Music Machine Learning

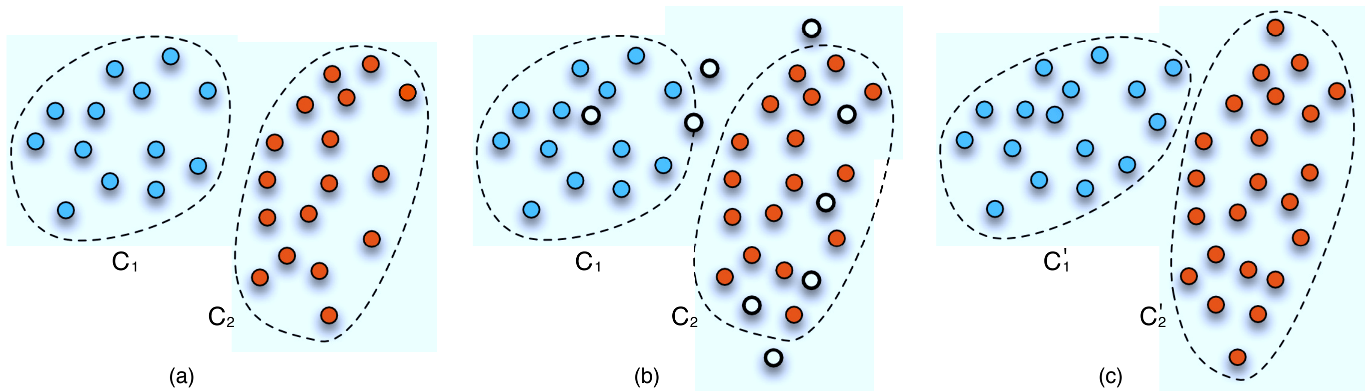
IV – Unsupervised clustering

Master ATIAM - Informatique

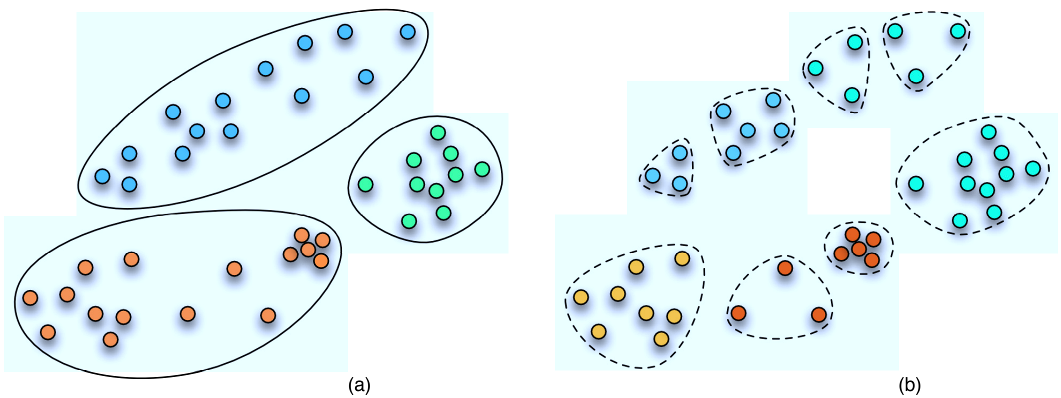
Philippe Esling (esling@ircam.fr)
Maître de conférences – UPMC
Equipe représentations musicales (IRCAM, Paris)

Clustering

- So far we have dealt with the **classification** problem
- Based on labeled training data, we want to find the class of unlabeled

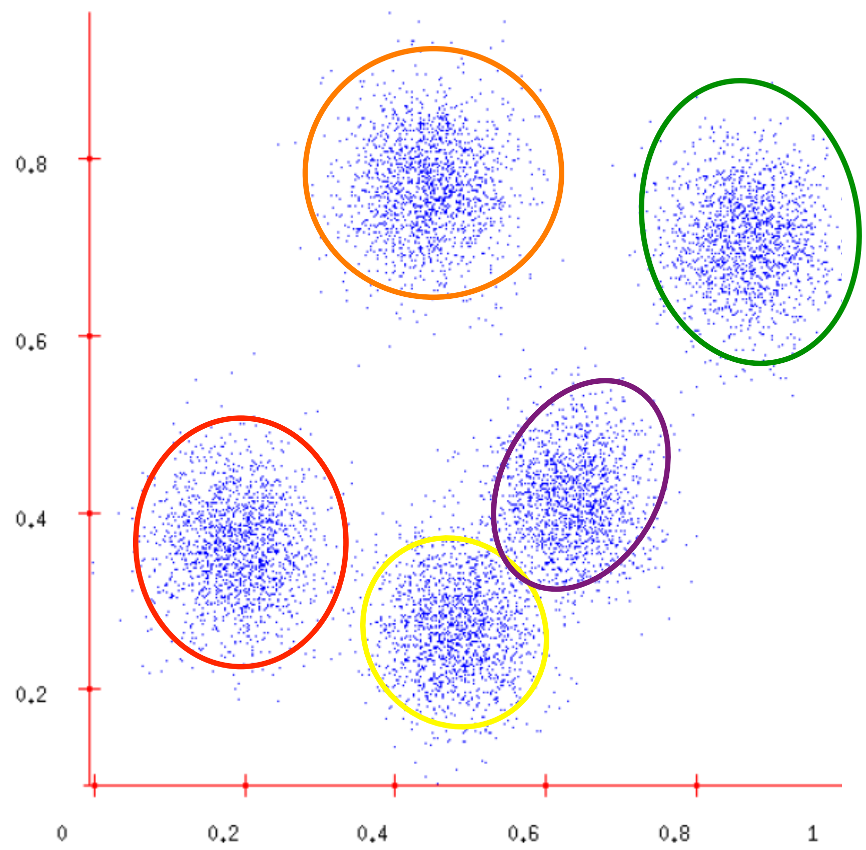


- But what happens if we don't have any prior knowledge on the data?



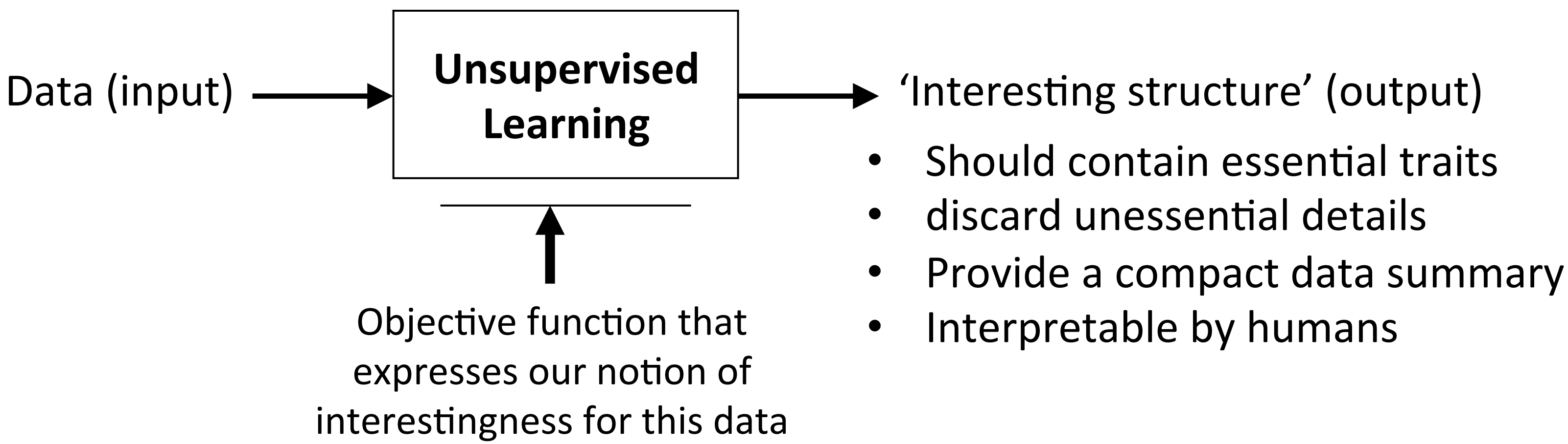
master ATiAM

Clustering example



Clustering

- Attach label to each observation or data points in a set
- You can say this is "unsupervised classification"
- Clustering is alternatively called as "grouping"
- You want to assign same label to data points that are "close"
- Thus, clustering algorithms rely on a distance metric between data points
- Sometimes, it is said that the for clustering, the distance metric is more important than the clustering algorithm



What we need for clustering

Data matrix

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

Dissimilarity matrix

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

(Dis)similarity between objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include: *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \ldots + |x_{i_p} - x_{j_p}|^q)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two p-dimensional data objects, and q is a positive integer

- If $q = 1$, d is Manhattan distance

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \ldots + |x_{i_p} - x_{j_p}|$$

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

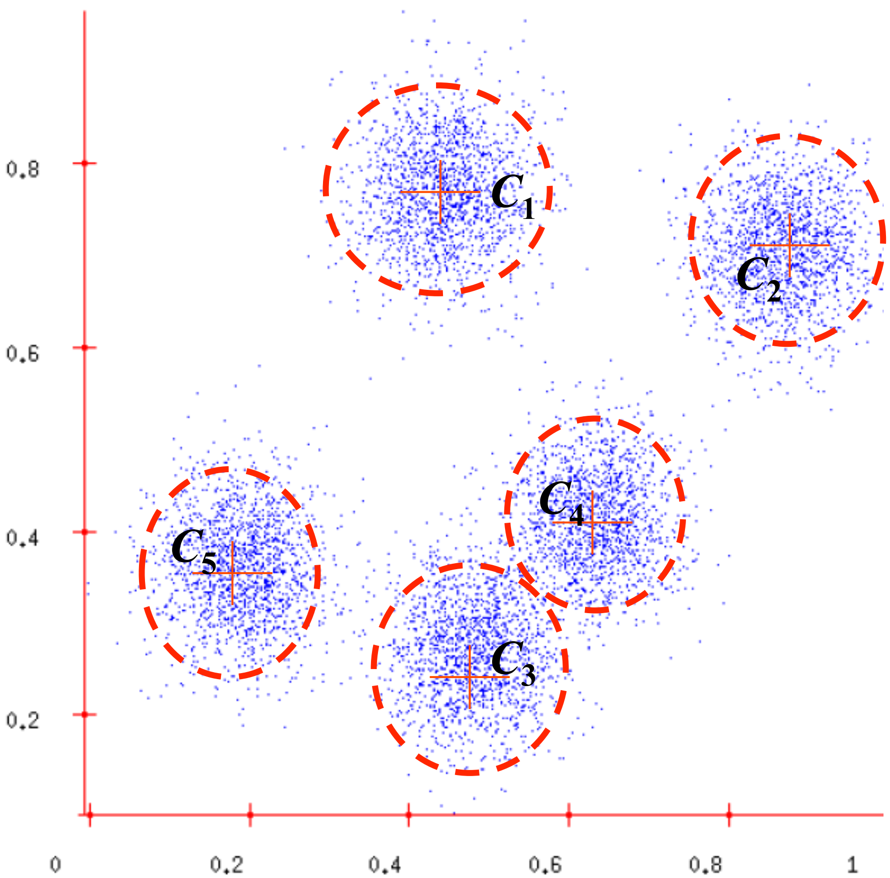
$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \ldots + |x_{i_p} - x_{j_p}|^2)}$$

- Also one can use weighted distance, parametric Pearson product moment correlation, or other disimilarity measures.

Distance-based clustering

- Assign a distance measure between data
- Find a partition such that:
  - Distance between objects within partition (I.e. same cluster) is minimized
  - Distance between objects from different clusters is maximised
- Issues :
  - Requires defining a distance (similarity) measure in situation where it is unclear how to assign it
  - What relative weighting to give to one attribute vs another?
  - Number of possible partition us superexponential

A « good » clustering ?



We can evaluate the distance **within-clusters**

$$argmin_{C_j, m_{i,j}} \left(\sum_j \sum_i (x_i - C^j) \right)$$

based on the centroid of each

With the membership functions and the conditions

$$m_{i,j} = \begin{cases} 1 & x_i \in \text{the j-th cluster} \\ 0 & x_i \notin \text{the j-th cluster} \end{cases}$$

$$\sum_j m_{i,j} = 1$$

$\rightarrow$ any $x_i \in$ a single cluster

How to efficiently cluster ?

$$argmin_{C_j, m_{i,j}} (\sum_j \sum_i (x_i - C^j))$$

based on the centroid of each

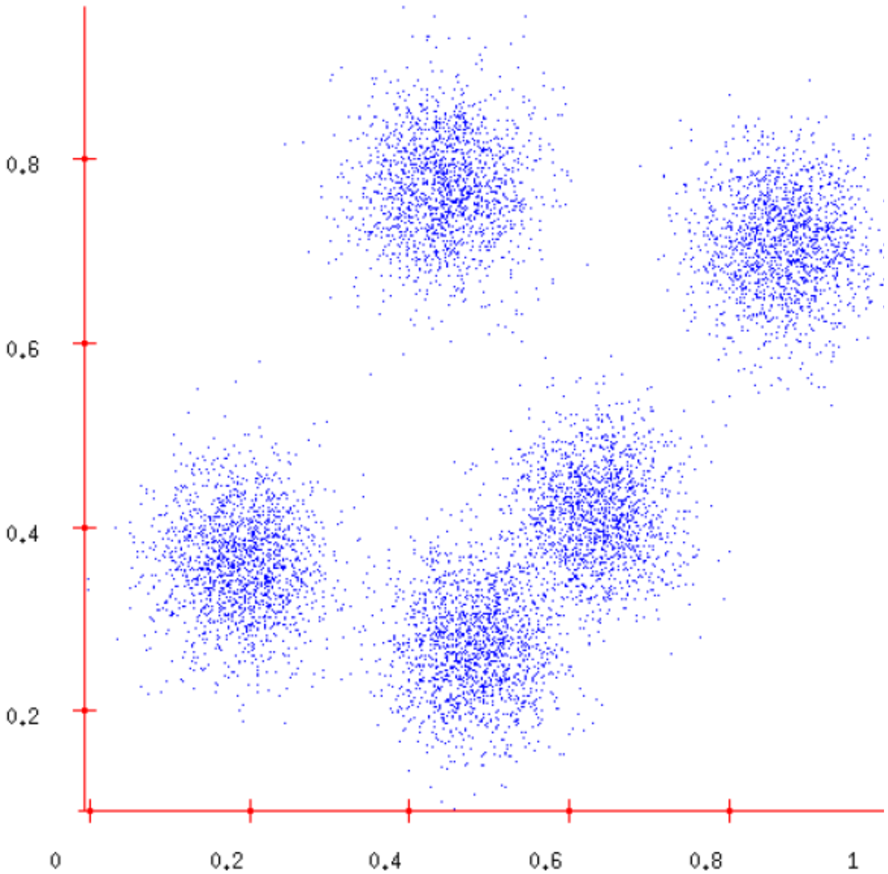
Memberships $\{m_{i,j}\}$ and centroids $\{C_j\}$ are correlated.

So we could somehow reverse the paradigm

$$\text{Given centroids } \{C_j\},\ m_{i,j} = \begin{cases} 1 & j = \arg\min_k (x_i - C_j)^2 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Given memberships } \{m_{i,j}\},\ C_j = \frac{\sum_{i=1}^{n} m_{i,j} x_i}{\sum_{i=1}^{n} m_{i,j}}$$

K-Means for clustering



- **K-means algorithm**

1. Start with a random guess of cluster centers
2. Determine the membership of each data points
3. Adjust the cluster centers

Loop with stop criterion based on

1. Iterations number
2. Quality criterion
3. Evolution of quality

Formalising *stop criterions*

- Define a **measure of cluster compactness** (total distance from the cluster mean)

$$\sum_{\mathbf{x}_n \in \mathcal{C}_k} ||\mathbf{x}_n - \mathbf{m}_k||^2 = \sum_{n=1}^{N} z_{kn} ||\mathbf{x}_n - \mathbf{m}_k||^2$$

where the cluster mean is defined as

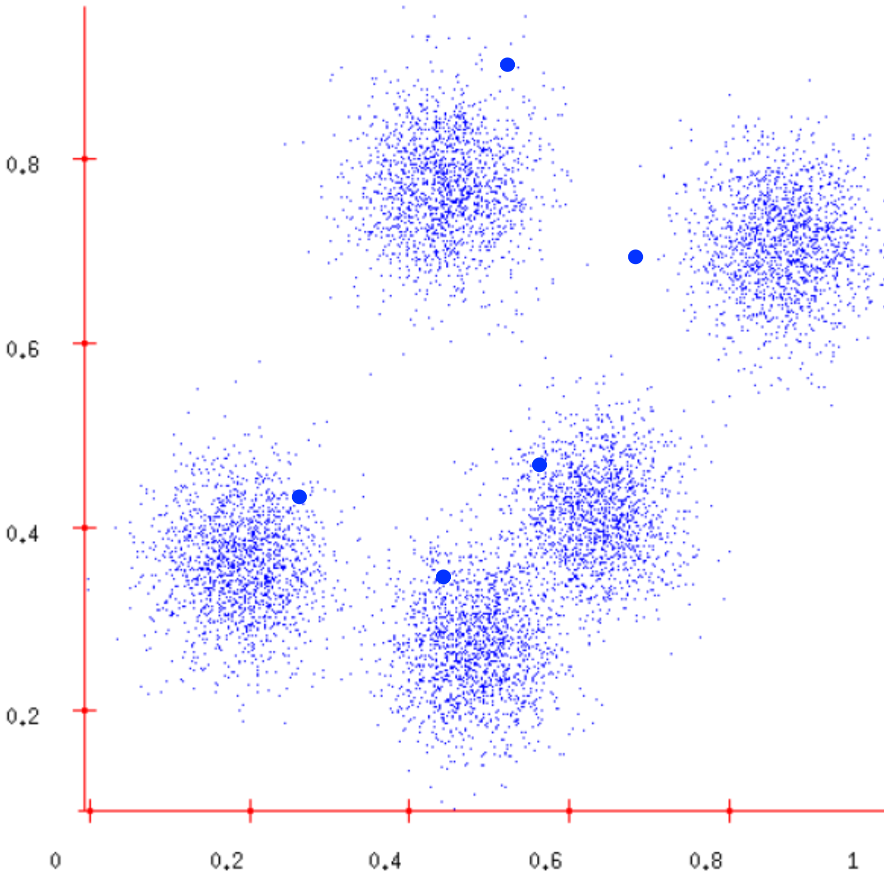
$$\mathbf{m}_k = \frac{1}{N_k} \sum_{\mathbf{x}_n \in \mathcal{C}_k} \mathbf{x}_n$$

and $N_k = \sum_{n=1}^{N} z_{kn}$ is the total number of points allocated to cluster K

- Define a **measure of cluster quality**

$$\mathcal{E}_K = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} ||\mathbf{x}_n - \mathbf{m}_k||^2$$

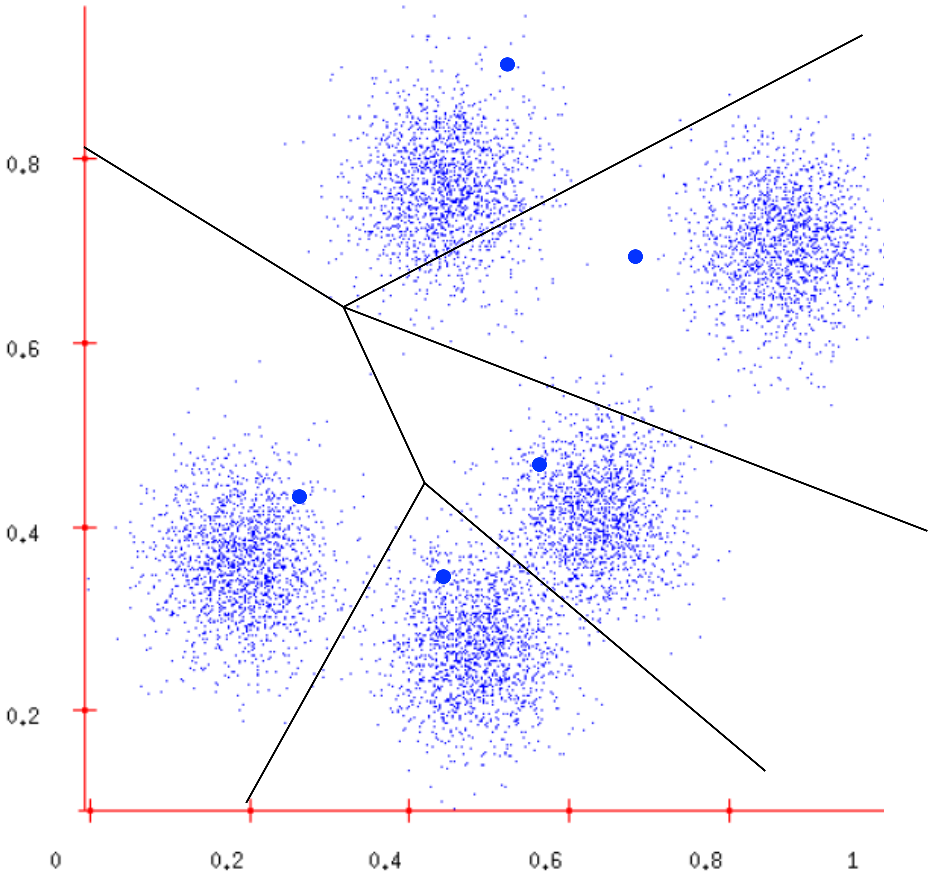
K-Means for clustering



- **K-means algorithm**

  1. Ask user how many clusters (here we set K=5)
  2. Start with a random guess of cluster centers

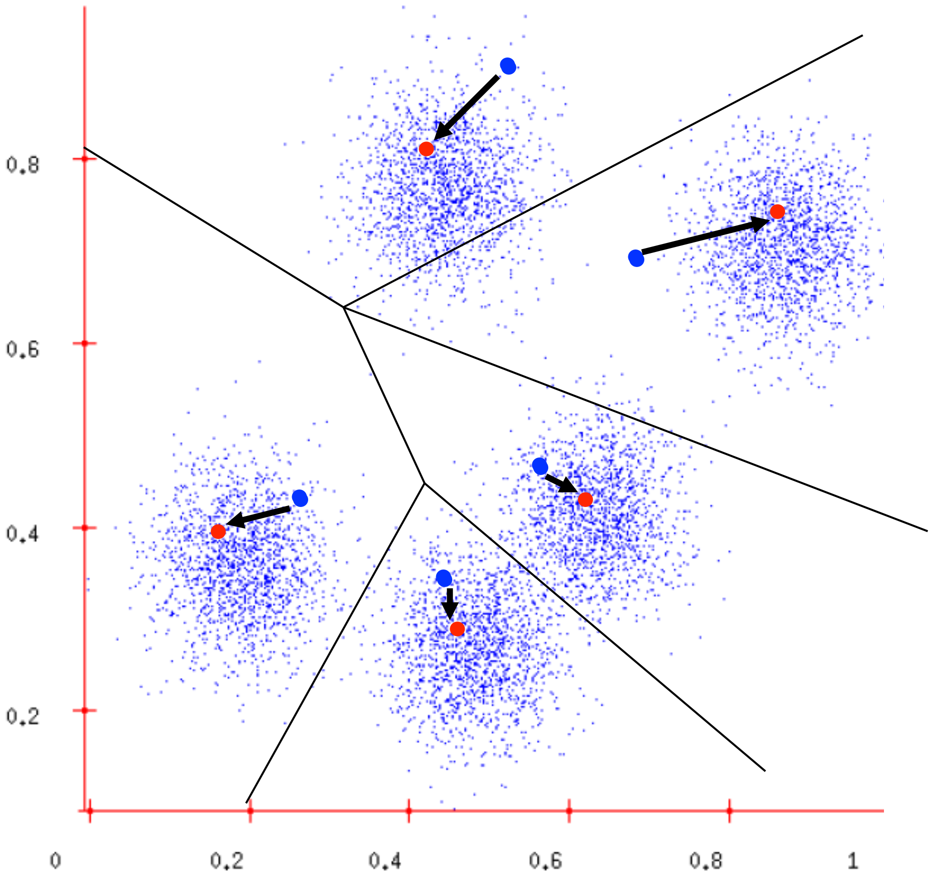
K-Means for clustering



- **K-means algorithm**

1. Ask user how many clusters (here we set K=5)
2. Start with a random guess of cluster centers
3. Each datapoint finds out which Center its closest to. (each Center "owns" a set of points)

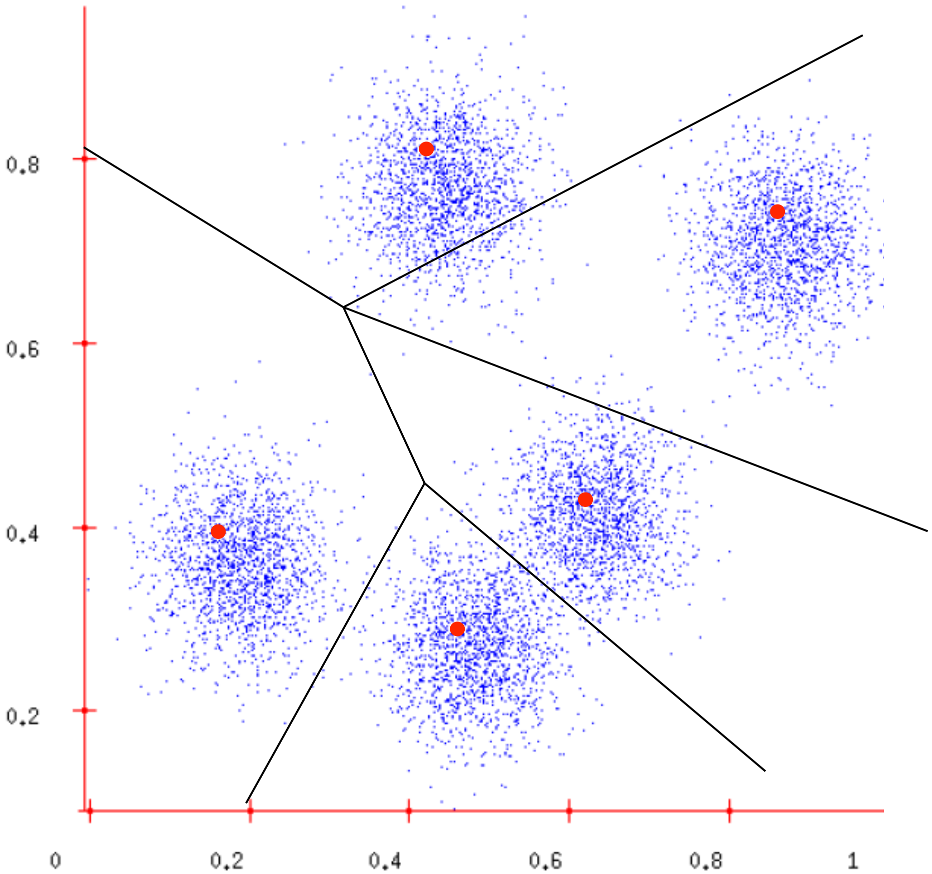
K-Means for clustering



- **K-means algorithm**

1. Ask user how many clusters (here we set K=5)
2. Start with a random guess of cluster centers
3. Each datapoint finds out which Center its closest to. (each Center "owns" a set of points)
4. Adjust the center by computing the median of the points set

K-Means for clustering



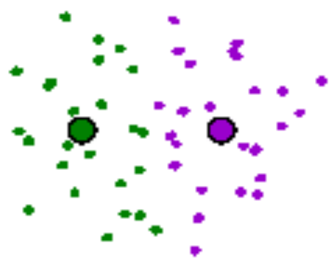
- **K-means algorithm**

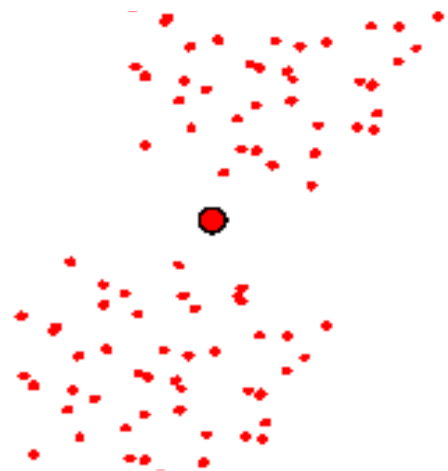
1. Ask user how many clusters (here we set K=5)
2. Start with a random guess of cluster centers
3. Each datapoint finds out which Center its closest to. (each Center "owns" a set of points)
4. Adjust the center by computing the median of the points set

Computational Complexity: O(N) where N is the number of points?

Problems of K-means

1. Obviously the number of clusters K
2. But even with the right number, will we find a good optima?





3. Also highly depends on the random start
4. We could perform several runs of K-means

K-Means for clustering

- <u>Strength</u>
  - *Relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- <u>Weakness</u>
  - 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*

Variations of K-means

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes* (Huang'98)
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data: *k-prototype* method

K-medoids clustering

- *K*-means is appropriate when we can work with Euclidean distances
- Thus, *K*-means can work only with numerical, quantitative variable types
- Euclidean distances do not work well in at least two situations
  - Some variables are categorical
  - Outliers can be potential threats
- A general version of *K*-means algorithm called *K*-medoids can work with any distance measure
- *K*-medoids clustering is computationally more intensive

K-medoids algorithm

- Step 1: For a given cluster assignment C, find the observation in the cluster minimizing the total distance to other points in that cluster: $i_k^* = \arg\min_{\{i:C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j).$

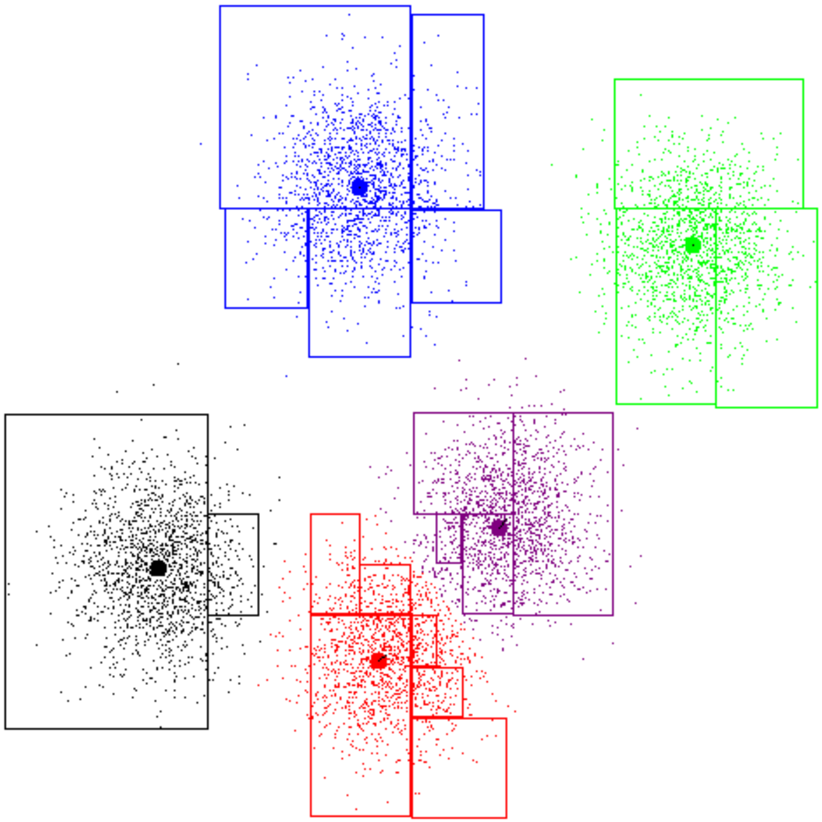
- Step 2: Assign $m_k = x_{i_k^*},\ k = 1,2,\ldots,K$

- Step 3: Given a set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:

$$C(i) = \arg\min_{1 \le k \le K} d(x_i, m_k),\ i = 1,\ldots,N$$

- Iterate steps 1 to 3

Improving K-means



- Group points by region
  - KD tree
  - SR tree
- **Key difference**
  - Find the closest center for each rectangle
  - Assign all the points within a rectangle to one cluster

master ATiAM

Choice of K?

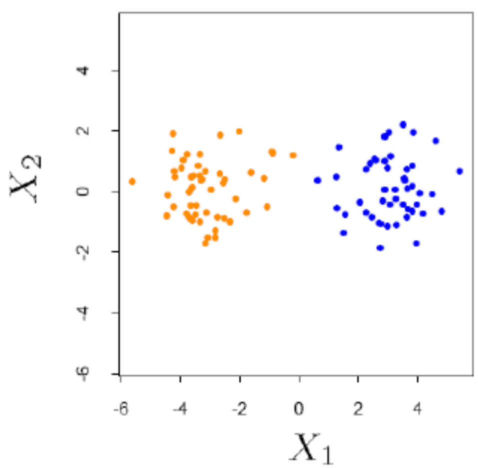
- Can $W_K(C)$, *i.e.*, the within cluster distance as a function of K serve as any indicator?
- Note that $W_K(C)$ decreases monotonically with increasing K. That is the within cluster scatter decreases with increasing centroids.
- Instead look for gap statistics (successive difference between $W_K(C)$):

$$\{W_K - W_{K+1} : K < K^*\} >> \{W_K - W_{K+1} : K \geq K^*\}$$

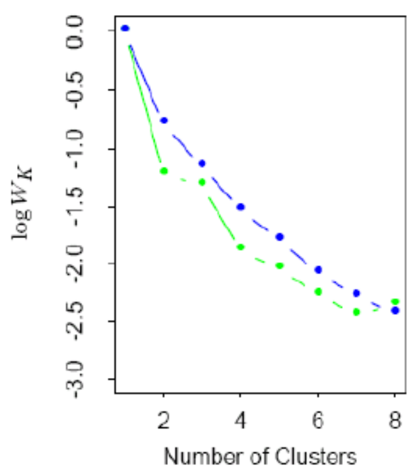
Choice of K?

- Can $W_K(C)$, *i.e.*, the within cluster distance as a function of K serve as any indicator?
- Note that $W_K(C)$ decreases monotonically with increasing K. That is the within cluster scatter decreases with increasing centroids.
- Instead look for gap statistics (successive difference between $W_K(C)$):

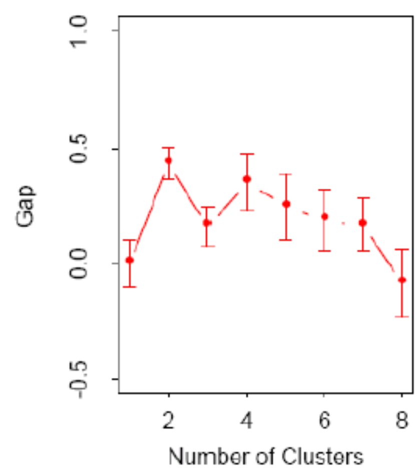
$$\{W_K - W_{K+1} : K < K^*\} >> \{W_K - W_{K+1} : K \geq K^*\}$$



Data points simulated from two pdfs



Log(W_K) curve

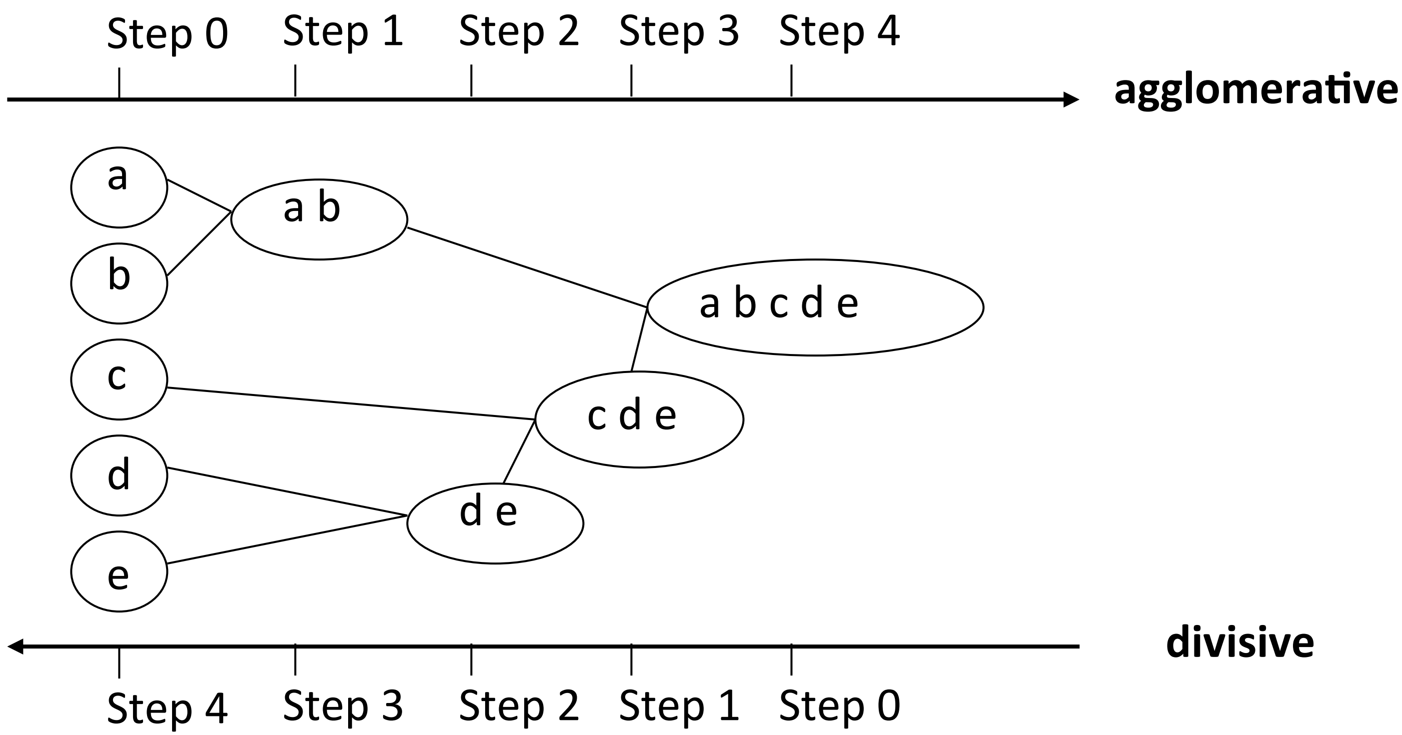


Gap curve

masterATIAM

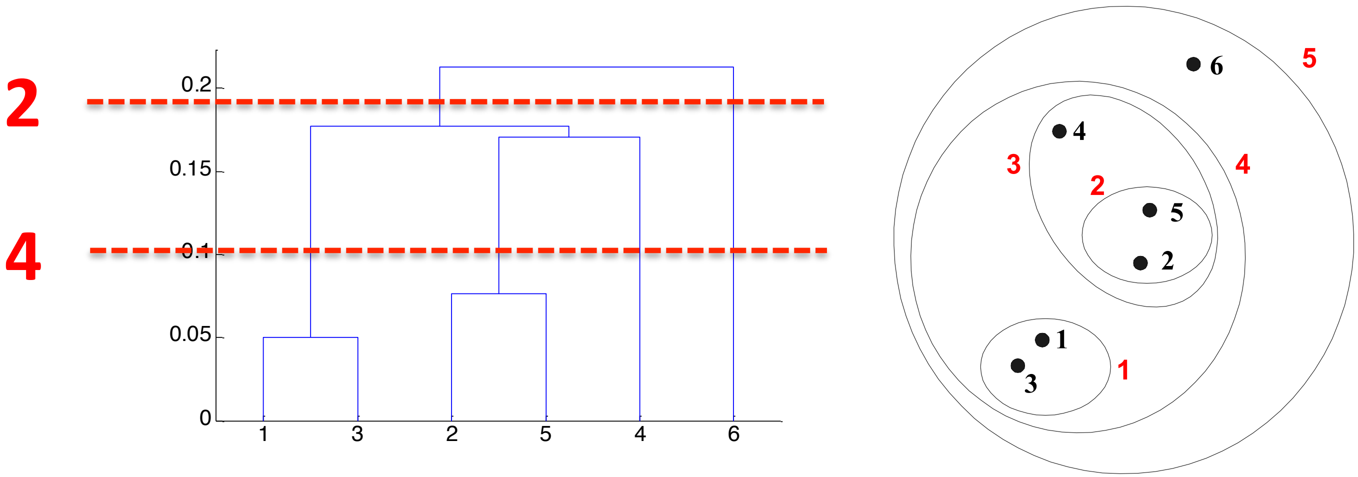
Hierarchical clustering

- Use distance matrix as clustering criteria. This method does not require the number of clusters ***k*** as an input, but needs a termination condition



Hierarchical clustering

- Produces set of ***nested clusters*** organized as hierarchical tree
- Can be visualized as a **dendrogram**
  - A tree-like diagram that records the sequences of merges or splits
  - A clustering can be obtained by trimming the tree



Hierarchical clustering

- Two main types of hierarchical clustering
  - **Agglomerative:**
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or **k** clusters) left
  - **Divisive:**
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are **k** clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Hierarchical clustering

Strength

- No assumptions on the number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- Hierarchical clusterings may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., phylogeny reconstruction, etc), web (e.g., product catalogs) etc

Complexity

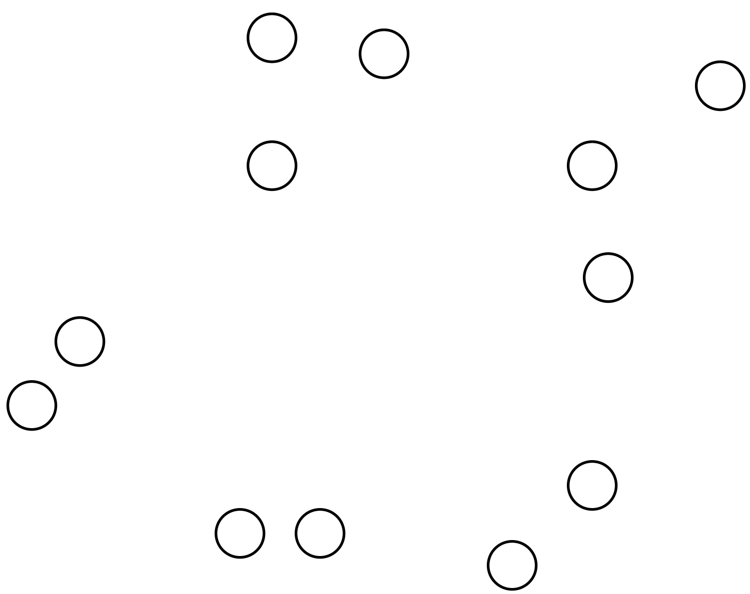
- Distance matrix is used for deciding which clusters to merge/split
- At least quadratic in the number of data points
- Not usable for large datasets

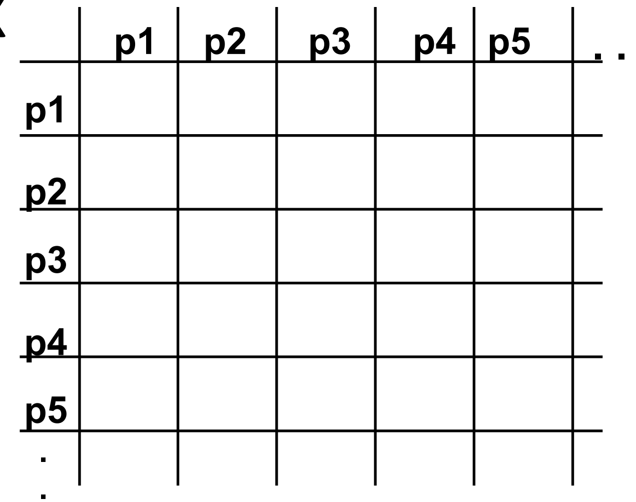
Agglomerative clustering

- Most popular hierarchical clustering technique
- Basic algorithm
  1. Compute the distance matrix between the input data points
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the distance matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the distance between two clusters
  - Different definitions of the distance between clusters lead to different algorithms

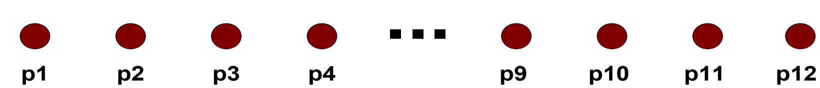
Hierarchical clustering

- Start with clusters of individual points and a distance/proximity matrix



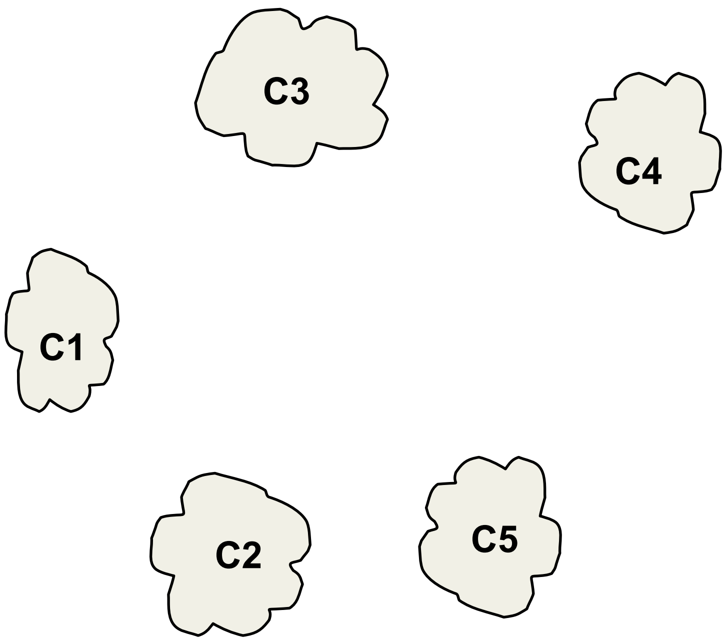


Distance/Proximity Matrix



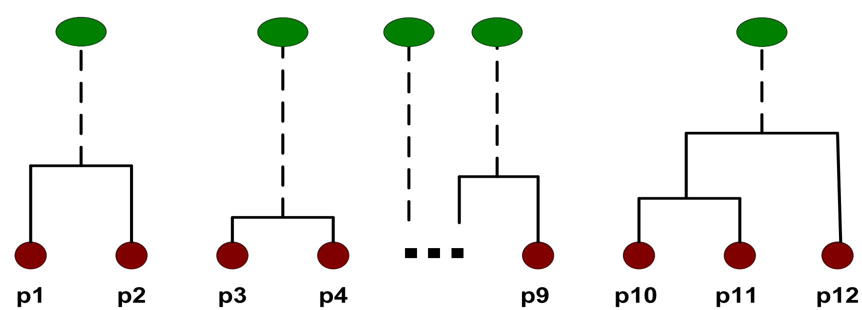
Hierarchical clustering

- After some merging steps, we have some clusters



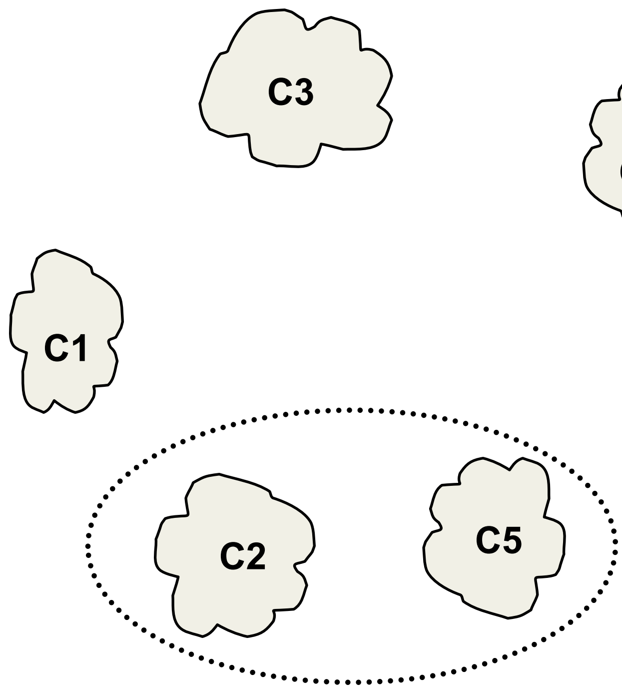
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

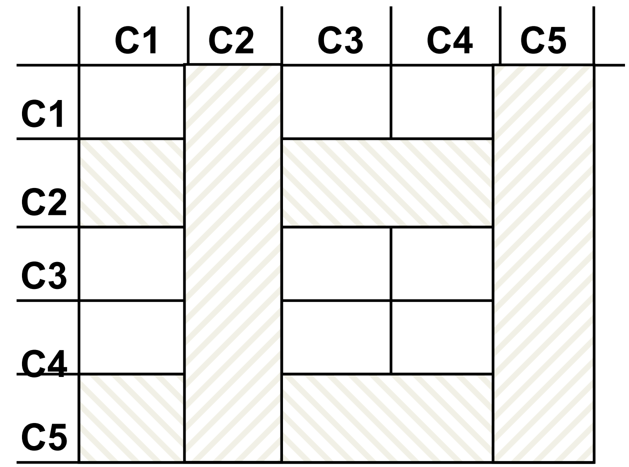
Distance/Proximity Matrix



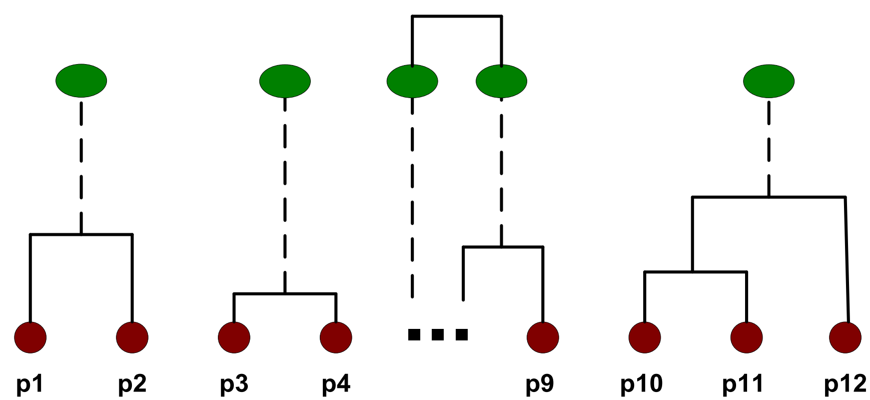
Hierarchical clustering

- Merge the two closest clusters (C2 and C5) and update the distance matrix.



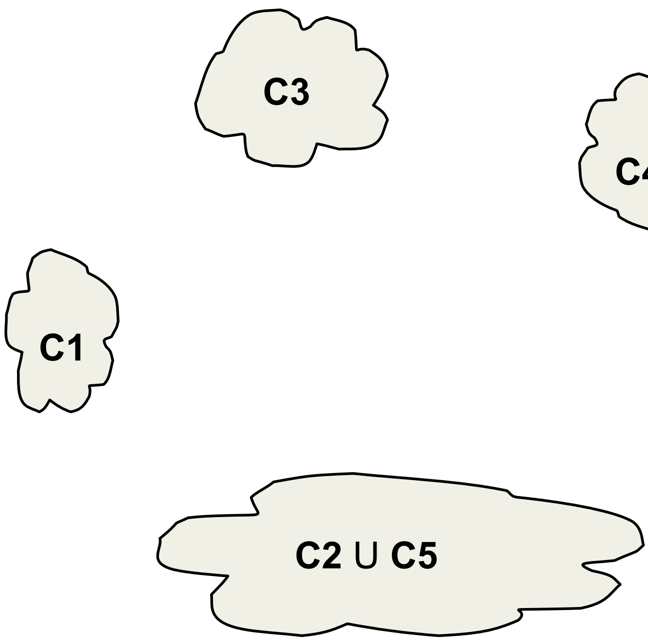


Distance/Proximity Matrix

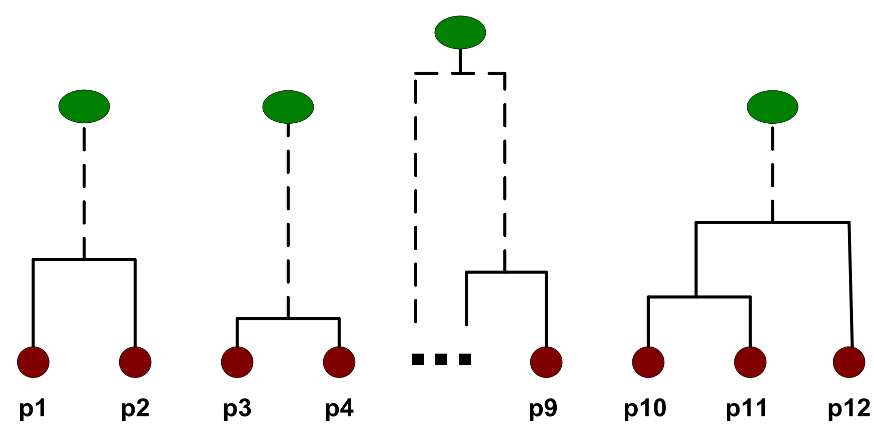


Hierarchical clustering

- "How do we update the distance matrix?"



	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		



Single-link

- **Single-link distance** between clusters C_i and C_j is the ***minimum distance*** between any object in C_i and any object in C_j

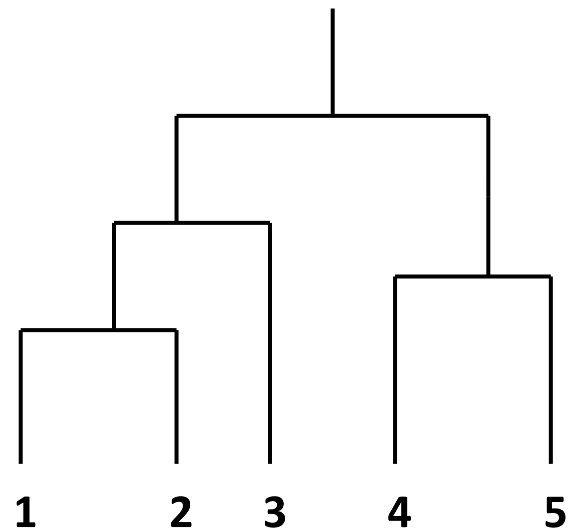
- The distance is **defined by the two most similar objects**

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

Single-link: example

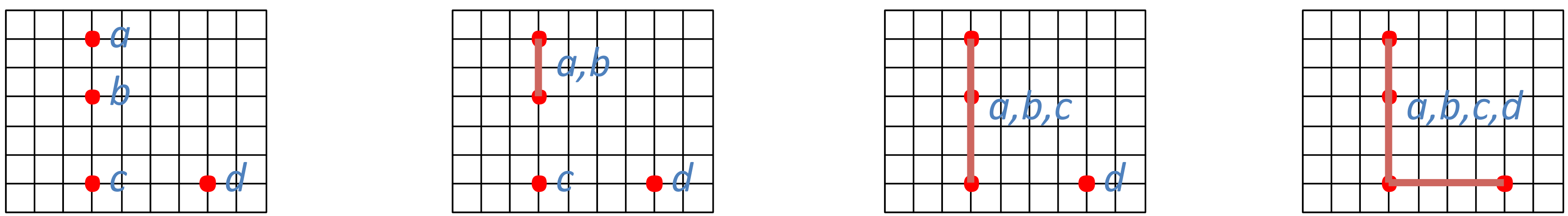
- Determined by one pair of points, i.e., by one link in the proximity graph.

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Single-link: evolution

Euclidean Distance



(1) (2) (3)

	b	c	d
a	2	5	6
b		3	5
c			4

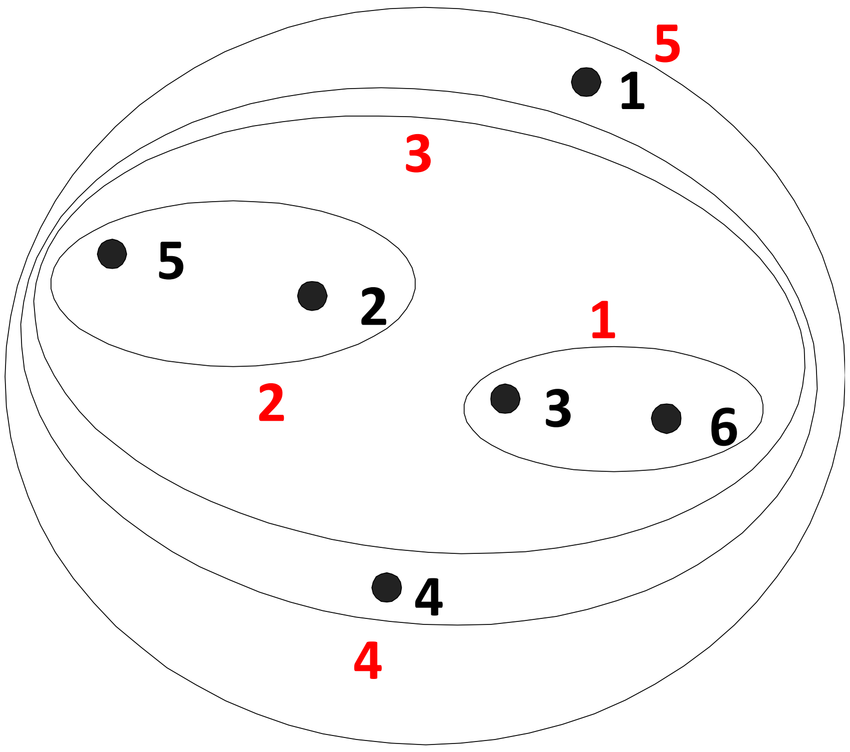
	b	c	d
a	2	5	6
b		3	5
c			4

	c	d
a,b	3	5
c		4

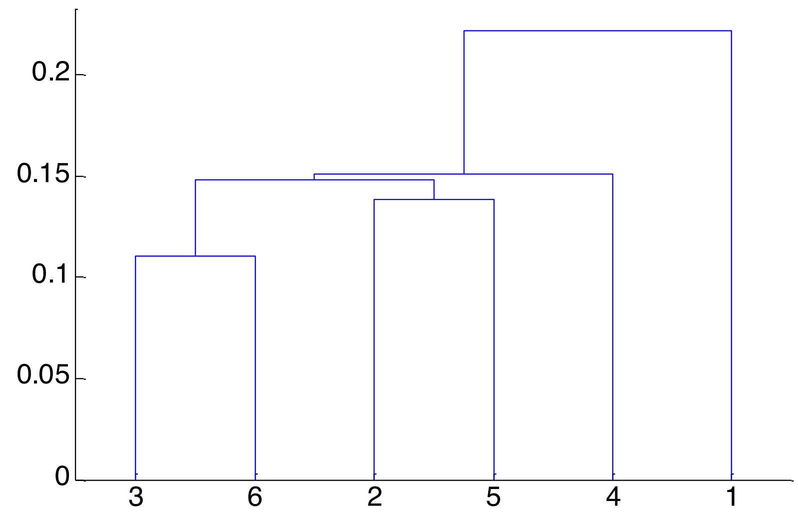
	d
a,b,c	4

Distance Matrix

Single-link: example

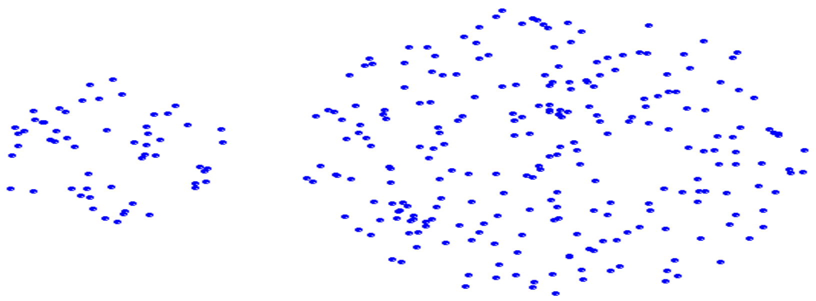


Nested Clusters

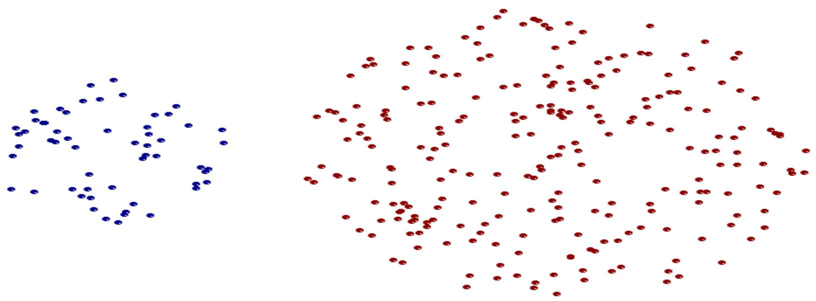


Dendrogram

Single-link: strength



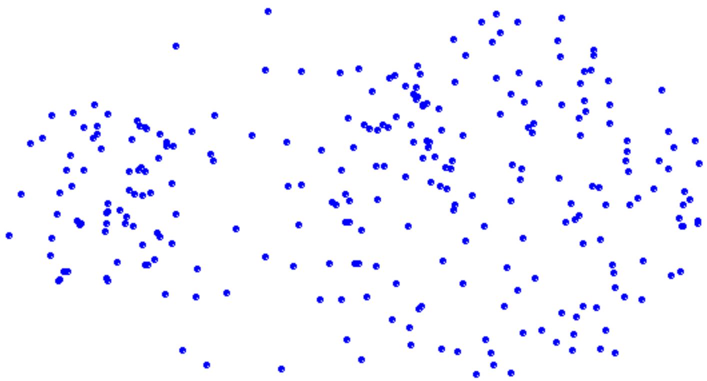
Original Points



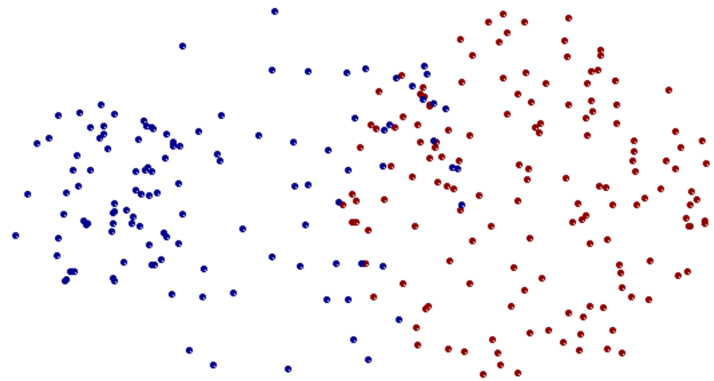
Two Clusters

- **Can handle non-elliptical shapes**

Single-link: limitations



Original Points



Two Clusters

- **Sensitive to noise and outliers**
- **It produces long, elongated clusters**

Complete-link

- **Complete-link distance** between clusters C_i and C_j is the ***maximum distance*** between any object in C_i and any object in C_j

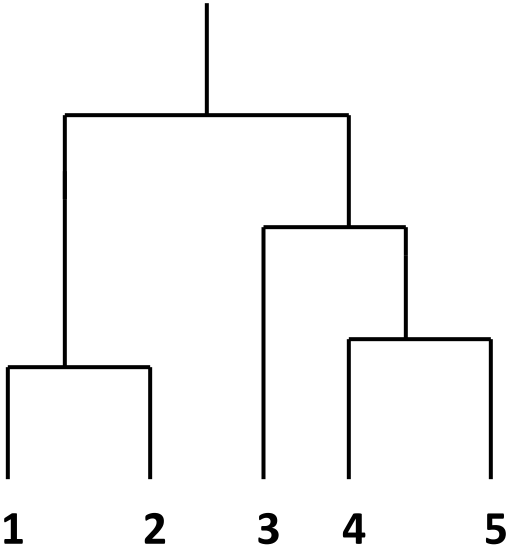
- The distance is **defined by the two most dissimilar objects**

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

Complete-link: example

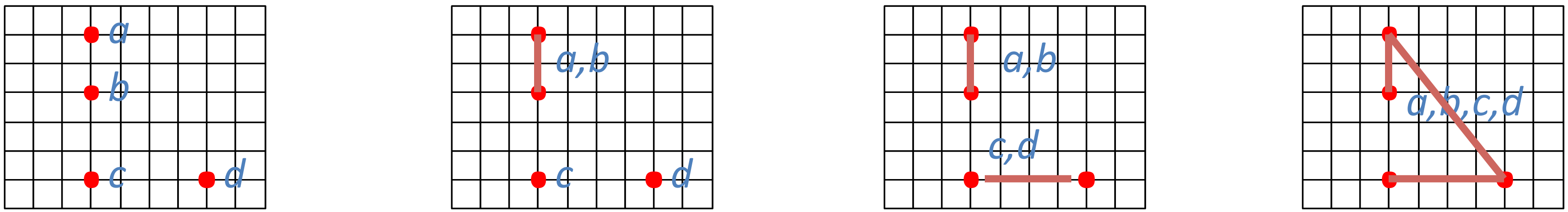
- Distance between clusters is determined by the two most distant points in the different clusters

	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Complete-link: example

Euclidean Distance



(1) (2) (3)

	b	*c*	*d*
a	2	5	6
b		3	5
c			4

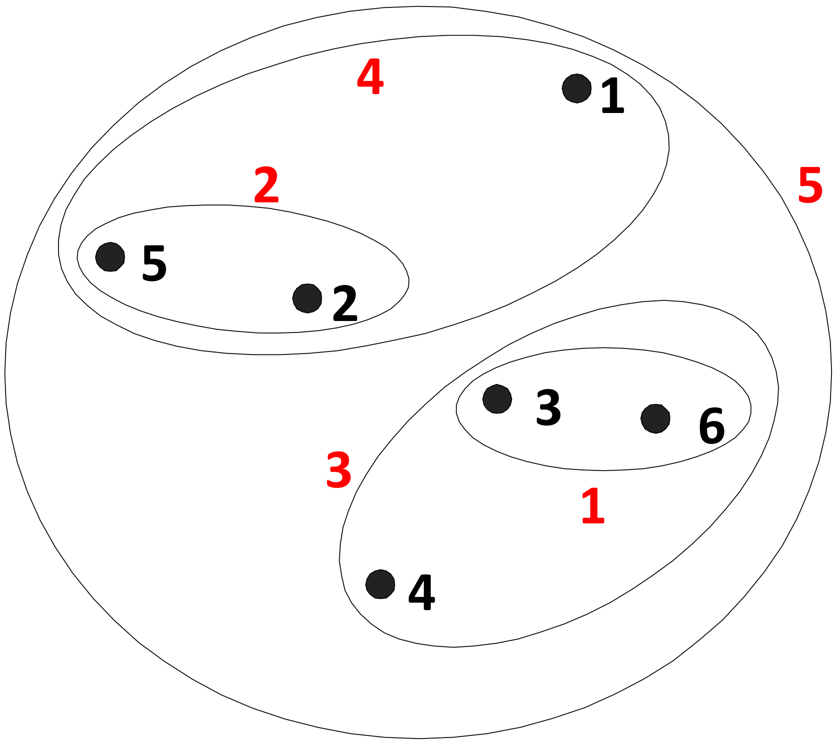
	b	*c*	*d*
a	2	5	6
b		3	5
c			4

	c	*d*
a,b	5	6
c		4

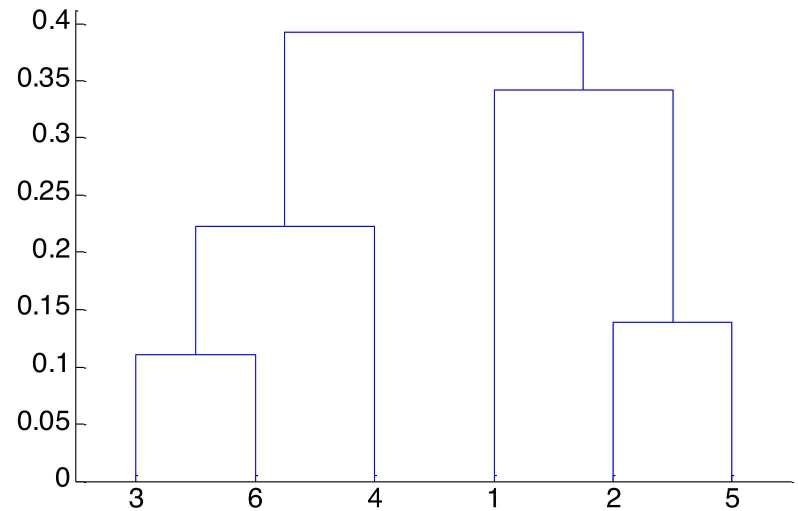
	c,d
a,b	6

Distance Matrix

Complete-link: example

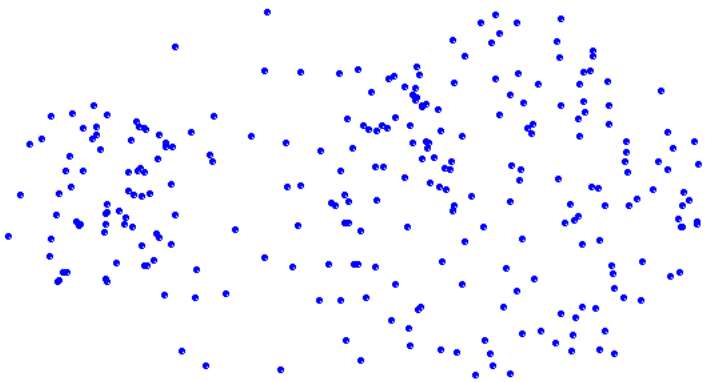


Nested Clusters

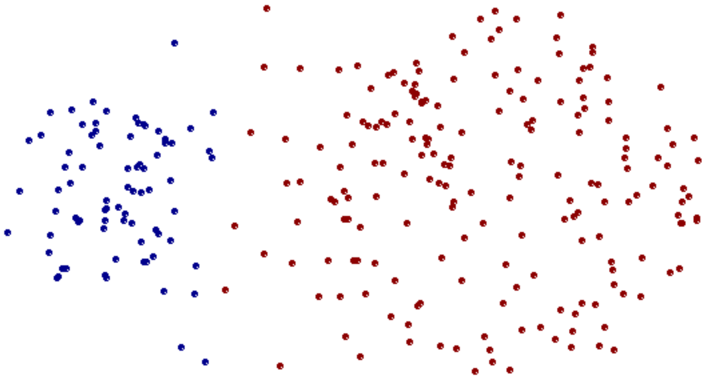


Dendrogram

Complete-link: strength



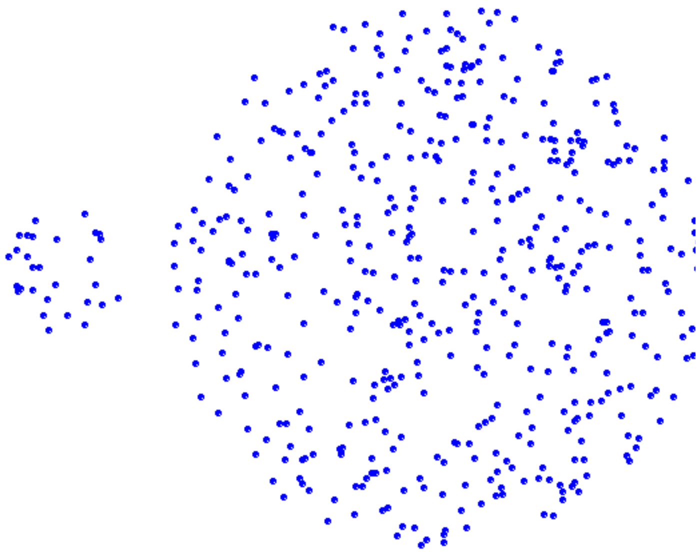
Original Points



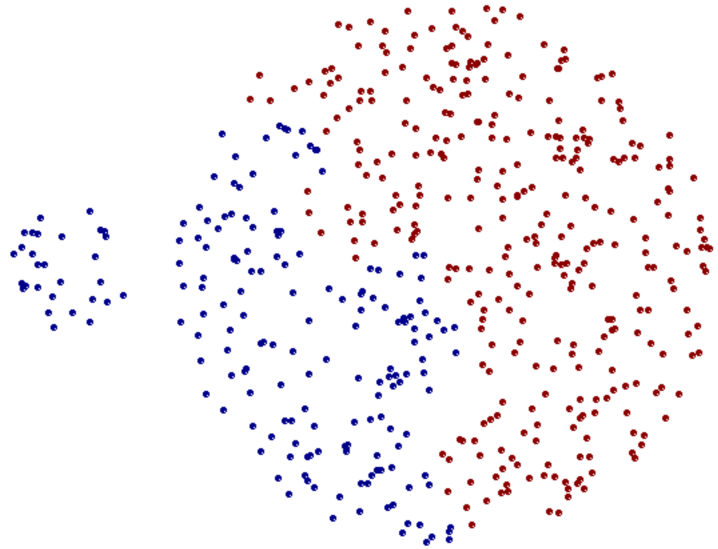
Two Clusters

- **More balanced clusters (with equal diameter)**
- **Less susceptible to noise**

Complete-link: limitations



Original Points



Two Clusters

- Tends to break large clusters
- All clusters tend to have the same diameter – small clusters are merged with larger ones

Average-link

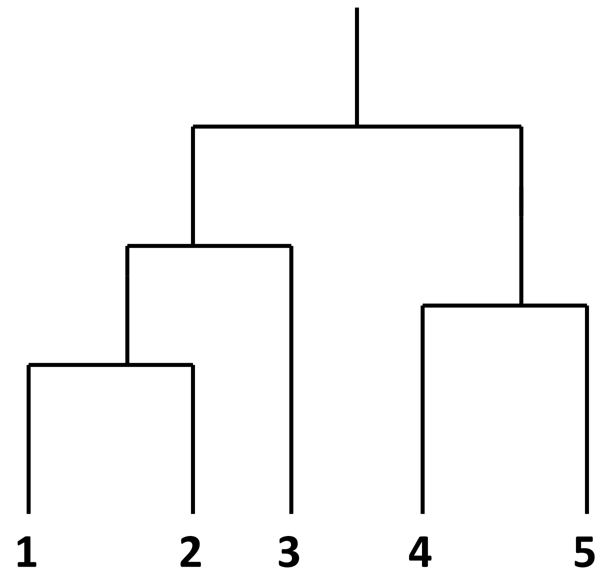
- **Group average distance** between clusters C_i and C_j is the ***average distance*** between any object in C_i and any object in C_j

$$D_{avg}(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$$

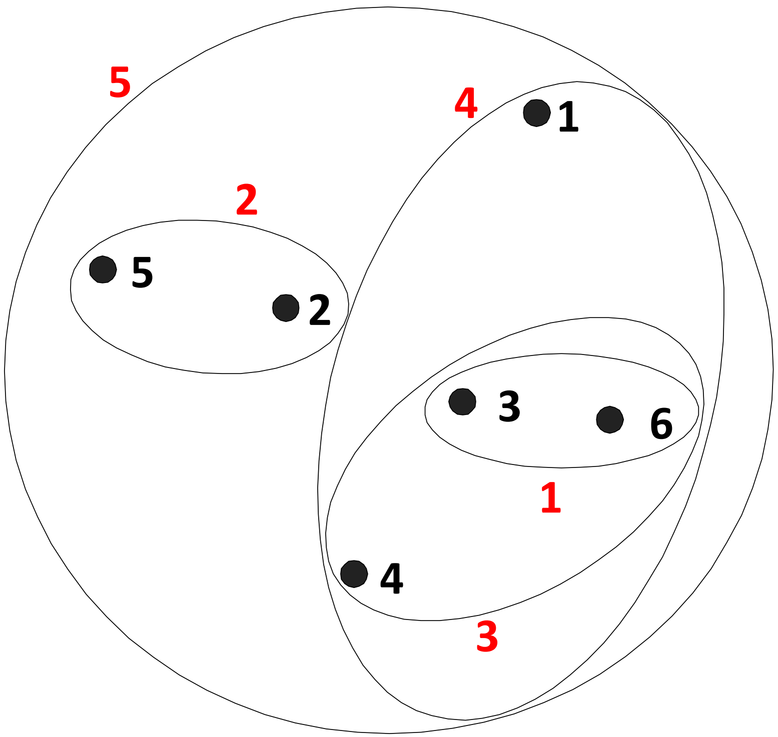
Average-link example

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

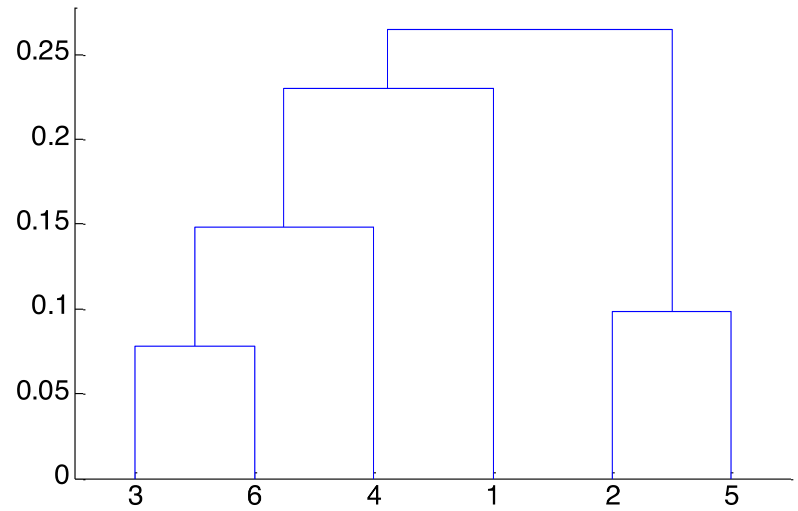
	I1	I2	I3	I4	I5
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00



Average-link example



Nested Clusters



Dendrogram

Average-link

- Compromise between Single and Complete
- Strengths
  - Less susceptible to noise and outliers
- Limitations
  - Biased towards globular clusters

Centroïd distance

- **Centroid distance** between clusters C_i and C_j is the distance between the centroid r_i of C_i and the centroid r_j of C_j

$$D_{centroids}\left(C_i, C_j\right) = d\left(r_i, r_j\right)$$

Ward's distance

- **Ward's distance** between clusters C_i and C_j is the ***difference*** between the ***total within cluster sum of squares for the two clusters separately***, and the ***within cluster sum of squares resulting from merging the two clusters*** in cluster C_{ij}

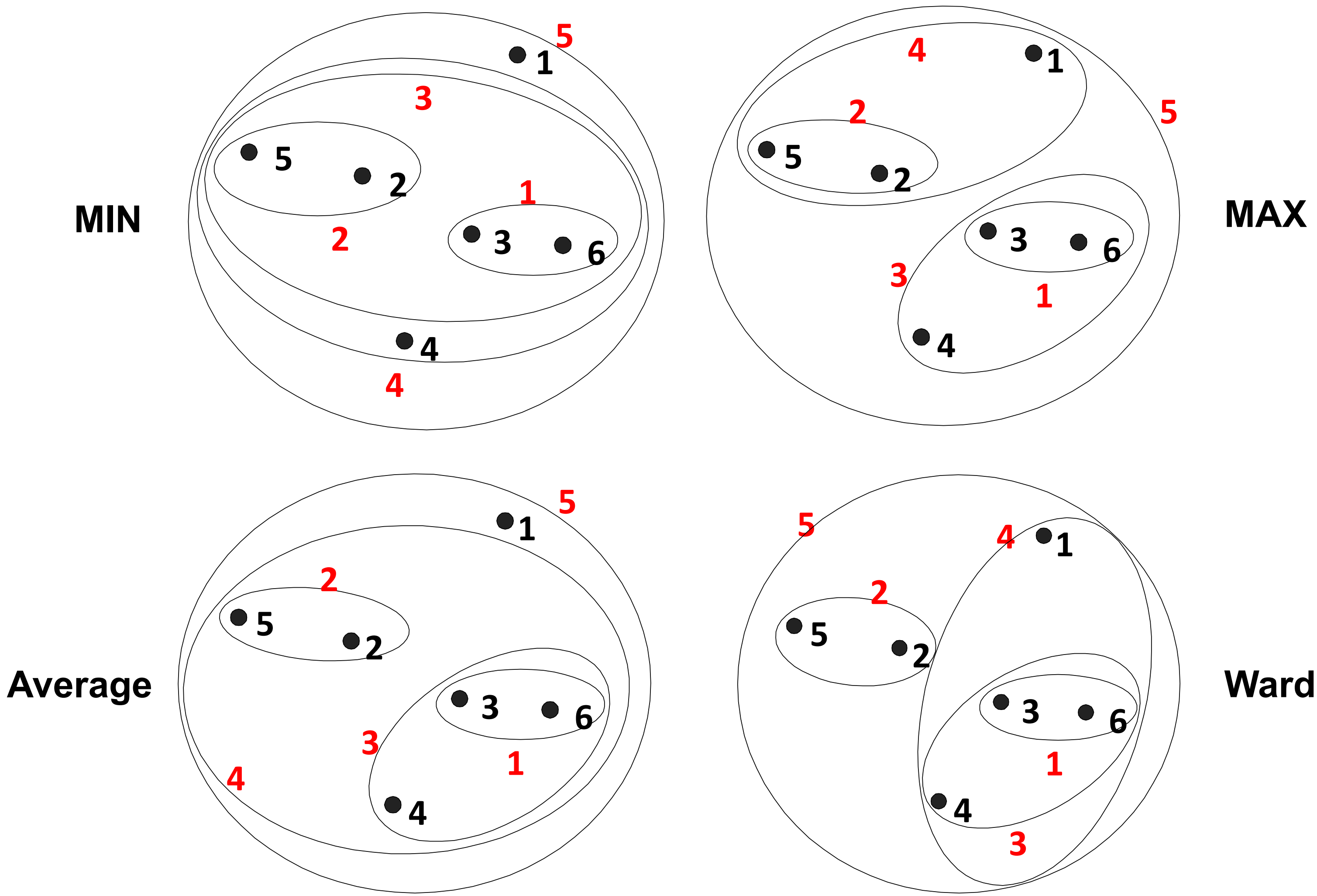
$$D_w(C_i, C_j) = \sum_{x \in C_i} (x - r_i)^2 + \sum_{x \in C_j} (x - r_j)^2 - \sum_{x \in C_{ij}} (x - r_{ij})^2$$

- r_i: centroid of C_i
- r_j: centroid of C_j
- r_{ij}: centroid of C_{ij}

Ward's distance

- Similar to group average and centroid distance
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of k-means
  - Can be used to initialize k-means

Comparisons



Time and space complexity

- For a dataset **X** consisting of **n** points
- **$O(n^2)$ space**; it requires storing the distance matrix
- **$O(n^3)$ time** in most of the cases
  - There are **n** steps and at each step the size **n^2** distance matrix must be updated and searched
  - Complexity can be reduced to **$O(n^2 \log(n))$** time for some approaches by using appropriate data structures

Divisive hierarchical clustering

- Start with a single cluster composed of all data points
- Split this into components
- Continue recursively
- *Monothetic* divisive methods split clusters using one variable/dimension at a time
- *Polythetic* divisive methods make splits on the basis of all variables together
- Any intercluster distance measure can be used
- Computationally intensive, less widely used than agglomerative methods

Prim / Kruskal