Clustering

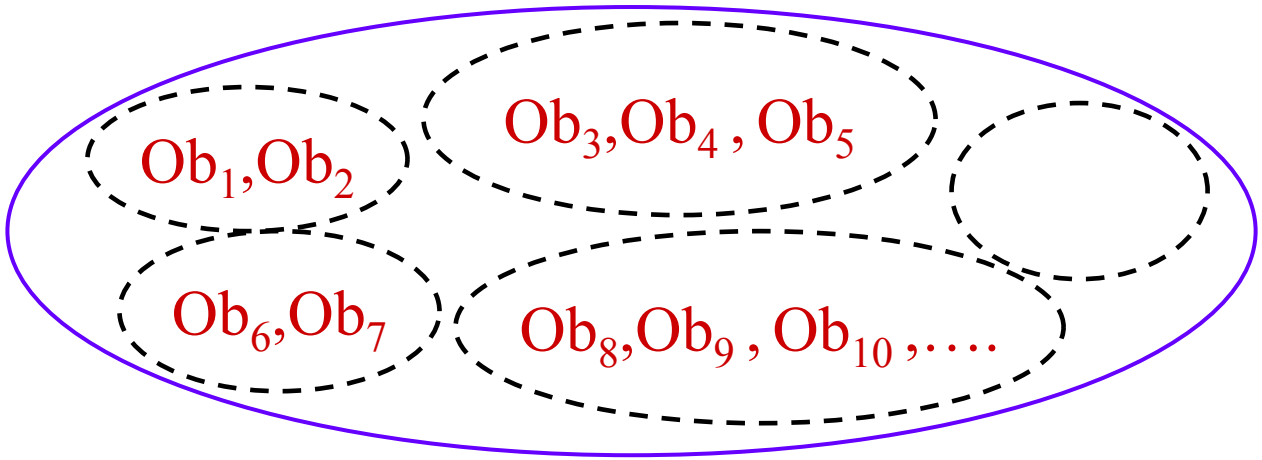
Wang Houfeng
ICL, Peking Univ.

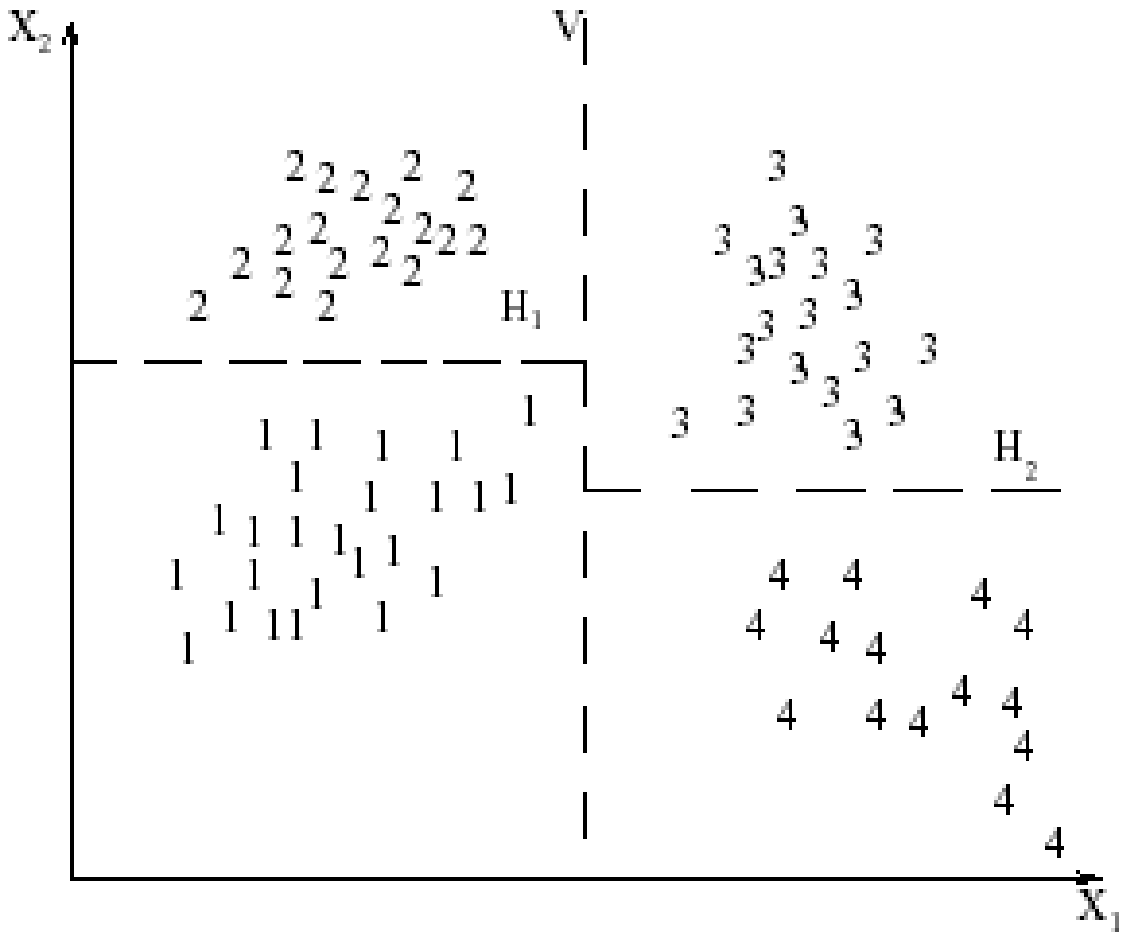
Outline

- **Definition**
- Similarities Measure
- Clustering Algorithm

Definition

- Unsupervised Learning Mode: partition a set of **unlabeled** objects into groups or clusters[C.D. Manning & Hinrich Schutze]
- Another viewpoint: Clustering is not a Learning Problem. It's an Optimization Problem. Give a distance metric, devise an algorithm that splits the data in such a way the optimizes some criteria.




X2
V
H1
H2
X1

Clustering

- Each data point is assumed to be an n-dimensional vector, represented as a column vector:

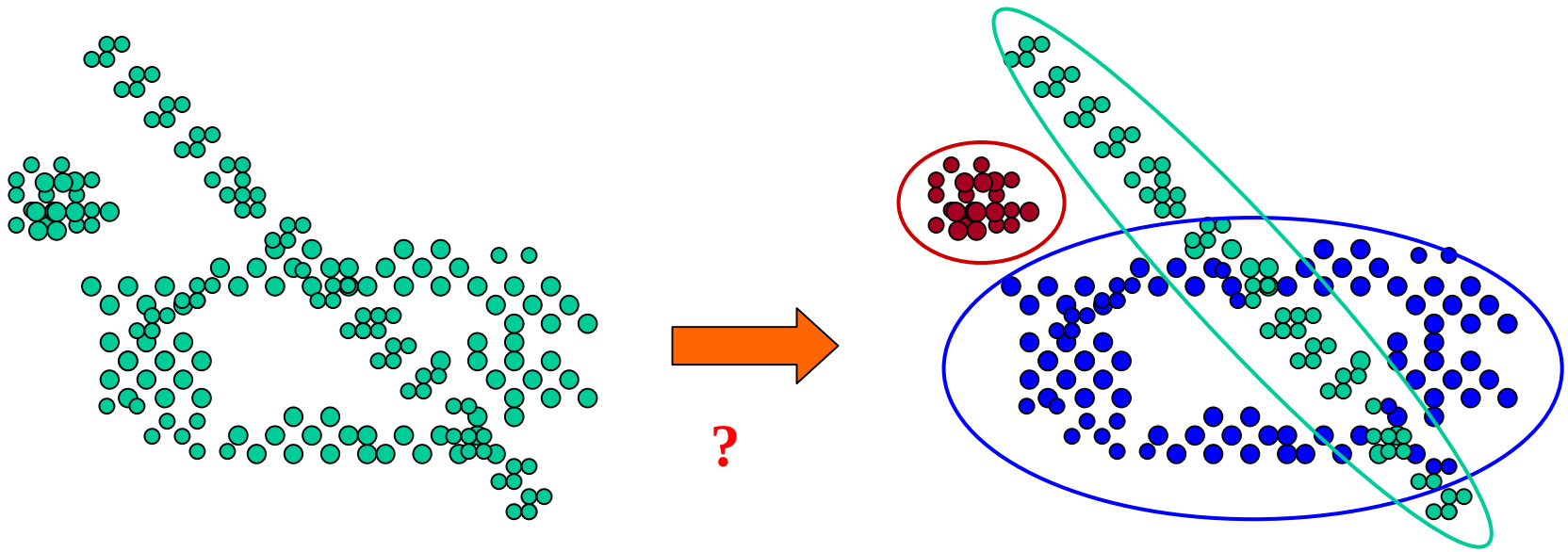
$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$$

• A cluster is a set of points which are **alike**, and points in different clusters are **not alike**.
• Finding groups of points such that the points in a group will be similar (or related) to one another and different from (or unrelated to) the points in other groups

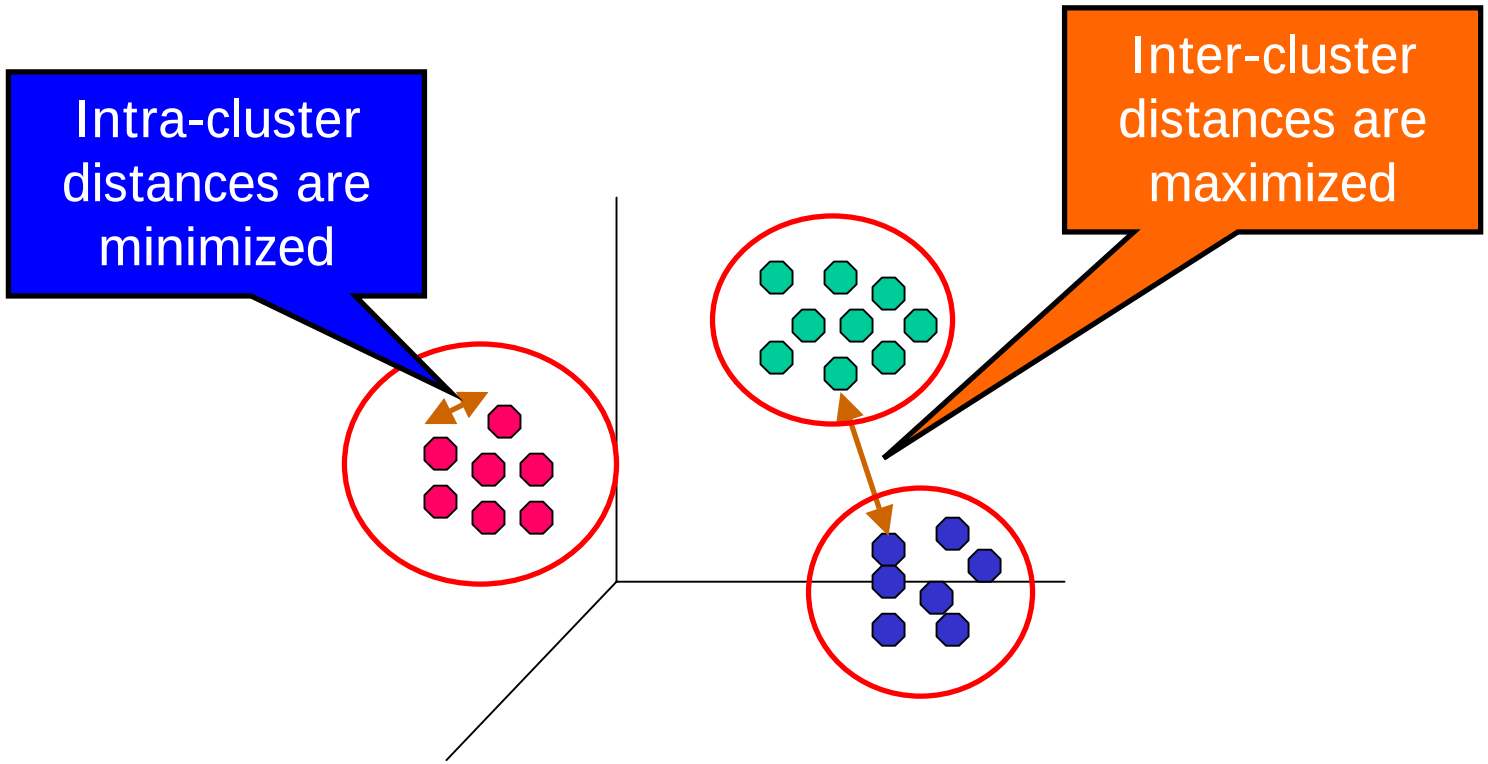
Clustering

- Clusters may be described as connected regions of a multi dimensional space containing a relatively **high density** of points, separated from other regions by regions containing a **low density** of points.

How to separate these points Into different group?



Intra vs. Inter-cluster



Data Representation

- A set of data points(objects): $\mathbf{D} = \{ d_1, d_2, ..., d_m \}$
- Each data point is represented as:

$$d_i = \{ x_{i1}, x_{i2}, \ldots, x_{in} \}$$

- The dot product (or inner product) between two point d_i and d_j is:

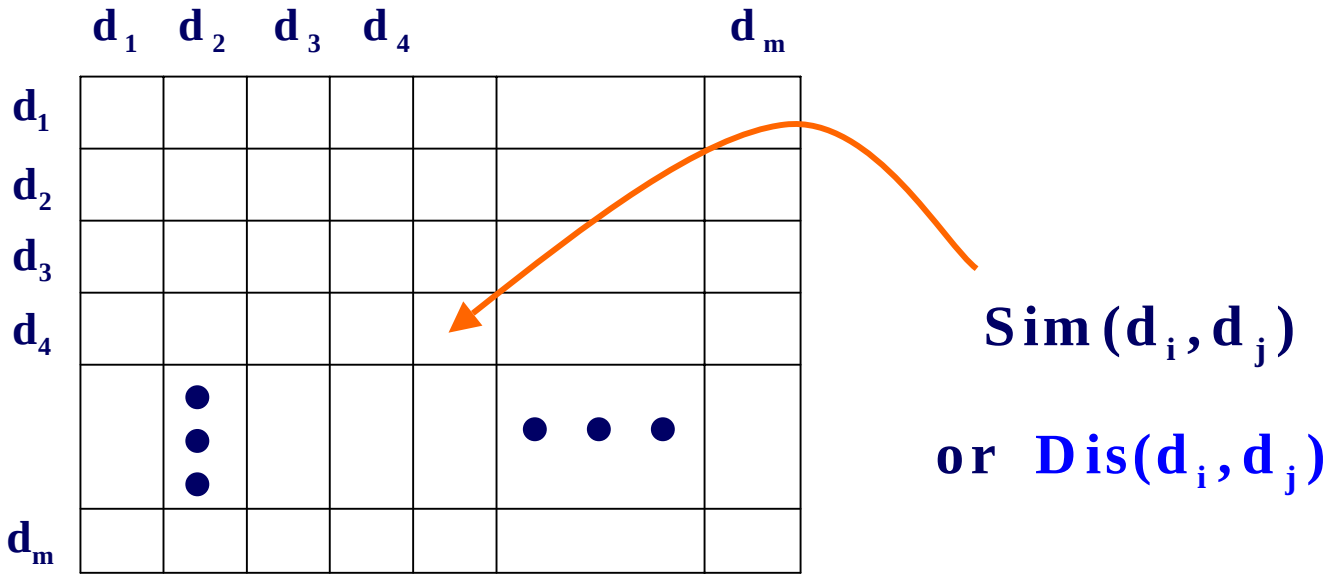
$$d_i \bullet d_j = \sum_{k=1}^{n} x_{ik} x_{jk}$$

Outline

- Definition
- **Similarities Measure**
- Clustering Algorithm

Similarity Measures

- Given a matrix of similarities(distances) between all pairs of data points.
- The input can be described as:



Similarity Measures

- The larger the similarity, the less the distance.
- Distance between two points is usually used to measure the **Similarity**.
- A distance measure (metric) is a function: $\mathbf{dis : R^n \times R^n \rightarrow R}$

in which,

1) $\mathbf{dis(x,y) \geq 0,\ dis(x,y) = 0 \Leftrightarrow x = y}$
2) $\mathbf{dis(x,y) + dis(y,z) \geq dis(x,z)}$ (Triangle Inequality)
3) $\mathbf{dis(x,y) = dis(y,x)}$ (Symmetry)

- For the purpose of clustering, sometimes the Triangle Inequality and Symmetry is not required to be a metric.

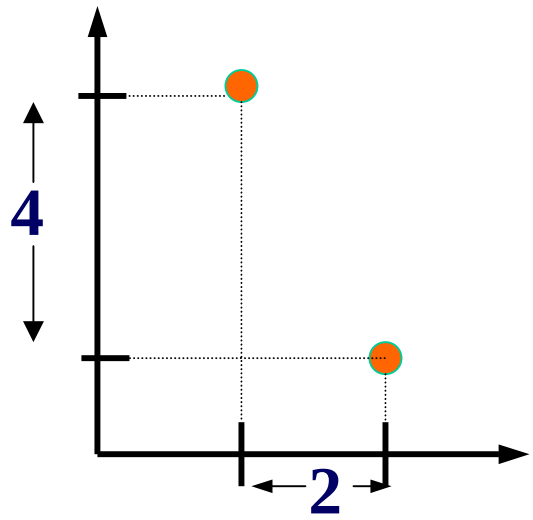
Typical Distance Measures

- **Euclidean Distance:**

$$dis(x,y) = \sqrt{(x-y)^2} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- **Manhattan Distance:** $dis(x,y) = |x-y| = \sum_{i=1}^{d}|x_i - y_i|$

- **Sup Distance:** $dis(x,y) = \max_{1\le i\le d}|x_i - y_i|$



Euclidean $= (4^2 + 2^2)^{1/2} = 4.47$

Manhattan $: 4 + 2 = 6$

Sup $= \text{Max}(4,2) = 4$

Distance Measures

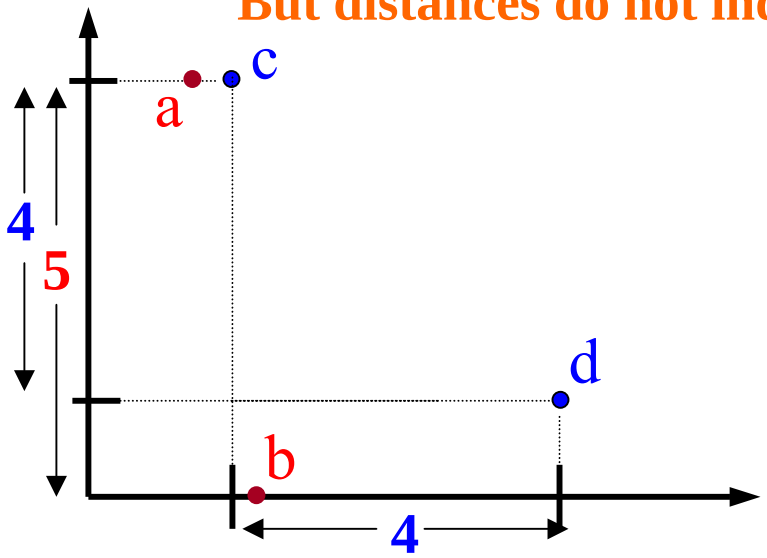
- **Sup Distance < Euclidean Distance <Manhattan Distance:**

$$L_1 = |x - y| = \sum_{i=1}^{n} |x_i - y_i| \qquad \text{(Manhattan Distance)}$$

$$L_2 = \sqrt{(x - y)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \qquad \text{(Euclidean Distance)}$$

$$L_\infty = \max_{1 \le i \le n} |x_i - y_i| \qquad \text{(Sup Distance)}$$

But distances do not induce same order on pairs of points



$$L_\infty(a,b) = 5$$

$$L_2(a,b) = (5^2 + \varepsilon^2)^{1/2} = 5 + \varepsilon$$

$$L_\infty(c,d) = 4$$

$$L_2(c,d) = (4^2 + 4^2)^{1/2} = 4\sqrt{2} = 5.66$$

$$L_\infty(c,d) < L_\infty(a,b)$$

$$L_2(c,d) > L_2(a,b)$$

Distance Measures

- In general, (L_p Norm):

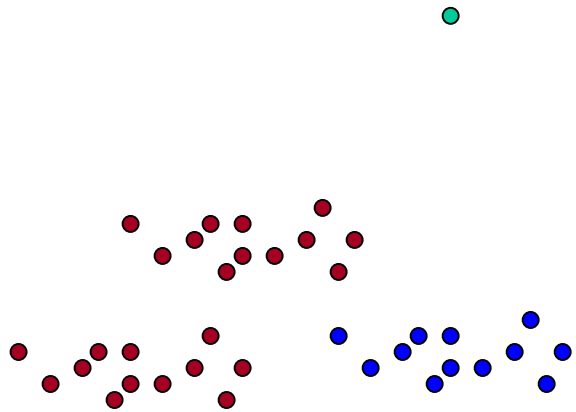
$$L_p(\mathbf{x},\mathbf{y}) = \sqrt[p]{\sum_{i=1}^{n} \left| x_i - y_i \right|^p}$$

$$= \| x - y \|_p$$

Distance Measures

- Sometime it is useful to define distance between a data point **x** and a set **A** of points:

$$\mathbf{dis(x, A) = \frac{1}{|A|} \sum_{y \in A} dis(x, y)}$$



- distance between sets of points A, B:

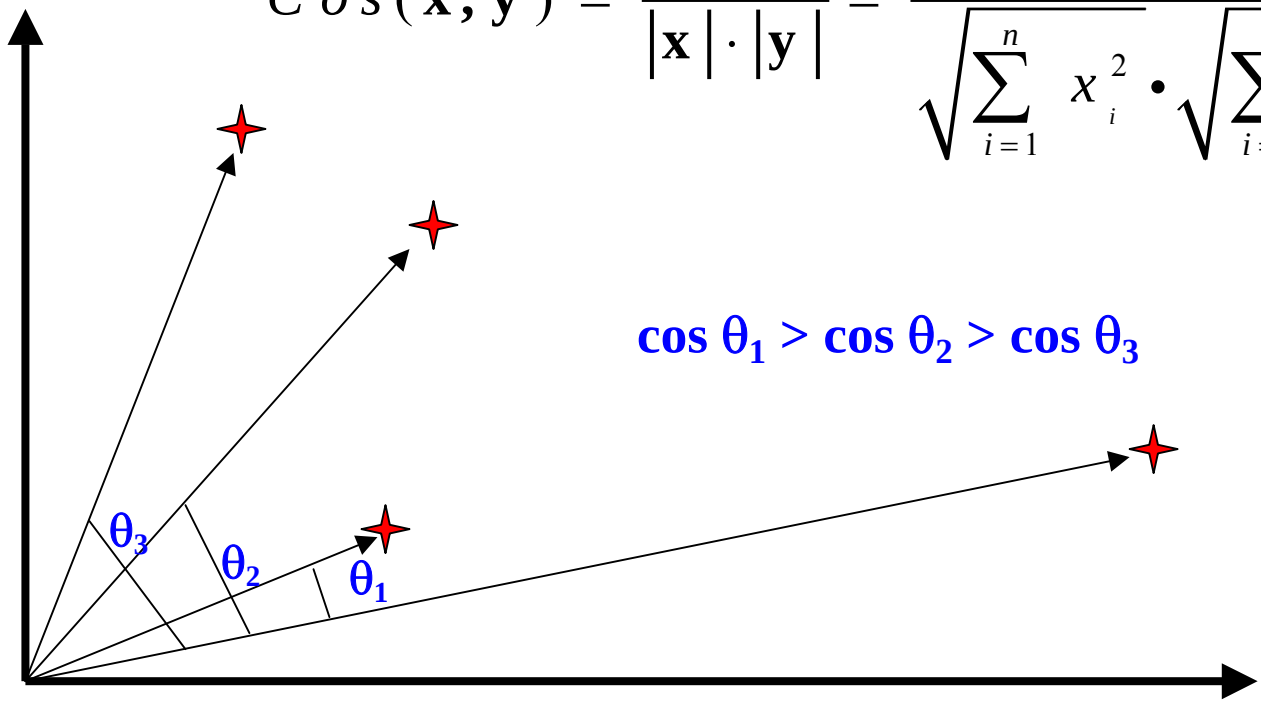
$$\mathbf{dis(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} dis(x, y)}$$

- There are many other ways to do it; may depend on the application.

Others (Similarity)

- Cosine of the angle of two vector

$$Cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}| \cdot |\mathbf{y}|} = \frac{\sum_{i=1}^{n} (x_i \bullet y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2} \bullet \sqrt{\sum_{i=1}^{n} y_i^2}}$$



Other Distance

- Kullback-Leibler divergence(KL-divergence)

$$P_1 = < P_1(x_1), P_1(x_2), \ldots, P_1(x_n) >$$
$$P_2 = < P_2(x_1), P_2(x_2), \ldots, P_2(x_n) >$$
$$KL(P_1, P_2) = \sum_{x \in X} P_1(x) \log \frac{P_1(x)}{P_2(x)}$$

- KL-divergence is not a true metric since it not symmetric and does not obey triangle inequality

$$\frac{1}{2}(KL(P_1, P_2) + KL(P_1, P_2))$$

Outline

- Definition
- Similarities Measure
- **Clustering Algorithm**

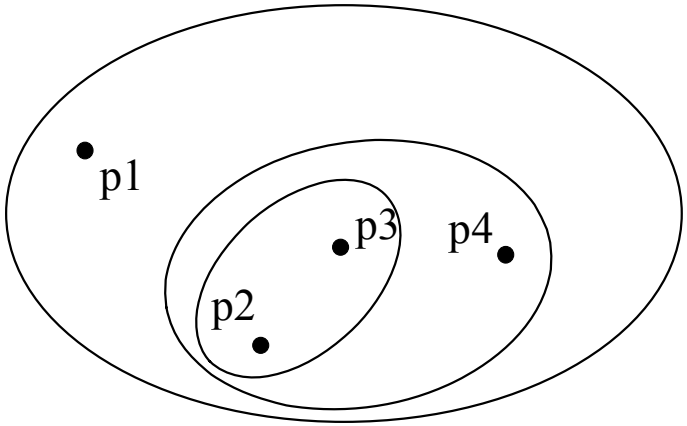
Hard/Soft clustering

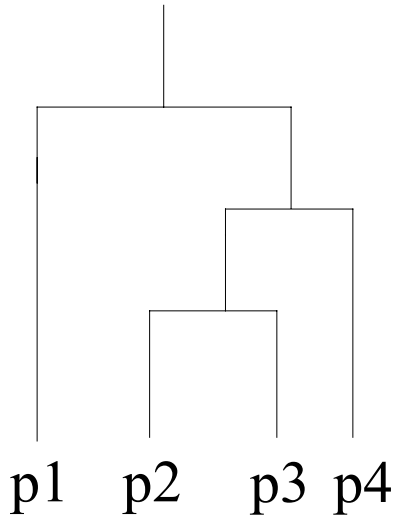
- Hard Clustering
  - Each point belongs to just one cluster
- Soft clustering
  - Allowing degrees of membership and membership in multiple clusters
    - Probabilistic framework, a point x has a probability distribution $P(.|x)$ over cluster c_j: $P(c_j|x)$

Hierarchical/Partitional Clustering

- Partitional Clustering
  - A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Hierarchical clustering
  - A set of nested clusters organized as a hierarchical tree

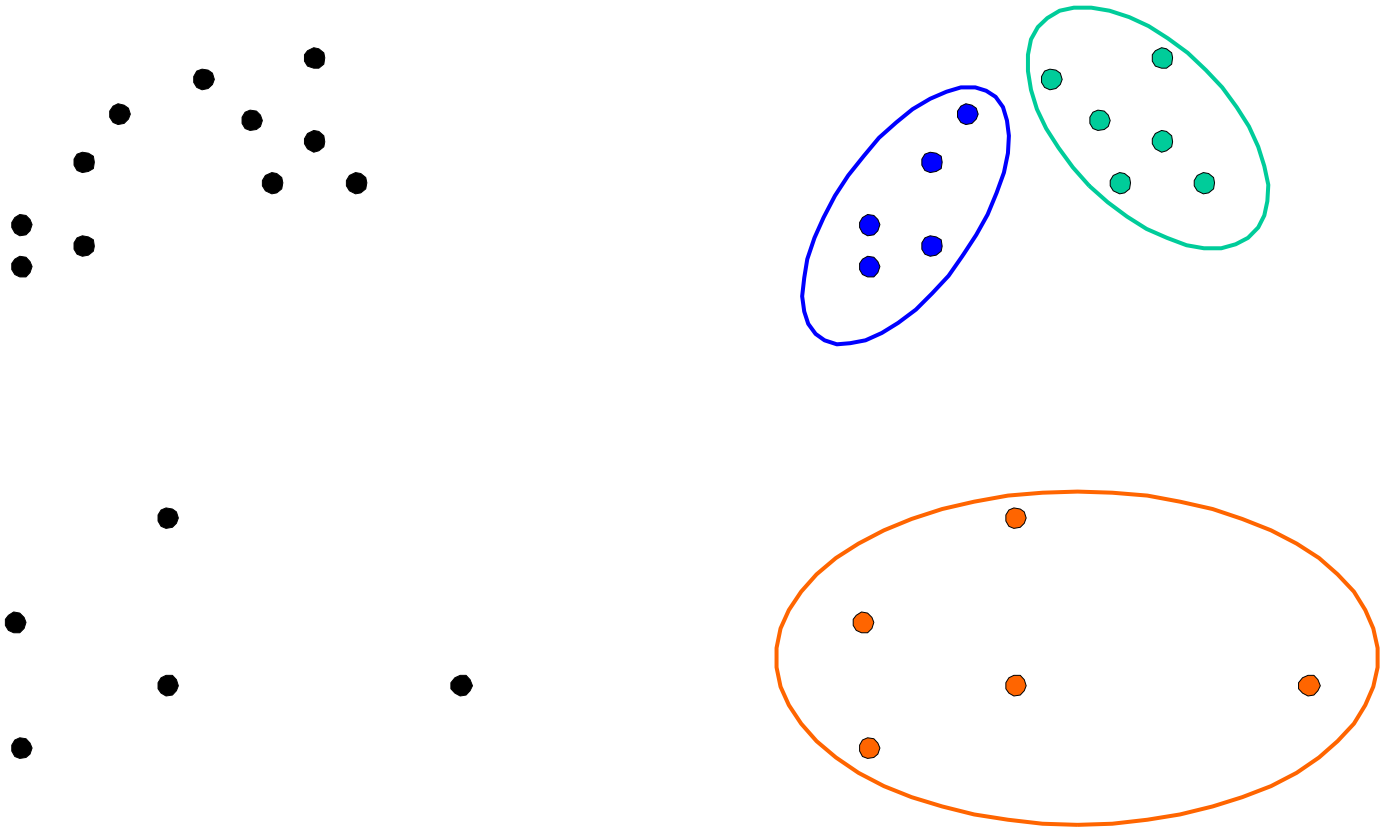
Hierarchical Clustering





Hierarchical Clustering **Dendrogram**

Partitional Clustering



Original Points

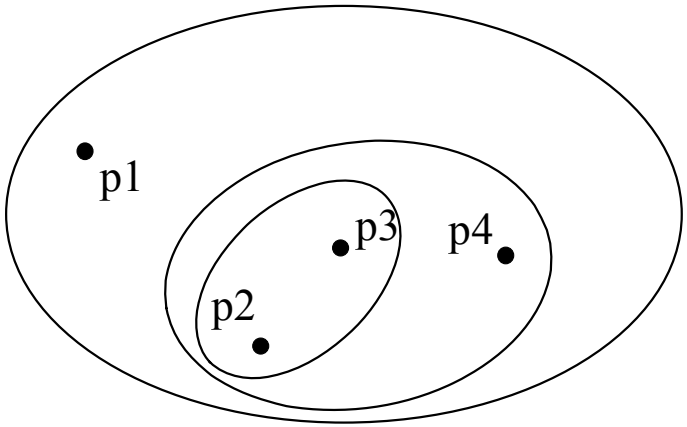
A Partitional Clustering

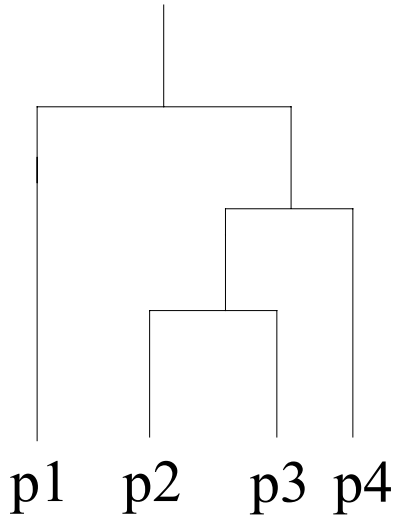
Clustering Algorithms

- ✓ **Hierarchical clustering**
- K-means
- Graph based clustering

Hierarchical Clustering

- Produces a set of nested clusters visualized as a dendrogram

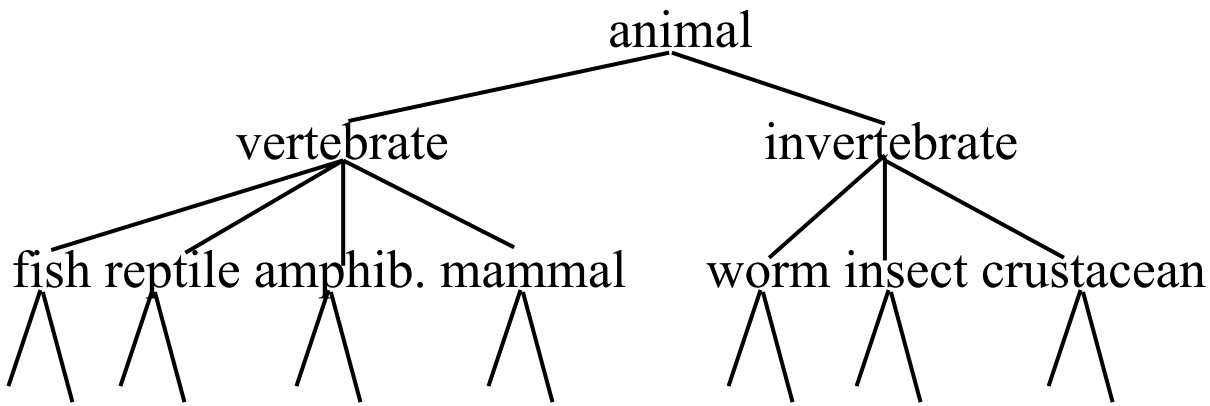




Hierarchical Clustering **Dendrogram**

An Example

- Build a tree-based hierarchical taxonomy from a set of unlabeled objects



Hierarchical Clustering

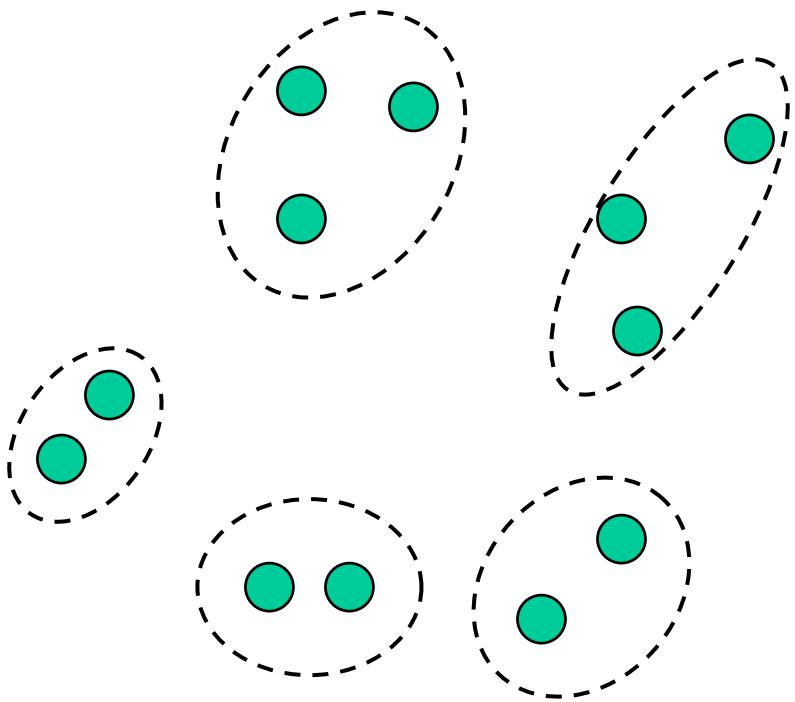
- Two main types of hierarchical clustering
  - Aglommerative Clustering (*bottom-up*)
    - start with each example in its own cluster and iteratively combine them to form larger and larger clusters.
  - Divisive Clustering (*partitional*, *top-down*)
    - separate all examples immediately into clusters.
- similarity or distance matrix
  - Merge or split one cluster at a time

Aglommerative Clustering

- More popular hierarchical clustering technique :
- Basic algorithm is straightforward
  1. Compute the **proximity** matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the **proximity** matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix.

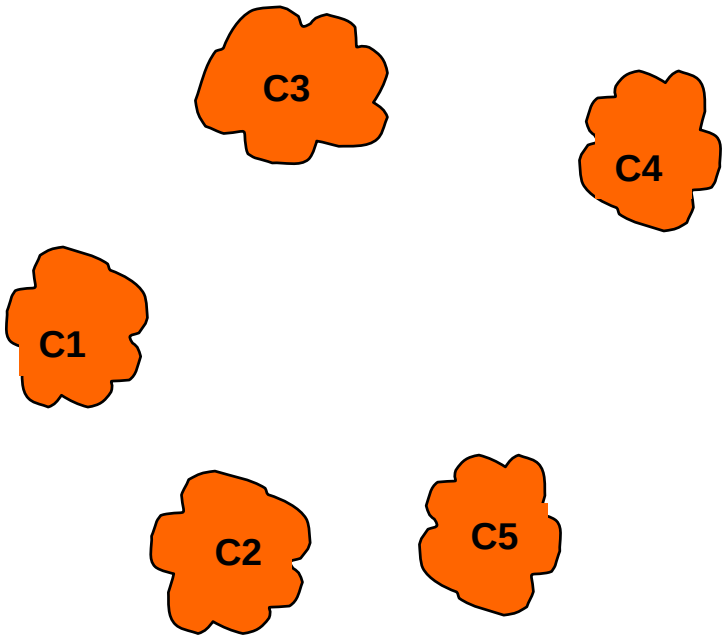


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

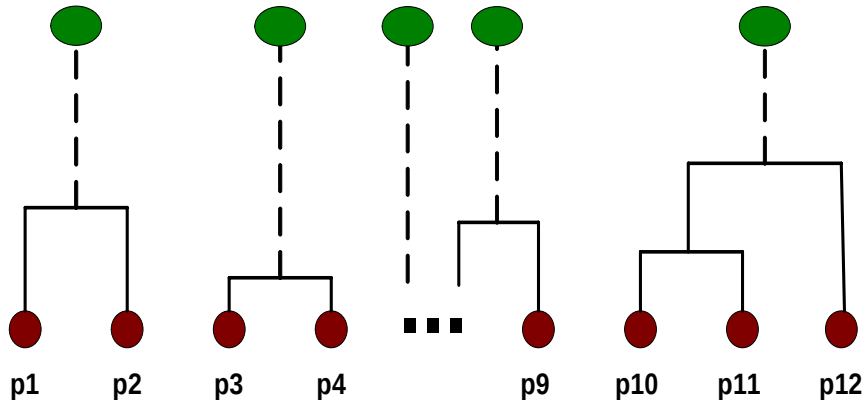
Intermediate Situation

- After some merging steps, we have some clusters



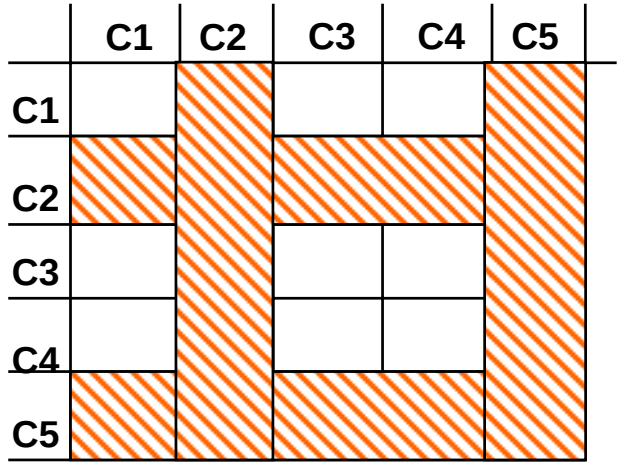
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

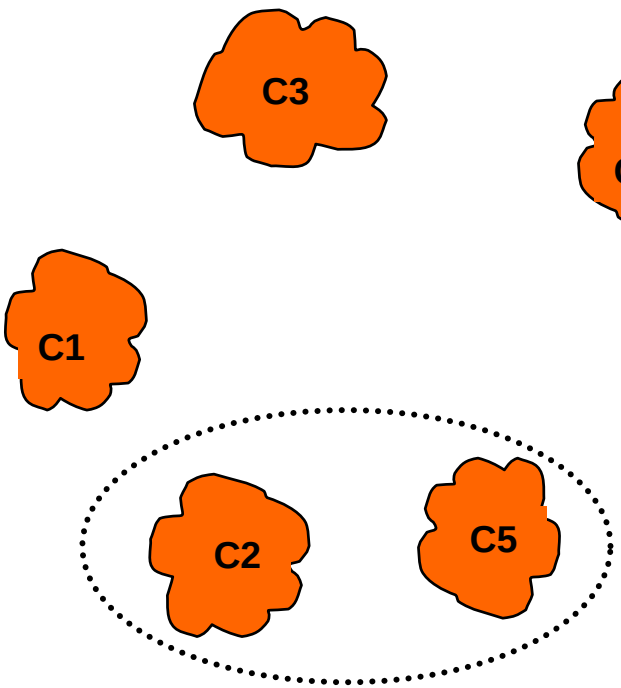


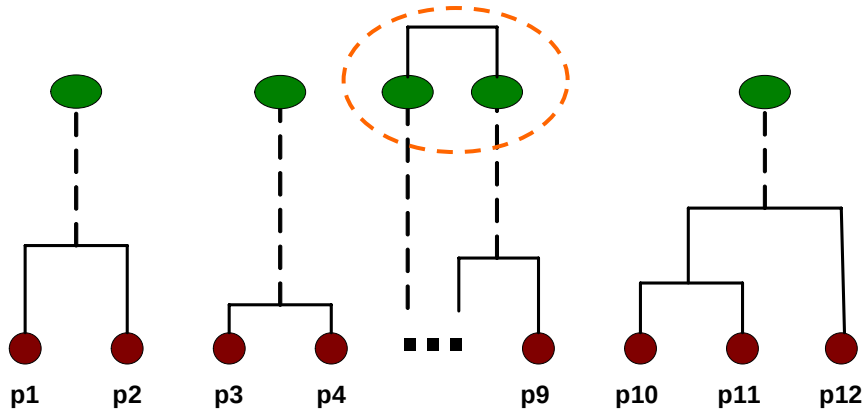
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



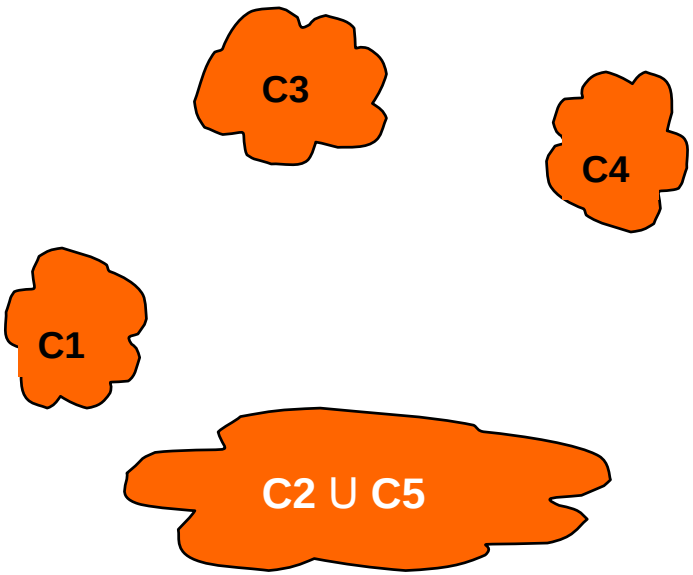
Proximity Matrix





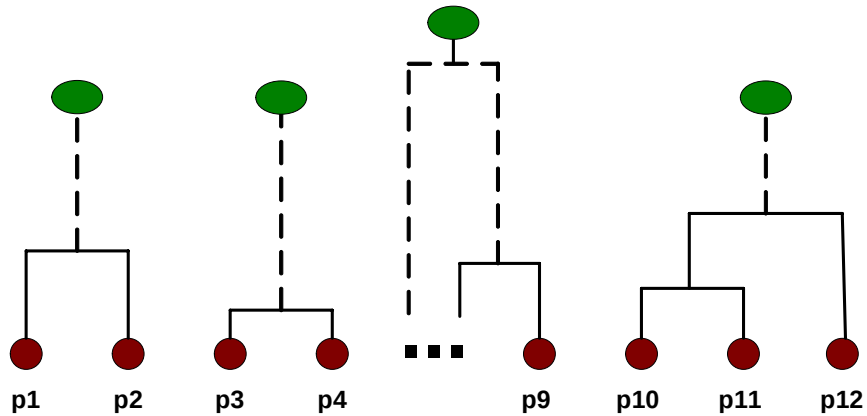
After Merging

- How do we update the proximity matrix?

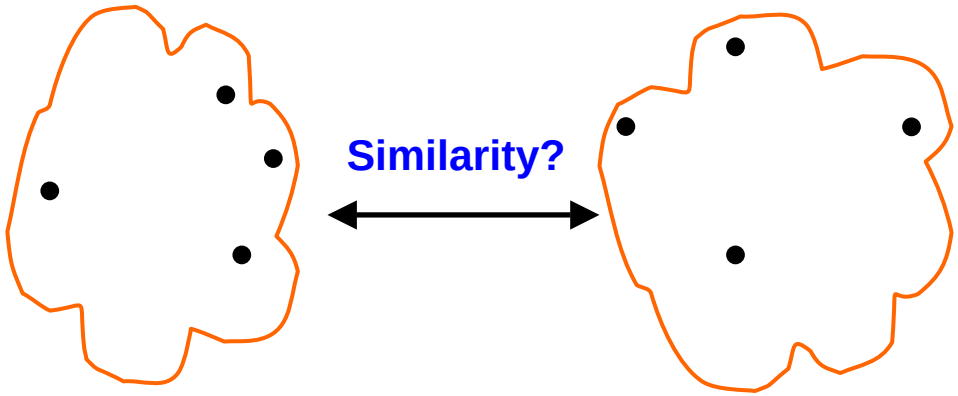


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

Cluster Similarity

- Assume a similarity function that determines the similarity of two points: *sim*(*x*,*y*).
- How to compute similarity of two clusters each possibly containing multiple points?
  - Single Link: Similarity of two most similar members.
  - Complete Link: Similarity of two least similar members.
  - Group Average: Average similarity between members.
  - Distance Between Centroids

Single Link Agglomerative Clustering

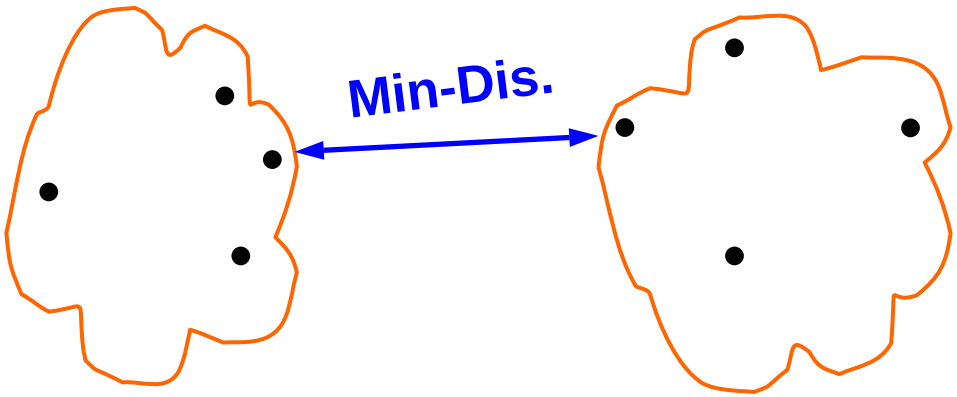
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points, i.e., by one link in the proximity graph.

$$sim(c_i, c_j) = \max_{x \in c_i, y \in c_j} sim(x, y)$$

$$or \quad dis(c_i, c_j) = \min_{x \in c_i, y \in c_j} dis(x, y)$$

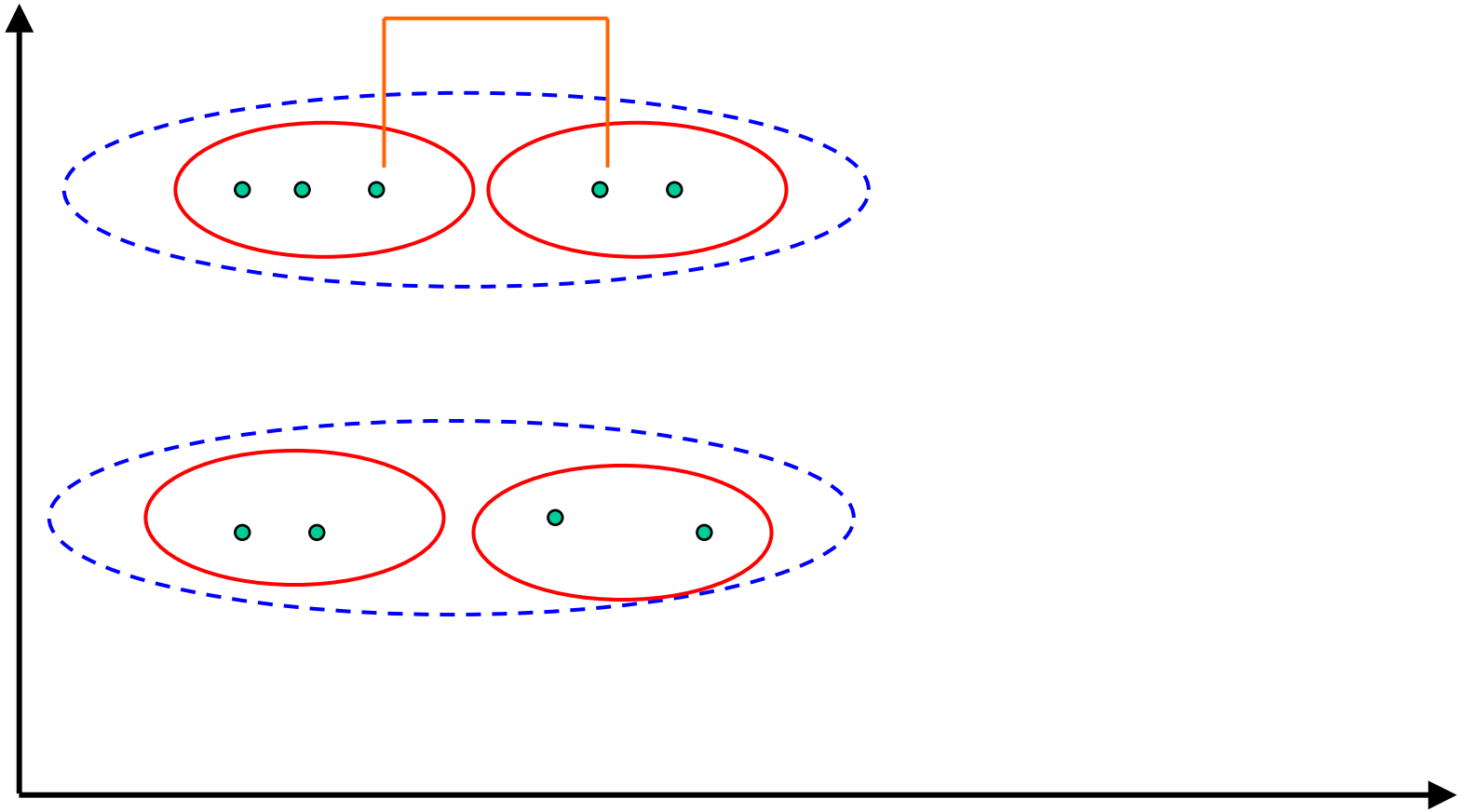
- Can result in "straggly" (long and thin) clusters due to *chaining effect*.

Inter-Cluster Similarity



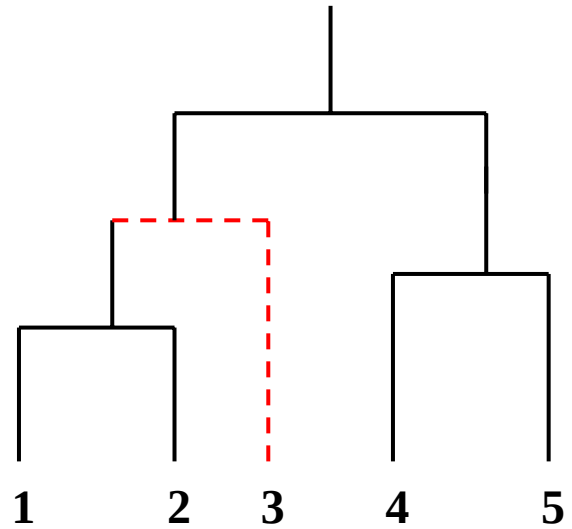
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



Max-Sim. (Single Link)

	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



0.7 > 0.4

Complete Link Agglomerative Clustering

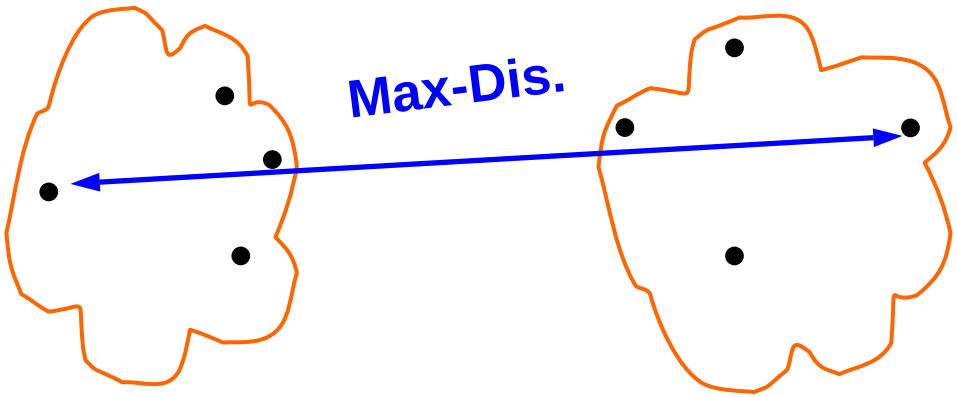
- Use minimum similarity of pairs: Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

$$or \quad dis(c_i, c_j) = \max_{x \in c_i, y \in c_j} dis(x, y)$$

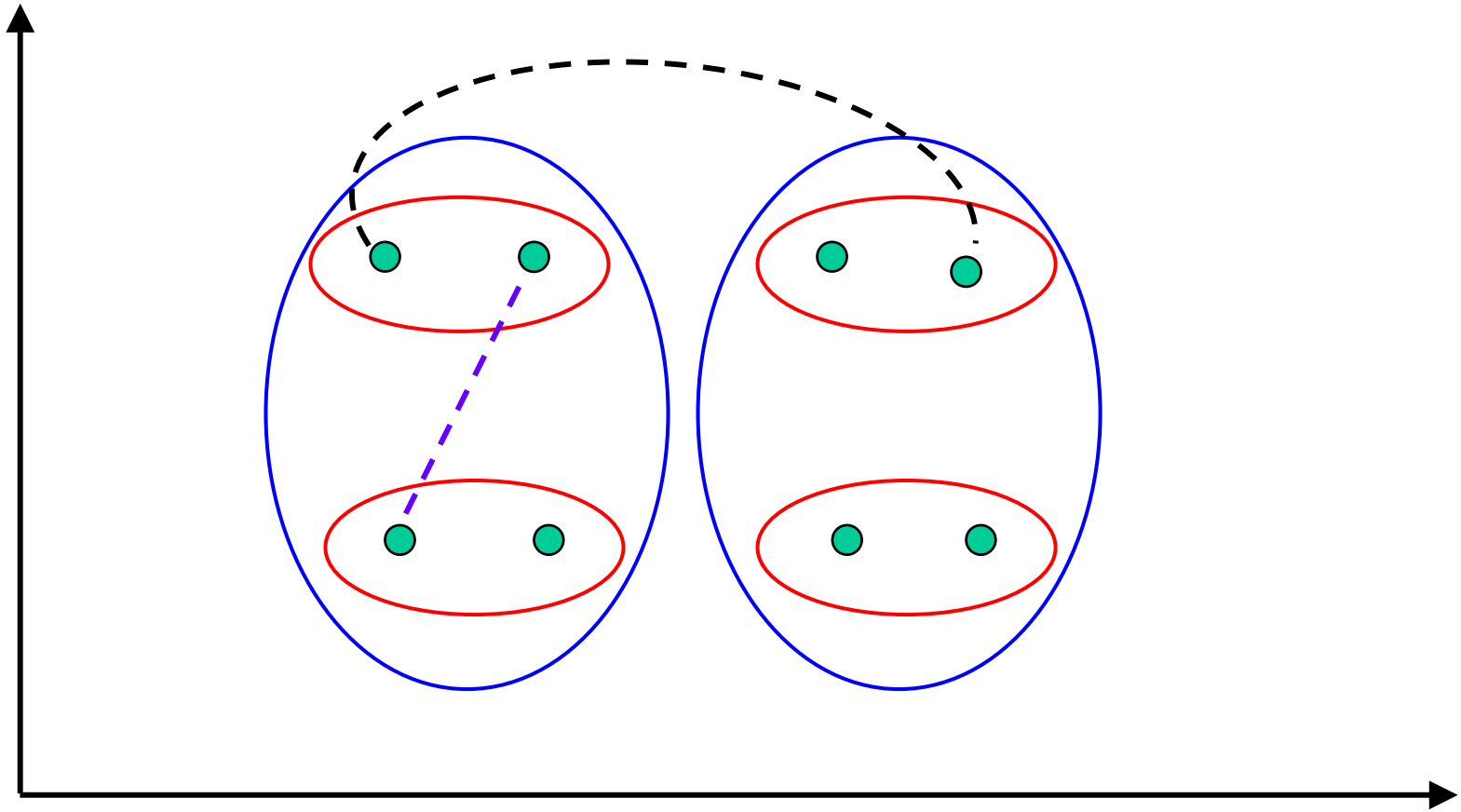
- Makes more "tight," spherical clusters that are typically preferable.

Inter-Cluster Similarity



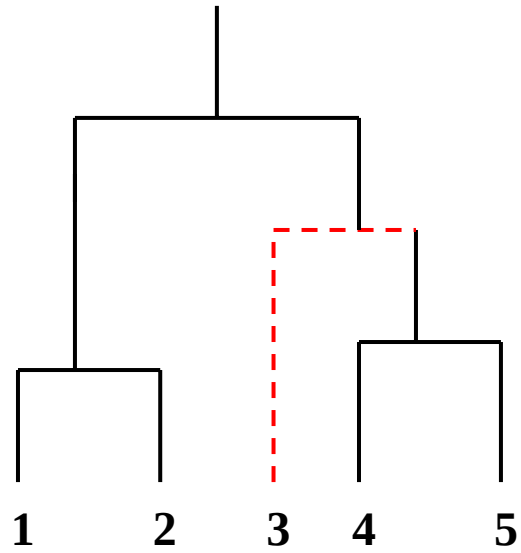
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



MAX-Dis. (Complete Linkage)

	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



0.3 > 0.1

Computing Cluster Similarity

- After merging c_i and c_j, the similarity of the resulting cluster to any other cluster, c_k, can be computed by:
  - Single Link:

$$sim((c_i \cup c_j), c_k) = \max(sim(c_i, c_k), sim(c_j, c_k))$$

  - Complete Link:

$$sim((c_i \cup c_j), c_k) = \min(sim(c_i, c_k), sim(c_j, c_k))$$

Group Average Agglomerative Clustering

- Use average similarity across all pairs within the merged cluster to measure the similarity of two clusters.

$$sim(c_i, c_j) = \frac{1}{\left|c_i \cup c_j\right|\left(\left|c_i \cup c_j\right| - 1\right)} \sum_{x \in (c_i \cup c_j)} \sum_{y \in (c_i \cup c_j): y \neq x} sim(x, y)$$

- Suppose: complete graph with n vertexes having n*(n-1)/2 edges.

Computing Group Average Similarity

- Assume cosine similarity and normalized vectors with unit length.
- Always maintain sum of vectors in each cluster.

$$\vec{s}(c_j) = \sum_{\vec{x} \in c_j} \vec{x}$$

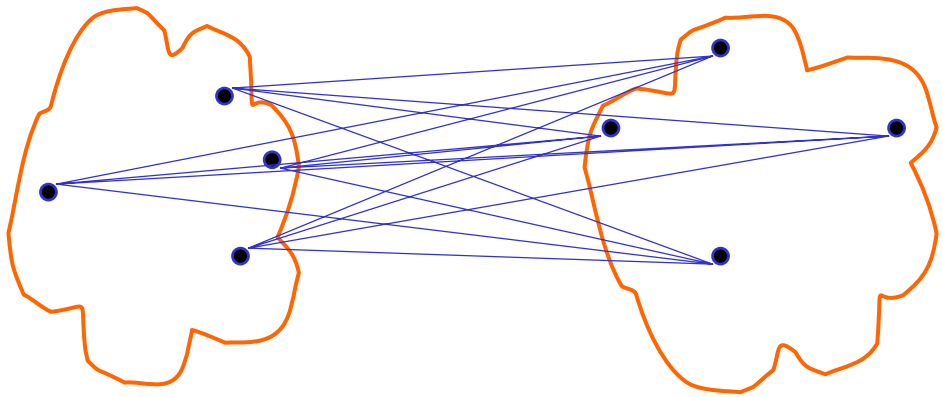
- Compute similarity of clusters in constant time:

$$sim(c_i, c_j) = \frac{(\vec{s}(c_i) + \vec{s}(c_j)) \bullet (\vec{s}(c_i) + \vec{s}(c_j)) - (|c_i| + |c_j|)}{(|c_i| + |c_j|)(|c_i| + |c_j| - 1)}$$

Group Average: another computing

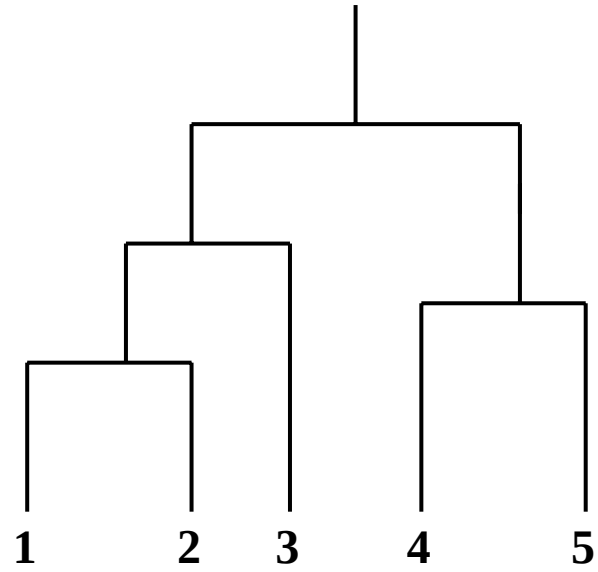
- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$



Group Average

	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



Group Average

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

Complexity on Hierarchical Clustering

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

Clustering Algorithms

- Hierarchical clustering
- ✓ **K-means**
- Graph based clustering

K-Means Clustering

- Randomly choose *k* points as *seeds(centroids)*, *k* must be specified.
- Form initial clusters based on these **centroids**.
- Iterate, repeatedly assigning each point to different clusters according to the closest centroid.
- Recompute the centroid of each cluster.
- Stop when clustering converges or after a fixed number of iterations.

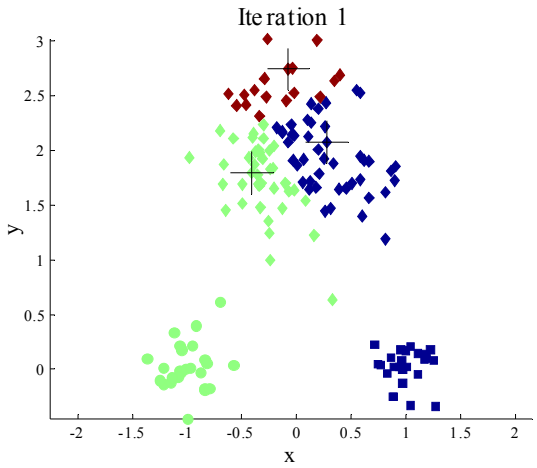
K-Means

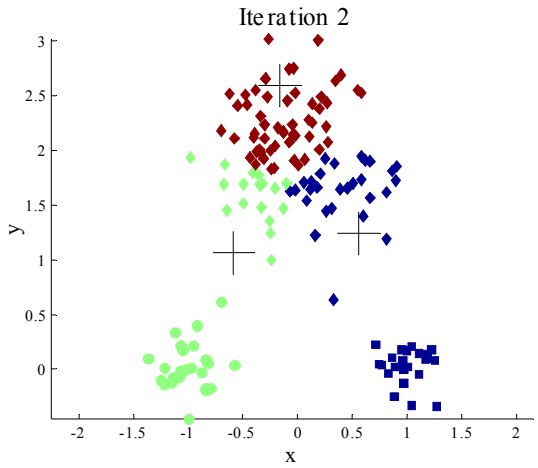
- *Centroids*, *center of gravity*, or mean of points in a cluster, ***c***, *is calculated as follows*:

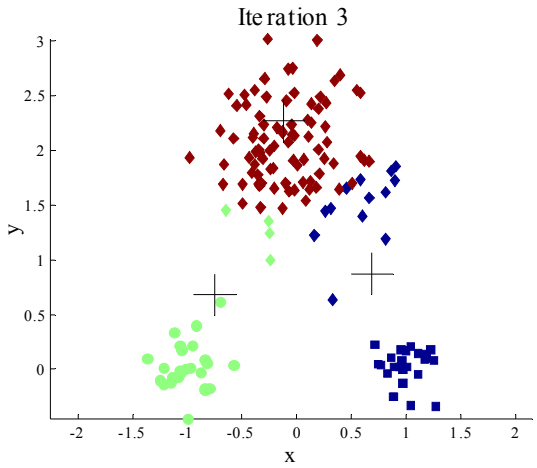
$$\mu(\mathbf{c}) = \frac{1}{|c|}\sum_{\mathbf{x}\in c}\mathbf{x}$$

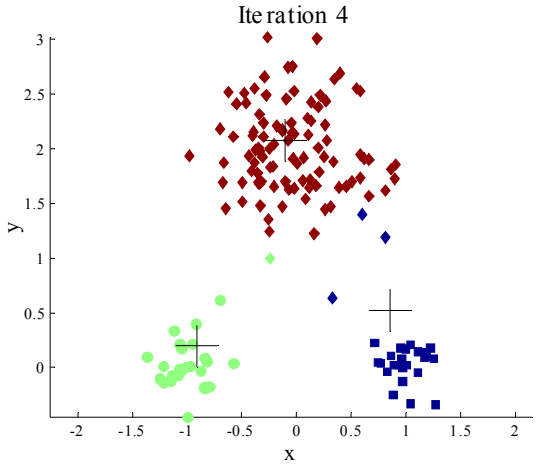
- Reassignment of instances to clusters is based on distance to the current cluster centroids.

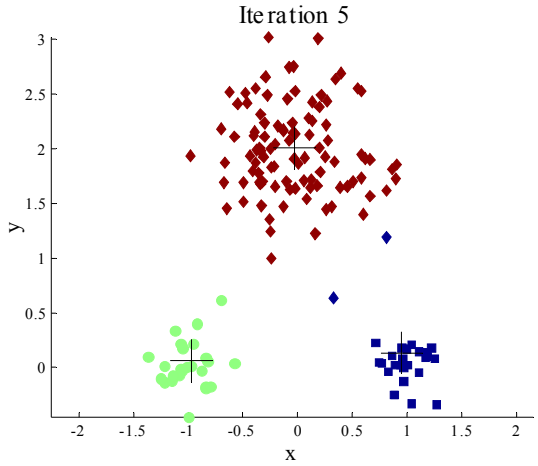
Example

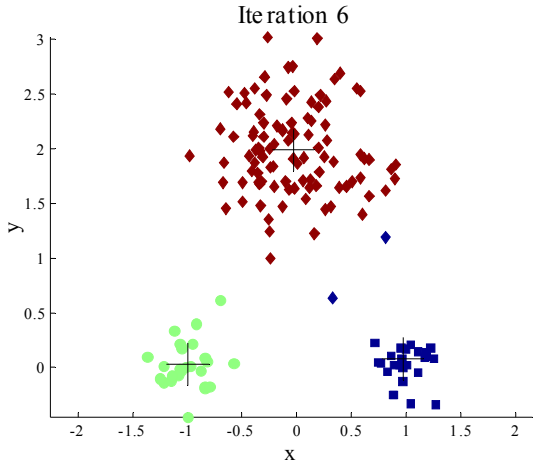

Iteration 1
x
y


Iteration 2
x
y


Iteration 3
x
y


Iteration 4
x
y


Iteration 5
x
y


Iteration 6
x
y

K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} \mathbf{dis}^2(\mu_i, x)$$

where, x is a point in cluster C_i, μ_i is the centroid of C_i

  - Given two clusters, we can choose the one with the smaller error
  - One easy way to reduce SSE is to increase K, the number of clusters

Limitation of K-Means Algorithm

- The number of clusters (K) is difficult to determine.

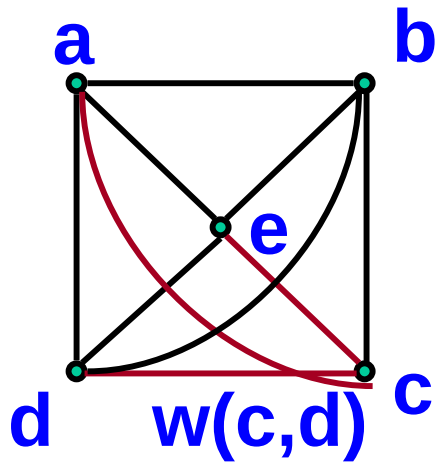
Clustering Algorithms

- Hierarchical clustering
- K-means
- ✓ **Graph based clustering**

Weighted graph

- A weighted graph includes a set of vertices, edges, and the weights corresponding to the edges.
  - G = (V,W,E)

	a	b	c	d	e
a	0	6	8	2	7
b	6	0	2	5	3
c	8	2	0	10	9
d	2	5	10	0	4
e	7	3	9	4	0

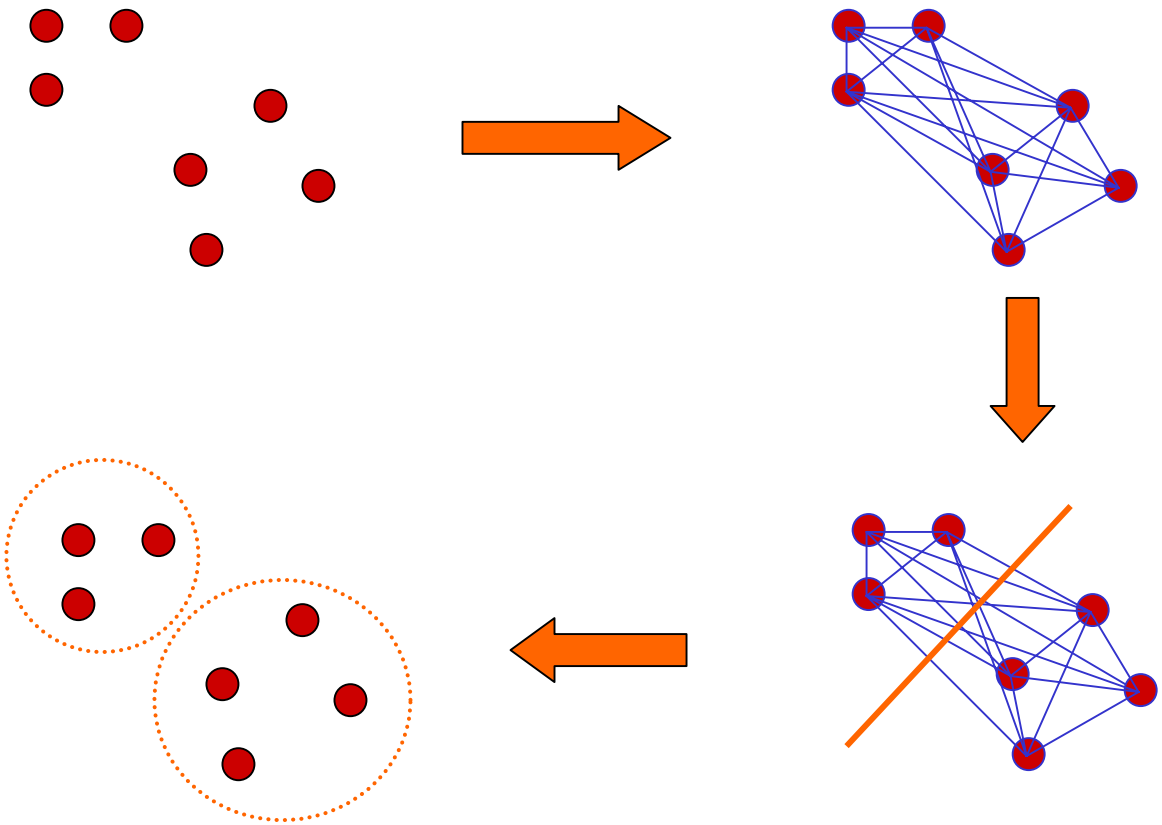


- each data point is represented as a vertex in a weighted graph, where the weights (i.e length) on the edges between elements are large if the two points are similar and small if they are not.

Graph based clustering

- Seek a partition of the set of vertices V into disjoint sets $\mathbf{V_1, V_2, ..., V_k}$ where: some measure of the similarity among the vertices in each V_i is large , and across sets V_i and V_j is small.
- Clustering becomes a graph cut problem: cut the graph into connected components with relatively large interior weights by cutting edges with relatively small weights.

Example



General Graph Method

- General Method: Decompose the graph into connected component by identifying and deleting inconsistent ("bad") edges.

Algorithm:

  - **Construct the Maximum Spanning Tree (MST);**
  - **Identify inconsistent edges in the MST;**
  - **Remove the inconsistent edges to form connected components and call them clusters.**

- Hierarchical clustering belongs to this method.

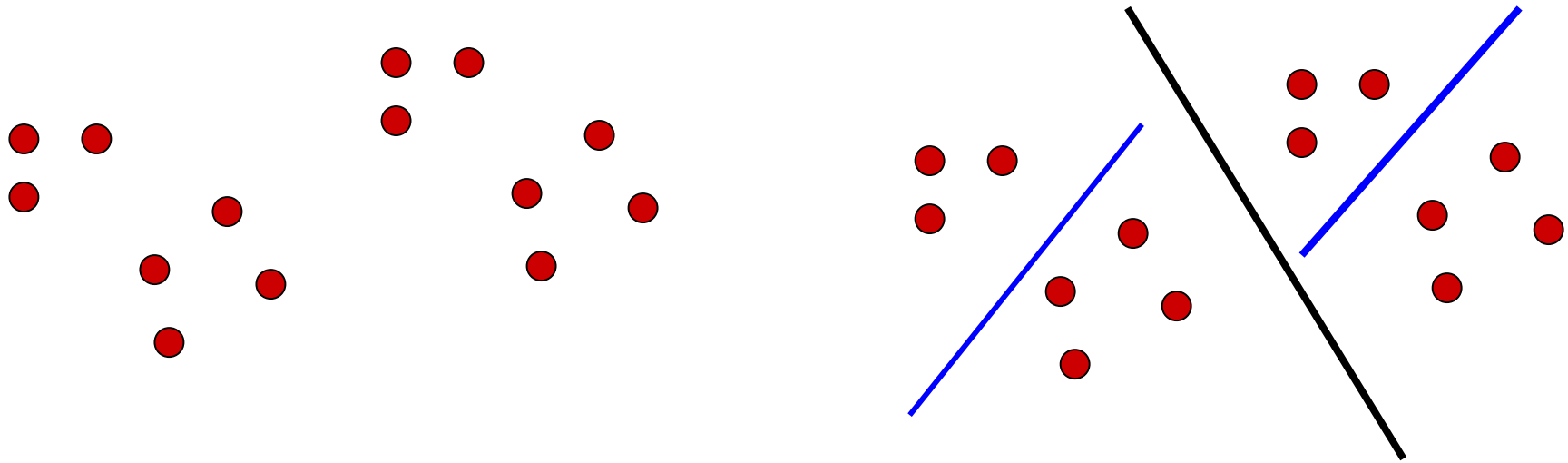
Graph Method

- MST and neighborhood approaches are very efficient but are based on local properties of the graph.
- Many applications needs a partition criterion that depends on global properties.
- A graph G=(V,W,E) can be partitioned into two disjoint sets A, B, with $A \cap B = \emptyset$.
- The degree of dissimilarity between these two sets can be computed as total weight of the edges that connect these two sets. In graph theoretic language, it is called the cut:

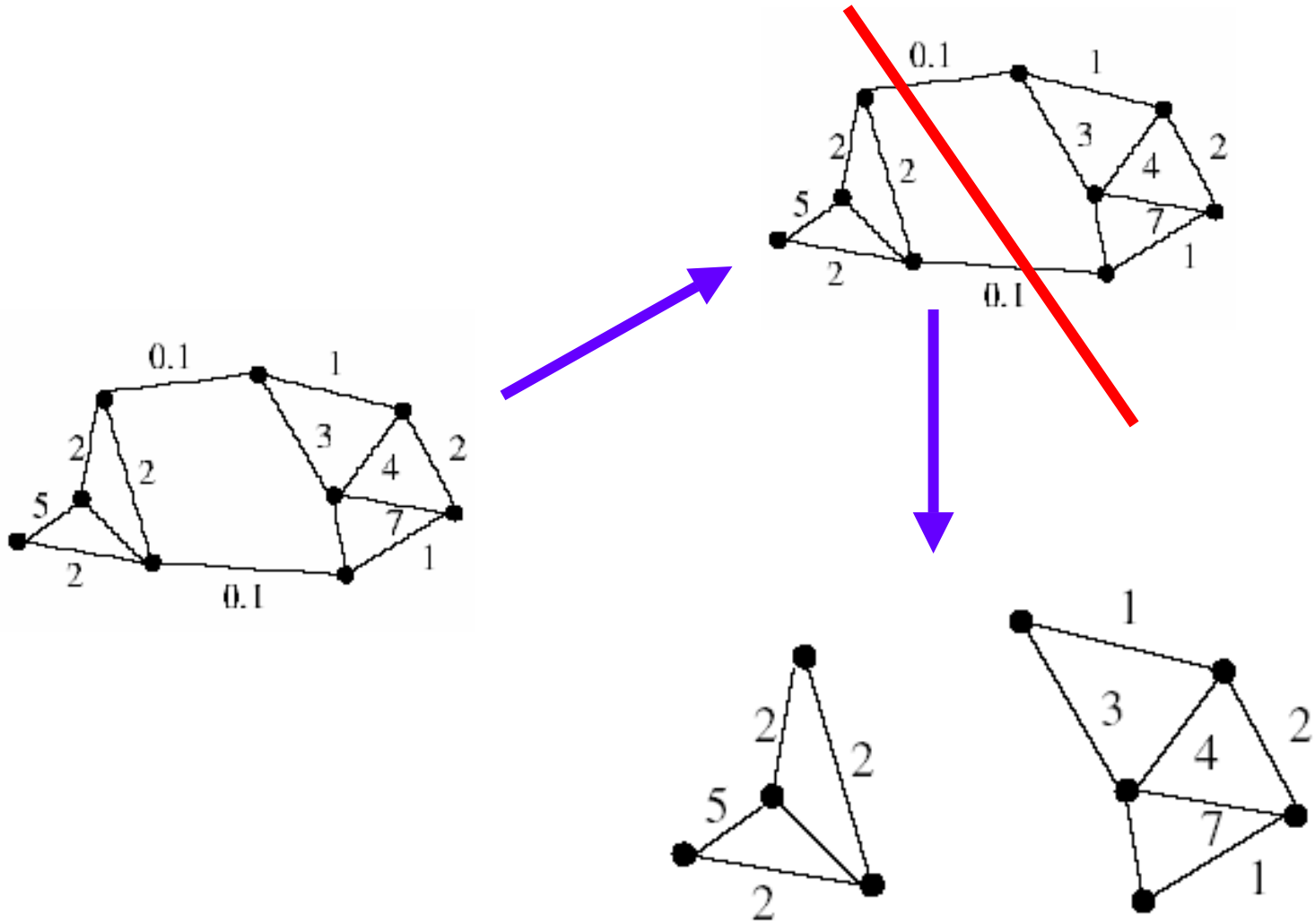
$$Cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

Cut Methods

- Partition into two clusters
- Use procedure recursively

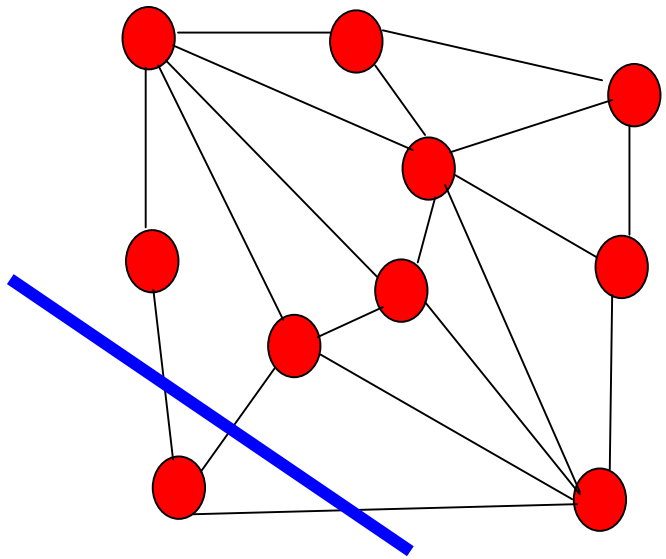


Illustration



Minimal Cut

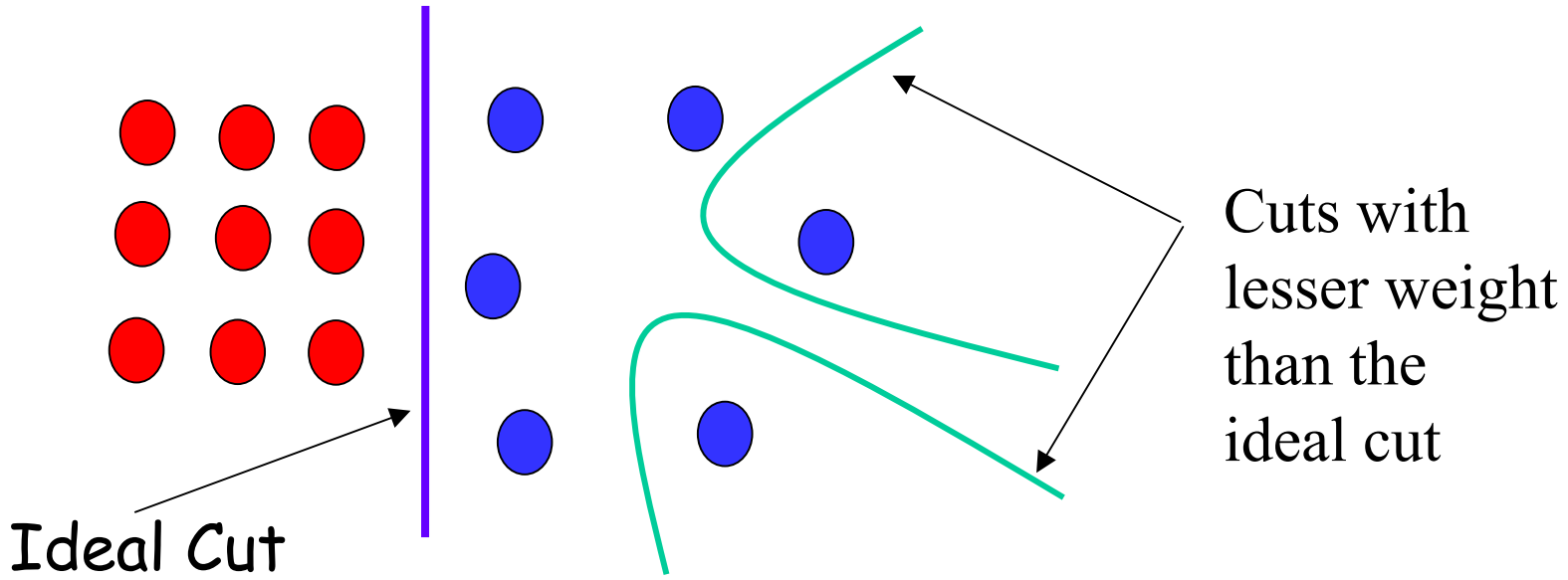
- A cut <A,B> of a graph *G* is the set of edges *S* that connects vertices of A to vertices of B.
- Minimum cut is the cut of minimum weight, where weight of cut <A,B> is given as



$$w(\langle A, B\rangle) = \sum_{x \in A, y \in B} w(x, y)$$

Drawbacks of Minimum Cut

- The cut value increases with the number of edges going across the partitions.



Normalized Cut

$$Ncuts(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

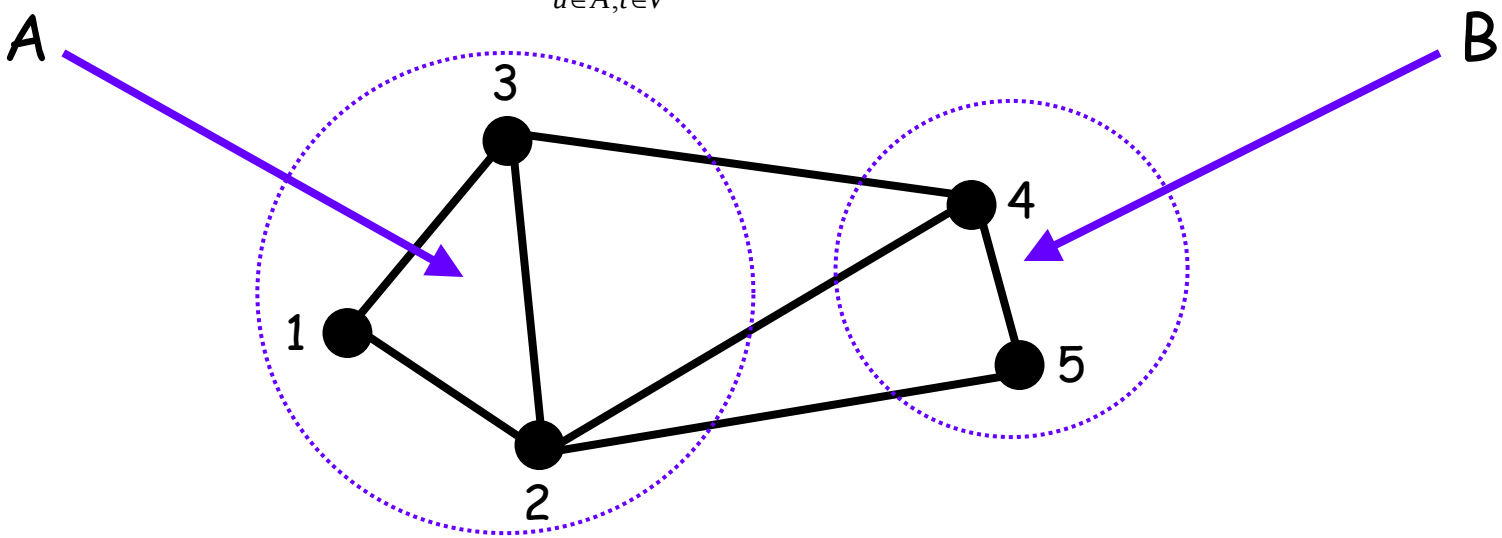
$$assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$

- This is a measure of dissociation **between clusters** in the graph.
- Assoc(A,V) is called **Normalization**

Illustration

$$Ncuts(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$



Cut(A,B) = w(3,4) + w(2,4) + w(2,5)

Assoc(A,V) = w(1,3) + w(1,2) + w(2,3) + w(3,4) + w(2,4) + w(2,5)

Assoc(B,V) = w(4,5) + w(3,4) + w(2,4) + w(2,5)

Normalized Cut: another measure

- The previous measure is to **minimize** the **dissociation** between **two clusters**;
- We also want to maximize the connection between all points **in the same cluster**, that is:

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$

$$assoc(A,A) = \sum_{u \in A, t \in A} w(u,t), \quad assoc(B,B) = \sum_{u \in B, t \in B} w(u,t)$$

$$assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$

Relation: the two measures

$$Ncut(A,B)$$

$$= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

$$= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)}$$

$$= 2 - (\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}) = 2 - Nassoc(A,B)$$

The minimal the Ncut(A,B), the maximal the Nassoc(A,B)

Normalized Cut: two criteria

- minimizing the disassociation measure and maximizing the within cluster association measure are related and can be satisfied simultaneously.
- The problem of Normalized Cut is NP hard.