Unsupervised Learning and Clustering

Muhammad Sarim

Contents

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised* .

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised* .
- In this chapter, we will study *clustering* as an *unsupervised* procedure that uses unlabeled samples.

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised* .
- In this chapter, we will study *clustering* as an *unsupervised* procedure that uses unlabeled samples.
- Unsupervised procedures are used for several reasons:

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised* .
- In this chapter, we will study *clustering* as an *unsupervised* procedure that uses unlabeled samples.
- Unsupervised procedures are used for several reasons:
  - Collecting and labeling a large set of sample patterns can be costly.

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised*.
- In this chapter, we will study *clustering* as an *unsupervised* procedure that uses unlabeled samples.
- Unsupervised procedures are used for several reasons:
  - Collecting and labeling a large set of sample patterns can be costly.
  - One can train with large amount of unlabeled data, and then use supervision to label the groupings found.

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised* .
- In this chapter, we will study *clustering* as an *unsupervised* procedure that uses unlabeled samples.
- Unsupervised procedures are used for several reasons:
  - Collecting and labeling a large set of sample patterns can be costly.
  - One can train with large amount of unlabeled data, and then use supervision to label the groupings found.
  - Unsupervised methods can be used for feature extraction.

Introduction

- Until now we have assumed that the training examples were labeled by their class membership.
- Procedures that use labeled samples are said to be *supervised* .
- In this chapter, we will study *clustering* as an *unsupervised* procedure that uses unlabeled samples.
- Unsupervised procedures are used for several reasons:
    - Collecting and labeling a large set of sample patterns can be costly.
    - One can train with large amount of unlabeled data, and then use supervision to label the groupings found.
    - Unsupervised methods can be used for feature extraction.
    - Exploratory data analysis can provide insight into the nature or structure of the data.

Contents

Data Description

- Assume that we have a set of unlabeled multi-dimensional patterns.

Data Description

- Assume that we have a set of unlabeled multi-dimensional patterns.
- One way of describing this set of patterns is to compute their sample mean and covariance.

Data Description

- Assume that we have a set of unlabeled multi-dimensional patterns.
- One way of describing this set of patterns is to compute their sample mean and covariance.
- This description uses the assumption that the patterns form a cloud that can be modeled with a hyperellipsoidal shape.

Data Description

- Assume that we have a set of unlabeled multi-dimensional patterns.
- One way of describing this set of patterns is to compute their sample mean and covariance.
- This description uses the assumption that the patterns form a cloud that can be modeled with a hyperellipsoidal shape.
- However, we must be careful about any assumptions we make about the structure of the data.

Data Description

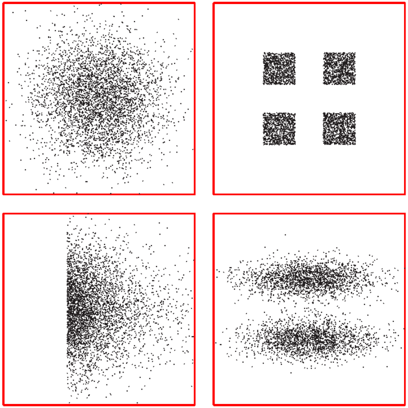


Figure: These four data sets have identical first-order and second-order statistics. We need to find other ways of modeling the structure. Clustering is an alternative way of describing the data in terms of groups of patterns.

Contents

Clusters

- A *cluster* is comprised of a number of similar objects collected or grouped together.

Clusters

- A *cluster* is comprised of a number of similar objects collected or grouped together.
- Other definitions of clusters include (from Jain and Dubes, 1988):

Clusters

- A *cluster* is comprised of a number of similar objects collected or grouped together.
- Other definitions of clusters include (from Jain and Dubes, 1988):
    - A cluster is a set of entities which are alike, and entities from different clusters are not alike.

Clusters

- A *cluster* is comprised of a number of similar objects collected or grouped together.
- Other definitions of clusters include (from Jain and Dubes, 1988):
    - A cluster is a set of entities which are alike, and entities from different clusters are not alike.
    - A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.

Clusters

- A *cluster* is comprised of a number of similar objects collected or grouped together.
- Other definitions of clusters include (from Jain and Dubes, 1988):
    - A cluster is a set of entities which are alike, and entities from different clusters are not alike.
    - A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.
    - Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other such regions by a region containing a relatively low density of points.

Contents

Clustering

- *Cluster analysis* organizes data by abstracting the underlying structure either as a grouping of individuals or as a hierarchy of groups.

Clustering

- *Cluster analysis* organizes data by abstracting the underlying structure either as a grouping of individuals or as a hierarchy of groups.
- These groupings are based on measured or perceived similarities among the patterns.

Clustering

- *Cluster analysis* organizes data by abstracting the underlying structure either as a grouping of individuals or as a hierarchy of groups.
- These groupings are based on measured or perceived similarities among the patterns.
- Clustering is unsupervised. Category labels and other information about the source of data influence the interpretation of the clusters, not their formation.

Clustering

- Clustering is a very difficult problem because data can reveal clusters with different shapes and sizes.

Clustering

- Clustering is a very difficult problem because data can reveal clusters with different shapes and sizes.

Figure: The number of clusters in the data often depend on the resolution (fine vs. coarse) with which we view the data. How many clusters do you see in this figure? 5, 8, 10, more?

Contents

Clustering

- Clustering algorithms can be divided into several groups:

Clustering

- Clustering algorithms can be divided into several groups:
  - *Exclusive* (each pattern belongs to only one cluster) vs. *nonexclusive* (each pattern can be assigned to several clusters)

Clustering

- Clustering algorithms can be divided into several groups:
  - *Exclusive* (each pattern belongs to only one cluster) vs. *nonexclusive* (each pattern can be assigned to several clusters)
  - *Hierarchical* (nested sequence of partitions) vs. *partitional* (a single partition)

Clustering

- Clustering algorithms can be divided into several groups:
  - *Exclusive* (each pattern belongs to only one cluster) vs. *nonexclusive* (each pattern can be assigned to several clusters)
  - *Hierarchical* (nested sequence of partitions) vs. *partitional* (a single partition)
- Implementations of clustering algorithms can also be grouped:

Clustering

- Clustering algorithms can be divided into several groups:
  - *Exclusive* (each pattern belongs to only one cluster) vs. *nonexclusive* (each pattern can be assigned to several clusters)
  - *Hierarchical* (nested sequence of partitions) vs. *partitional* (a single partition)
- Implementations of clustering algorithms can also be grouped:
  - *Agglomerative* (merging atomic clusters into larger clusters) vs. *divisive* (subdividing large clusters into smaller ones)

Clustering

- Clustering algorithms can be divided into several groups:
    - *Exclusive* (each pattern belongs to only one cluster) vs. *nonexclusive* (each pattern can be assigned to several clusters)
    - *Hierarchical* (nested sequence of partitions) vs. *partitional* (a single partition)
- Implementations of clustering algorithms can also be grouped:
    - *Agglomerative* (merging atomic clusters into larger clusters) vs. *divisive* (subdividing large clusters into smaller ones)
    - *Serial* (processing patterns one by one) vs. *simultaneous* (processing all patterns at once)

Clustering

- Clustering algorithms can be divided into several groups:
  - *Exclusive* (each pattern belongs to only one cluster) vs. *nonexclusive* (each pattern can be assigned to several clusters)
  - *Hierarchical* (nested sequence of partitions) vs. *partitional* (a single partition)
- Implementations of clustering algorithms can also be grouped:
  - *Agglomerative* (merging atomic clusters into larger clusters) vs. *divisive* (subdividing large clusters into smaller ones)
  - *Serial* (processing patterns one by one) vs. *simultaneous* (processing all patterns at once)
  - *Graph-theoretic* (based on connectedness) vs. *algebraic* (based on error criteria)

Clustering

- Hundreds of clustering algorithms have been proposed in the literature.

Clustering

- Hundreds of clustering algorithms have been proposed in the literature.
- Most of these algorithms are based on the following two popular techniques:

Clustering

- Hundreds of clustering algorithms have been proposed in the literature.
- Most of these algorithms are based on the following two popular techniques:
  - Iterative squared-error partitioning

Clustering

- Hundreds of clustering algorithms have been proposed in the literature.
- Most of these algorithms are based on the following two popular techniques:
  - Iterative squared-error partitioning
  - Agglomerative hierarchical clustering

Clustering

- Hundreds of clustering algorithms have been proposed in the literature.
- Most of these algorithms are based on the following two popular techniques:
  - Iterative squared-error partitioning
  - Agglomerative hierarchical clustering
- One of the main challenges is to select an appropriate measure of similarity to define clusters that is often both data (cluster shape) and context dependent.

Contents

Similarity Measures

- The most obvious measure of *similarity* (or dissimilarity) between two patterns is the distance between them.

Similarity Measures

- The most obvious measure of *similarity* (or dissimilarity) between two patterns is the distance between them.
- If distance is a good measure of dissimilarity, then we can expect the distance between patterns in the same cluster to be significantly less than the distance between patterns in different clusters.

Similarity Measures

- The most obvious measure of *similarity* (or dissimilarity) between two patterns is the distance between them.
- If distance is a good measure of dissimilarity, then we can expect the distance between patterns in the same cluster to be significantly less than the distance between patterns in different clusters.
- Then, a very simple way of doing clustering would be to choose a threshold on distance and group the patterns that are closer than this threshold.

Similarity Measures

- The most obvious measure of *similarity* (or dissimilarity) between two patterns is the distance between them.
- If distance is a good measure of dissimilarity, then we can expect the distance between patterns in the same cluster to be significantly less than the distance between patterns in different clusters.
- Then, a very simple way of doing clustering would be to choose a threshold on distance and group the patterns that are closer than this threshold.

Similarity Measures

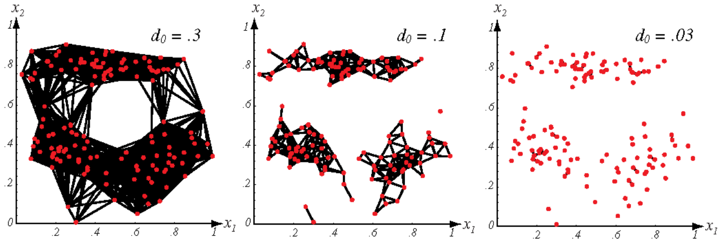


Figure: The distance threshold affects the number and size of clusters that are shown by lines drawn between points closer than the threshold.

Contents

Criterion Functions

- The next challenge after selecting the similarity measure is the choice of the criterion function to be optimized.

Criterion Functions

- The next challenge after selecting the similarity measure is the choice of the criterion function to be optimized.
- Suppose that we have a set $D = \{x_1, \ldots, x_n\}$ of n samples that we want to partition into exactly k disjoint subsets $D_1, \ldots, D_k$.

Criterion Functions

- The next challenge after selecting the similarity measure is the choice of the criterion function to be optimized.
- Suppose that we have a set $D = \{x_1, \ldots, x_n\}$ of n samples that we want to partition into exactly k disjoint subsets $D_1, \ldots, D_k$.
- Each subset is to represent a cluster, with samples in the same cluster being somehow more similar to each other than they are to samples in other clusters.

Criterion Functions

- The next challenge after selecting the similarity measure is the choice of the criterion function to be optimized.
- Suppose that we have a set $D = \{x_1, \ldots, x_n\}$ of n samples that we want to partition into exactly k disjoint subsets $D_1, \ldots, D_k$.
- Each subset is to represent a cluster, with samples in the same cluster being somehow more similar to each other than they are to samples in other clusters.
- The simplest and most widely used criterion function for clustering is the *sum-of-squared-error* criterion.

Contents

Squared-error Partitioning

- Suppose that the given set of n patterns has somehow been partitioned into k clusters $D_1, \ldots, D_k$.

Squared-error Partitioning

- Suppose that the given set of n patterns has somehow been partitioned into k clusters $D_1, \ldots, D_k$.
- Let n_i be the number of samples in D_i and let m_i be the mean of those samples

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

Squared-error Partitioning

- Suppose that the given set of n patterns has somehow been partitioned into k clusters $D_1, \ldots, D_k$.
- Let n_i be the number of samples in D_i and let m_i be the mean of those samples

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

- Then, the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - m_i\|^2$$

Squared-error Partitioning

- Suppose that the given set of n patterns has somehow been partitioned into k clusters $D_1, \ldots, D_k$.
- Let n_i be the number of samples in D_i and let m_i be the mean of those samples

$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

- Then, the sum-of-squared errors is defined by

$$J_e = \sum_{i=1}^{k} \sum_{x \in D_i} \|x - m_i\|^2$$

- For a given cluster D_i, the mean vector m_i (centroid) is the best representative of the samples in D_i.

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
    1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
  1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
  2. Generate a new partition by assigning each pattern to its closest cluster center.

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
  1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
  2. Generate a new partition by assigning each pattern to its closest cluster center.
  3. Compute new cluster centers as the centroids of the clusters.

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
  1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
  2. Generate a new partition by assigning each pattern to its closest cluster center.
  3. Compute new cluster centers as the centroids of the clusters.
  4. Repeat steps 2 and 3 until an optimum value of the criterion function is found (e.g., when a local minimum is found or a predefined number of iterations are completed).

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
  1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
  2. Generate a new partition by assigning each pattern to its closest cluster center.
  3. Compute new cluster centers as the centroids of the clusters.
  4. Repeat steps 2 and 3 until an optimum value of the criterion function is found (e.g., when a local minimum is found or a predefined number of iterations are completed).
  5. Adjust the number of clusters by merging and splitting existing clusters or by removing small or outlier clusters.

Squared-error Partitioning

- A general algorithm for iterative squared-error partitioning:
    1. Select an initial partition with k clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
    2. Generate a new partition by assigning each pattern to its closest cluster center.
    3. Compute new cluster centers as the centroids of the clusters.
    4. Repeat steps 2 and 3 until an optimum value of the criterion function is found (e.g., when a local minimum is found or a predefined number of iterations are completed).
    5. Adjust the number of clusters by merging and splitting existing clusters or by removing small or outlier clusters.
- This algorithm, without step 5, is also known as the *k-means* algorithm.

Squared-error Partitioning

- k-means is computationally efficient and gives good results if the clusters are compact, hyperspherical in shape and well-separated in the feature space.

Squared-error Partitioning

- k-means is computationally efficient and gives good results if the clusters are compact, hyperspherical in shape and well-separated in the feature space.
- However, choosing k and choosing the initial partition are the main drawbacks of this algorithm.

Squared-error Partitioning

- k-means is computationally efficient and gives good results if the clusters are compact, hyperspherical in shape and well-separated in the feature space.
- However, choosing k and choosing the initial partition are the main drawbacks of this algorithm.
- The value of k is often chosen empirically or by prior knowledge about the data.

Squared-error Partitioning

- k-means is computationally efficient and gives good results if the clusters are compact, hyperspherical in shape and well-separated in the feature space.
- However, choosing k and choosing the initial partition are the main drawbacks of this algorithm.
- The value of k is often chosen empirically or by prior knowledge about the data.
- The initial partition is often chosen by generating k random points uniformly distributed within the range of the data, or by randomly selecting k points from the data.

Squared-error Partitioning

- Numerous attempts have been made to improve the performance of the basic k-means algorithm:

Squared-error Partitioning

- Numerous attempts have been made to improve the performance of the basic k-means algorithm:
  - incorporating a fuzzy criterion resulting in fuzzy k-means,

Squared-error Partitioning

- Numerous attempts have been made to improve the performance of the basic k-means algorithm:
  - incorporating a fuzzy criterion resulting in fuzzy k-means,
  - using genetic algorithms, simulated annealing, deterministic annealing to optimize the resulting partition,

Squared-error Partitioning

- Numerous attempts have been made to improve the performance of the basic k-means algorithm:
  - incorporating a fuzzy criterion resulting in fuzzy k-means,
  - using genetic algorithms, simulated annealing, deterministic annealing to optimize the resulting partition,
  - using iterative splitting to find the initial partition.

Squared-error Partitioning

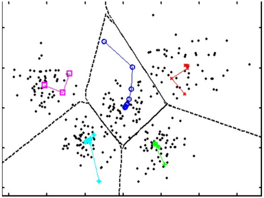
- Numerous attempts have been made to improve the performance of the basic k-means algorithm:
  - incorporating a fuzzy criterion resulting in fuzzy k-means,
  - using genetic algorithms, simulated annealing, deterministic annealing to optimize the resulting partition,
  - using iterative splitting to find the initial partition.
- Another alternative is to use model-based clustering using Gaussian mixtures to allow more flexible shapes for individual clusters (k-means with Euclidean distance assumes spherical shapes).

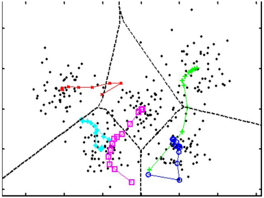
Squared-error Partitioning

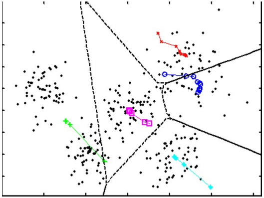
- Numerous attempts have been made to improve the performance of the basic k-means algorithm:
  - incorporating a fuzzy criterion resulting in fuzzy k-means,
  - using genetic algorithms, simulated annealing, deterministic annealing to optimize the resulting partition,
  - using iterative splitting to find the initial partition.
- Another alternative is to use model-based clustering using Gaussian mixtures to allow more flexible shapes for individual clusters (k-means with Euclidean distance assumes spherical shapes).
- In model-based clustering, the value of k corresponds to the number of components in the mixture.

Contents

Examples


Good initialization


Good initialization


Bad initialization

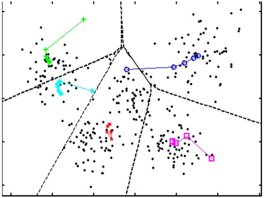

Bad initialization

Figure: Examples for k-means with different initializations of five clusters for the same data.

Contents

Hierarchical Clustering

- The k-means algorithm produces a *flat* data description where the clusters are disjoint and are at the same level.

Hierarchical Clustering

- The k-means algorithm produces a *flat* data description where the clusters are disjoint and are at the same level.
- In some applications, groups of patterns share some characteristics when looked at a particular level.

Hierarchical Clustering

- The k-means algorithm produces a *flat* data description where the clusters are disjoint and are at the same level.
- In some applications, groups of patterns share some characteristics when looked at a particular level.
- Hierarchical clustering tries to capture these multi-level groupings using *hierarchical* representations rather than flat partitions.

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,
  - the first level consists of n clusters (each cluster containing exactly one sample),

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,
  - the first level consists of n clusters (each cluster containing exactly one sample),
  - the second level contains $n-1$ clusters,

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,
  - the first level consists of n clusters (each cluster containing exactly one sample),
  - the second level contains $n - 1$ clusters,
  - the third level contains $n - 2$ clusters,

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,
  - the first level consists of n clusters (each cluster containing exactly one sample),
  - the second level contains $n - 1$ clusters,
  - the third level contains $n - 2$ clusters,
  - and so on until the last (n'th) level at which all samples form a single cluster.

Hierarchical Clustering

- In hierarchical clustering, for a set of n samples,
  - the first level consists of n clusters (each cluster containing exactly one sample),
  - the second level contains $n - 1$ clusters,
  - the third level contains $n - 2$ clusters,
  - and so on until the last (n'th) level at which all samples form a single cluster.
- Given any two samples, at some level they will be grouped together in the same cluster and remain together at all higher levels.

Contents

Hierarchical Clustering

- The most natural representation of hierarchical clustering is a tree, also called a *dendrogram*, which shows how the samples are grouped.

Hierarchical Clustering

- The most natural representation of hierarchical clustering is a tree, also called a *dendrogram*, which shows how the samples are grouped.
- If there is an unusually large gap between the similarity values for two particular levels, one can argue that the level with fewer number of clusters represents a more natural grouping.

Hierarchical Clustering

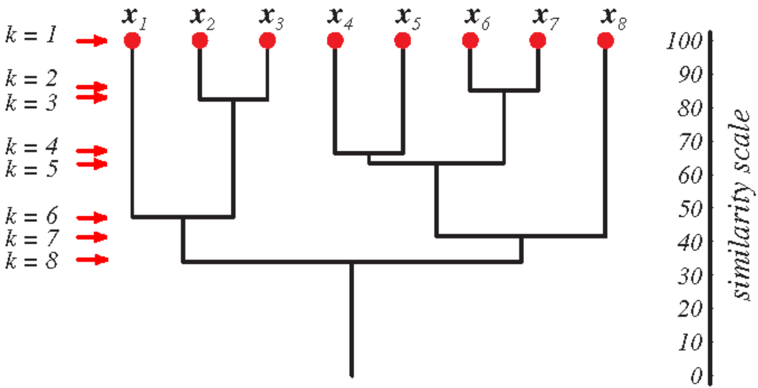


Figure: A dendrogram can represent the results of hierarchical clustering algorithms. The vertical axis shows a generalized measure of similarity among clusters.

Contents

Hierarchical Clustering

- *Agglomerative Hierarchical Clustering:*

Hierarchical Clustering

- *Agglomerative Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.

Hierarchical Clustering

- *Agglomerative Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the closest clusters according to a distance measure.

Hierarchical Clustering

- *Agglomerative Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the closest clusters according to a distance measure.
  3. Merge these two clusters.

Hierarchical Clustering

- *Agglomerative Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the closest clusters according to a distance measure.
  3. Merge these two clusters.
  4. Return the resulting clusters.

Hierarchical Clustering

- Popular distance measures (for two clusters D_i and D_j):

$$d_{\text{min}}(D_i, D_j) = \min_{\substack{x \in D_i \\ x' \in D_j}} \|x - x'\|$$

$$d_{\text{max}}(D_i, D_j) = \max_{\substack{x \in D_i \\ x' \in D_j}} \|x - x'\|$$

$$d_{\text{avg}}(D_i, D_j) = \frac{1}{\#D_i \, \#D_j} \sum_{x \in D_i} \sum_{x' \in D_j} \|x - x'\|$$

$$d_{\text{mean}}(D_i, D_j) = \|m_i - m_j\|$$

Hierarchical Clustering

- When d_{min} is used to measure the distance between clusters, the algorithm is called the nearest neighbor clustering algorithm.

Hierarchical Clustering

- When d_{min} is used to measure the distance between clusters, the algorithm is called the nearest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *single linkage algorithm* where

Hierarchical Clustering

- When d_{min} is used to measure the distance between clusters, the algorithm is called the nearest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *single linkage algorithm* where
    - patterns represent the nodes of a graph,

Hierarchical Clustering

- When d_{min} is used to measure the distance between clusters, the algorithm is called the nearest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *single linkage algorithm* where
  - patterns represent the nodes of a graph,
  - edges connect patterns belonging to the same cluster,

Hierarchical Clustering

- When d_{min} is used to measure the distance between clusters, the algorithm is called the nearest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *single linkage algorithm* where
  - patterns represent the nodes of a graph,
  - edges connect patterns belonging to the same cluster,
  - merging two clusters corresponds to adding an edge between the nearest pair of nodes in these clusters.

Hierarchical Clustering

- When $d_{\max}$ is used to measure the distance between clusters, the algorithm is called the farthest neighbor clustering algorithm.

Hierarchical Clustering

- When d_{max} is used to measure the distance between clusters, the algorithm is called the farthest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *complete linkage algorithm* where

Hierarchical Clustering

- When d_{max} is used to measure the distance between clusters, the algorithm is called the farthest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *complete linkage algorithm* where
  - patterns represent the nodes of a graph,

Hierarchical Clustering

- When d_{max} is used to measure the distance between clusters, the algorithm is called the farthest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *complete linkage algorithm* where
  - patterns represent the nodes of a graph,
  - edges connect all patterns belonging to the same cluster,

Hierarchical Clustering

- When d_{max} is used to measure the distance between clusters, the algorithm is called the farthest neighbor clustering algorithm.
- Moreover, if the algorithm is terminated when the distance between nearest clusters exceeds a threshold, it is called the *complete linkage algorithm* where
  - patterns represent the nodes of a graph,
  - edges connect all patterns belonging to the same cluster,
  - merging two clusters corresponds to adding edges between every pair of nodes in these clusters.

Hierarchical Clustering

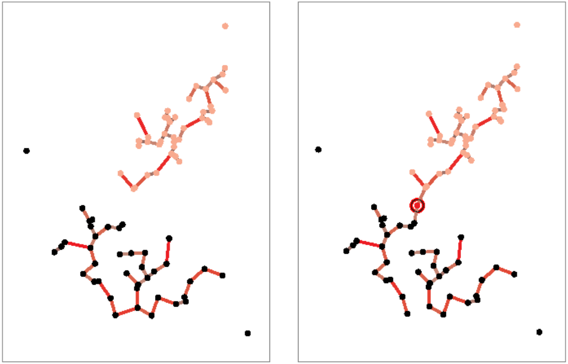


Figure: Examples for single linkage clustering.

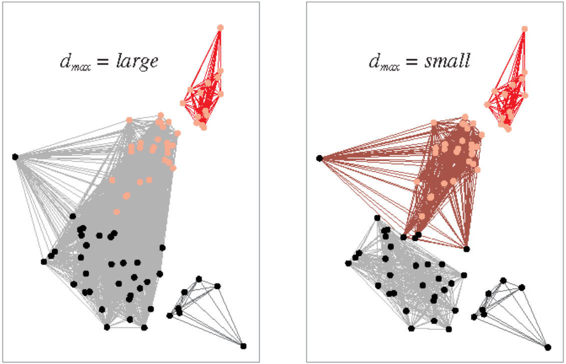


Figure: Examples for complete linkage clustering.

Contents

Hierarchical Clustering

- *Stepwise-Optimal Hierarchical Clustering:*

Hierarchical Clustering

- *Stepwise-Optimal Hierarchical Clustering:*
    1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.

Hierarchical Clustering

- *Stepwise-Optimal Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the clusters whose merger increases an error criterion the least.

Hierarchical Clustering

- *Stepwise-Optimal Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the clusters whose merger increases an error criterion the least.
  3. Merge these two clusters.

Hierarchical Clustering

- *Stepwise-Optimal Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the clusters whose merger increases an error criterion the least.
  3. Merge these two clusters.
  4. Return the resulting clusters.

Hierarchical Clustering

- *Stepwise-Optimal Hierarchical Clustering:*
  1. Specify the number of clusters. Place every pattern in a unique cluster and repeat steps 2 and 3 until a partition with the required number of clusters is obtained.
  2. Find the clusters whose merger increases an error criterion the least.
  3. Merge these two clusters.
  4. Return the resulting clusters.
- When the sum-of-squared-error criterion J_e is used, the pair of clusters whose merger increases J_e as little as possible is the pair for which the distance

$$d_e(D_i, D_j) = \frac{\#D_i \, \#D_j}{\#D_i + \#D_j} \|m_i - m_j\|^2$$

is minimum.

Graph-Theoretic Clustering

- **Graph**: (S, R)
  - S: Set of nodes
  - R: Set of edges, $R \subseteq S \times S$

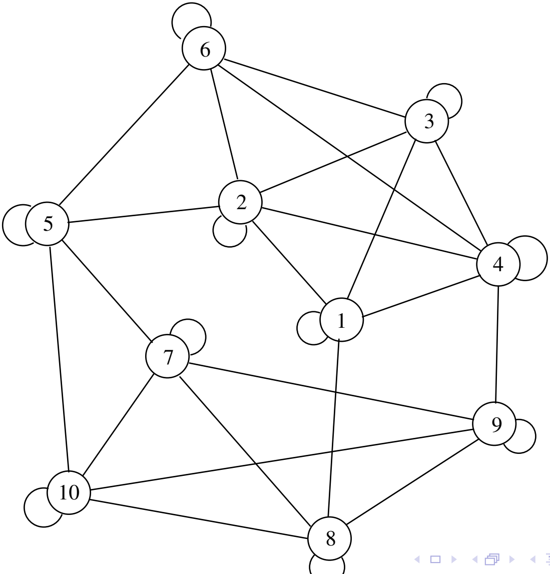
Graph-Theoretic Clustering

- **Graph**: (S, R)
  - S: Set of nodes
  - R: Set of edges, $R \subseteq S \times S$
- **Clique**: Set of nodes that are all connected to each other, $\{P \subseteq S | P \times P \subseteq R\}$.

Graph-Theoretic Clustering

- **Graph**: (S, R)
  - S: Set of nodes
  - R: Set of edges, $R \subseteq S \times S$
- **Clique**: Set of nodes that are all connected to each other, $\{P \subseteq S | P \times P \subseteq R\}$.
- **Goal**: Find clusters in a graph that are not as dense as cliques but are compact enough within user specified thresholds.

Graph-Theoretic Clustering



Graph-Theoretic Clustering

- $(X, Y) \in R$ means Y is a **neighbor** of X,

$$\text{Neighborhood}(X) = \{Y \mid (X, Y) \in R\}.$$

Graph-Theoretic Clustering

- $(X, Y) \in R$ means Y is a **neighbor** of X,

$$\text{Neighborhood}(X) = \{Y \mid (X, Y) \in R\}.$$

- **Conditional density** $D(Y|X)$ is the number of nodes in the neighborhood of X which have Y as a neighbor,

$$\begin{aligned} D(Y|X) &= \#\{N \in S \mid (N, Y) \in R \text{ and } (X, N) \in R\} \\ &= D(X|Y) \\ &= \#\{\text{Neighborhood}(X) \cap \text{Neighborhood}(Y)\}. \end{aligned}$$

Graph-Theoretic Clustering

- Given an integer K, a **dense region** Z around a node $X \in S$ is defined as

$$Z(X, K) = \{Y \in S \mid D(Y|X) \geq K\}.$$

Graph-Theoretic Clustering

- Given an integer K, a **dense region** Z around a node $X \in S$ is defined as

$$Z(X, K) = \{Y \in S \mid D(Y|X) \geq K\}.$$

- $Z(X) = Z(X, J)$ is a **dense region candidate** around X where

$$J = \max\{K \mid \#Z(X, K) \geq K\}$$

  because if M is a major clique of size L, then $X, Y \in M$ implies that $D(Y|X) \geq L$. Thus $M \subseteq Z(X, L)$ and $K \leq L \leq \#Z(X, K)$.

Graph-Theoretic Clustering

- **Association** of a node X to a subset B of S is

$$A(X|B) = \frac{\#\{\text{Neighborhood}(X) \cap B\}}{\#B}$$

where $0 \leq A(X|B) \leq 1$.

Graph-Theoretic Clustering

- **Association** of a node X to a subset B of S is

$$A(X|B) = \frac{\#\{\text{Neighborhood}(X) \cap B\}}{\#B}$$

where $0 \leq A(X|B) \leq 1$.

- **Compactness** of a subset B of S is

$$C(B) = \frac{1}{\#B} \sum_{X \in B} A(X|B)$$

where $0 \leq C(B) \leq 1$.

Graph-Theoretic Clustering

- A **dense region** B of the graph (S, R) should satisfy
  1. $B = \{N \in Z(X) \mid A(N|Z(X)) \geq \tau_a\}$ for some $X \in S$,
  2. $C(B) \geq \tau_c$,
  3. $\#B \geq \tau_s$

  where τ_a, τ_c and τ_s are thresholds supplied by the user for minimum association, minimum compactness, and minimum size, respectively.

Graph-Theoretic Clustering

- Algorithm for finding a dense region around a node X:

Graph-Theoretic Clustering

- Algorithm for finding a dense region around a node X:
  1. Compute $D(Y|X)$ for every other node Y in S.

Graph-Theoretic Clustering

- Algorithm for finding a dense region around a node X:
  1. Compute $D(Y|X)$ for every other node Y in S.
  2. Find a dense region candidate $Z(X, K')$ where

$$K' = \max\{K \mid \#\{Y|D(Y|X) \geq K\} \geq K\}.$$

Graph-Theoretic Clustering

- Algorithm for finding a dense region around a node X:
  1. Compute $D(Y|X)$ for every other node Y in S.
  2. Find a dense region candidate $Z(X, K')$ where

$$K' = \max\{K \mid \#\{Y|D(Y|X) \geq K\} \geq K\}.$$

  3. Remove the nodes with a low association from the candidate set. Iterate until all of the nodes have high enough association.

Graph-Theoretic Clustering

- Algorithm for finding a dense region around a node X:
  1. Compute $D(Y|X)$ for every other node Y in S.
  2. Find a dense region candidate $Z(X, K')$ where

$$K' = \max\{K \mid \#\{Y|D(Y|X) \geq K\} \geq K\}.$$

  3. Remove the nodes with a low association from the candidate set. Iterate until all of the nodes have high enough association.
  4. Check whether the remaining nodes have high enough average association.

Graph-Theoretic Clustering

- Algorithm for finding a dense region around a node X:
  1. Compute $D(Y|X)$ for every other node Y in S.
  2. Find a dense region candidate $Z(X, K')$ where

$$K' = \max\{K \mid \#\{Y|D(Y|X) \geq K\} \geq K\}.$$

  3. Remove the nodes with a low association from the candidate set. Iterate until all of the nodes have high enough association.
  4. Check whether the remaining nodes have high enough average association.
  5. Check whether the candidate set is large enough.

Graph-Theoretic Clustering

- Given the dense regions, the algorithm for graph-theoretic clustering proceeds as follows:

Graph-Theoretic Clustering

- Given the dense regions, the algorithm for graph-theoretic clustering proceeds as follows:
  1. Define the **dense-region relation** F as

$$F = \{(B_1, B_2) \mid B_1, B_2 \text{ are dense regions of } R, \\ \frac{\#B_1 \cap B_2}{\#B_1} \geq \tau_o \text{ or } \frac{\#B_1 \cap B_2}{\#B_2} \geq \tau_o\}$$

where τ_o is a threshold supplied by the user for minimum overlap.

Graph-Theoretic Clustering

- Given the dense regions, the algorithm for graph-theoretic clustering proceeds as follows:
  1. Define the **dense-region relation** F as
  $$F = \{(B_1, B_2) \mid B_1, B_2 \text{ are dense regions of } R, \frac{\#B_1 \cap B_2}{\#B_1} \geq \tau_o \text{ or } \frac{\#B_1 \cap B_2}{\#B_2} \geq \tau_o\}$$
  where τ_o is a threshold supplied by the user for minimum overlap.
  2. Merge the regions that have enough overlap if all of the nodes in the set resulting after merging have high enough associations.

Graph-Theoretic Clustering

- Given the dense regions, the algorithm for graph-theoretic clustering proceeds as follows:
  1. Define the **dense-region relation** F as

$$F = \{(B_1, B_2) \mid B_1, B_2 \text{ are dense regions of } R, \\ \frac{\#B_1 \cap B_2}{\#B_1} \geq \tau_o \text{ or } \frac{\#B_1 \cap B_2}{\#B_2} \geq \tau_o\}$$

     where τ_o is a threshold supplied by the user for minimum overlap.
  2. Merge the regions that have enough overlap if all of the nodes in the set resulting after merging have high enough associations.
  3. Iterate until no regions can be merged.

Graph-Theoretic Clustering

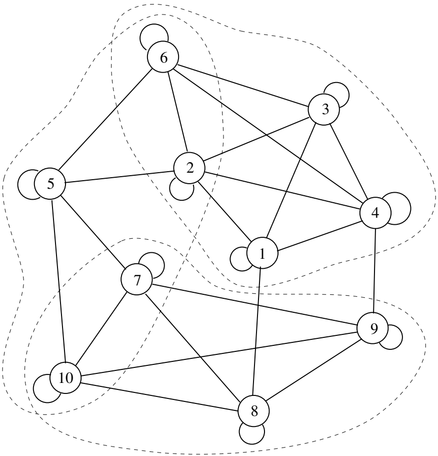


Figure: Clusters found in the example graph using the thresholds $\tau_a = 0.5, \tau_c = 0.6, \tau_s = 3, \tau_o = 0.9$: $\{1, 2, 3, 4, 6\}$ (compactness=0.92), $\{7, 8, 9, 10\}$ (compactness=1.00), $\{2, 5, 6, 7, 10\}$ (compactness=0.68).

Graph-Theoretic Clustering

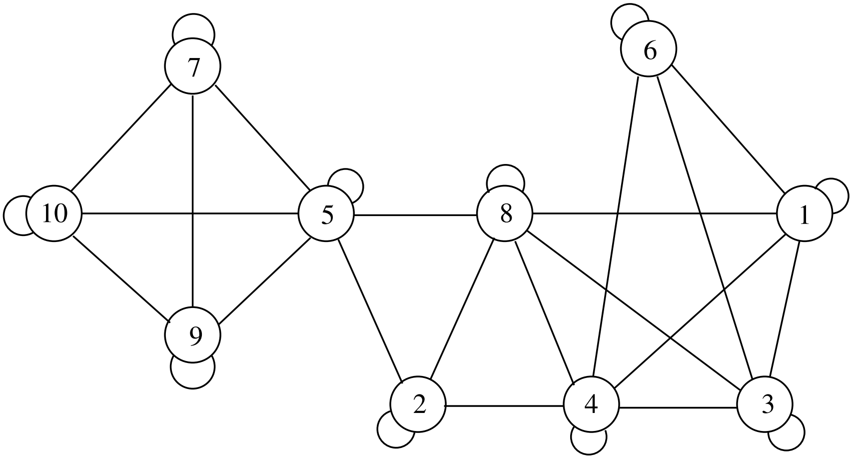


Figure: Another example graph.

Graph-Theoretic Clustering

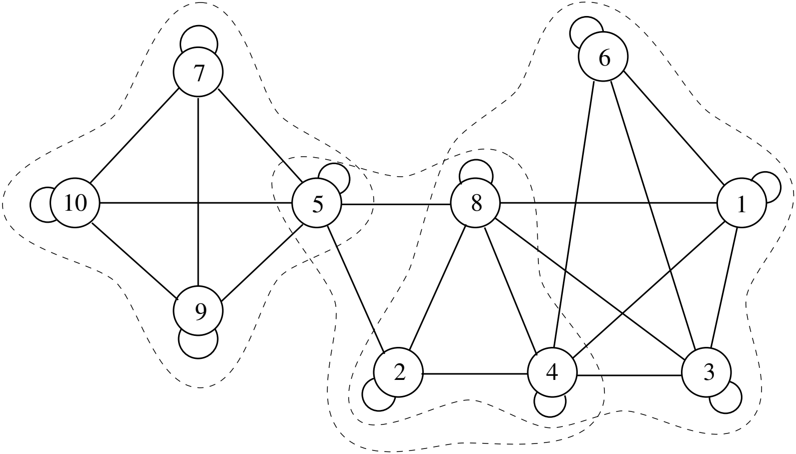


Figure: Clusters found in the second example graph using the thresholds $\tau_a = 0.5, \tau_c = 0.8, \tau_s = 3, \tau_o = 0.75$: $\{1, 2, 3, 4, 6, 8\}$ (compactness=0.78), $\{2, 4, 5, 8\}$ (compactness=0.88), $\{5, 7, 9, 10\}$

Cluster Validity

- The procedures we have considered so far either assume that the number of clusters is known or use some thresholds to decide how the clusters are formed.

Cluster Validity

- The procedures we have considered so far either assume that the number of clusters is known or use some thresholds to decide how the clusters are formed.
- These may be reasonable assumptions for some applications but are usually unjustified if we are exploring a data set whose properties and structure are unknown.

Cluster Validity

- The procedures we have considered so far either assume that the number of clusters is known or use some thresholds to decide how the clusters are formed.
- These may be reasonable assumptions for some applications but are usually unjustified if we are exploring a data set whose properties and structure are unknown.
- Furthermore, most of the iterative algorithms that we use may find a local extremum and may not give the best result.

Cluster Validity

- Methods for validating the results of a clustering algorithm include:

Cluster Validity

- Methods for validating the results of a clustering algorithm include:
    - Repeating the clustering procedure for different values of the parameters, and examining the resulting values of the criterion function for large jumps or stable ranges.

Cluster Validity

- Methods for validating the results of a clustering algorithm include:
  - Repeating the clustering procedure for different values of the parameters, and examining the resulting values of the criterion function for large jumps or stable ranges.
  - Evaluating the goodness-of-fit using measures such as the chi-squared or Kolmogorov-Smirnov statistics.

Cluster Validity

- Methods for validating the results of a clustering algorithm include:
  - Repeating the clustering procedure for different values of the parameters, and examining the resulting values of the criterion function for large jumps or stable ranges.
  - Evaluating the goodness-of-fit using measures such as the chi-squared or Kolmogorov-Smirnov statistics.
  - Formulating hypothesis tests that check whether multiple clusters found have been formed by chance, and whether the observed change in the error criterion has any significance.