

COMP8400: Algorithms and Techniques for Data Mining

Introduction to cluster analysis

What is cluster analysis?

- A cluster is a collection of data objects
 - Similar to one another in the same cluster
 - Disimilar to the objects in other cluster
- Cluster analysis (or *clustering*) is finding similarities between data objects according to the characteristics in the data and grouping similar data objects into clusters
- Cluster analysis is unsupervised, *descriptive* data mining
 - No predefined classes
- Typical applications
 - As a stand-alone tool to get insight into data distribution
 - As a pre-processing step (data cleaning and data reduction) for other data mining algorithms

Lecture outline

- What is cluster analysis?
- Applications and examples
- What is good clustering?
- Clustering requirements in data mining
- Measurements of cluster quality
- Similarities between data objects
- Main clustering approaches
- Partitioning algorithms
- *K-means* clustering approach

Applications of cluster analysis

- Pattern recognition
- Spatial data analysis
 - Create thematic maps in geographical information systems (GIS) by clustering feature spaces
 - Detect spatial clusters for use in other spatial data mining tasks
- Image processing
- Economic science (especially market research)
 - Groupings of similar customers
- Internet / WWW
 - Document / Web page categorisation
 - Cluster Web log data to discover groups of similar access patterns

Examples of cluster analysis

- *Marketing*: Help marketers to discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- *Land use*: identification of areas of similar land use in an earth observation database (satellite images, etc.)
- *Insurance*: Identify groups of (for example, motor insurance) policy holders with a high average claim cost
- *City planning*: Identifying groups of houses according to their house type, value, and geographical location

Clustering requirements in data mining

- Scalability
- Ability to deal with different attribute types
- Ability to handle dynamic data
- Discovery of clusters of arbitrary shapes
- Minimal domain knowledge required to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- Handle high dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

What is good clustering?

- A good clustering will produce clusters with
 - High intra-class similarity
 - Low inter-class similarity
- The quality of a clustering result depends upon both the similarity measure and the algorithm used for searching
 - Different algorithms deliver different clusterings
- The quality of a clustering is also measured by its ability to discover some or all of the hidden patterns in the data
- Clustering may not be the best way to discover interesting groups in data sets
 - Visualisation often works well, allowing human experts to identify useful groups
 - This becomes problematic with very large data sets

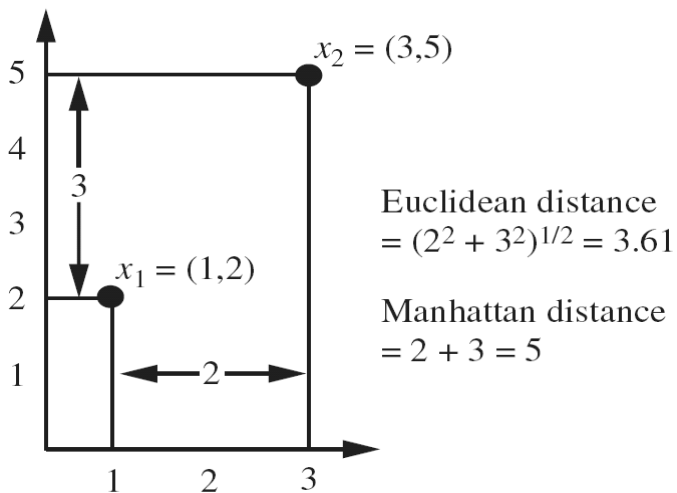
Measurements of cluster quality

- Dissimilarity/similarity metric: Similarity is expressed in terms of a distance function, typically a metric: $d(a, b)$
- There is a separate “quality” function that measures the “goodness” of a cluster
- The definitions of distance functions are usually very different for interval-scaled, boolean, categorical, ordinal, ratio-scaled, and vector variables
- Weights can be associated with different variables (attributes) based on applications and data semantics
- It is hard to define “similar enough” or “good enough”
 - The answer is typically highly subjective

Similarity and dissimilarity between objects

- Distances are normally used to measure similarity and dissimilarity between two data objects
 - Two objects: $a = (a_1, a_2, \dots, a_n)$ and $b = (b_1, b_2, \dots, b_n)$
- Properties of a *distance measure* $d(i, j)$:
 - $d(a, a) = 0$
 - $d(a, b) \geq 0$
 - $d(a, b) = d(b, a)$
 - $d(a, c) \leq d(a, b) + d(b, c)$ Triangular inequality

Euclidean and Manhattan distance example



Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Minkowski distance

- Popular distance measure includes *Minkowski* distance:

$$d(a, b) = \sqrt[q]{(|a_1 - b_1|^q + |a_2 - b_2|^q + \dots + |a_n - b_n|^q)}$$

where $a = (a_1, a_2, \dots, a_n)$ and $b = (b_1, b_2, \dots, b_n)$ are two n -dimensional data objects, and q is a positive integer

- If $q = 1$, d is the *Manhattan* distance:

$$d(a, b) = |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$

- If $q = 2$, d is the *Euclidean* distance:

$$d(a, b) = \sqrt{(|a_1 - b_1|^2 + |a_2 - b_2|^2 + \dots + |a_n - b_n|^2)}$$

Similarities for other data types

- Many different ways to measure similarities between objects
 - Binary data: contingency tables
 - Nominal variables (e.g. colors): Count number of matches divided by total number of possible matches
 - Strings: Exact or approximate string similarities (edit-distance, q-gram based, longest common sub-string, etc.)
 - Vector objects (document words, micro array gene features): cosine measure based on term-frequency/inverse document frequency (TF-IDF)
- A database might contain different types of attributes
- One might use a weighted sum to calculate final similarity between objects
 - For example, $d(a, b) = 0.3 d_{name}(name_a, name_b) + 0.7 d_{salary}(salary_a, salary_b)$

Calculate the distance between clusters

- *Single link*: Smallest distance between a data object in one cluster and a data object in the other: $d(K_i, K_j) = \min(t_{ip}, t_{jq})$
- *Complete link*: Largest distance between a data object in one cluster and a data object in the other: $d(K_i, K_j) = \max(t_{ip}, t_{jq})$
- *Average*: Average distance between a data object in one cluster and a data object in the other: $d(K_i, K_j) = \text{avg}(t_{ip}, t_{jq})$
(same as *Centroid* - Distance between the centroids of two clusters)
- *Medoid*: Distance between the medoids of two clusters:
 $d(K_i, K_j) = d(M_i, M_j)$
 - A *medoid* is a data object centrally located in the cluster

Major clustering approaches (1)

- **Partitioning approaches**
 - Construct various partitions and then evaluate them by some criterion, for example, minimising cluster radius or diameter, or the sum of square errors
 - A fixed number, k , of clusters is generated
 - Typical methods: *k-means*, *k-medoids*, *CLARANS*
- **Hierarchical approaches**
 - Create a hierarchical decomposition of the data objects using some criterion
 - Typical methods: *Diana*, *Agnes*, *BIRCH*, *ROCK*, *CAMELEON*
- **Density based approaches**
 - Based on connectivity and density functions
 - Typical methods: *DBSCAN*, *OPTICS*, *DenClue*

Centroid, radius and diameter of a cluster

- For numerical data objects t_{ip} in cluster i
$$C_i = \frac{\sum_{p=1}^N \sum (t_{ip})}{N}$$
- Centroid: the “middle” of a cluster
- Radius: square root of average distance from any data object of the cluster to its centroid
$$R_i = \sqrt{\frac{\sum \sum_{p=1}^N (t_{ip} - c_i)^2}{N}}$$
- Diameter: square root of average mean squared distance between all pairs of data objects in the cluster
$$D_i = \sqrt{\frac{\sum_{p=1}^N \sum_{q=1}^N (t_{ip} - t_{iq})^2}{N(N-1)}}$$

Major clustering approaches (2)

- **Grid-based approaches**
 - Based on a multi-level granularity structure
 - Typical methods: *STING*, *WaveCluster*, *CLIQUE*
- **Model-based approaches**
 - A model is hypothesised for each of the clusters and the idea is to find the best fit of that model
 - Typical methods: *EM (Expectation-Maximisation)*, *SOM (Self-organising maps)*, *COBWEB*
- **Frequent-pattern based approaches**
 - Based on analysis of frequent patterns
 - Typical method: *pCluster*
- **User-guided or constrain-based approaches**
 - Clustering by considering user- or application-specific constraints
 - Typical method: *COD (obstacles)*, *constrained clustering*

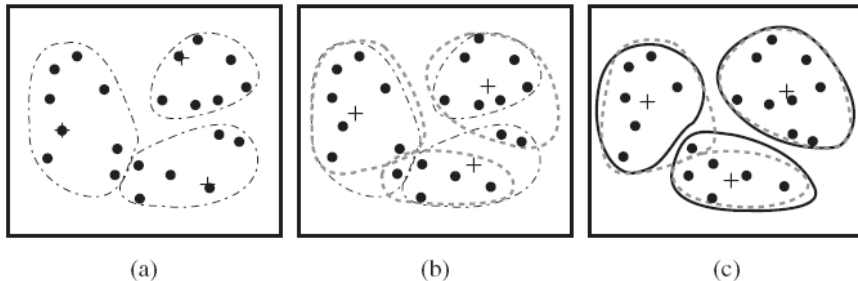
Partitioning algorithms: Basic concept

- Construct a partition of a database D of n objects into a set of k clusters, such that minimum sum of squared distance

$$\sum_{i=1}^k \sum_{t_{ip} \in K_i} (C_i - t_{ip})^2$$

- Given a k , find a partition of k clusters that optimises the chosen partitioning criterion
 - Global optimal: exhaustively enumerate all partitions
 - Heuristic methods: k -means and k -medoids algorithms
 - k -means (MacQueen'67): Each cluster is represented by the center of the cluster
 - k -medoids or PAM (Partition Around Medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The k -means clustering algorithm



+ = centroids

Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

The k -means clustering algorithm

Algorithm: k -means. The k -means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

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 from D as the initial cluster centers;
- (2) repeat
- (3) (re)assign each object to the cluster to which the object is the most similar based on the mean value of the objects in the cluster;
- (4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
- (5) until no change;

Source: Han and Kamber, DM Book, 2nd Ed. (Copyright © 2006 Elsevier Inc.)

Comments on the k -means algorithm

- **Strength:** Relatively efficient: $O(tkn)$, where n is the number of data objects, k is the number of clusters, and t is the number of iterations. Normally, k and $t \ll n$
 - In comparison: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$ (s = sample size)
- **Comment:** Often terminates at a local optimum!
 - The global optimum may be found using techniques such as deterministic annealing and genetic algorithms
 - Basic idea: running k -means many times with different starting configurations
- **Weaknesses**
 - Applicable only when *mean* is defined, then what about categorical data?
 - Need to specify k , the number of clusters, in advance
 - Unable to handle noisy data and outliers
 - Not suitable to discover clusters with non-convex shapes