Cluster Analysis

Module - 6

Clustering

Clustering is the unsupervised learning task, where the goal is to segment the data into a set of homogeneous clusters of records for the purpose of generating insight.

Separating a dataset into clusters of homogeneous records is also useful for improving performance of supervised methods, by modeling each cluster separately rather than the entire, heterogeneous dataset.

sample

Cluster/group

Cluster analysis

Cluster analysis is used to form groups or clusters of similar records based on several measurements made on these records.

The key idea is to characterize the clusters in ways that would be useful for the aims of the analysis. This idea has been applied in many areas, including astronomy, archaeology, medicine, chemistry, education, psychology, linguistics, and sociology.

One popular use of cluster analysis in marketing is for *market segmentation:*

customers are segmented based on demographic and transaction history information, and a marketing strategy is tailored for each segment.

Current Applications

- **Document Classification:** Cluster documents in multiple categories based on tags, topics, and the content of the document. This is a very standard classification problem and k-means is a highly suitable algorithm for this purpose.
- **Identifying Crime Localities:** With data related to crimes available in specific localities in a city, the category of crime, the area of the crime, and the association between the two can give quality insight into crime-prone areas within a city

Current Applications

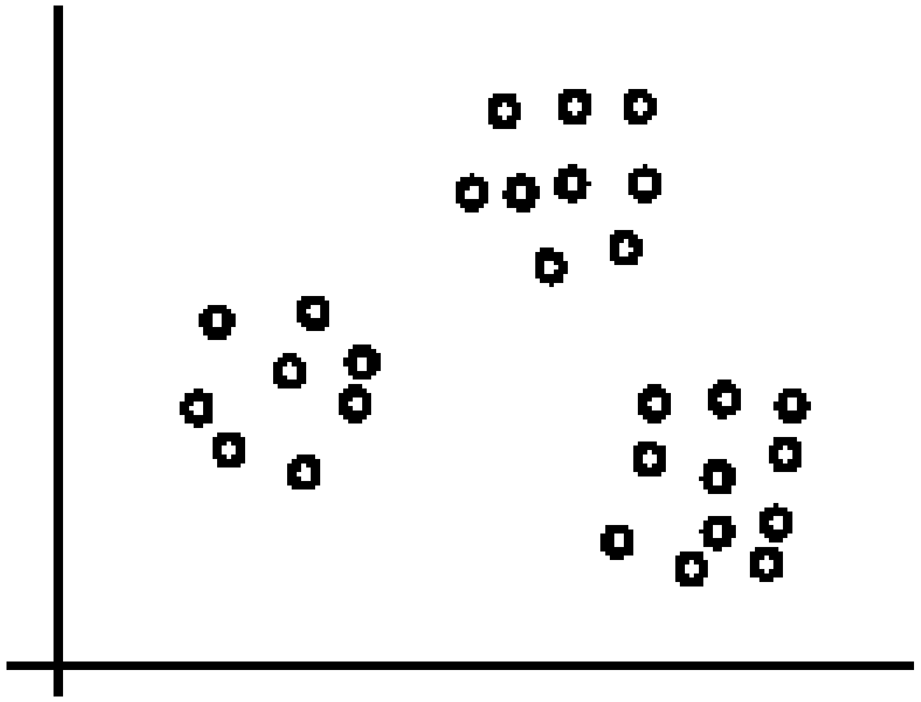
- **Customer Segmentation:** Clustering helps marketers improve their customer base, work on target areas, and segment customers based on purchase history, interests, or activity monitoring.
- **Insurance Fraud Detection:** Utilizing past historical data on fraudulent claims, it is possible to isolate new claims based on its proximity to clusters that indicate fraudulent patterns.

Current Applications

Rideshare Data Analysis: The publicly available Uber ride information dataset provides a large amount of valuable data around traffic, transit time, peak pickup localities, and more. Analyzing this data is useful not just in the context of Uber but also in providing insight into urban traffic patterns and helping us plan for the cities of the future.

Illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



Types of clustering algorithms

Two general types of clustering algorithms for a dataset of *n* records are hierarchical and non-hierarchical clustering:

- **Hierarchical methods** can be either *agglomerative* or *divisive*. Agglomerative methods begin with *n* clusters and sequentially merge similar clusters until a single cluster is obtained. Divisive methods work in the opposite direction, starting with one cluster that includes all records. Hierarchical methods are especially useful when the goal is to arrange the clusters into a natural hierarchy.

- **Non-hierarchical methods,** such as *k*-means. Using a prespecified number of clusters, the method assigns records to each cluster. These methods are generally less computationally intensive and are therefore preferred with very large datasets.
- In both cases, we need to define two types of distances: distance between two records and distance between two clusters.

Measuring distance between two records

We denote by *dij* a *distance metric*, or *dissimilarity measure*, between records *i* and *j*.

For record *i* we have the vector of *p* measurements (*xi*1*; xi*2*; : : : ; xip*),

while for record *j* we have the vector of measurements (*xj*1*; xj*2*; : : : ; xjp*).

Euclidean Distance

- The most popular distance measure is the *Euclidean distance*, d_{ij} , wh between two records, *i* and *j*, is defined by

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots + (x_{ip} - x_{jp})^2}.$$

Correlation based Similarity

- A popular similarity measure is the square of the Pearson correlation coefficient, $r_{ij}{}^2$, where the correlation coefficient is defined by

$$r_{ij} = \frac{\sum_{m=1}^{p} (x_{im} - \overline{x}_m)(x_{jm} - \overline{x}_m)}{\sqrt{\sum_{m=1}^{p} (x_{im} - \overline{x}_m)^2 \sum_{m=1}^{p} (x_{jm} - \overline{x}_m)^2}}.$$

$$d_{ij} = 1 - r_{ij}^2.$$

Statistical distance (Mahalanobis distance)

This metric has an advantage over the other metrics. It takes into account the correlation between measurements. The statistical distance between records i and j is defined as

$$d_{i,j} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)' S^{-1} (\mathbf{x}_i - \mathbf{x}_j)},$$

Manhattan Distance

The distance looks at the absolute differences rather than squared differences and is defined by

$$d_{ij} = \sum_{m=1}^{p} | \, x_{im} - x_{jm} \, | \, .$$

Maximum coordinate distance

This distance looks only at the measurement on which records i and j deviate most. It is define by

$$d_{ij} = \max_{m=1,2,\ldots,p} \mid x_{im} - x_{jm} \mid .$$

Distance measure for categorical data

In the case of measurements with binary values, it is more intuitively appealing to use similarity measures than distance measures. Suppose that we have binary values for all the x_{ij} ’s, and for records i and j we have the following 2X2 table:

		Record j		
		0	1	
Record i	0	a	b	$a+b$
	1	c	d	$c+d$
		$a+c$	$b+d$	n

where a denotes the number of variables for which records i and j do not have that attribute (they each have value 0 on that attribute), d is the number of variables for which the two records have the attribute present, and so on. The most useful similarity measures in this situation are:

Matching coefficient: (a+d)/n

Jaquard's coefficient: d/(b+c+d)

Distance measures for mixed data

When the measurements are mixed (some continuous and some binary), a similarity coefficient suggested by Gower is very useful. *Gower's similarity measure* is a weighted average of the distances computed for each variable, after scaling each variable to a [0,1] scale. It is defined as

$$s_{ij} = \frac{\sum_{m=1}^{p} w_{ijm} s_{ijm}}{\sum_{m=1}^{p} w_{ijm}},$$

where s_{ijm} is the similarity between records *i* and *j* on measurement *m*, and w_{ijm} is a binary weight given to the corresponding distance.

The similarity measures s_{ijm} and weights w_{ijm} are computed as follows:

1. For continuous measurements, $s_{ijm} = 1 - \frac{|x_{im} - x_{jm}|}{\max(x_m) - \min(x_m)}$ and $w_{ijm} = 1$ unless the value for measurement m is unknown for one or both of the records, in which case $w_{ijm} = 0$.
2. For binary measurements, $s_{ijm} = 1$ if $x_{im} = x_{jm} = 1$ and 0 otherwise. $w_{ijm} = 1$ unless $x_{im} = x_{jm} = 0$.
3. For nonbinary categorical measurements, $s_{ijm} = 1$ if both records are in the same category, and otherwise $s_{ijm} = 0$. As in continuous measurements, $w_{ijm} = 1$ unless the category for measurement m is unknown for one or both of the records, in which case $w_{ijm} = 0$.

Non-hierarchical Clustering: The *k*-Means Algorithm

A non-hierarchical approach to forming good clusters is to pre-specify a desired number of clusters, *k*, and assign each case to one of the *k* clusters so as to minimize a measure of dispersion within the clusters. In other words, the goal is to divide the sample into a predetermined number *k* of non-overlapping clusters so that clusters are as homogeneous as possible with respect to the measurements used.

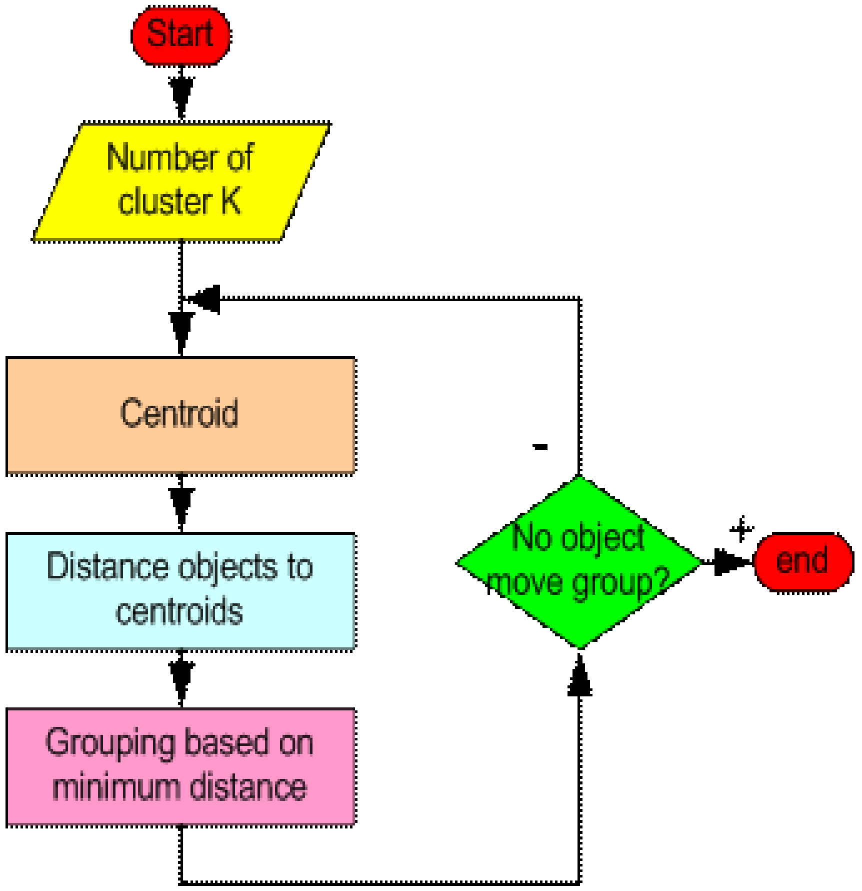
A common measure of within-cluster dispersion is the sum of distances (or sum of squared Euclidean distances) of records from their cluster centroid.

The *k*-means algorithm starts with an initial partition of the records into *k* clusters. Subsequent steps modify the partition to reduce the sum of the distances of each record from its cluster centroid. The modification consists of allocating each record to the nearest of the *k* centroids of the previous partition. This leads to a new partition for which the sum of distances is smaller than before. The means of the new clusters are computed and the improvement step is repeated until the improvement is very small.

k-MEANS CLUSTERING ALGORITHM:

1. Start with k initial clusters (user chooses k).
2. At every step, each record is reassigned to the cluster with the "closest" centroid.
3. Recompute the centroids of clusters that lost or gained a record, and repeat Step 2.
4. Stop when moving any more records between clusters increases cluster dispersion.

How it works?


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Algorithm

- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:
    1. Take the first k training sample as single-element clusters
    2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

Algorithm

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

Partitional - K-means clustering and K-Mode Clustering

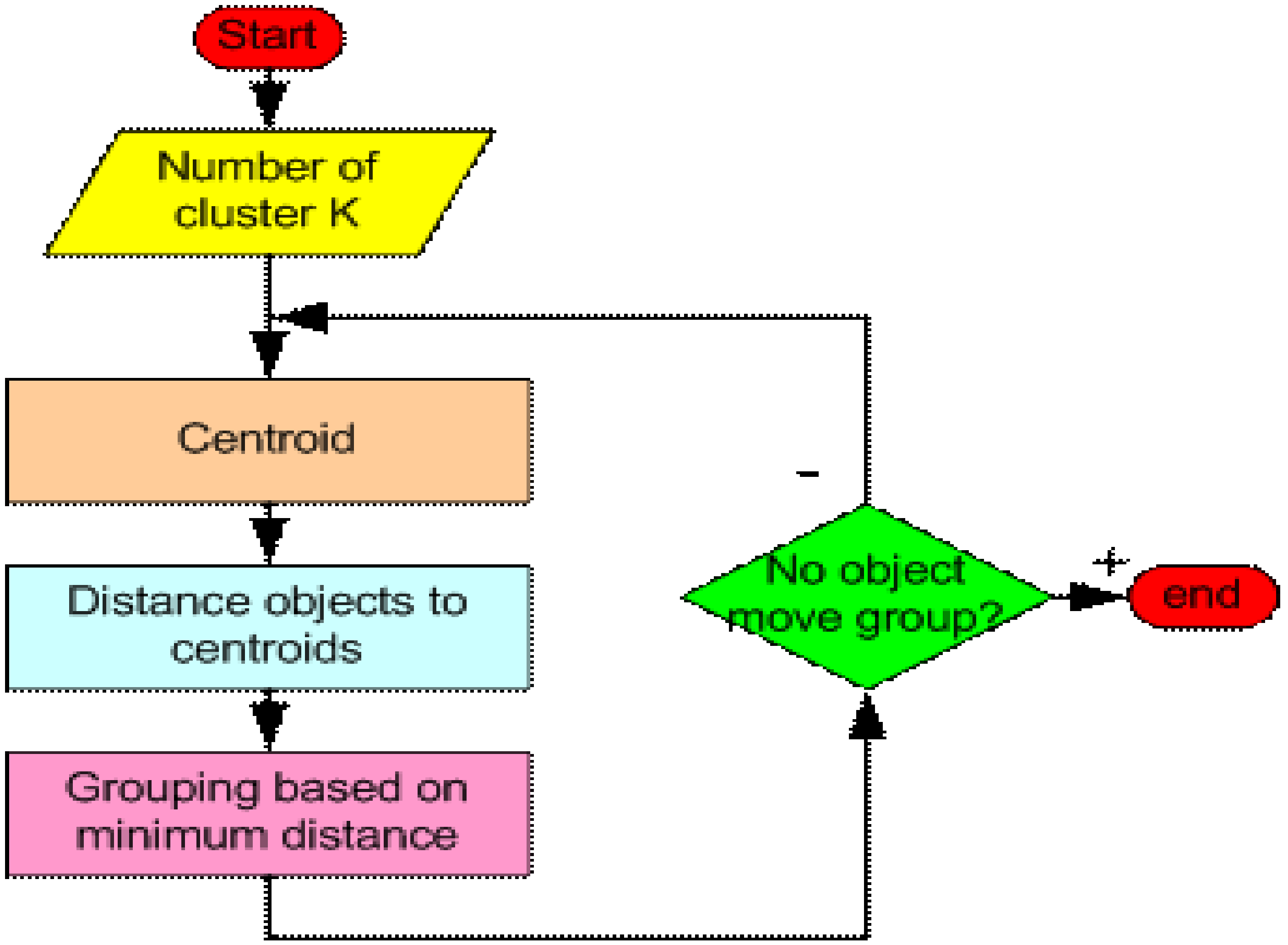
Partitioning approach (k-means)

- Each cluster is represented by the center of the cluster.
- Compute seed points as the centroids of the clusters of the current partition.
- The centroid is the center, i.e., *mean point*, of the cluster.

Comments on the *K-Means* Method

- <u>Strengths</u>
    - *Relatively efficient*: *O*(*tkn*), where *n* is number of objects, *k* is number of clusters, and *t* is number of iterations. Normally, *k*, *t* << *n*.
    - Often terminates at a *local optimum*.
- <u>Weaknesses</u>
    - Applicable only when *mean* is defined
    - Need to specify *k,* the *number* of clusters, in advance
    - Trouble with noisy data and *outliers*

How the K-Mean Clustering algorithm works?


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Contd…

- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:

  1.Take the first k training sample as single - element clusters

  2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid.

     After each assignment, recompute the centroid of

     the gaining cluster.

Contd…

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

Problem

- Consider eight data points with two dimensions X and Y as candidates. Find three clusters using the k means algorithm.

	X	Y
P1	1	1
P2	6	7
P3	4	6
P4	5	7
P5	5	2
P6	2	3
P7	1	2
P8	3	1

Step 1

- No. of clusters = 3 (i.e.) K = 3

Step 2

- Find the centroid

  Initial Seeds:

    Seed1(P1)=(1,1)

    Seed2(P4)=(5,7)

    Seed3(P7)=(1,2)

Step 3

- Find the distance between objects (points) and centroids using Manhattan distance

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

Data points	Distance with seed1 (P1) (1,1)	Distance with seed2 (P4) (5,7)	Distance with seed3 (P7) (1,2)
P1(1,1)	0	7.211	1
P2(6,7)	7.8	1	7.07
P3(4,6)	5.831	1.414	5
P4(5,7)	7.211	0	6.4
P5(5,2)	4.1231	5	4
P6(2,3)	2.23	5	1.414
P7(1,2)	1	6.4	0
P8(3,1)	2	6.32	2.23

Step 4

- Grouping based on minimum distance

Data points	Distance with seed1 (P1) (1,1)	Distance with seed2 (P4) (5,7)	Distance with seed3 (P7) (1,2)	Cluster No
P1(1,1)	**0**	7.211	1	**I**
P2(6,7)	7.8	**1**	7.07	**II**
P3(4,6)	5.831	**1.414**	5	**II**
P4(5,7)	7.211	**0**	6.4	**II**
P5(5,2)	4.1231	5	**4**	**III**
P6(2,3)	2.23	5	**1.414**	**III**
P7(1,2)	1	6.4	**0**	**III**
P8(3,1)	**2**	6.32	2.23	**I**

- **New Clusters members are:**

Cluster 1 = {P1, P8}
Cluster2 = {P2, P3, P4}
Cluster3 = {P5,P6,P7}

- **Seed Calculation:**

Seed1 = (1/2)[(1+3),(1+1)] =(2,1)
Seed2= (1/3)[(6+4+5),(7+6+7)] = (5,6.66)
Seed3 = (1/3)[(5+2+1),(2+3+2)] =(2.66,2.33)

Repeat Step 4 - 1

	Distance with seed1 (2,1)	Distance with seed2 (5,6.66)	Distance with seed3 (2.66,2.33)	Cluster No
P1(1,1)	1	6.9308	2.127	I
P2(6,7)	7.211	1.0562	5.741	II
P3(4,6)	5.385	1.1982	3.907	II
P4(5,7)	6.7	0.34	5.22	II
P5(5,2)	3.1623	4.66	2.3632	III
P6(2,3)	2	4.73	0.94	III
P7(1,2)	1.414	6.1413	1.6925	I
P8(3,1)	1	6	1.328	I

- **New clusters members are:**

Cluster1={P1,P7,P8}

Cluster2={P2,P3,P4}

Cluster3={P5,P6}

- **Seed Calculation:**

Seed1 = (1/3)[(1+1+1),(1+2+1)] =(1.67,1.34)

Seed2= (1/3)[(6+4+5),(7+6+7)] = (5,6.66)

Seed3 = (1/2)[(5+2),(2+3)] =(3.5,2.5)

- Compare previous clusters members with current clusters members. Both members are same means stop the process otherwise repeat the step 4.
- Previous clusters members

Cluster 1 = {P1, P8}

Cluster2 = {P2, P3, P4}

Cluster3 = {P5,P6,P7}

- Current or New clusters members are:

Cluster1={P1,P7,P8}

Cluster2={P2,P3,P4}

Cluster3={P5,P6}

Repeat Step 4 - 2

Data points	Distance with seed1 (1.67,1.34)	Distance with seed2 (5,6.66)	Distance with seed1 (3.5,2.5)	Cluster No
P1(1,1)	0.577	6.9308	2.915	I
P2(6,7)	7.13	1.0562	5.15	II
P3(4,6)	5.21	1.1982	3.53	II
P4(5,7)	6.57	0.34	4.743	II
P5(5,2)	3.4	4.66	1.581	III
P6(2,3)	1.7	4.73	1.58	III
P7(1,2)	0.94	6.1413	2.55	I
P8(3,1)	1.372	6	1.58	I

- Compare previous clusters members with current clusters members.
- Previous clusters members

Cluster1={P1,P7,P8}

Cluster2={P2,P3,P4}

Cluster3={P5,P6}

- Current or New clusters members are:

Cluster1={P1,P7,P8}

Cluster2={P2,P3,P4}

Cluster3={P5,P6}

- Both members are same. Stop the process.