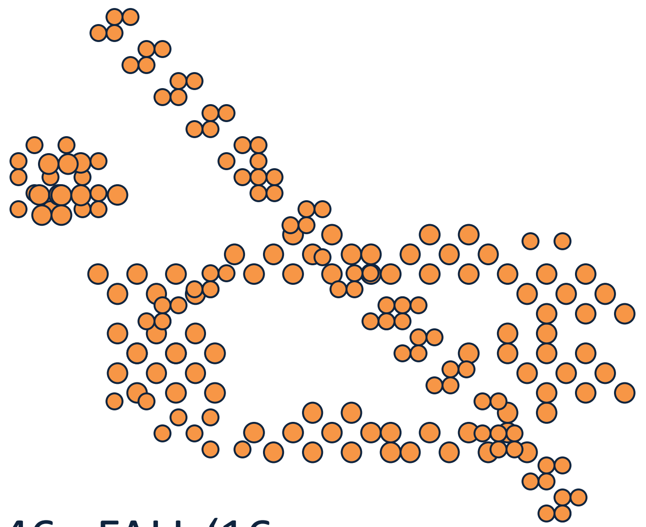
Administration

- Final Exam: Next Tuesday, 12/6 12:30, in class.
  - Material: Everything covered from the beginning of the semester
  - Format: Similar to mid-term; closed books
  - Review session on Thursday
- HW 7: Due on Thursday 12/1.
  - Only 24 hours of no additional time if needed.
- Final Projects:
  - Due on Tuesday, 12/13;
  - Follow Piazza and web site for submission insturctions
  - The final report should look like a conference paper.
  - A report guideline is available from the class info page.

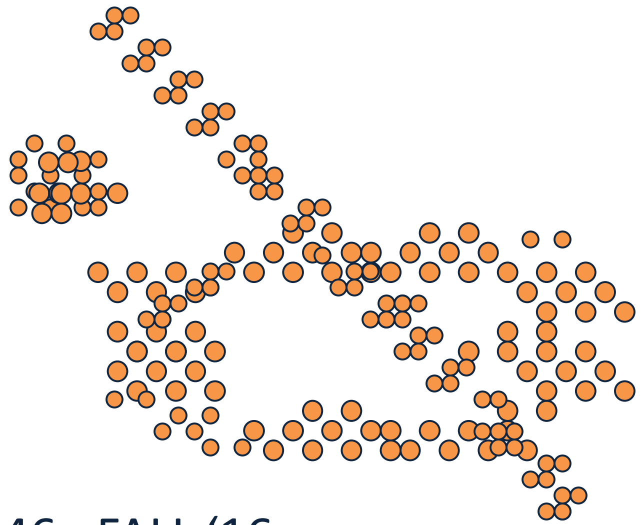
No office hours this week.

How many are there ?



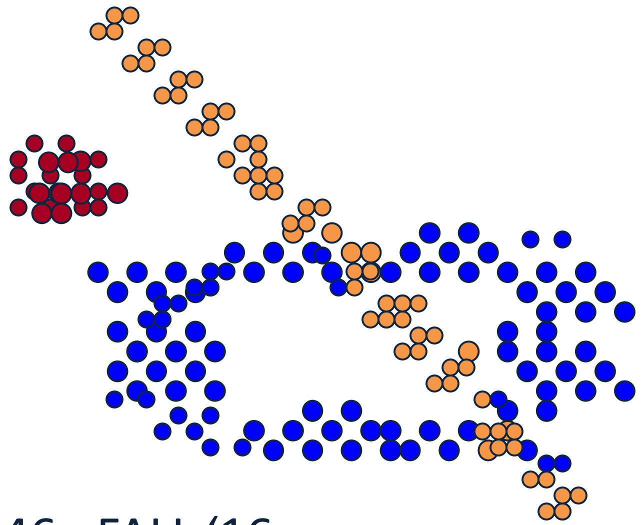
Clustering

- Clustering is a mode of unsupervised learning.
- Given a collection of data points, the goal is to find structure in the data: organize that data into sensible groups.
- We are after a convenient and valid organization of the data, not after a rule for separating future data into categories.
- Cluster analysis is the formal study of algorithms and methods for doing that.
- How many are there ?



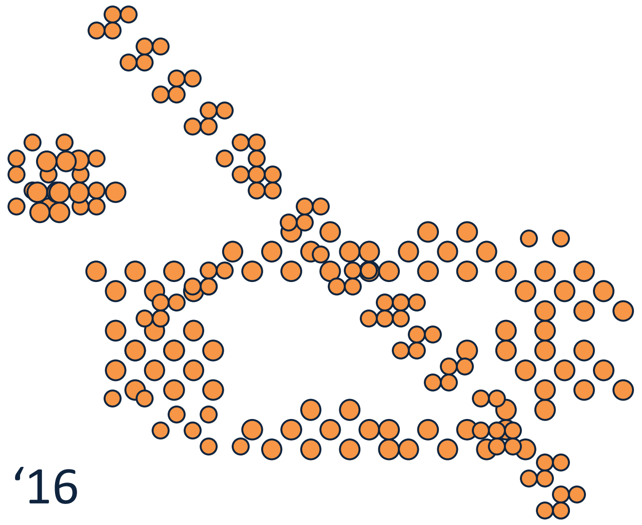
Clustering

- Clustering is a mode on unsupervised learning.
- Given a collection of data points, the goal is to find structure in the data: organize that data into sensible groups.
- We are after a convenient and valid organization of the data, not after a rule for separating future data into categories.
- Cluster analysis is the formal study of algorithms and methods for doing that.



Clustering

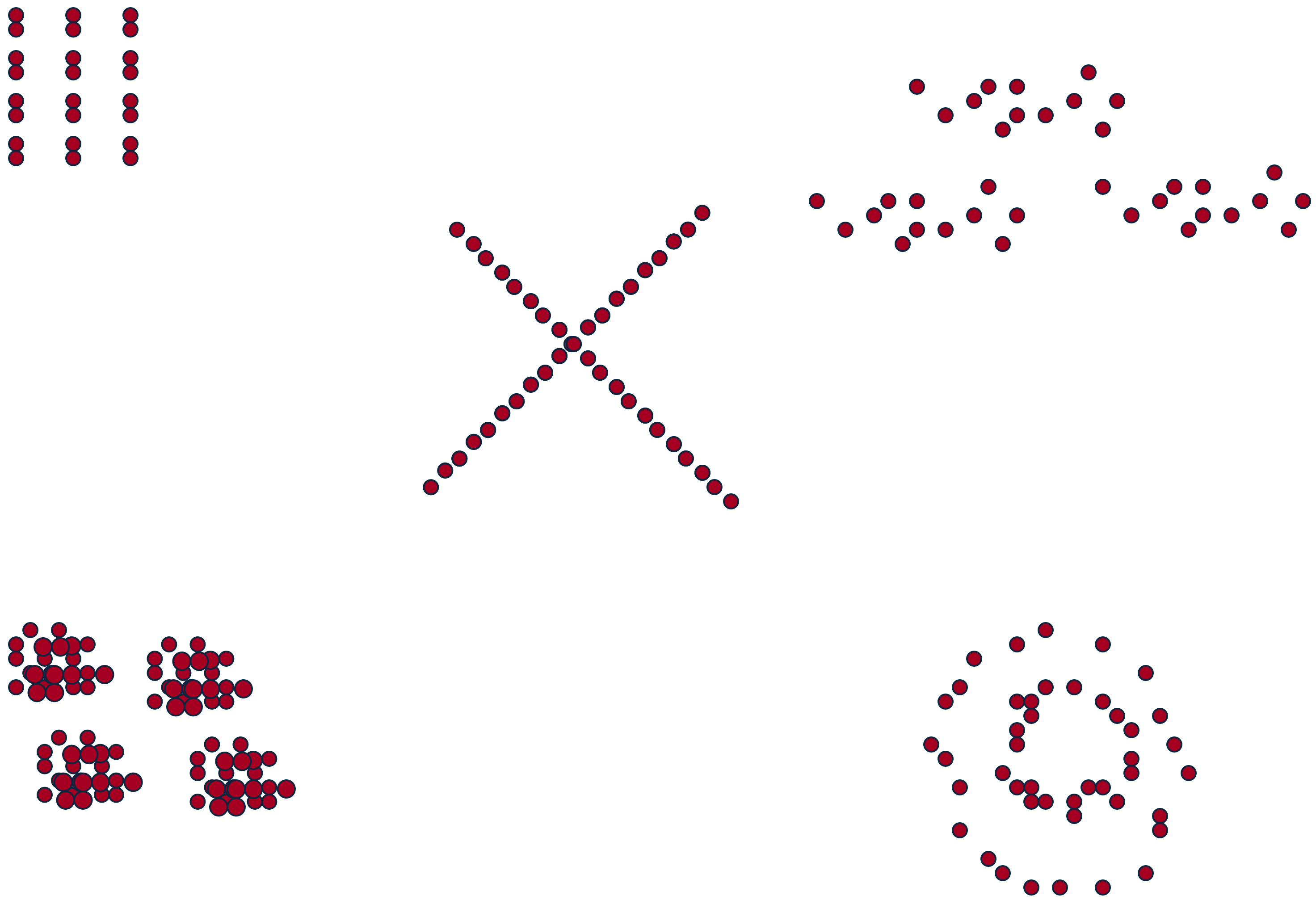
- A cluster is a set of entities which are alike, and entities in different clusters are not alike.
- A cluster is an aggregation of points in the test space such that the distance between any two points in the cluster is less than the distance between any point in the cluster and any point not in it.
- Clusters may be described as connected regions of a multi-dimensional space containing a relatively high density of points, separated from other regions by regions containing a low density of points.



Clustering

- The last definitions assume that the objects to be clustered are represented as points in some measurements space.
- “We recognize a cluster when we see it”.
- It is easy to give a functional definition for a cluster, but a lot harder to give an operational definition.
- One reason may be that objects can be clustered into groups with a purpose in mind (shape, size, time, resolution,….)

Clustering



Theorem: That is no clustering function that maps a set of points into a partition of it, that satisfies all three conditions. [Klienberg, NIPS 2002]

Clustering

- Clustering is not a Learning Problem. It's an Optimization Problem. Given a set of points and a pairwise distance, devise an algorithm f that splits the data so that it optimizes some natural conditions.
- Scale-Invariance.
  - For any distance function d; for any $\alpha > 0$, we have $f(d) = f(\alpha \cdot d)$.
- Richness.
  - Range(f) is equal to the set of all partitions of S.
  - In other words, suppose we are given the names of the points only (i.e. the indices in *S*) but not the distances between them. Richness requires that for any desired partition Γ, it should be possible to construct a distance function *d* on *S* for which *f*(*d*) = Γ
- Consistency.
  - Let d and d' be two distance functions. If f(d) = Γ, and *d'* is a Γ-transformation of *d*, then *f*(*d'*) = Γ. In other words, suppose that the clustering Γ arises from the distance function *d*. If we now produce *d'* by reducing distances within the clusters and enlarging distances between clusters then the same clustering Γ should arise from *d'*.

Clustering

- Clustering is not a Learning Problem. It's an Optimization Problem. Given a set of points and a pairwise distance, devise an algorithm f that splits the data so that it optimizes some natural conditions.

- So, what do we do?
  - Different optimization heuristics that make sense.
- Clustering can be done under generative model assumptions, or without any statistical assumptions
- A key component in clustering is the measurement space:
  - What is a reasonable distance/similarity measure ?
  - What are the important dimensions of the data ?
- We will discuss:
  - Clustering methods → Metric Learning methods
  - Dimensionality reduction methods

The Clustering Problem

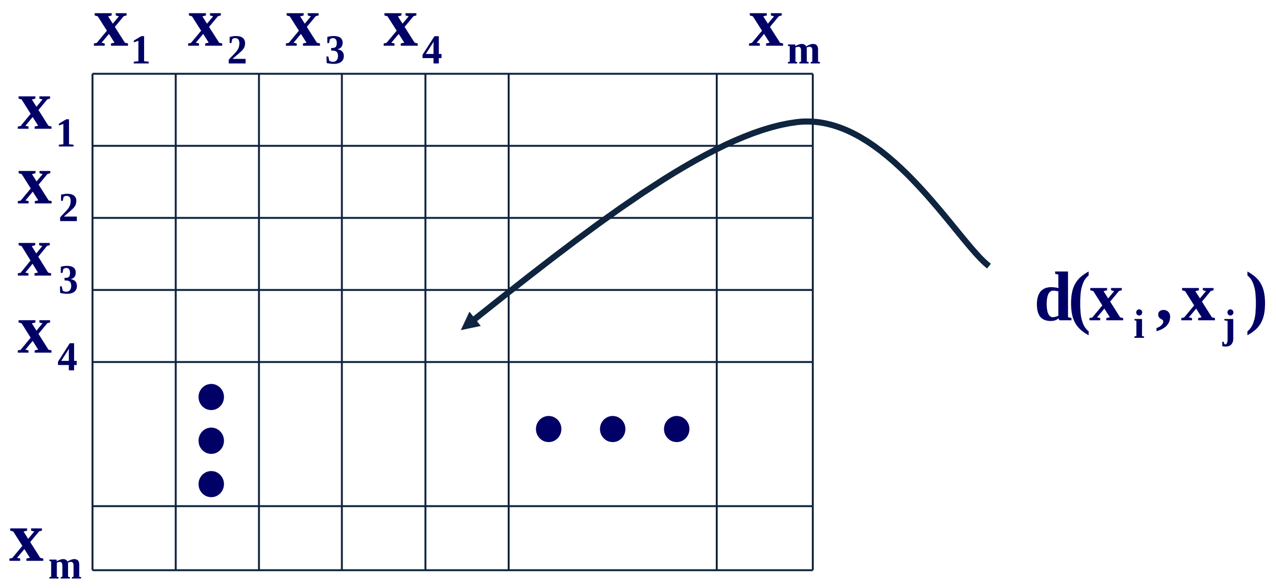
- We are given a set of data points $x_1, x_2, \ldots x_m$ that we would like to cluster.
- Each data point is assumed to be an d-dimensional vector, that we will write as a column vector:

$$x = (x_1, x_2, \ldots x_d)^T$$

- We do not make any statistical assumptions on the given data, nor on the number of clusters.

Distance Measures

- In studying Clustering techniques we will assume that we are given a matrix of distances between all pairs of data points.
- We can assume that the input to the problem is:



Distance Measures

- In studying Clustering techniques we will assume that we are given a matrix of distances between all pairs of data points.
- A distance measure (metric) is a function $d: R^d \times R^d \rightarrow R$ that satisfies:

$$1.\ \mathbf{d(x, y)} \geq 0,\ \mathbf{d(x, y) = 0} \Leftrightarrow \mathbf{x = y}$$

$$2.\ \mathbf{d(x, y)} + \mathbf{d(y, z)} \geq \mathbf{d(x, z)}$$

$$3.\ \mathbf{d(x, y)} = \mathbf{d(y, x)}$$

- For the purpose of clustering, sometimes the distance (similarity) is not required to be a metric
  - No Triangle Inequality
  - No Symmetry

Distance Measures

<u>Examples</u>:

- Euclidean Distance:

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^2} = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})} = \sqrt{\sum_{i=1}^{d}(\mathbf{x}_i - \mathbf{y}_i)^2}$$

- Manhattan Distance:

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = |\mathbf{x}-\mathbf{y}| = \sum_{i=1}^{d}|\mathbf{x}_i - \mathbf{y}_i|$$

- Infinity (Sup) Distance:

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \max_{1\le i\le d}|\mathbf{x}_i - \mathbf{y}_i|$$

- Notice that if $\mathbf{d}(\mathbf{x},\mathbf{y})$ is the Euclidean metric, $\mathbf{d}^2(\mathbf{x},\mathbf{y})$ is not a metric but can be used as a measure (no triangle inequality)

Distance Measures

Examples:

- Euclidean Distance:

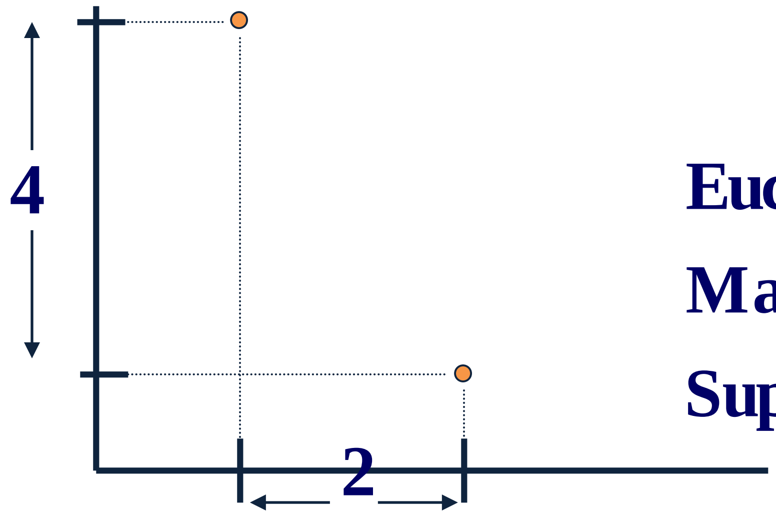
$$d(x, y) = \sqrt{(x - y)^2} = \sqrt{(x - y)^T (x - y)} = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- Manhattan Distance:

$$d(x, y) = | x - y | = \sum_{i=1}^{d} | x_i - y_i |$$

- Infinity (Sup) Distance:

$$d(x, y) = \max_{1 \le i \le d} | x_i - y_i |$$



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

$$\text{Manhattan} : 4 + 2 = 6$$

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

Distance Measures

Notice that:

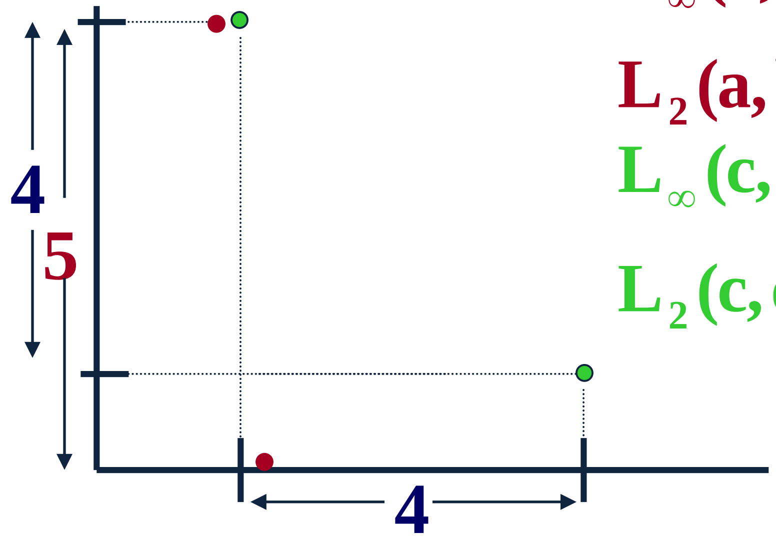
- Infinity (Sup) Distance < Euclidean Distance <Manhattan Distance:

$$L_\infty = \max_{1 \le i \le d} |x_i - y_i|$$

$$L_1 = |x - y| = \sum_{i=1}^{d} |x_i - y_i|$$

$$L_2 = \sqrt{(x - y)^2} = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

- But distances do not induce same order on pairs of points



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

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

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

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

$$\boxed{\begin{aligned} L_\infty(c,d) &< L_\infty(a,b) \\ L_2(c,d) &> L_2(a,b) \end{aligned}}$$

Distance Measures

- The clustering may be sensitive to the similarity measure.
- Sometimes this can be avoided by using a distance measure that is invariant to some of the transformations that are natural to the problem.
- Mahalanobis Distance: $\mathbf{d(x, y)} = \sqrt{(\mathbf{x} - \mathbf{y})^{\mathrm{T}} \Sigma (\mathbf{x} - \mathbf{y})}$

  where Σ is a symmetric matrix.

  Covariance Matrix: Translates all the axes so that they have Mean=0 and Variance=1 (Shift and Scale invariance)

$$\mu = \frac{1}{\mathbf{m}} \sum_{\mathbf{i}=1}^{\mathbf{m}} \mathbf{x}_{\mathbf{i}}, \text{ a column vector, average of the data}$$

$$\Sigma = \frac{1}{\mathbf{m}} \sum_{\mathbf{i}=1}^{\mathbf{m}} (\mathbf{x} - \mu)(\mathbf{x} - \mu)^{\mathrm{T}}, \text{matrix of size m x m}$$

Distance Measures

- The clustering may be sensitive to the similarity measure.
- Sometimes this can be avoided by using a distance measure that is invariant to some of the transformations that are natural to the problem.
- Mahalanobis Distance: $$\mathbf{d(x, y)} = \sqrt{\mathbf{(x - y)^T \Sigma (x - y)}}$$ where Σ is a symmetric matrix.

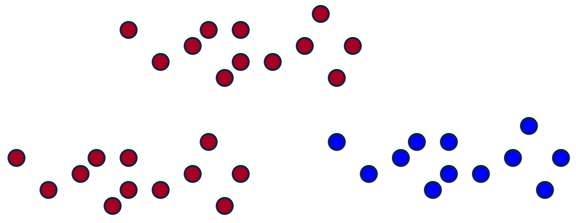
  Covariance Matrix: Translates all the axes so that they have Mean=0 and Variance=1 (Shift and Scale invariance)

- It is possible to get rotation invariance by rotating the axes so that they coincide with the eigenvectors of the covariance matrix. This is a transformation to the principle components.

Distance Measures

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

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



- and distance between sets of points A, B:

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

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

Basic Algorithms

- Given: a set $x_1, x_2, \ldots x_m$ of data points,
  a distance function $d(x,y)$ and
  a threshold T
- C_i will represent clusters, z_i their representative
- i index into data points, j index into clusters

Problems: Outcome depends on the order of the data points both in assigning points to a cluster and in determining distance of a point from a cluster

$$\text{Initialize}: \mathbf{x_1 = z_1 \in C_1}$$

$$\mathbf{k = 1}$$

Do sequentially for all i:

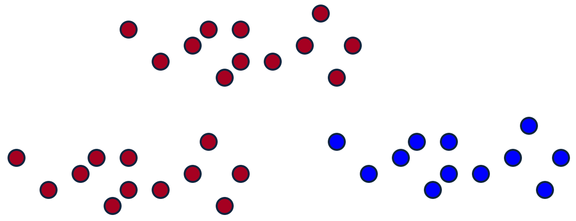
$$\mathbf{Let}: D_{ij} = d(x_i, z_j), \quad \text{for all } j = 1, \ldots k$$

$$D_i = \mathrm{Min}_j D_{ij}, \quad I_i = \{j \mid D_i = D_{ij}\}$$

$\rightarrow$ **For each point i, find its cluster**

$$\text{If } D_i < T \quad \Rightarrow x_i \in C_{I_i}$$

$$\text{Otherwise}: \Rightarrow k = k+1, z_{k+1} = x_i \in C_{k+1}$$



process data point i
(where to place it?)

Association-Dissociation

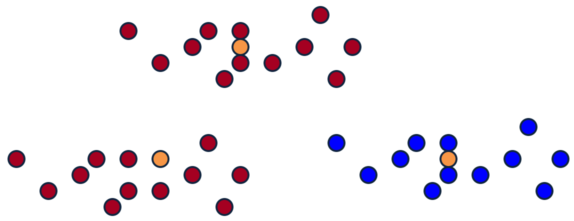
- Given a collection of points, one way to define the goal of a clustering process is to use the following two measures:

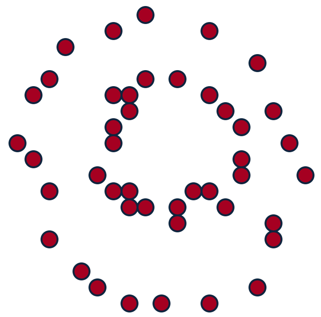
- A measure of similarity within a group of points
- A measure of similarity between different groups

- Ideally, we would like to define these so that:

  The within similarity can be maximized
  The between similarity can be minimized
  at the same time.

- This turns out to be a hard task.

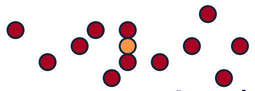




Quality

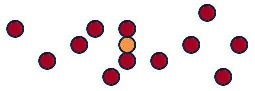
- Given: a set $X = \{x_1, x_2, \ldots x_m\}$ of points, a distance function $d(x,y)$
- X is split into k clusters C_j, each with a representative $z_j \in X$

- Definitions: (scatter measures)



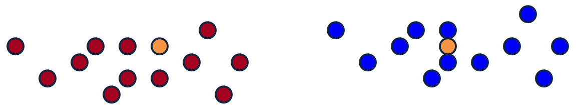
Within cluster: (average distance to representative)

$$D_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} d(x_i, z_j)$$



Global Clustering Scatter:

$$D = \frac{1}{|m|} \sum_{i=1.m} \min_{j=1,k} d(x_i, z_j)$$



For each x_i choose the closest representative z_j.

D_j measures the scatter of the jth cluster; we want to minimize it.

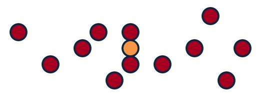
D measures the quality of the clustering;

Quality

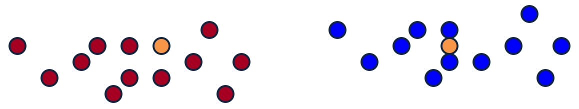
- Definitions: (scatter measures)

Within cluster: (average distance to representative)

$$D_j = \frac{1}{|C_j|}\sum_{x_i \in C_j} d(x_i, z_j)$$



Global Clustering Scatter:



$$D = \frac{1}{|m|}\sum_{i=1.m} \min_{j=1,k} d(x_i, z_j)$$

For each x_i choose the closest representative z_j.

D_j measures the scatter of the jth cluster; we want to minimize it.

D measures the quality of the clustering;

For optimal clustering: $\textbf{If } x_i \in C_r : \operatorname{argmin}_{j=1,k} d(x_i, z_j) = z_r$

$$D = \frac{1}{|m|}\sum_1^m \min_{j=1}^{k} d(x_i, z_j) = \frac{1}{|m|}\sum_{j=1}^{k} |C_j| D_j = \sum_{j=1}^{k} \frac{|C_j|}{m} D_j$$

K-Means

- Given: a set $X = \{x_1, x_2, \ldots x_m\}$ of points, a distance function $d(x,y)$
- X is split into k clusters C_j, each with a representative $z_j \in X$

- Algorithm:
  1. Initialize centers randomly $\mathbf{z_1, z_2, \ldots z_k}$ round: $r=1$
  2. Cluster $x_1, x_2, \ldots x_m$ w.r.t centers using Nearest Neighbor
     $$\mathbf{x_i \in C_j \Leftrightarrow j = \text{argmin}\ {}_j d(x_i, z_j)}$$
  3. Choose new centers: Choose z_j to minimize D_i .
     Compute the clustering total scatter for this round: $\mathbf{D(r)}$
  4. Stopping Criterion: Check if $$\mathbf{\frac{D(r-1) - D(r)}{D(r-1)} < T}$$
     If not, iterate: $r = r+1$, go back to 2.

K-Means

•Given: a set $X = \{x_1, x_2, \ldots x_m\}$ of points, a distance function $d(x,y)$
• X is split into k clusters C_j, each with a representative $z_j \in X$

• Will it converge ? It can be shown that the scatter mean goes down.
    • Note that this is a Hard EM algorithm (see K-Means in the EM lecture)
• We do not know how fast it will converge -- bound # of iterations.

• Why should the center be an element in the set ?
  Using the Euclidean Distance, minimizing is achieved by computing the average, which need not be a data element.

• What is k ? Can try with different values, and measure the quality of the clustering.

Improving K-Means

- The main problem with k-means is the initial conditions -- determining k and the centers.

- Bad initial conditions may generate unimportant cells and may degrade overall performance.
- There are various ways to get around it.
- Methods for splitting centers:
  Start with k=1; for k=i use the centers of k=i-1, or a simple function of them.
- ISODATA: k-means with provisions for
  - deleting clusters (if they are too small)
  - splitting clusters (if their mean scatter is too large)
  - Adapting k; stopping criterion

Limitations

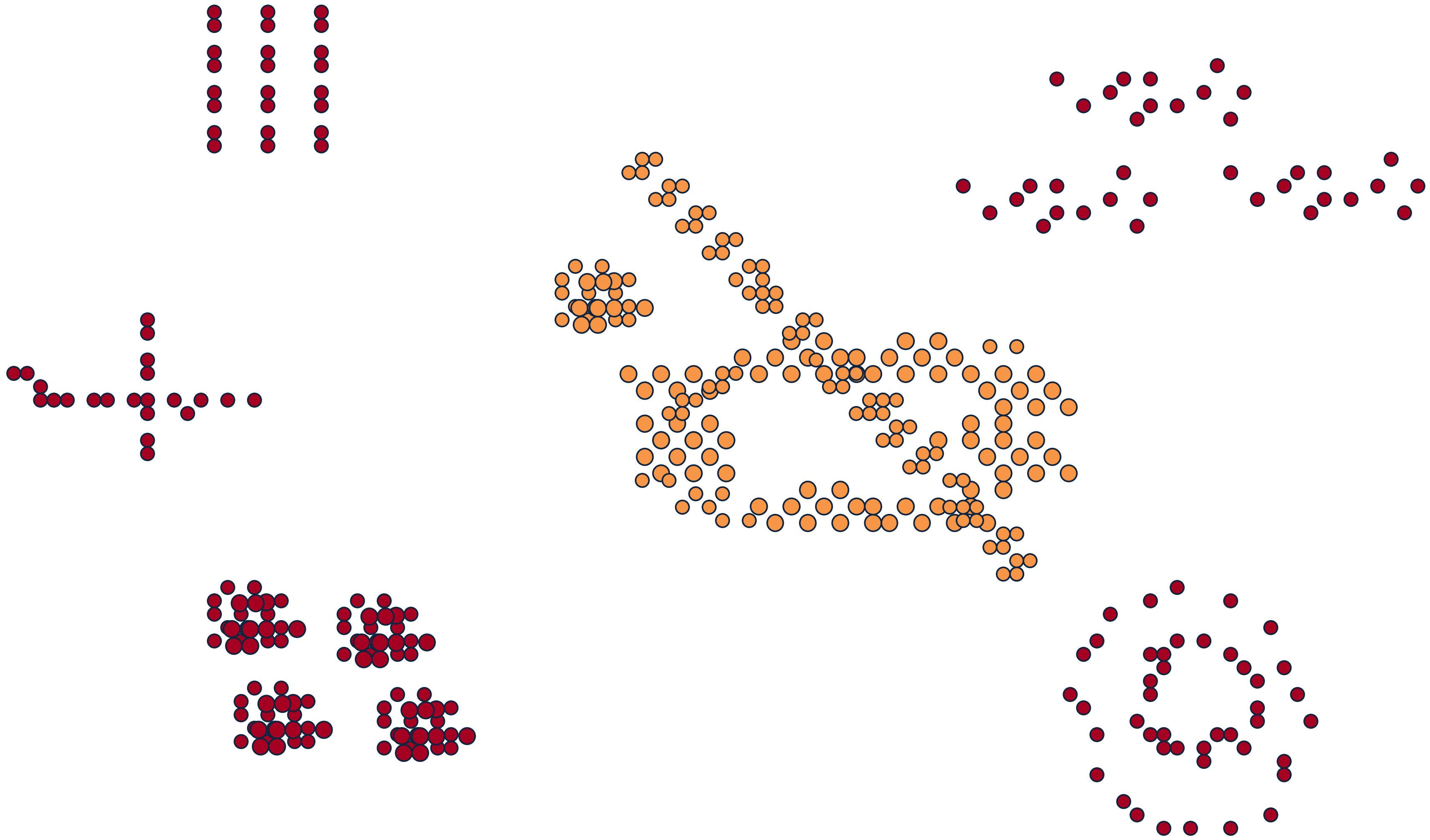
- k-means/ISODATA will work well in cases where all the clusters behaves similarly statistically.

- K-means can be shown to be optimal when the distance function is derived from the probability distribution that generates the data.

- E.g., for a mixture of Normal distribution, the Euclidean metric yields optimal performance.
  This is the EM algorithms studied earlier.

- These methods are not so effective when the data has some internal structure, especially if different clusters have different structures.

Limitations



Model Based Methods

- One advantage of K-means is that it is a principled method – it has a probabilistic interpretation.

This allows a principle investigation of the algorithm; a better understanding of what it does, and a way to modify it in a principled way.

Can this be done for other algorithms?

Agglomerative Clustering

- Assume a distance measure between points $d(x_1,x_2)$
- Define a distance measure between Clusters $D(c_1,c_2)$
- Algorithm:
  - Initialize: Each point in a separate cluster.
  - At each stage, merge the two closest clusters according to D.
    (I.e., merge the two D-closest clusters).

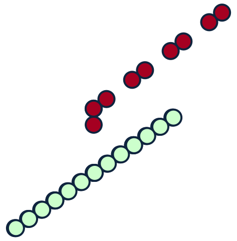
Different definitions of D, for the same d, give rise to radically different partitions of the data.

Examples (I)

- Assume a distance measure between points $d(x_1,x_2)$
- Define a distance measure between Clusters $D(c_1,c_2)$
- Algorithm:
  - Initialize: Each point in a separate cluster.
  - At each stage, merge the two closest clusters according to D. (I.e., merge the two D-closest clusters).

Single Link Clustering:

$$D_{SL}(C_1,C_2) = \min\{x_i \in C_i\}\ d(x_1,x_2)$$



Complete Link Clustering:

$$D_{CL}(C_1,C_2) = \max\{x_i \in C_i\}\ d(x1,x2)$$

Examples (II)

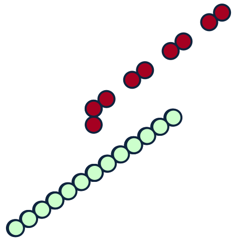
- Assume a distance measure between points $d(x_1,x_2)$
- Define a distance measure between Clusters $D(c_1,c_2)$
- Algorithm:
  - Initialize: Each point in a separate cluster.
  - At each stage, merge the two closest clusters according to D. (I.e., merge the two D-closest clusters).

Ward's Method:

$$D(ward) = ESS(C1 \cup C2) - ESS(C1) - ESS(C2)$$

Where: $ESS(C) = \Sigma(x-m)^2$

m – mean of data point in cluster C



Group Average Clustering:

$$D_{GA}(C_1,C_2) = \text{mean}\{C_i, C_j\}\ d(x1,x2)$$

Model Based Methods

Claim: Common heuristics for agglomerative clustering algorithms are Each equivalent to a hierarchical model-based (probabilistic) method.

This interpretation gives a theoretical explanation for the empirical behavior of these algorithms, as well as a principled approach to practical issues: no. of clusters, choice of methods, etc.

Model based clustering views clustering as the problem of computing the (approximate) maximum for the classification likelihood of the data X.

The classification likelihood of the data X:

$$L(\theta_1,\ldots,\theta_k ; l_1,\ldots,l_n \mid X) = \Pi p\,(x_i \mid \theta_i)$$

Where: l_i is the label (cluster id) of the point x_i

θ_i are the model parameters.

Notice that this is a model of *hard* clustering. It is also possible to model soft clustering, as a mixture model.

Model Based Agglomerative Methods

Model based clustering views clustering as the problem of computing the (approximate) maximum for the classification likelihood of the data X.

Agglomerative approach:

- Start with a partition P of the data in which each sample is in its own singleton cluster.

- At each stage, two clusters are chosen from P and merged, forming a new partition P'.

- The pair which is merged is the one which gives the highest resulting likelihood. (merges typically reduce the likelihood)

- The process is greedy. The best choice at a certain stage need not develop into the best strategy.

Model Based Agglomerative Methods

Agglomerative approach:

- Start with a partition P of the data in which each sample is in its own singleton cluster.
- At each stage, two clusters are chosen from P and merged, forming a new partition P'.
- The pair which is merged is the one which gives the highest resulting likelihood. (merges typically reduce the likelihood)
- The process is greedy. The best choice at a certain stage need not develop into the best strategy.

At each stage of the algorithm we are choosing new labels; we don't explicitly choose new parameters. Implicitly, it is assumed we have the best parameters. The quality of the current labeling:

$$J(l_1,\ldots,l_n \mid X) = \max_\Theta L(\Theta, l_1,\ldots,l_{n-} \mid X)$$

Relative cost of a merge:

$$\Delta J(P.P') = J(P)/J(P')$$

Rather than maximizing J(P'), can maximize the relative cost.

Model Based Methods

The classification likelihood of the data X:

$$L(\theta_1,\ldots,\theta_k ;l_1,\ldots,l_n \; |X) = \Pi p\,(x_i|\; \theta_i)$$

Where: l_i is the label (cluster id) of the point x_i
θ_i are the model parameters.

Notice that this is a model of *hard* clustering. It is also possible to model soft clustering, as a mixture model.

Model Based Interpretation

Ward's Method:

- If the probability model is multivariate normal with uniform spherical covariance matrix σI, then

$$\Delta J \sim D(\text{ward})$$

In this case we assume the component density is:

Rather than maximizing J(P'), can maximize the relative cost.

$$p(x_i \mid \sigma, \eta_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x_i-\eta_i)^2} / 2\sigma^2$$

Model Based Interpretation

Single-Link clustering:

- The corresponding probability model is a mixture of branching random walks (BRWs). A BRW is a stochastic process which generates a tree of data points x as follows:

  - The process starts with a single root x_0 in the placed according to some distribution p_0

  - Each node in the frontier of the tree produces zero or more children. The position of a child is generated according to a multivariate normal distribution, with variance σI centered around the parent's location.

Claim: If the probability model is a mixture of BRWs, then:

$$\Delta J \sim D(SL)$$

Model Based Methods

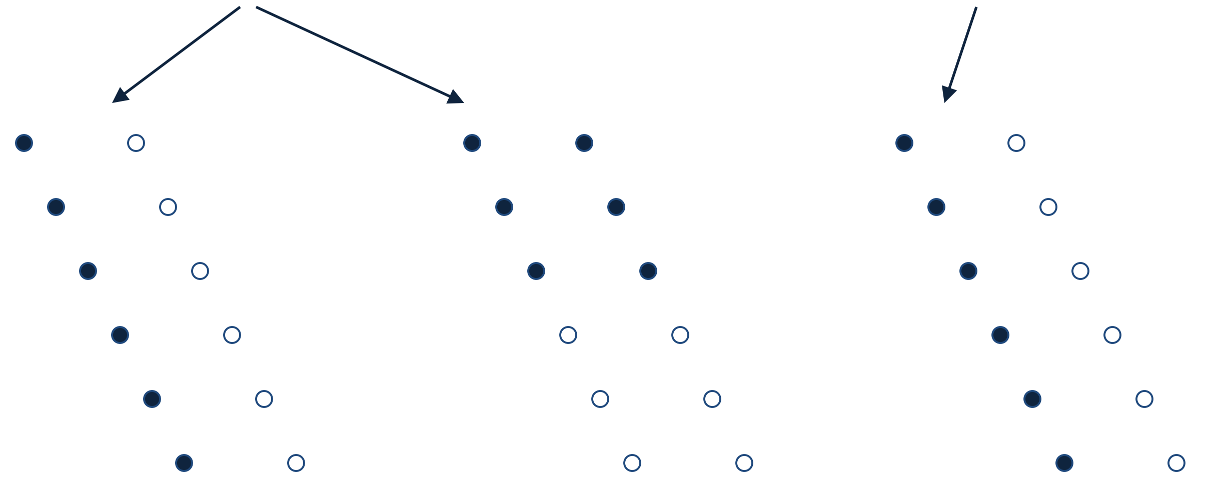
- One advantage of K-means is that it is a principled method – it has a probabilistic interpretation.

This allows a principle investigation of the algorithm; a better understanding of what it does, and a way to modified it in a principled way.

Several Heuristics can be given probabilistic interpretation.

Importance of a Metric for a Clustering Algorithm

$d^1(x,x') = [(f_1 - f_1')^2+(f_2 - f_2')^2]^{1/2}$ $d^2(x,x') = |f_1 - f_1'|+|f_2-f_2'|$



(a) Single-Linkage with Euclidean

(b) K-Means with Euclidean

(c) K-Means with a Linear Metric

There is no 'universal' distance metric that is good for any clustering algorithms and for any problems.

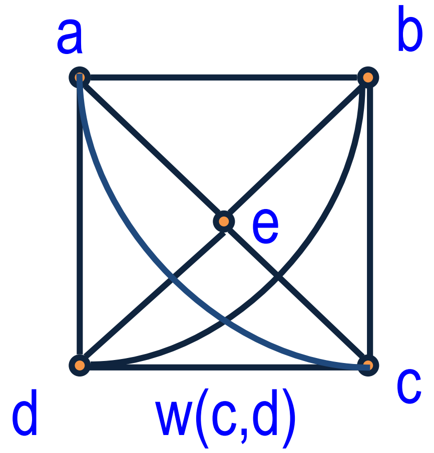
Graph Theoretic Methods

- Points in an arbitrary feature space are represented as a weighted graph G=(V,E)

- Nodes represent the points in the feature space.
- Edges are drawn between every pair of nodes. The weight of the edge w(i,j) is a function of the similarity between nodes i and j.

Proximity Matrix:

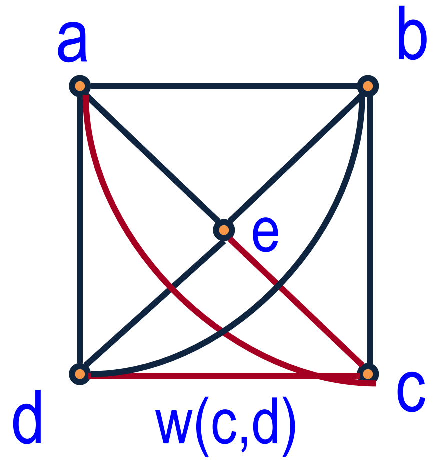
	a	b	c	d	e
a	0	6	8	2	7
b	6	0	2	5	3
c	8	2	0	10	9
d	2	5	10	0	4
e	7	3	9	4	0



Graph Theoretic Methods

- Points in an arbitrary feature space are represented as a weighted graph G=(V,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



- We seek a partition of the set of $\mathbf{V}$ vertices into disjoint sets $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_k$ where: some measure of the similarity
  among the vertices in each $\mathbf{V}_i$ is high , and
  across sets $\mathbf{V}_i, \mathbf{V}_j$ is low.

(Notice that we assume a similarity measure, but it need not be metric)

Graph Theoretic Methods

- What is the precise criterion for a good partition ?
- How can such a partition be computed efficiently ?

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

Algorithm:

- Construct the Maximum Spanning Tree (recall: we work with similarity)
- Identify inconsistent edges in the MST
- Remove the inconsistent edges to form connected components and call them clusters.

Graph Theoretic Methods

- Algorithm:
- Construct the Maximum Spanning Tree
- Identify inconsistent edges in the MST
- Remove the inconsistent edges to form connected components and call them clusters.

What are inconsistent edges ?

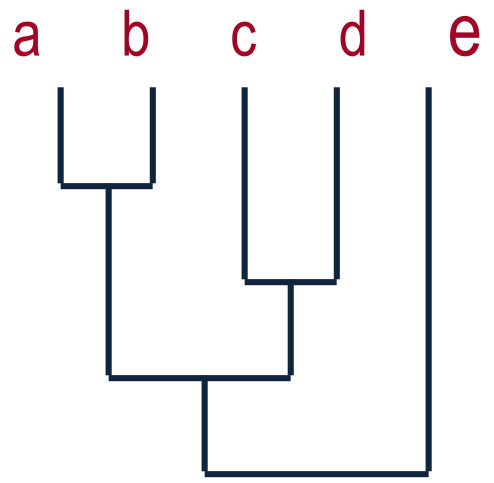
- Use a threshold (delete the light edges)
- Delete an edge if its weight is significantly lower than that of nearby edges.

Notice: in any case -- methods are local and thus not very different from the distance-based methods used before.

Example: Hierarchical Clustering

- Hierarchical clustering is a nested sequence of partitions
- <u>Agglomerative:</u>
  Places each object in its own cluster and gradually merge the atomic clusters into larger and larger clusters.
- <u>Divisive:</u> Start with all objects in one cluster and subdivide into smaller clusters.

{(a) ,(b),(c),(d),(e)}
{(a,b),(c),(d),(e)}
{(a,b),(c,d),(e)}
{(a,b,c,d),(e)}
{(a,b,c,d,e)}



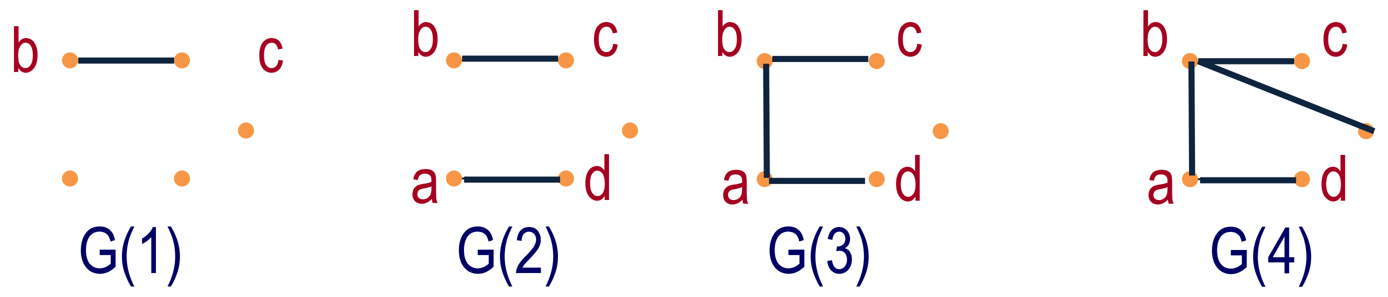
Example: Hierarchical Clustering

- Form a Threshold Graph G(k): $(i,j) \in G(k)$ iff $k \geq d(i,j)$
- If less clusters then before:
  - Name each <u>connected component</u> of G(k) a cluster or
  - Name each <u>clique</u> of G(k) a cluster

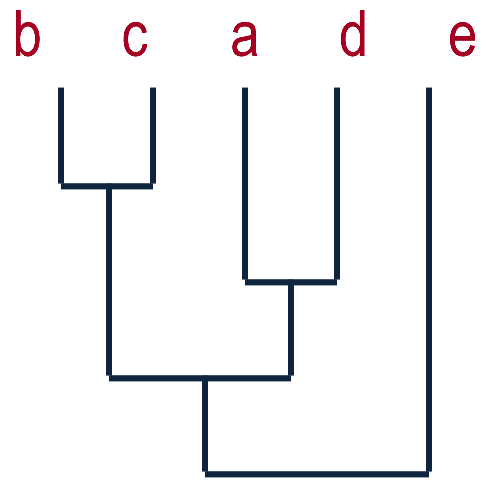
	a	b	c	d	e
a	0	3	8	2	7
b	3	0	1	5	4
c	8	1	0	10	9
d	2	5	10	0	4
e	7	4	9	4	0

Example: Hierarchical Clustering

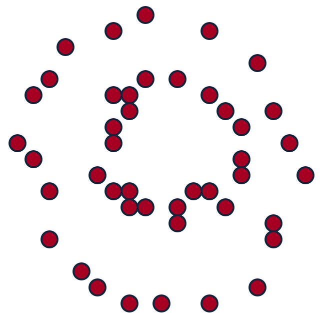
- Form a Threshold Graph G(k): $(i,j) \in G(k)$ iff $k \geq d(i,j)$
- If less clusters then before:
  - Name each connected component of G(k) a cluster or
  - Name each clique of G(k) a cluster



	a	b	c	d	e
a	0	3	8	2	7
b	3	0	1	5	4
c	8	1	0	10	9
d	2	5	10	0	4
e	7	4	9	4	0

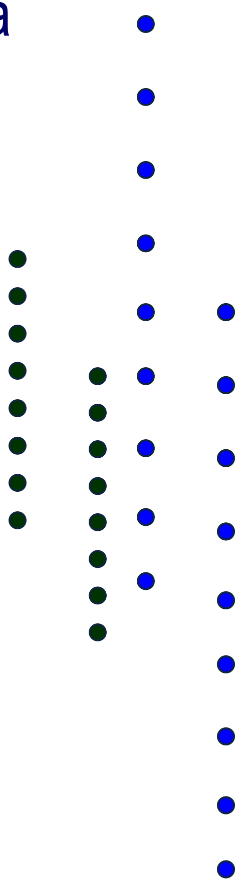


Clustering



Global Algorithms

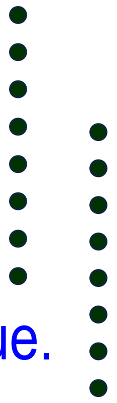
- MST and neighborhood approaches are very efficient but are based on local properties of the graph.
- In many applications (e.g., image segmentation) we need a partition criterion that depends on global properties.
- How to partition the graph G(V,E) into the "natural" disjoint sets A,B?
- Try to define a global degree of similarity between parts of the graph.



Cut Algorithms

- MST and neighborhood approaches are very efficient but are based on local properties of the graph.
- In many applications (e.g., image segmentation) we need a partition criterion that depends on global properties.

- A Graph G(V,E) can be partitioned into two disjoint sets A,B.
- The degree of similarity between the two parts:



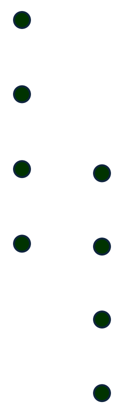
$$\mathbf{cut(A, B)} = \sum_{\mathbf{u \in A, v \in B}} \mathbf{w(u, v)}$$

- The optimal bi-partition of G is one that minimizes the cut value.

- There exist efficient algorithms for computing the minimal cut.

Cut Algorithms

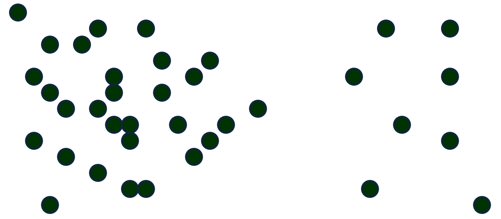
$$\mathrm{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$



- Cut algorithms can be extended to k-partitions by recursively finding the minimal cuts that bisects the existing groups.

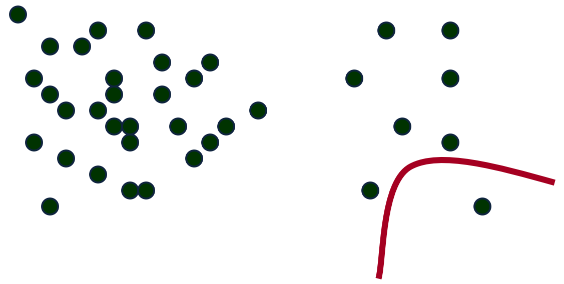
Cut Algorithms

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



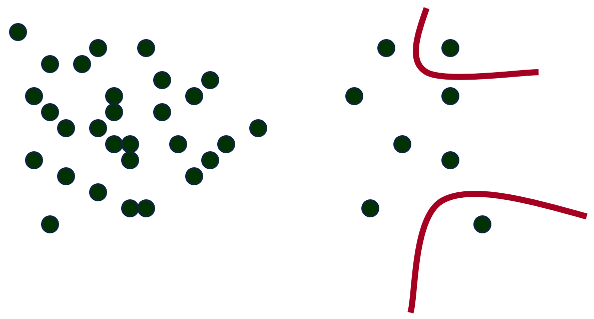
Cut Algorithms

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



Cut Algorithms

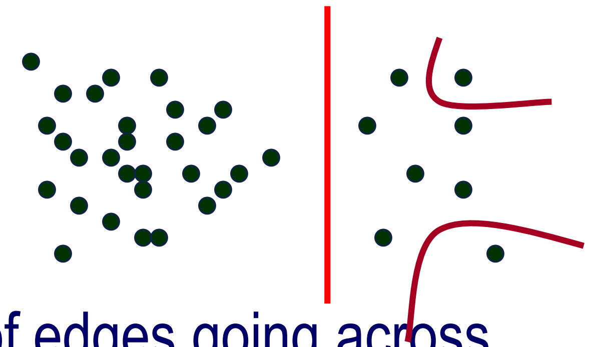
$$\mathrm{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$



Cut Algorithms

- Minimal cut favors cutting small sets of isolated nodes in the graph G(V,E)

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



The cut value increases with the number of edges going across the partitions. (The drawn partition assumes that distances are inversely proportional to the similarity).

Improvement: Normalization -

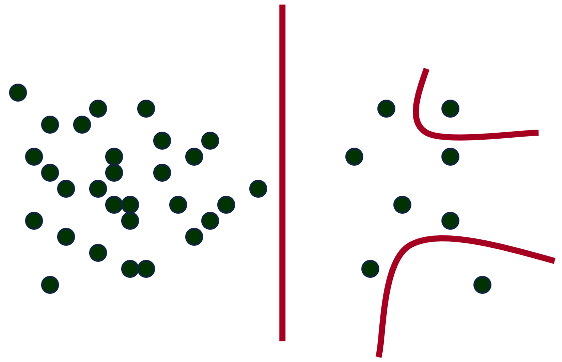
$$\mathrm{asso}(A, V) = \sum_{u \in A, v \in V} w(u, v)$$

measures the total connection from the nodes in A to the graph V.

Cut Algorithms

- The normalized measure would be:

$$\mathrm{Ncut}(A, B) = \frac{\mathrm{cut}(A, B)}{\mathrm{asso}(A, V)} + \frac{\mathrm{cut}(A, B)}{\mathrm{asso}(B, V)}$$

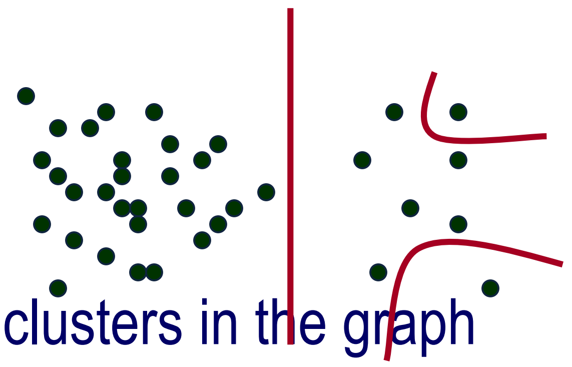


This is a measure of dissociation between clusters in the graph

Cut Algorithms

- The normalized measure would be:

$$\mathbf{Ncut(A, B)} = \frac{\mathbf{cut(A, B)}}{\mathbf{asso(A, V)}} + \frac{\mathbf{cut(A, B)}}{\mathbf{asso(B, V)}}$$



This is a measure of dissociation between clusters in the graph

We can also define the normalized association within clusters:

Let $\mathbf{asso(A, A)}$ be as before (total weights edges with A)

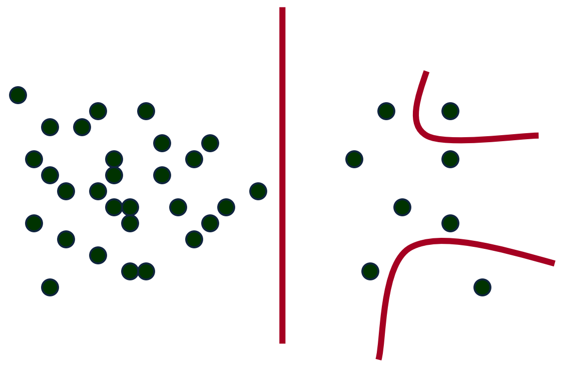
$$\mathbf{Nasso(A,B)} = \frac{\mathbf{asso(A, A)}}{\mathbf{asso(A, V)}} + \frac{\mathbf{asso(B,B)}}{\mathbf{asso(B, V)}}$$

Cut Algorithms

- We have two measure:

  The disassociation measure

$$\mathbf{Ncut(A, B)} = \frac{\mathbf{cut(A, B)}}{\mathbf{asso(A, V)}} + \frac{\mathbf{cut(A, B)}}{\mathbf{asso(B, V)}}$$



which we want to minimize.

and a measure of association within clusters:

$$\mathbf{Nasso(A,B)} = \frac{\mathbf{asso(A, A)}}{\mathbf{asso(A, V)}} + \frac{\mathbf{asso(B,B)}}{\mathbf{asso(B, V)}}$$

which reflects how tightly, on average, nodes within the groups are connected to each other and we want to maximize.

Cut Algorithms

- The disassociation measure (want to minimize)

$$\underline{\mathrm{Ncut}(A,B)} = \frac{\mathrm{cut}(A,B)}{\mathrm{asso}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{asso}(B,V)} =$$

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

$$2 - \left(\frac{\mathrm{asso}(A,A)}{\mathrm{asso}(A,V)} + \frac{\mathrm{asso}(B,B)}{\mathrm{asso}(B,V)}\right) = \underline{2 - \mathrm{Nasso}(A,B)}$$

Within cluster association measure (want to maximize).

Normalized Cut Algorithms

- The two partition criteria that we seek:

  minimizing the disassociation measure and

  maximizing the within cluster association measure

  are related and can be satisfied simultaneously.

- How to compute it efficiently:

  The problem of Normalized Cut is NP hard.

  Approximation algorithms are based on

  Spectral Methods - solving an eigenvalue problem

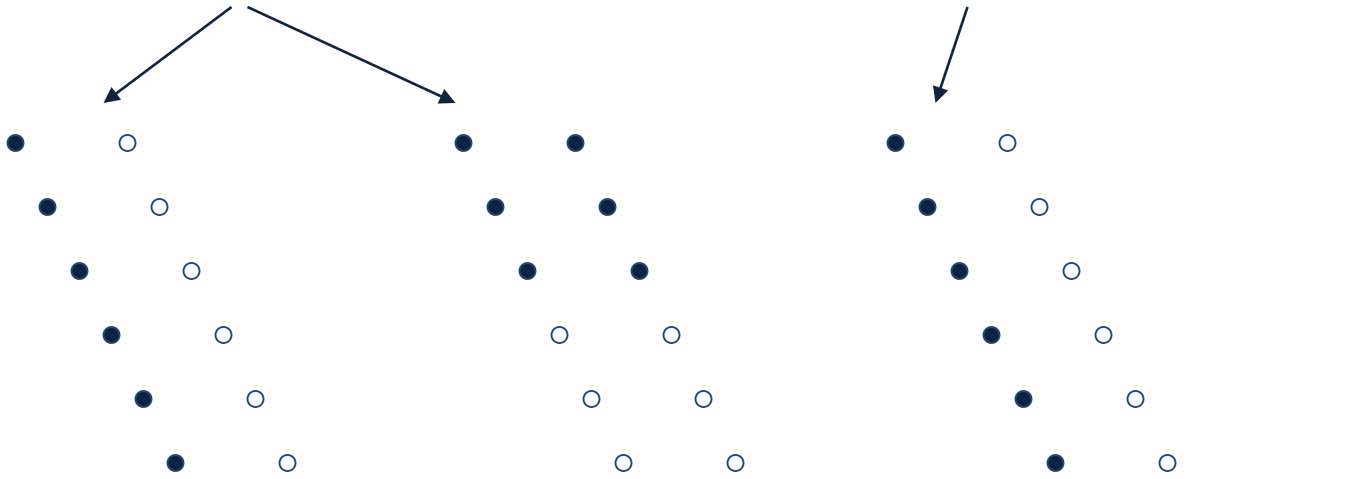
Clustering: Summary

- The problem of partitioning a set of point into k groups is ill defined.
- Determining the features space and the similarity measure may be application dependent and are crucial in many cases.

- Standard approaches:
  k-means; agglomerative methods
- Graph Theoretic methods:
- MST algorithms
- Cut algorithm
- Normalized Cut/Spectral Methods
- Key questions in current research:
  - Scalability; Metric Learning

Importance of a Metric for a Clustering Algorithm

$d^1(x,x') = [(f_1 - f_1')^2 + (f_2 - f_2')^2]^{1/2}$ $d^2(x,x') = |(f_1 + f_2) - (f_1' + f_2')|$



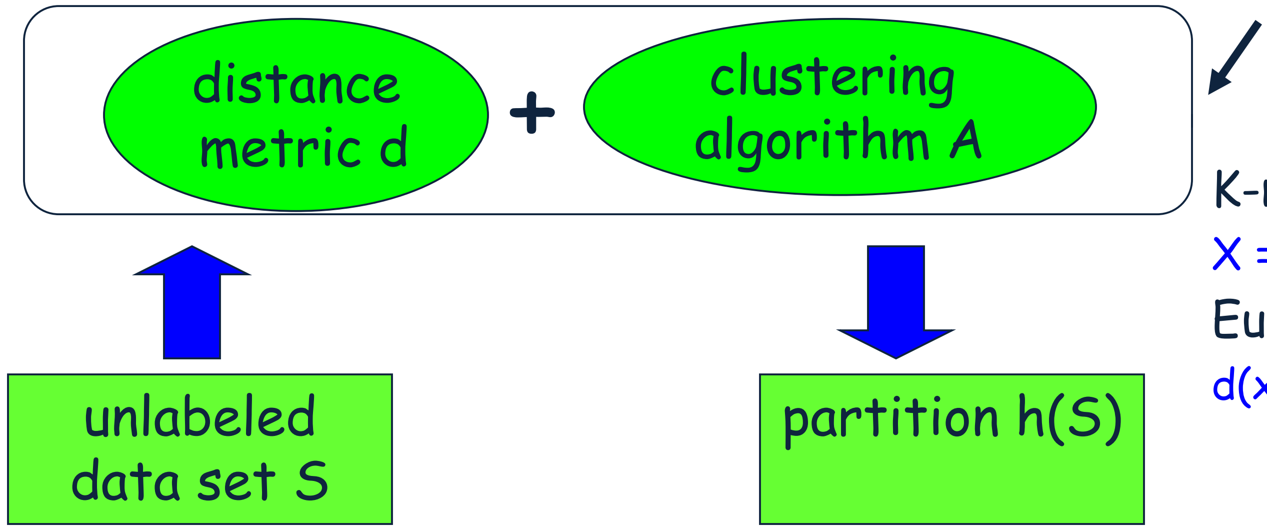
(a) Single-Linkage with Euclidean

(b) K-Means with Euclidean

(c) K-Means with a Linear Metric

There is no 'universal' distance metric good for any clustering algorithms and for any problems.

Traditional Clustering



A partition function

$h(S) = A_d(S)$

K-means

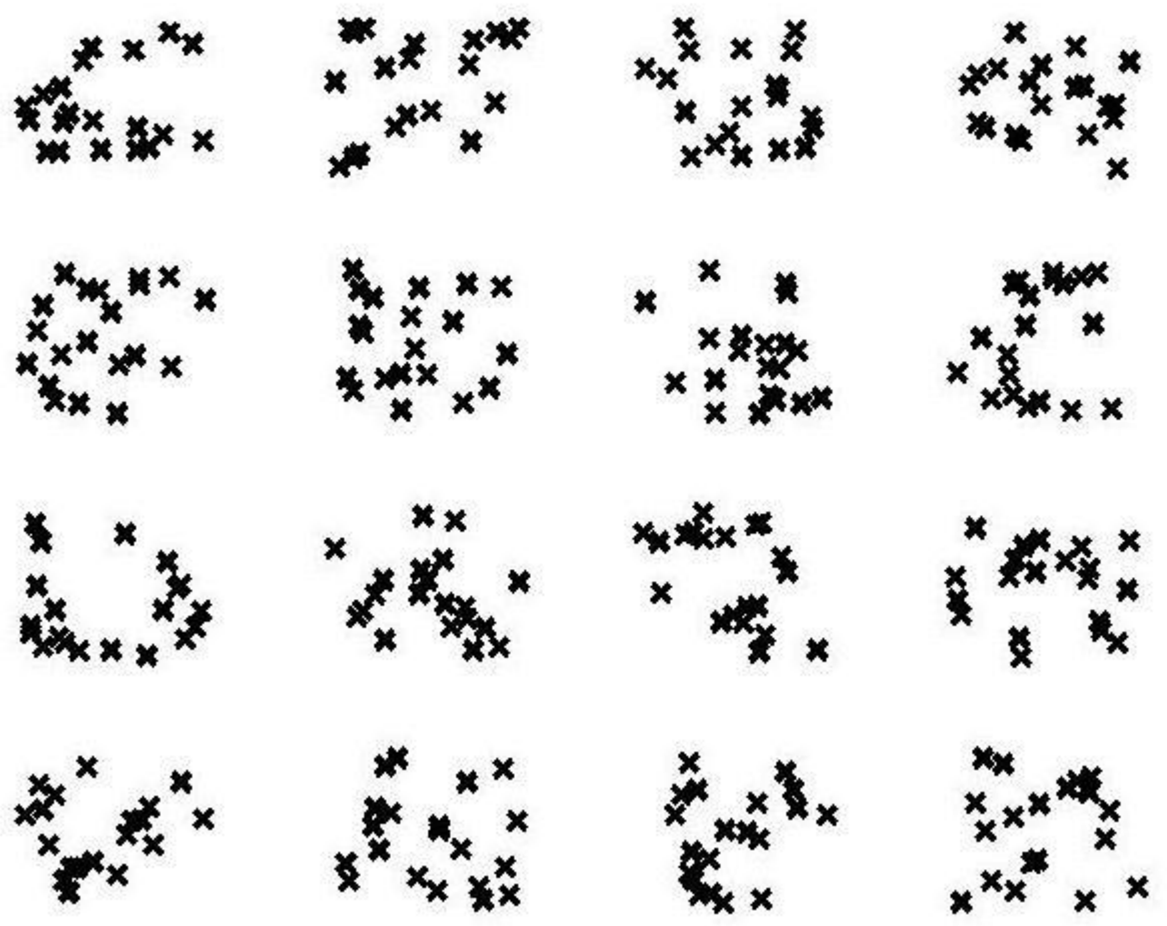
$X = \{x_1, x_2, \ldots\}$, $C = \{c_1, c_2, \ldots, c_k\}$

Euclidean Distance:

$d(x, x') = [(x - x')^T (x - x')]^{1/2}$

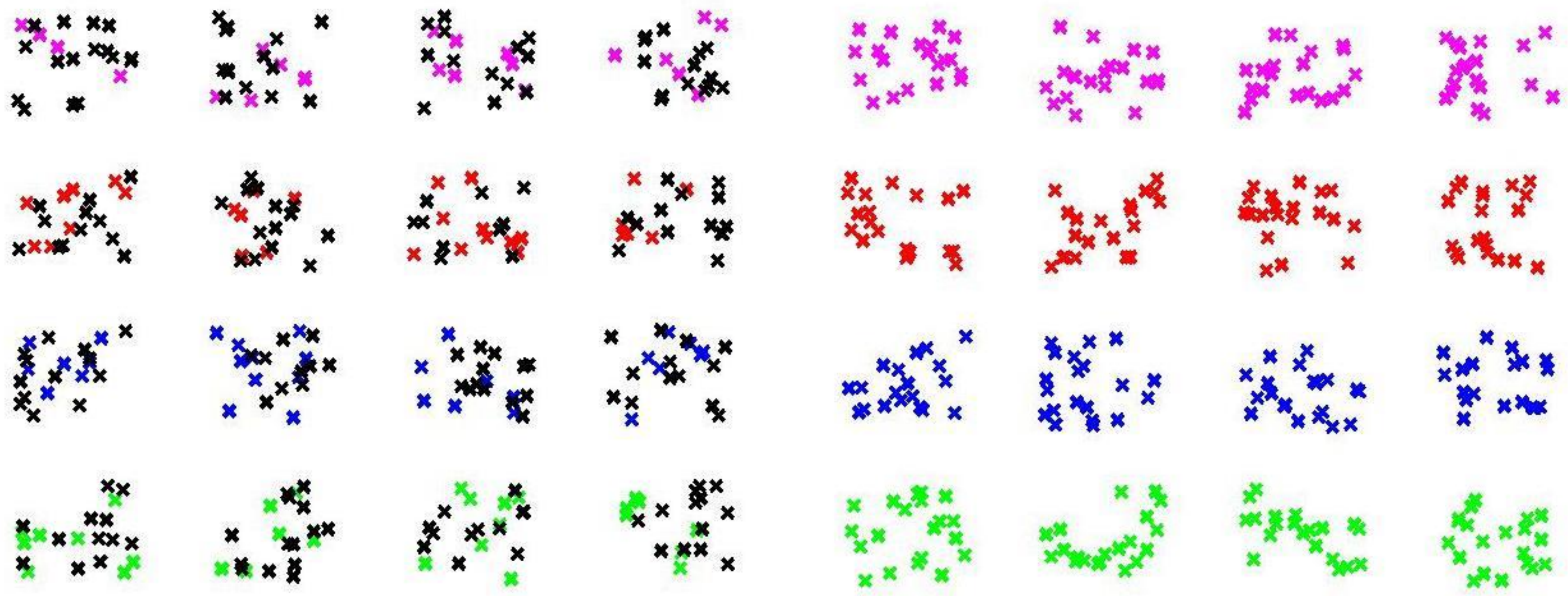
- Unsupervised, without learning.
- Metric learning and supervision: (Mooney etc. 03, 04, Xing etc. 03, Schultz & Joachims 03, Bach & Jordan03) Li & Roth'05

Supervision in Clustering



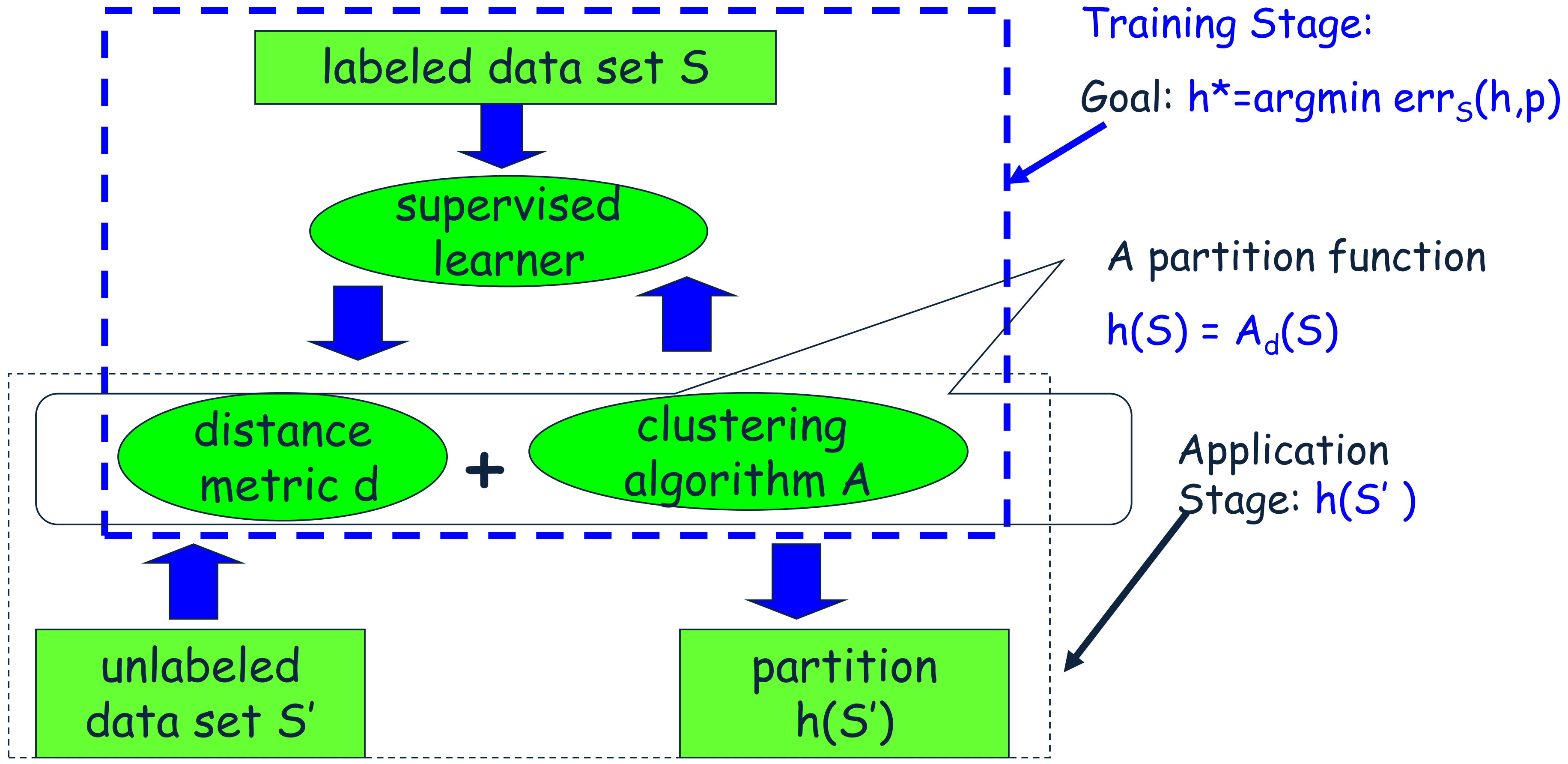
K=4

Supervision in Clustering



Supervised Discriminative Clustering (SDC)

(based on Li & Roth 2005)



Learning partitioning function h by learning metric d

Supervised Metric Learning:

- Given a data set S,
- a fixed clustering algorithm A
- supervision $p(S) = \{(x_i, c_i)\}_1^m$,

the training process tries to find d^*, minimizing the clustering error:

$$d^* = \text{argmin}_d \, err_S(h,p), \quad \text{where } h(S) = A_d(S).$$

Clustering Error

Pairwise error:

$$B_{ij} \equiv I[p(x_i) \neq p(x_j) \;\&\; h(x_i) = h(x_j)]$$

$$err_S^1(h,p) \equiv \frac{1}{|S|^2} \sum_{x_i,x_j \in S} \left[d(x_i,x_j)^2 \cdot A_{ij} + (d_{max}^2 - d(x_i,x_j)^2) \cdot B_{ij} \right]$$

$$A_{ij} \equiv I[p(x_i) = p(x_j) \;\&\; h(x_i) \neq h(x_j)]$$

K-Means Intra-cluster Error:

$$err_S^2(h,p) \equiv \frac{1}{|S|} \left| \sum_{k=1}^{K} \sum_{x \in S'_k} d(x, \mu'_k)^2 - \sum_{k=1}^{K} \sum_{x \in S_k} d(x, \mu''_k)^2 \right|$$

Mean of h

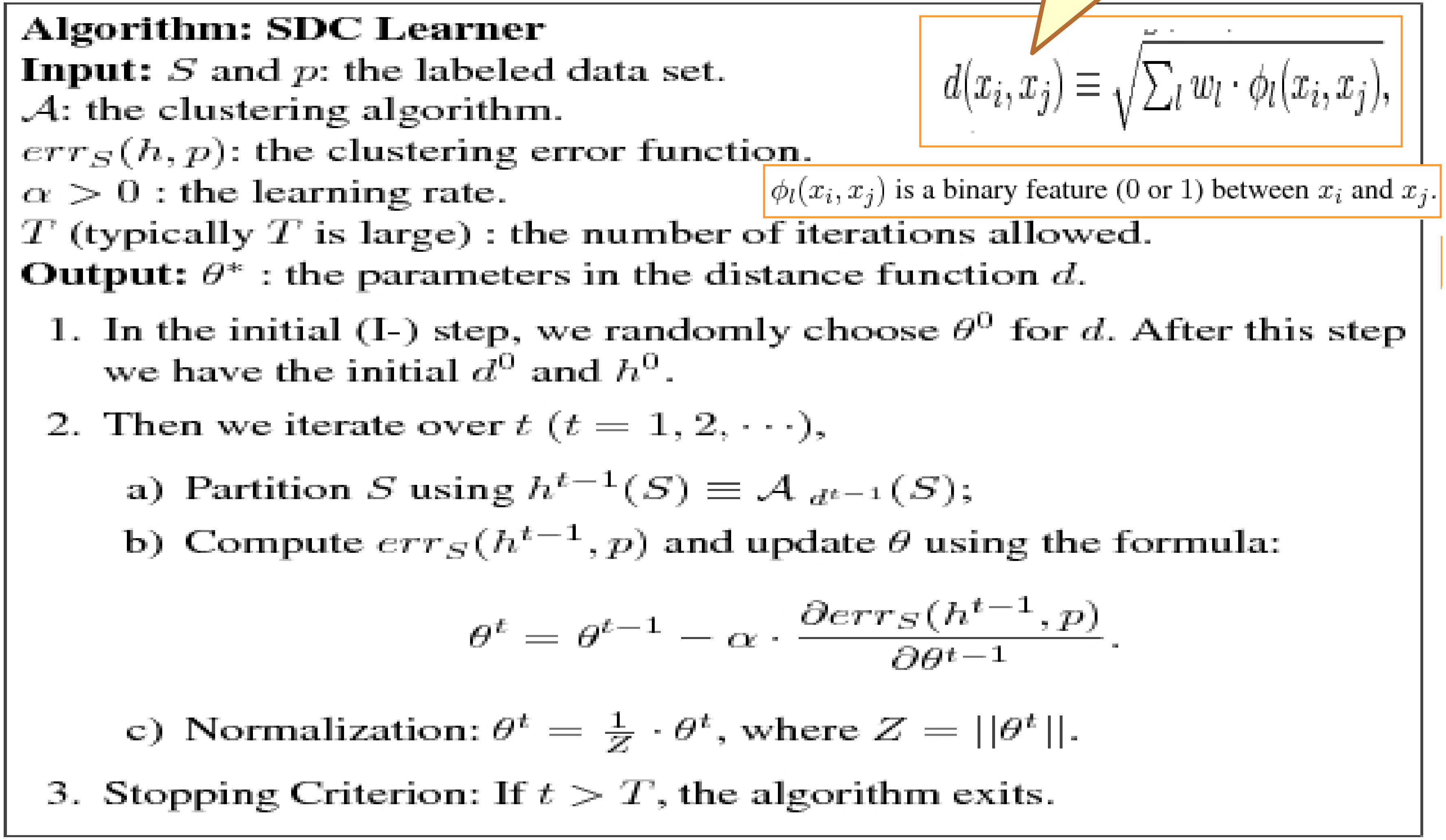
Mean of P

Pairwise intra-cluster error:

$$err_S^3(h,p) \equiv \frac{1}{|S|^2} \left| \sum_{k=1}^{K} \sum_{x_i,x_j \in S'_k} d(x_i,x_j)^2 - \sum_{k=1}^{K} \sum_{x_i,x_j \in S_k} d(x_i,x_j)^2 \right|$$

Algorithm

How to parameterize the distance function



$$d(x_i, x_j) \equiv \sqrt{\textstyle\sum_l w_l \cdot \phi_l(x_i, x_j)},$$

$\phi_l(x_i, x_j)$ is a binary feature (0 or 1) between x_i and x_j.

```
Algorithm: SDC Learner
Input: $S$ and $p$: the labeled data set.
$\mathcal{A}$: the clustering algorithm.
$err_S(h, p)$: the clustering error function.
$\alpha > 0$ : the learning rate.
$T$ (typically $T$ is large) : the number of iterations allowed.
Output: $\theta^*$ : the parameters in the distance function $d$.
  1. In the initial (I-) step, we randomly choose $\theta^0$ for $d$. After this step
     we have the initial $d^0$ and $h^0$.
  2. Then we iterate over $t$ ($t = 1, 2, \cdots$),
     a) Partition $S$ using $h^{t-1}(S) \equiv \mathcal{A}_{d^{t-1}}(S)$;
     b) Compute $err_S(h^{t-1}, p)$ and update $\theta$ using the formula:
        $\theta^t = \theta^{t-1} - \alpha \cdot \frac{\partial err_S(h^{t-1}, p)}{\partial \theta^{t-1}}$.
     c) Normalization: $\theta^t = \frac{1}{Z} \cdot \theta^t$, where $Z = ||\theta^t||$.
  3. Stopping Criterion: If $t > T$, the algorithm exits.
```