Lecture 6:

Data clustering (I)

Outline

- Clustering
  - Main concepts
  - Clustering validation measures
- Partitional algorithms
  - kMeans
  - fuzzy cMeans
- Hierarchical algorithms
  - Agglomerative
  - Divisive

Aim of clustering (reminder)

What is known?

- A set of data (not necessarily structured)
- A similarity/dissimilarity measure between data (the measure is specific to the problem) based on which is constructed the similarity/dissimilarity matrix

What is desired?

- A model describing the grouping of data in clusters such that data belonging to the same cluster are more similar than data belonging to different clusters

Which is the final aim?

- Check if two data belong to the same cluster
- Find the most appropriate cluster for a new data

Remark: for some clustering methods it is enough to know the matrix of (dis)similarity values

Aim of clustering (reminder)

Examples:

- Customer segmentation = identify groups of customers with similar shopping behaviors
- Data summarization / document clustering = identify groups of electronic documents based on their content
- User profiles extraction = identify groups of users of an e-commerce system or a web service characterized by similar behaviors
- Image segmentation = identify homogeneous regions in an image

Related to

- Communities detection = identify tightly clustered nodes in a network

Clustering allows to:

- Summarize and/or visualize in a different form the data in order to understand them better

Particularities of clustering

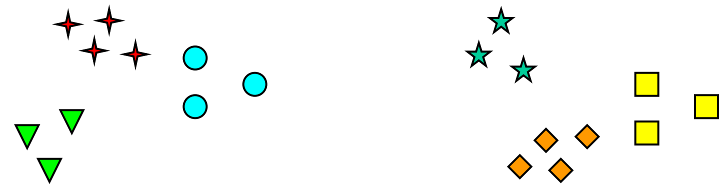
It is an unsupervised process:

- The training set contains only the values of the attributes
- The class labels are not known before clustering

The clustering task is ill-defined:

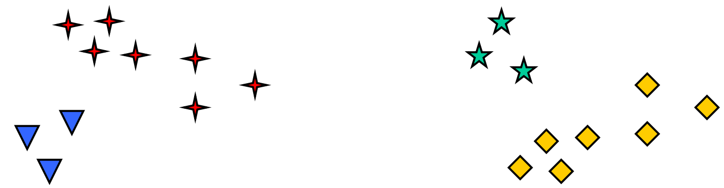
- identifying the clusters is not easy
- It can be a subjective decision

How many clusters?



Six Clusters

Two Clusters



Four Clusters

[images from slides by Kumar, 2004]

Main concepts

- Cluster = group of data which are “similar enough”
- (Di)similarity matrix for a set of n data instances = matrix of n rows and n columns with the (di)similarity between any two data instances
- Clustering

  = set of clusters

  = process of identifying the clusters
- Cluster prototype = “object” which is representative for the data in the cluster
  - Centroid = the mean of the data in the cluster – the centroid is not necessary a data from the cluster
  - Medoid = the data instance from the cluster which is closest to the mean of the cluster – the medoid is one of the data in the cluster
- Cluster radius = average of the distances between the data in the cluster and the cluster prototype
- Cluster diameter = maximum of the distance between two data in the cluster

Types of clustering

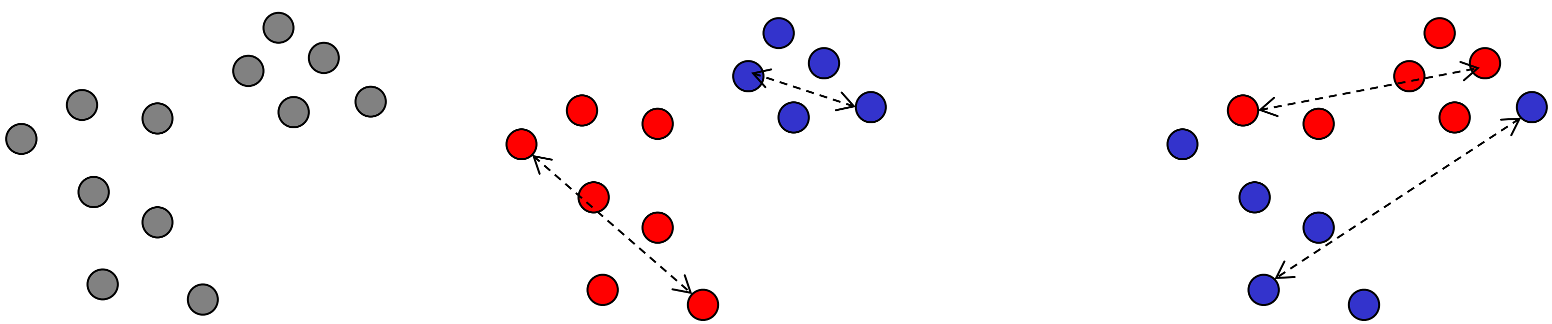
- Crisp vs fuzzy clustering
  - Crisp clustering = each data instance belongs to only one cluster
  - Fuzzy clustering = a data can belong to several clusters and for each cluster there is a membership degree

- Flat vs hierarchical clustering
  - Flat (partitional) clustering = the result is one partition (set of clusters)
  - Hierarchical clustering = the result is a hierarchy of partitions

- Variants of algorithms
  - Partitional algorithms (e.g: kMeans, Fuzzy cMeans)
  - Hierarchical algorithms (e.g. agglomerative algorithm, divisive algorithm)
  - Density based algorithms (e.g. DBSCAN)
  - Probabilistic algorithms (e.g. EM = Expectation Maximization)

Clustering validation measures

- There is no unique indicator which measures the quality of a clustering result
- The most straightforward approach is to estimate:
  - The compactness of data in one cluster (intra-cluster variability – it should be small)
  - The degree of separation between data belonging to different clusters (inter-cluster distance – it should be large)



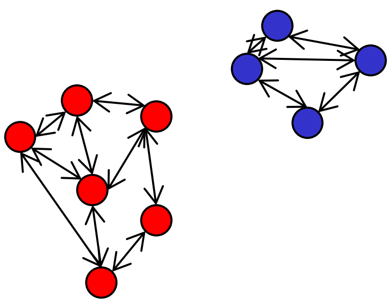
An acceptable clustering　　A lower quality clustering

Clustering validation measures

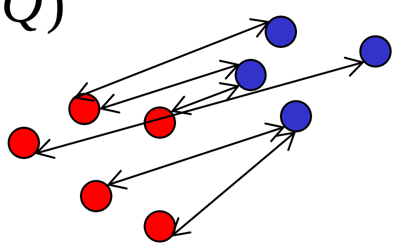
- Intra-cluster to inter-cluster distance ratio = Intra/Inter (smaller values correspond to better clustering)
- Let P be the set of pairs of data instances which belong to the same cluster and Q=DxD-P (the rest of pairs: one data belongs to one cluster and the other data belongs to another cluster)

$$Intra = \sum_{(x_i, x_j) \in P} d(x_i, x_j) / card(P)$$

$$Inter = \sum_{(x_i, x_j) \in Q} d(x_i, x_j) / card(Q)$$



Examples of paired distances involved in the computation of the intra measure



Examples of paired distances involved in the computation of the inter measure

Clustering validation measures

- Silhouette coefficient (it measures the difference between the similarity of an object and its own cluster (cohesion) and it similarity to other clusters (separation))

$$S_i = \frac{D\min_i^{out} - Davg_i^{in}}{\max\{D\min_i^{out}, Davg_i^{in}\}}$$

$$S = \frac{1}{n}\sum_{i=1}^{n} S_i$$

$$Davg_i^{in} = \text{the average of the distances between } x_i \text{ and all other data in the cluster of } x_i$$

$$Davg_i^{j} = \text{the average of the distances between } x_i \text{ and all data in another cluster } j \ (j \neq i)$$

$$D\min_i^{out} = \min_j Davg_i^{j}$$

Remark:

- S takes values in (-1,1)
- Larger values indicate better clustering

kMeans

- Input: data set $D=\{x_1,x_2,\ldots,x_N\}$, K = number of clusters
- Output: a partition $P=\{C_1,C_2,\ldots,C_K\}$ of D

kMeans (D,k)

initialize the centroids c_1, c_2, ..., c_K (by random selection from the data set or by using a pre-clustering method)

repeat

- assign each data from D to the cluster corresponding to the closest centroid (with respect to a similarity/distance)
- update each centroid as mean of the data belonging to the corresponding cluster

until <the partition does not change>

Remark: this is the so-called Lloyd variant

kMeans

Characteristics

- kMeans is a center based clustering method which aims to minimize the total sum of squared errors (SSE) – distances between data and their corresponding centroids

$$SSE = \sum_{k=1}^{K} \sum_{x \in C_k} d^2(x, c_k) = \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{j=1}^{n} (x_j - c_{kj})^2$$

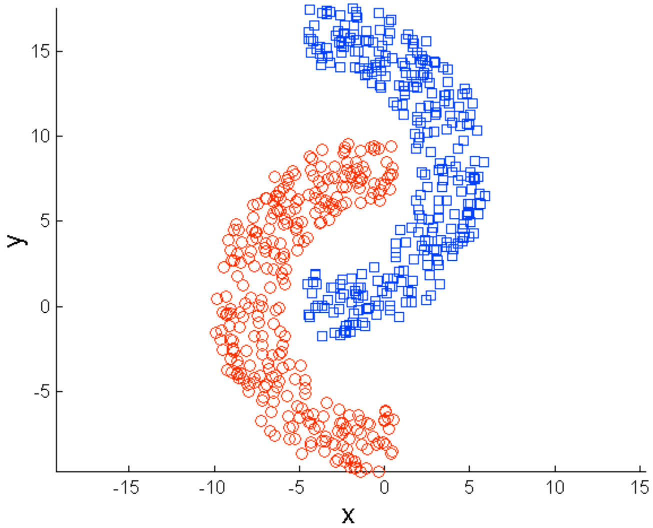
(in the case of Euclidean distance)

- Complexity: O(n*N*K*iterations) (n=number of attributes, N=number of data instances, K=number of clusters)
- Useful pre-processing: normalization
- Useful post-processing:
  - Remove the small clusters
  - Split the loose clusters
  - Merge the close clusters

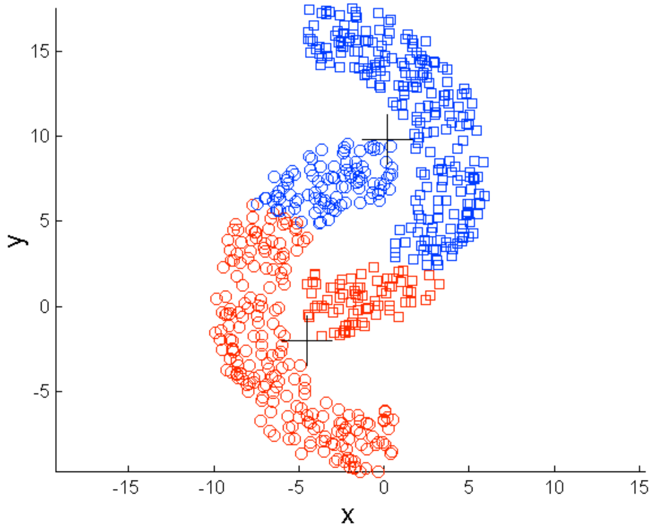
kMeans

Limits:

- It does not work well in the case when the clusters are not "spherical"
  - Solution: use other approaches (e.g. density based clustering)



True clusters



Kmeans result

[from slides Kumar, 2004]

kMeans

Limits: It requires the apriori knowledge of the number of clusters

- Solutions:
  - apply the algorithm for different values of K and select the variant with the best values of the validation criteria
  - Post-process the clustering results by splitting the clusters which are not compact enough and by merging clusters which are close one to each other (e.g. ISODATA algorithm)

ISODATA

Main ideas of ISODATA

- If a cluster size is smaller than Nmin then the cluster should merge with another cluster (the closest one)

- If the distance between two clusters (e.g. the distance between the clusters' prototypes) is smaller than Dmin then the clusters should be merged

- If the variance of a cluster is higher than Vmax and the number of data instances it contains is larger than 2*Nmin then the cluster should be divided in two other clusters:
  - Identify the attribute j for which the variance is maximal
  - From prototype c_k two other prototypes c' and c'' are constructed by replacing the value of attribute j from c_k with $c_k(j)-b$ and $c_k(j)+b$, respectively (b is a user parameter)

Fuzzy cMeans

Main idea of fuzzy (soft) clustering:

- A data instance does not belong only to one cluster but it can belong to several clusters (with a given membership degree for each cluster)

- The output of fuzzy clustering is a matrix M of size NxK

  (N= number of data instances, K= number of clusters);

  M(i,j) = a value in [0,1] which corresponds to the degree of membership of data i to cluster j

- To obtain a crisp clustering each data i is assigned to the cluster j characterized by the largest membership value M(i,j)

Fuzzy cMeans

Algorithm

- Initialize the membership matrix (M)
- **Repeat**
  - Compute the centroids(c_k, k=1,...K)
  - Update the membership values (m_{ij}, i=1,...,N, j=1,...,K

  until <no significant changes in the membership function>

Remark: at the end of the clustering process, the data are assigned to the cluster for which the membership value is maximal

Computation of centroids

$$c_j = \frac{\sum_{i=1}^{n} M_{ij}^{p} x_i}{\sum_{i=1}^{n} M_{ij}^{p}}, \quad j = \overline{1,K}$$

p>1 is a parameter (e.g. p=2)

Membership values computation

$$M_{ij} = \frac{1}{\left\| x_i - c_j \right\|^{2/(p-1)} \sum_{k=1}^{K} 1/\left\| x_i - c_k \right\|^{2/(p-1)}}$$

$$i = \overline{1,n},\ j = \overline{1,K}$$

Hierarchical algorithms

Remark: one of the main limits of partitional algorithms is the fact that the number of clusters should be known apriori.

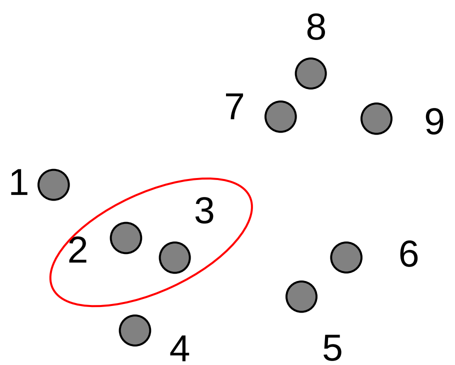
Another approach: construct a hierarchy of partitions

- In a bottom-up manner (agglomerative approach)
  - Start with a partition consisting of one-data clusters (each data belongs to its own cluster)
  - Merge the clusters which are "similar" enough, in an iterative way until all data belong to one cluster

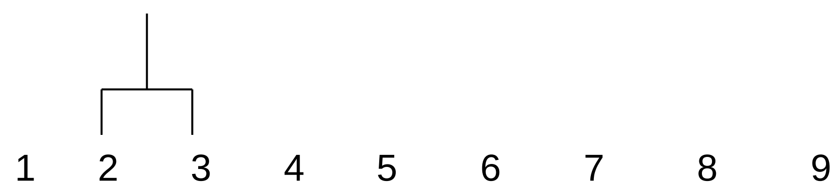
- In a top-down manner (divisive approach)
  - Start with a partition containing one cluster (which contains all data)
  - Divide the "large" clusters by applying a flat clustering (e.g. kMeans) iteratively until the partition consists of singletons (each cluster contains one data instance)

Agglomerative clustering

Idea: identify at each step the most similar clusters and merge them

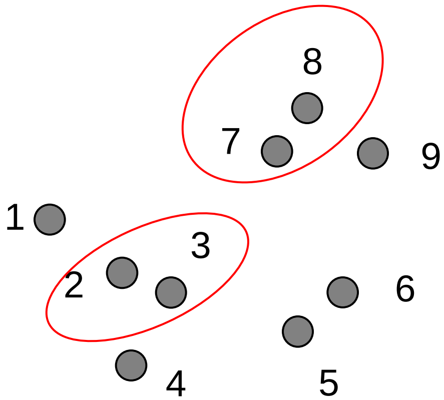


	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0

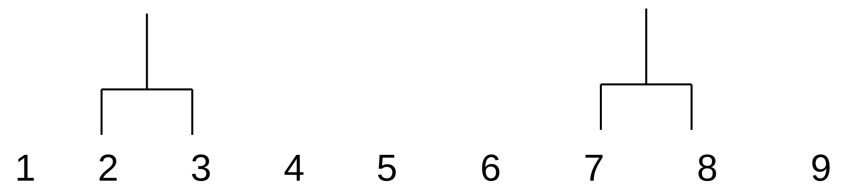


Agglomerative clustering

Idea: identify at each step the most similar clusters and merge them

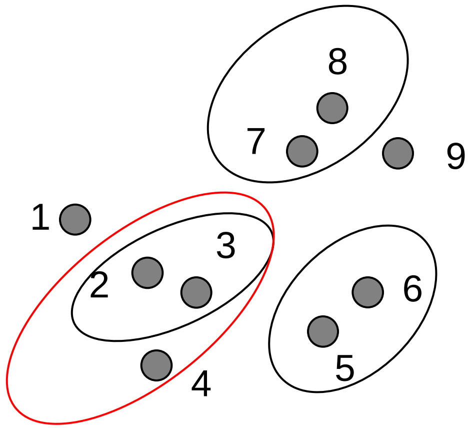


	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0

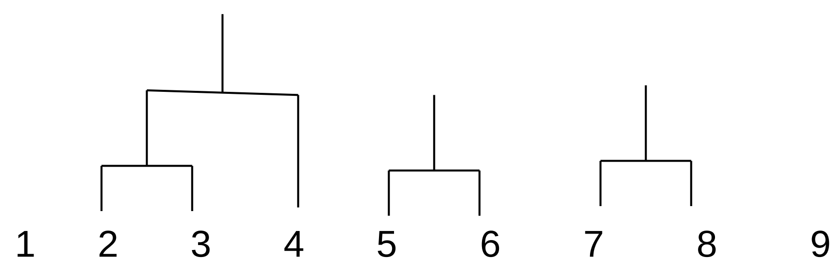


Agglomerative clustering

Idea: identify at each step the most similar clusters and merge them

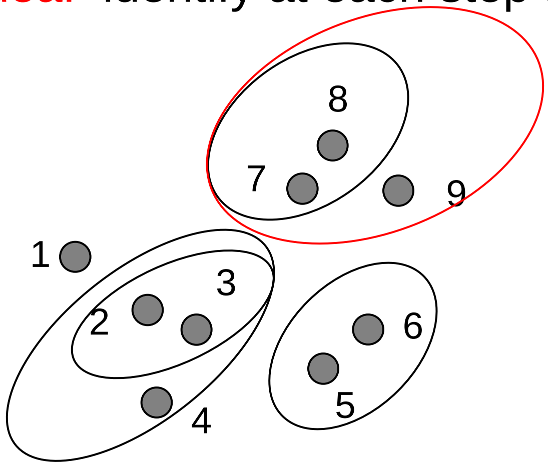


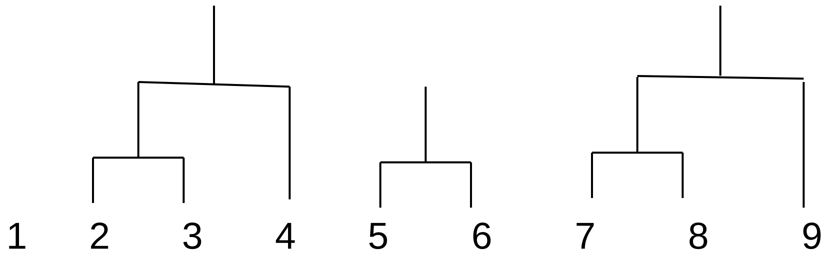
	1	2	3	4	5	6	7	8	9
1	0	2	3	4	7	8	6	8	10
2	2	0	1	2	4	6	7	8	9
3	3	1	0	2	3	5	6	8	9
4	4	2	2	0	3	6	9	10	11
5	7	4	3	3	0	1	4	6	5
6	8	6	5	6	1	0	3	4	3
7	6	7	6	9	4	3	0	1	2
8	8	8	8	10	6	4	1	0	2
9	10	9	9	11	5	3	3	2	0



Agglomerative clustering

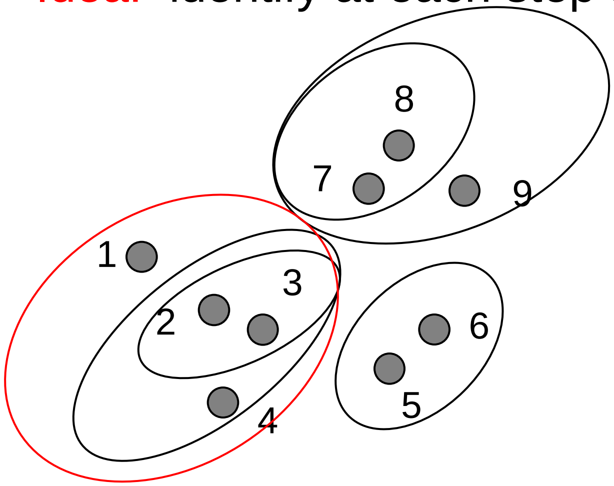
Idea: identify at each step the most similar clusters and merge them

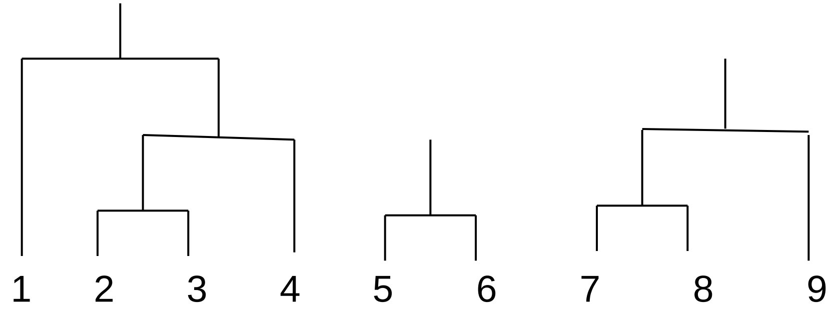




Agglomerative clustering

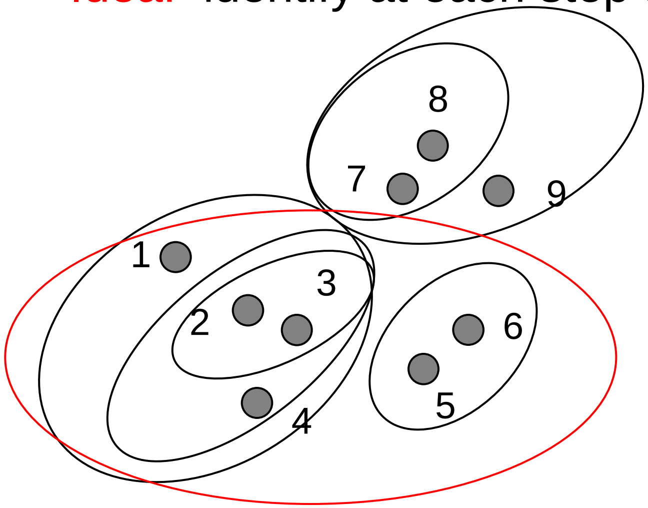
Idea: identify at each step the most similar clusters and merge them

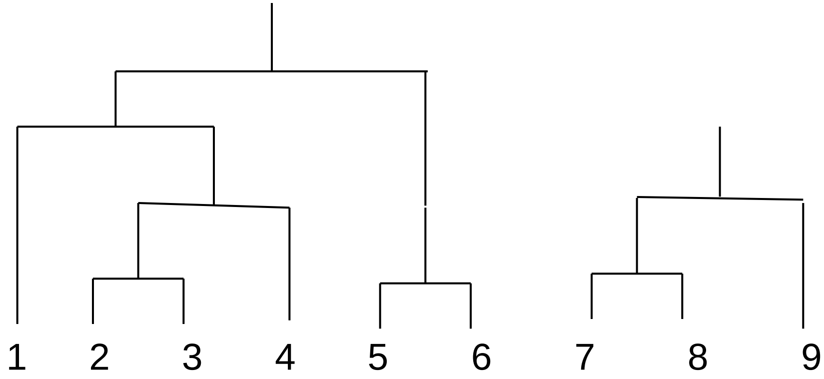




Agglomerative clustering

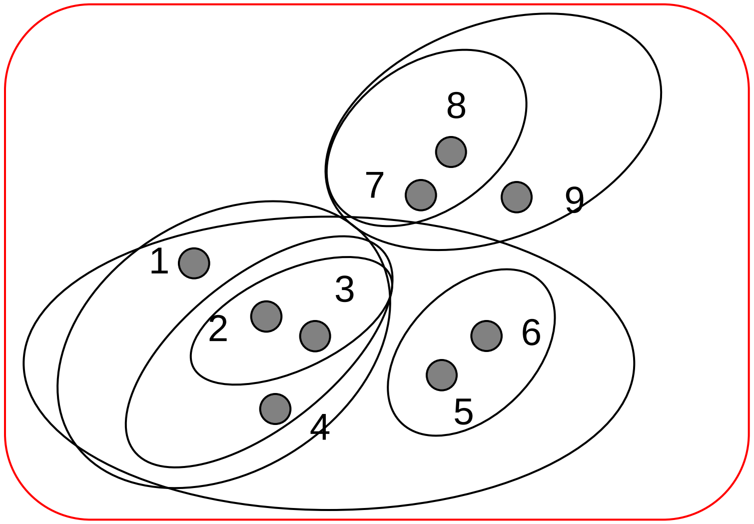
Idea: identify at each step the most similar clusters and merge them



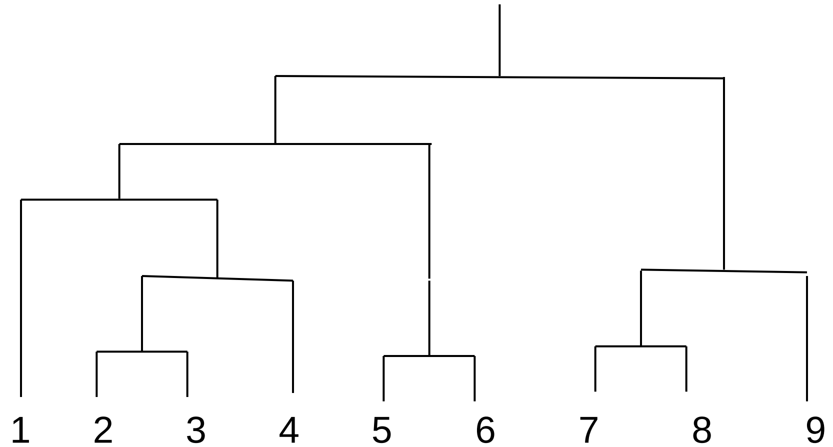


Agglomerative clustering

Idea: identify at each step the most similar clusters and merge them



- The resulting tree is called dendrogram

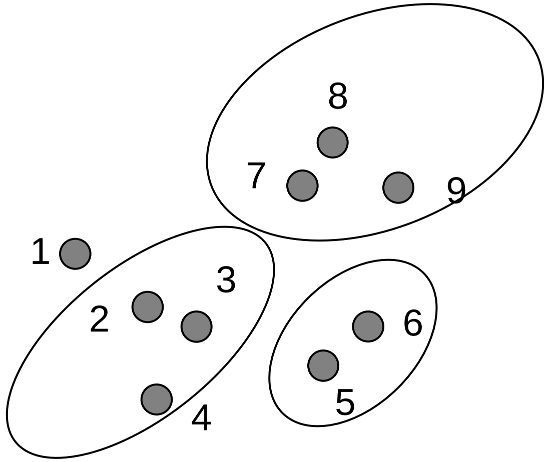


- Representation of the dendrogram: as a set of ordered triples (level, number of clusters, clusters)

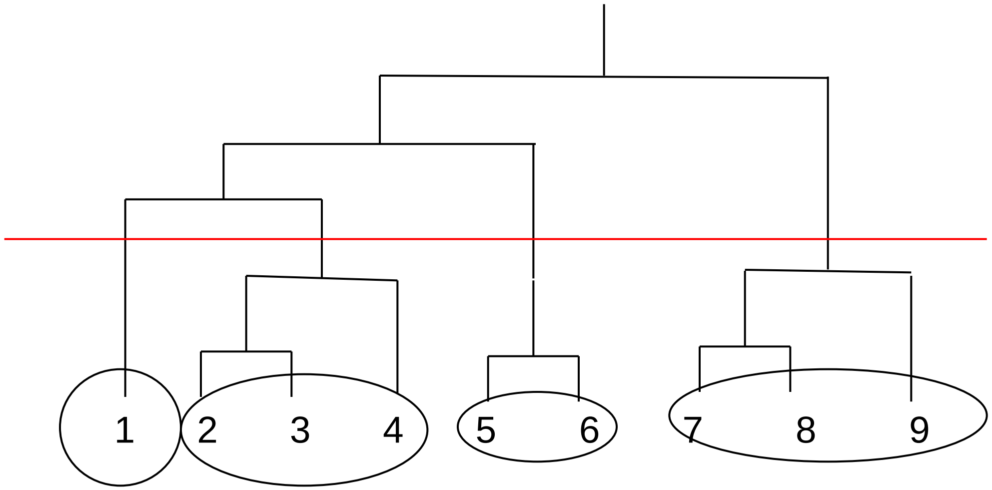
{(0,9,{{1},{2},…,{9}}) ,(1,6,{{1},{2,3},{4},{5,6},{7 ,8},{9}}),
(2,4,{{1},{2,3,4},{5,6},{7,8,9}}), (3,3,{{1,2,3,4},{{5,6},{7,8,9}}),
(4,2,{{1,2,3,4,5,6},{7,8,9}),(5,1,{{1,2,3,4,5,6,7,8,9}})}

Agglomerative clustering

Idea: identify at each step the most similar clusters and merge them



- The resulting tree is called dendrogram
- In order to obtain a partition the dendrogram should be cut at a given level



Partition:

C1={1}

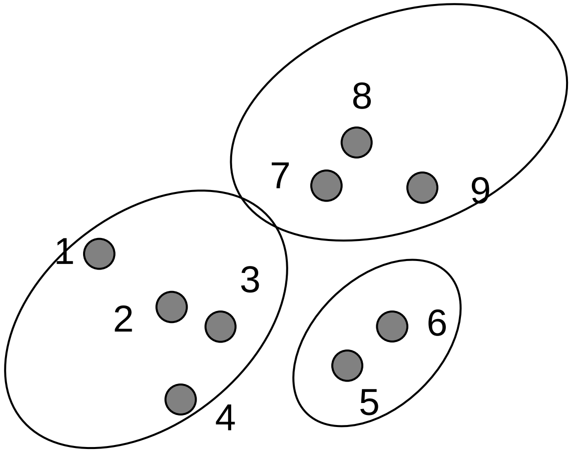
C2={2,3,4}

C3={5,6}

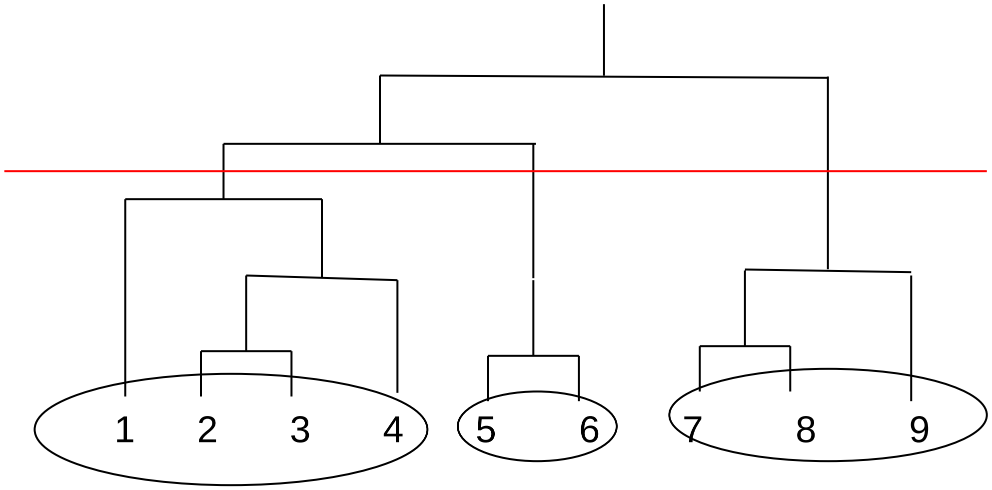
C4={7,8,9}

Agglomerative clustering

Idea: identify at each step the most similar clusters and merge them



- By changing the level one obtains a different partition



Partition:

C1={1,2,3,4}

C2={5,6}

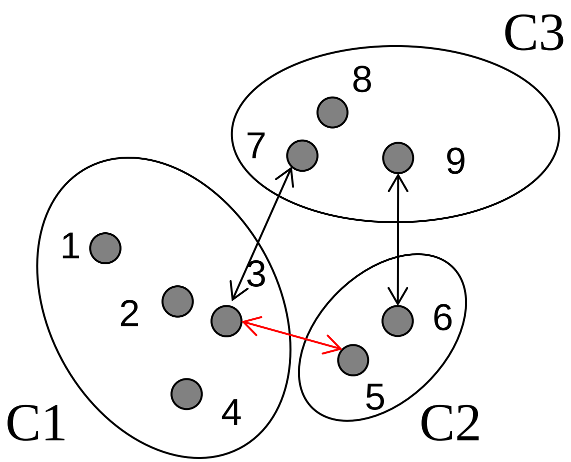
C3={7,8,9}

Agglomerative clustering

Question: how are selected the clusters for merging?

Answer: by using a dissimilarity measure between clusters; there are different ways of computing the dissimilarity measure:

- Single-linkage: the smallest distance between points belonging to different clusters



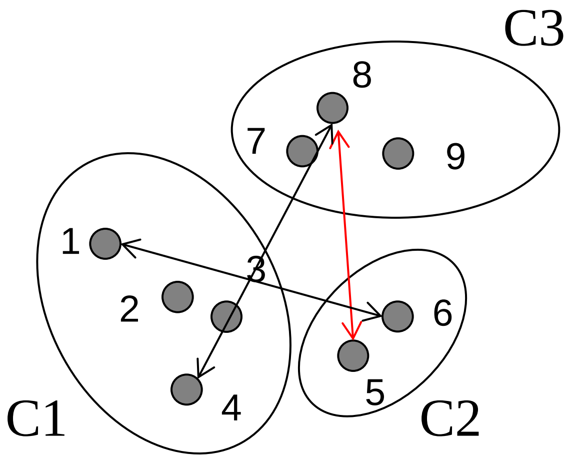
$$D_{SL}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$$

Agglomerative clustering

Question: how are selected the clusters for merging?

Answer: by using a dissimilarity measure between clusters; there are different ways of computing the dissimilarity measure:

- Complete-linkage: the largest distance between points belonging to different clusters



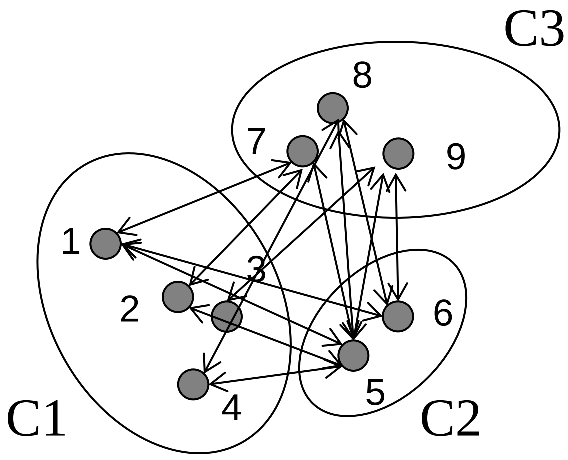
$$D_{CL}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$$

Agglomerative clustering

Question: how are selected the clusters for merging?

Answer: by using a dissimilarity measure between clusters; there are different ways of computing the dissimilarity measure:

- Average-linkage: the average distance between points belonging to different clusters



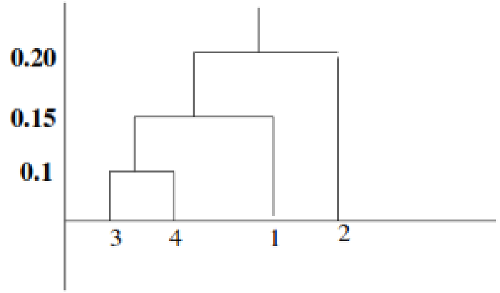
$$D_{AL}(C_i, C_j) = \frac{1}{card(C_i)card(C_j)} \sum_{x \in C_i, y \in C_j} d(x, y)$$

Agglomerative clustering

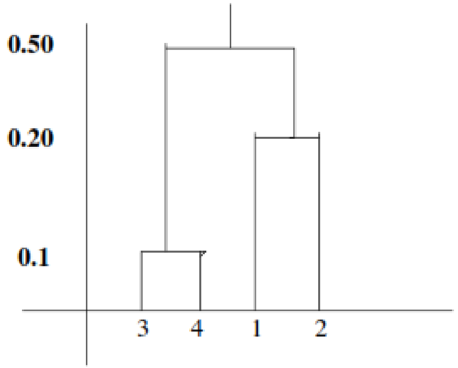
The dissimilarity between clusters influences the clustering result:

	1	2	3	4
1	0.0	0.20	0.15	0.30
2	0.20	0.0	0.40	0.50
3	0.15	0.40	0.0	0.10
4	0.30	0.50	0.10	0.0

(a) Dissimilarity Matrix



(b) Single Link



(c) Complete Link

Data Clustering: Algorithms and Applications, 2014

Agglomerative clustering

Algorithm

Input : data set with N instances

$X=\{x_1,x_2,\ldots,x_N\}$ + dissimilarity matrix D

Output: dendrogram (set of triples)

```
agglomerative(X,D)
level=0;  k=N
C={{x1},{x2},...,{xN}};  DE={(level,k,C)}
repeat
  oldk=k
  level=level+1
  (k,C)=mergeClusters(k,C,D)
  D=recompute the dissimilarity  matrix using
      single/complete/average linkage
  DE=union (DE, (level,k,C))
until k=1
```

Remarks:

- The mergeClusters function identifies the closest clusters and merge them
- The algorithm has a quadratic complexity with respect to the number of data instances ($O(N^2)$)
- The agglomerative algorithms are sensitive to the noise in data

Divisive clustering

Generic top-down clustering algorithm

Input : data set with N instances $X=\{x_1,x_2,\ldots,x_N\}$

Output: dendrogram (tree) T

divisive(X,D)

Initialize the tree T with a root node containing the entire data set

Repeat

select a leaf node L from T (based on a specific criterion)

use a flat clustering algorithm to split L into $L_1,L_2,\ldots L_k$

Add $L_1,L_2,\ldots L_k$ as children of L in T

until <a stopping criterion>

Remark: the flat clustering algorithm may be kMeans; a particular case is the bisecting kMeans which is based on splitting each node in two other nodes (by applying kMeans for k=2)

Bisecting Kmeans

- Bisecting K-means algorithm
  - Variant of K-means that can produce a partitional or a hierarchical clustering

```
Initialize the list of clusters to contain the cluster containing all points.
repeat
    Select a cluster from the list of clusters
    for i = 1 to number_of_iterations do
        Bisect the selected cluster using basic K-means
    end for
    Add the two clusters from the bisection with the lowest SSE to the list of clusters.
until Until the list of clusters contains K clusters
```