $k = 4$.

Value of k	Value of objective function
1	62.8
2	12.3
3	9.4
4	9.3
5	9.2
6	9.1
7	9.05

Figure 20: Value of Objective Function For Different Values of k

Conclusions

- It is possible that the value of the objective function drops sharply after $k = 7$. However, $k = 3$ is probably the best choice.
- We normally try for a fairly small number of clusters.
- Note that we are *not* trying to find the value of k with the smallest value of the objective function.
- That will occur when the value of k is the same as the number of objects, i.e. each object forms its own cluster of one (worthless)
- We usually want a fairly small number of clusters and accept that the objects in a cluster will be spread around the centroid (but ideally not too far away).

k-Medoids Method

- The k-means algorithm is *sensitive to outliers* because an object with an extremely large value may substantially distort the distribution of data.
- This effect is particularly exacerbated due to the use of the square-error function (Equation 9).
- To diminish such sensitivity, instead of taking the mean value of the objects in a cluster as a reference point, pick *actual objects* to represent the clusters, using one representative object per cluster.
- Each remaining object is clustered with the **representative** object to which it is the most similar.

k-Medoids: Algorithm

The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. This **absolute-error criterion** is defined as:

$$E = \sum_{j=1}^{k} \sum_{p \in c_j} | p - o_j | \qquad (10)$$

- where E is the sum of the absolute error for all objects in the data set;
- p is the point in space representing a given object in cluster C_j;
- and o_j is the representative object of C_j.

The Medoid

- In general, the algorithm iterates until each representative object is actually the medoid, or most centrally located object, of its cluster.
- This is the basis of the *k*-medoids method for grouping *n* objects into *k* clusters.

Examining *k*-medoids clustering

- The initial representative objects (or seeds) are chosen arbitrarily.
- The iterative process of replacing representative objects by *non representative* objects continues as long as the quality of the resulting clustering is improved.
- This quality is estimated using a cost function that measures the *average dissimilarity* between an object and the representative object of its cluster.

Reassigning the Representative Object

Is o_{random}, is a good replacement for o_j? There are four cases are as in figure 21.

- Case 1: p currently belongs to representative object o_j. If oj is replaced by o_{random} as a representative object and p is closest to one of the other representative objects o_i, $i \neq j$, then p is reassigned to o_i.
- Case 2: p currently belongs to representative object o_j. If o_j is replaced by o_{random} as a representative object and p is closest to o_{random}, then p is reassigned to o_{random}.
- Case 3: p currently belongs to representative object o_i, $i \neq j$. If o_j is replaced by o_{random} as a representative object and p is still closest to o_i, then the assignment does not change.
- Case 4: p currently belongs to representative object o_i, $i \neq j$. If o_j is replaced by o_{random} as a representative object and p is closest to o_{random}, then p is reassigned to o_{random}.

Cost Function for *k*-medoids clustering

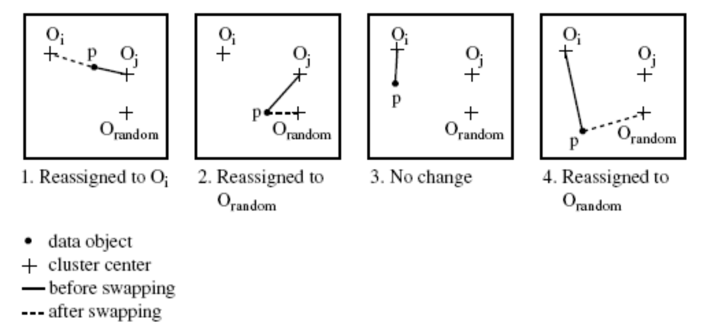

Figure 21: 4 Cases of the Cost Function

How it Works

- Each time a reassignment occurs, a *difference* in absolute error E is contributed to the cost function.
- Therefore, the cost function calculates the *difference* in absolute-error value if a current representative object is replaced by a non-representative object.
- The total cost of swapping is the *sum of costs* incurred by all non-representative objects.
- If the total cost is negative, then o_j is replaced or swapped with o_{random} since the actual absolute error E would be reduced.
- If the total cost is positive, the current representative object o_j, is considered acceptable and nothing is changed in the iteration.

Partitioning Around Medoids

PAM (Partitioning Around Medoids) attempts to determine *k* partitions for *n* objects.

Algorithm: ***k*-medoids.** PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose k objects in D as the initial representative objects or seeds;
(2) **repeat**
(3) assign each remaining object to the cluster with the nearest representative object;
(4) randomly select a nonrepresentative object, o_{random};
(5) compute the total cost, S, of swapping representative object, o_j, with o_{random};
(6) **if** $S < 0$ **then** swap o_j with o_{random} to form the new set of k representative objects;
(7) **until** no change;

Figure 22: PAM: k-medoids Partitioning Algorithm

Logic

- After an initial random selection of k representative objects, the algorithm repeatedly tries to make a better choice of cluster representatives.
- All of the possible pairs of objects are analysed, where one object in each pair is considered a representative object and the other is not.
- The quality of the resulting clustering is calculated for each such combination.
- An object o_j, is replaced with the object causing the greatest reduction in error.
- The set of best objects for each cluster in one iteration forms the representative objects for the next iteration.
- The final set of representative objects are the respective medoids of the clusters.

k-means or *k*-medoids?

- The *k*-medoids method is more robust than *k*-means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean.
- However, its processing is more costly than the *k*-means method.
- Both methods require the user to specify *k*, the number of clusters.

Agglomerative Hierarchical Clustering

- Another popular clustering technique is called *Agglomerative Hierarchical Clustering*.
- As with *k*-means clustering, one must choose a way of measuring the distance between two objects.
- Again, Euclidean distance is used.
- In two dimensions, Euclidean distance is just the *straight line* distance between two points.
- The idea behind Agglomerative Hierarchical Clustering is a simple one.

AHC: Basic Algorithm

We start with each object in a cluster of its own and then repeatedly merge the closest pair of clusters until we end up with just one cluster containing everything.

1. Assign each object to its own single-object cluster. Calculate the distance between each pair of clusters.
2. Choose the *closest pair* of clusters and merge them into a single cluster (so reducing the total number of clusters by one).
3. Calculate the distance between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all the objects are in a single cluster.

Example with 11 objects

- If there are *N* objects there will be *N-1* mergers of two objects needed to produce a single cluster.
- However, the method does not only produce a single large cluster, it generates a *hierarchy* of clusters.
- Suppose we start with eleven objects *A,B,C,. . . ,K* located as shown in Figure 23 and we merge clusters on the basis of Euclidean distance.
- It will take 10 passes through the algorithm (repetitions of Steps 2 and 3), to merge the initial 11 single object clusters into a single cluster.

- Let us assume the process starts by choosing objects A and B as the pair that are closest and merging them into a new cluster which we will call AB.
- The next step may be to choose clusters AB and C as the closest pair and to merge them.

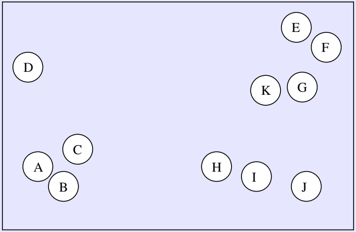

Figure 23: Original Data (11 Objects)

- After two passes the clusters look as shown in Figure 24.
- We will use notation such as A and B $\rightarrow$ AB to mean: clusters A and B are merged to form new cluster AB.

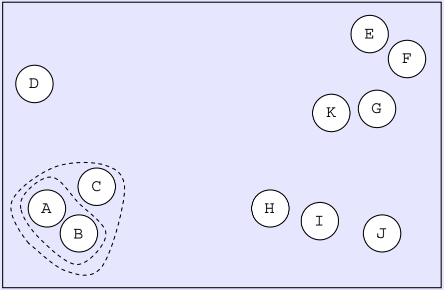

Figure 24: Clusters After Two Passes

Sequence of Operations (1)

Without knowing the precise distances between each pair of objects, a plausible sequence of events is as follows.

1. A and B $\rightarrow$ AB
2. AB and C $\rightarrow$ ABC
3. G and K $\rightarrow$ GK
4. E and F $\rightarrow$ EF
5. H and I $\rightarrow$ HI

Sequence of Operations (2)

Without knowing the precise distances between each pair of objects, a plausible sequence of events is as follows.

6. EF and GK $\rightarrow$ EFGK
7. HI and J $\rightarrow$ HIJ
8. ABC and D $\rightarrow$ ABCD
9. EFGK and HIJ $\rightarrow$ EFGKHIJ
10. ABCD and EFGKHIJ $\rightarrow$ ABCDEFGKHIJ

- The final result of this hierarchical clustering process is shown in Figure 25, which is called a dendrogram.
- A dendrogram is a binary tree (two branches at each node).

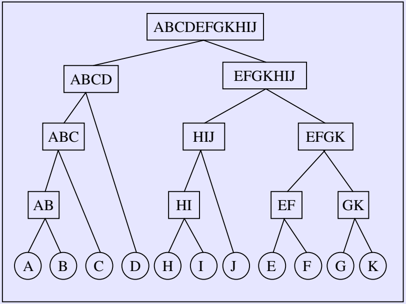

Figure 25: A Possible Dendrogram for Figure 23

Dendrogram Properties (1)

- However, the positioning of the clusters does not correspond to their physical location in the original diagram.
- All the original objects are placed at the same level (the bottom of the diagram), as leaf nodes.
- The root of the tree is shown at the top of the diagram.
- It is a **cluster** containing *all* the objects.

Dendrogram Properties (2)

- The other nodes show smaller clusters that were generated as the process proceeded.
- If we call the bottom row of the diagram **level 1** (with clusters A, B, C, . . . , K):
  - we can say that the **level 2** clusters are AB, HI, EF and GK;
  - the **level 3** clusters are ABC, HIJ and EFGK, and so on.
- The root node is at **level 5**.

Recording the Distance Between Clusters

- It would be very inefficient to calculate the distance between each pair of clusters for each pass through the algorithm, especially as the distance between those clusters not involved in the most recent merger cannot have changed.
- The usual approach is to generate and maintain a distance matrix giving the distance between each pair of clusters.

Sample Distance Matrix

- If we have six objects *a, b, c, d, e* and *f*, the initial distance matrix might look like Figure 26.

	a	b	c	d	e	f
a	0	12	6	3	25	4
b	12	0	19	8	14	15
c	6	19	0	12	5	18
d	3	8	12	0	11	9
e	25	14	5	11	0	7
f	4	15	18	9	7	0

Figure 26: Sample Distance Matrix

Using the Distance Matrix

- Note that the table is symmetric, so not all values have to be calculated (the distance from *c* to *f* is the same as the distance from *f* to *c* etc.).
- The values on the diagonal from the top-left corner to the bottom-right corner must always be zero (the distance from *a* to *a* is zero etc.).
- From the distance matrix of Figure 26, we can see that the closest pair of clusters (single objects) are *a* and *d*, with a distance value of 3.
- We combine these into a single cluster of two objects which we will call *ad*.

New Distance Matrix

- We can now rewrite the distance matrix with rows *a* and *d* replaced by a single row *ad* and similarly for the columns.
- The entries in the matrix for the various distances between *b, c, e* and *f* obviously remain the same, but how should we calculate the entries in row and column *ad*?

	ad	b	c	e	f
ad	0	?	?	?	?
b	?	0	19	14	15
c	?	19	0	5	18
e	?	14	5	0	7
f	?	15	18	7	0

Figure 27: Distance Matrix after First Merger (Incomplete)

Calculating for Merged Values

- We could calculate the position of the centroid of cluster *ad* and use that to measure the distance of cluster *ad* from clusters *b, c, e* and *f*.
- However, for hierarchical clustering a different approach, which involves less calculation, is generally used.
- In **single-link** clustering, the distance between two clusters is taken to be the *shortest distance* from any member of one cluster to any member of the other cluster.
- On this basis the distance from *ad* to *b* is 8, the shorter of the distance from *a* to *b* (12) and the distance from *d* to *b* (8) in the original distance matrix.

After Merger

- Alternatives to single-link clustering are **complete-link** and *average-link* clustering, where the distance between two clusters is taken to be the **longest distance** from any member of one cluster to any member of the other cluster, or the *average distance* respectively.
- Returning to the example and assuming that we are using single-link clustering, the position after the first merger is:

	ad	b	c	e	f
ad	0	8	6	11	4
b	8	0	19	14	15
c	6	19	0	5	18
e	11	14	5	0	7
f	4	15	18	7	0

Figure 28: Distance Matrix after First Merger

Distance Matrix after 2 Mergers

- The smallest (non-zero) value in the table is now 4, which is the distance between cluster *ad* and cluster *f*, so we next merge these clusters to form the three-object cluster *adf*.
- The distance matrix now becomes Figure 29.

	adf	b	c	e
adf	0	8	6	7
b	8	0	19	14
c	6	19	0	5
e	7	14	5	0

Figure 29: Distance Matrix after Two Mergers

Distance Matrix after 3 Mergers

- The smallest non-zero is now 5, the distance from cluster *c* to cluster *e*.
- These clusters are now merged into a single new cluster *ce* and the distance matrix is changed to Figure 30.

	adf	b	ce
adf	0	8	6
b	8	0	14
ce	6	14	0

Figure 30: Distance Matrix after Three Mergers

Distance Matrix after 4 Mergers

- Clusters *adf* and *ce* are now the closest, with distance 6 and are merged into a single cluster *adfce*.
- The distance matrix becomes Figure 31.

	$adfce$	b
$adfce$	0	8
b	8	0

Figure 31: Distance Matrix after Four Mergers

- Finally, clusters *adfce* and *b* are merged into a single cluster *adfceb* containing the original six objects.

Dendrogram for Hierarchical Clustering

- The dendrogram is shown in Figure 32.

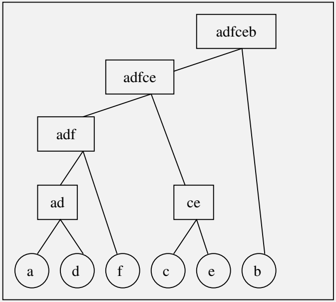

Figure 32: Dendrogram for Hierarchical Clustering

Terminating the Clustering Process

- Often we are content to allow the clustering algorithm to produce a complete cluster hierarchy.
- However, we may prefer to end the merger process when we have converted the original *N* objects to a *small enough* set of clusters.
- We can do this in several ways.
- For example, we can merge clusters until only some pre-defined number remain.
- Alternatively, we can stop merging when a newly created cluster fails to meet some criterion for its compactness, e.g. the average distance between the objects in the cluster is too high.

Outlier Analysis

- Many data mining algorithms try to minimize the influence of outliers or eliminate them all together.
- However, this may result in the loss of important *hidden* information because one person's noise could be another person's signal.
- In other words, the outliers may be of particular interest, such as in the case of fraud detection, where outliers may indicate fraudulent activity.
- Thus, outlier detection and analysis is an interesting data mining task, referred to as *outlier mining*.

Outlier Mining

- Given a set of n data points or objects and k expected number of outliers, find the top k objects that are considerably dissimilar, exceptional, or inconsistent with respect to the remaining data.
- The outlier mining problem can be viewed as two sub-problems:
  1. define what data can be considered as *inconsistent* in a given data set, and
  2. find an efficient method to mine the outliers as defined.

Defining outliers is Non-trivial!

- If a regression model is used for data modeling, analysis of the residuals (difference between the observed value and predicted value of x) can give a good estimation for data *extremeness*.
- The task becomes difficult when finding outliers in time-series data, as they may be hidden in trend, seasonal, or other cyclic changes.
- When multidimensional data are analyzed, not any particular one but rather a combination of dimension values may be extreme.
- For nonnumeric (i.e., categorical) data, the definition of outliers requires special consideration.

Statistical Distribution-Based Outlier Detection

- The statistical distribution-based approach to outlier detection assumes a *probability model* for the given data set (a normal or Poisson distribution), and then identifies outliers with respect to the model using a *discordancy test*.
- Application of the test requires knowledge of the data set parameters (such as the **assumed data distribution**), knowledge of distribution parameters (such as the **mean** and **variance**), and the *expected* number of outliers.
- A statistical discordancy test examines two hypotheses: a *working hypothesis* and an *alternative hypothesis*.
- A working hypothesis H, is a statement that the entire data set of n objects comes from an initial distribution model F:

$$H : o_i \in F, \text{ where } i = 1, 2, \ldots, n. \tag{11}$$

How does discordancy testing work?

- The hypothesis is retained if there is no statistically significant evidence supporting its rejection.
- A discordancy test verifies whether an object o_i, is significantly large (or small) in relation to the distribution *F*.
- Assuming that some statistic *T*, has been chosen for discordancy testing, and the value of the statistic for object o_i is v_i, then the distribution of *T* is constructed.
- Significance probability $SP(v_i)$=Prob(T > v_i), *is evaluated*.
- If SP(v_i) is sufficiently small, then o_i is discordant and the working hypothesis is rejected.

Alternative Hypothesis

- An alternative hypothesis H, which states that o_i comes from another distribution model G, is adopted.
- The result is dependent on which model F is chosen because o_i may be an outlier under one model and a perfectly valid value under another.
- The alternative distribution is very important in determining the power of the test: the probability that the working hypothesis is rejected when o_i is really an outlier.

Inherent Alternative Distribution

The working hypothesis that all of the objects come from distribution F is rejected in favor of the alternative hypothesis that all of the objects arise from another distribution G:

$$\bar{H} : o_i \in G, \textit{where } i = 1, 2, \ldots, n. \quad (12)$$

- F and G may be different distributions or differ only in parameters of the same distribution.
- There are constraints on the form of the G distribution in that it must have potential to produce outliers.
- For example, it may have a different mean or dispersion or a longer tail.

Alternatives

- The **mixture alternative** states that discordant values are not outliers in the F population, but contaminants from some other population G.
- This **slippage alternative** states that all of the objects (apart from some prescribed small number) arise independently from the initial model F, with its given parameters whereas the remaining objects are independent observations from a modified version of F in which the parameters have been shifted.

Procedures for Outlier Detection

1. **Block Procedures**: In this case, either *all* of the suspect objects are treated as outliers or all of them are accepted as consistent.
2. **Sequential Procedures**: An example of such a procedure is the *inside-out procedure*.
   Its main idea is that the object that is *least* likely to be an outlier is tested first.
   If it is found to be an outlier, then all of the more extreme values are also considered outliers; otherwise, the next most extreme object is tested, and so on.
3. Sequential tends to be more effective than block procedures.

Effectiveness of Statistical Approach

- A major drawback is that most tests are for *single* attributes, yet many data mining problems require finding outliers in multidimensional space.
- The statistical approach requires knowledge about parameters of the data set, such as the data distribution. However, in many cases, the data distribution may not be known.
- Statistical methods do not guarantee that all outliers will be found for the cases where no specific test was developed, or where the observed distribution cannot be adequately modeled with any standard distribution.