Unsupervised Clustering

Unsupervised Clustering

- Training samples are *not* labeled
- May not know
  - how many classes
  - *a prior* probability $P(\varpi_i)$
  - state-conditional probability $p(x|\varpi_i)$
- Automatic discovery of structures
  - Intuitively, objects in the same class stick together and form clusters

Unsupervised Clustering (cont)

- Locating groups (clusters) having similar measurements

$$Given \quad \aleph = \{x_1, x_2, ..., x_n\} (unlabelled)$$
$$partition\ in\ to\ \ C\ clusters\ \ \aleph_1, \aleph_2, ..., \aleph_C$$

Similarity Measure

- Need a similarity measurement s(**x**,**x**')
  - (e.g., distance between **x** and **x**')

Minkowski Metric

$$d(\mathbf{x},\mathbf{y}) = \left(\sum_{k=1}^{d} |\mathbf{x}_k - \mathbf{y}_k|^q\right)^{\frac{1}{q}}, q \geq 1$$

Mahalanobi s Distance

$$d(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^t \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\mathbf{y})$$

Normalized inner product

$$d(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

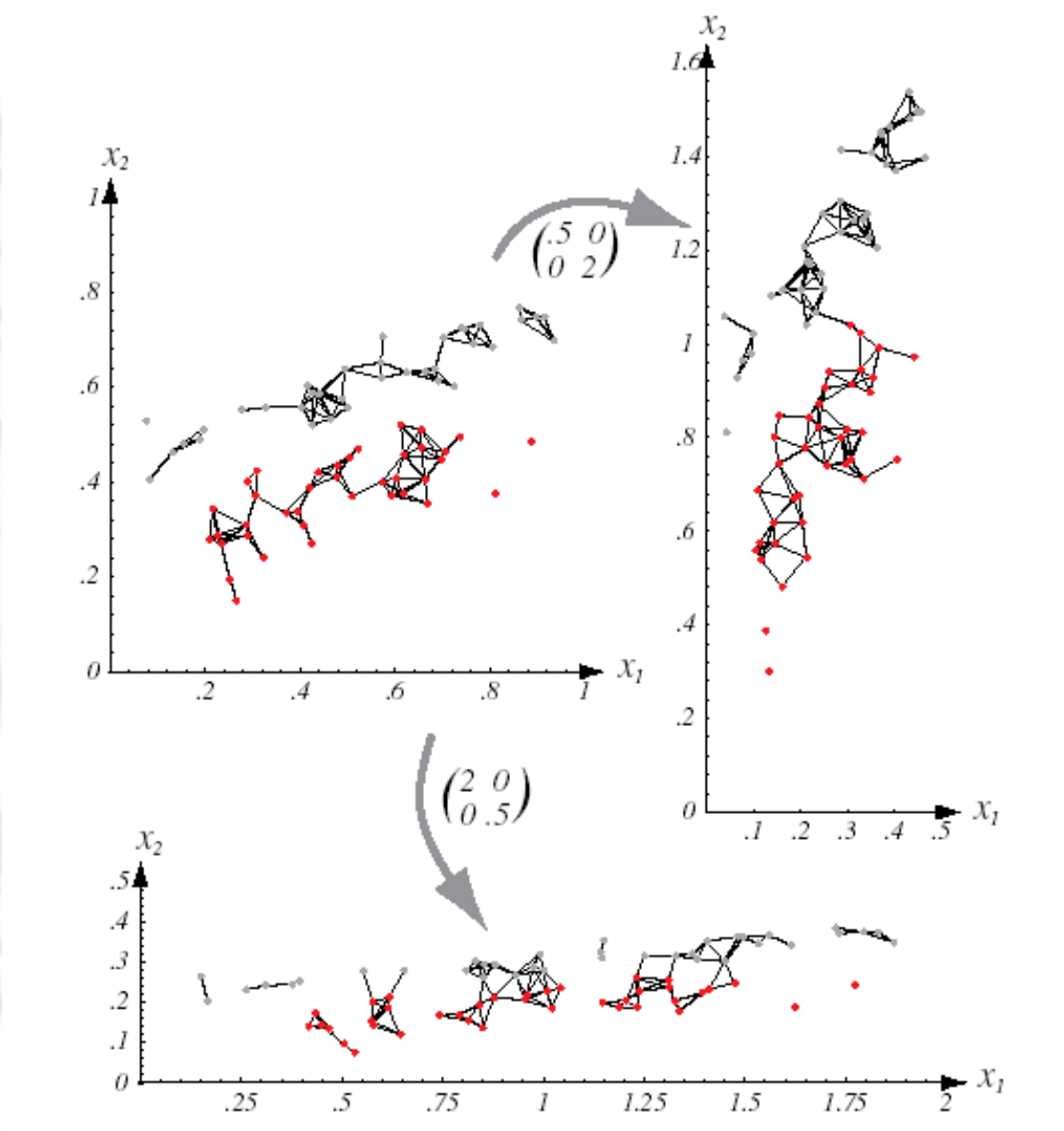
Commonalit y (for binary features)

$$d(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{d}$$

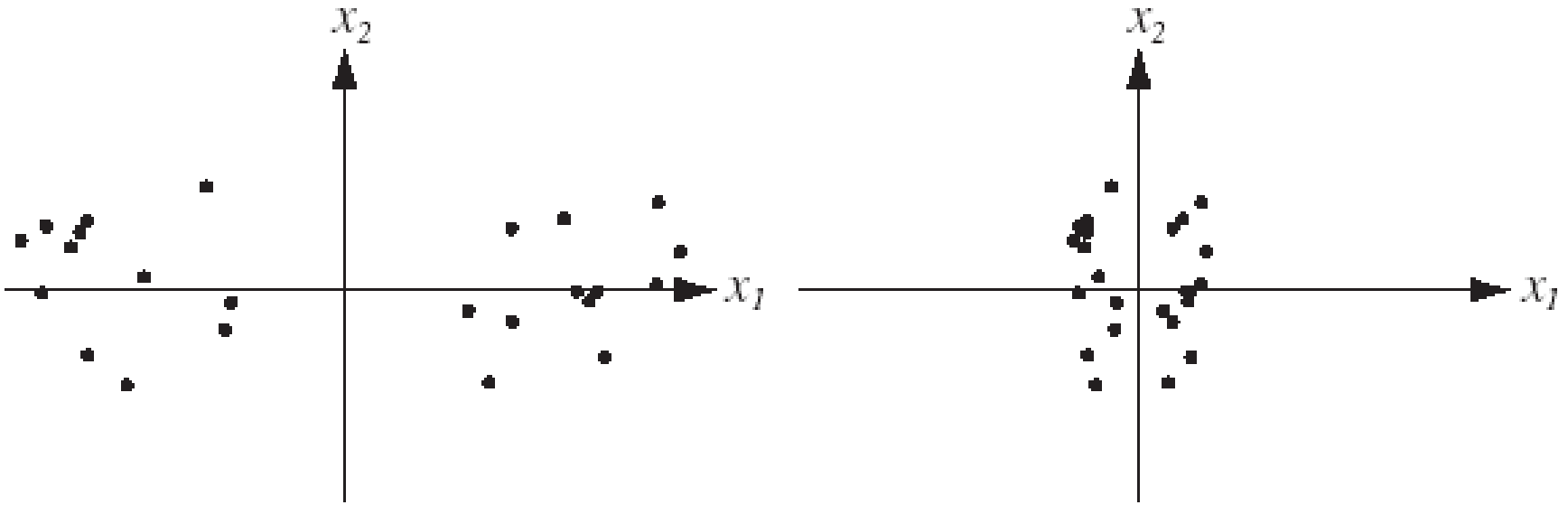
Similarity Measure (cont.)

- How to set the threshold?
  - Too large all samples assigned into a single class
  - Too small each sample in its own class
- How to properly weight each feature?
  - How do brightness of a fish and length of a fish relate?
  - How to scale (or normalize) different measures?

Axis Scaling

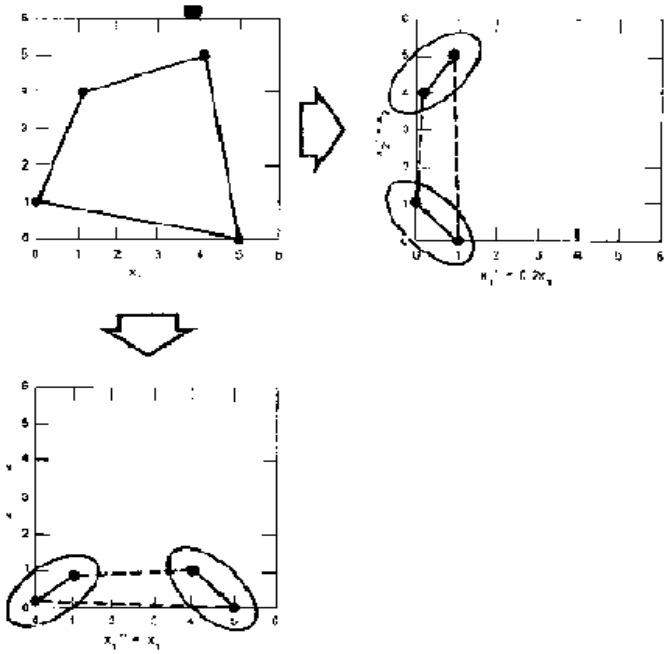


Axis Scaling (cont.)



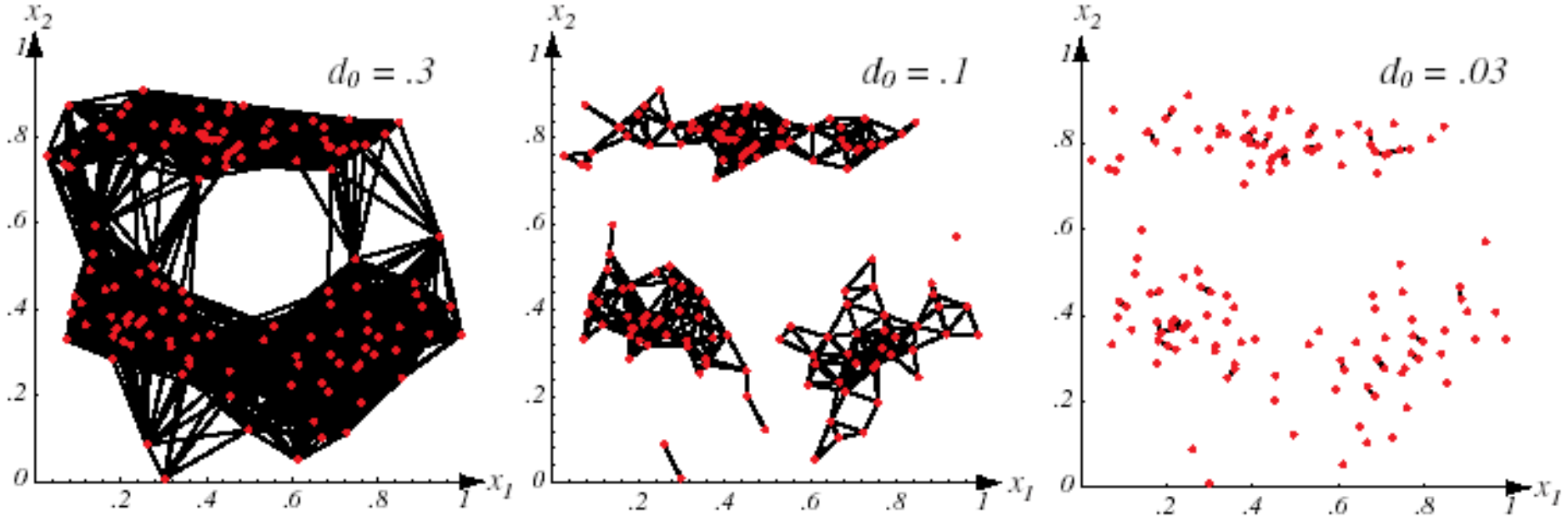
Pathology of Similarity Measure

1. Scaling the samples changes the apparent clustering.



How do you compare the brightness of fish in one dimension with the length of the fish in another dimension?

Threshold



Criteria for Clustering

- A criterion function for clustering

$$J_e = \sum_{i=1}^{c} \sum_{x \in \aleph_i} |\mathbf{x} - \mathbf{m}_i|^2 \qquad \mathbf{m}_i = \frac{1}{n_i} \sum_{x \in \aleph_i} \mathbf{x}$$

$$J = \sum_{i=1}^{c} \frac{1}{n_i} \sum_{x \in \aleph_i} \sum_{x' \notin \aleph_i} |\mathbf{x} - \mathbf{x}'|^2$$

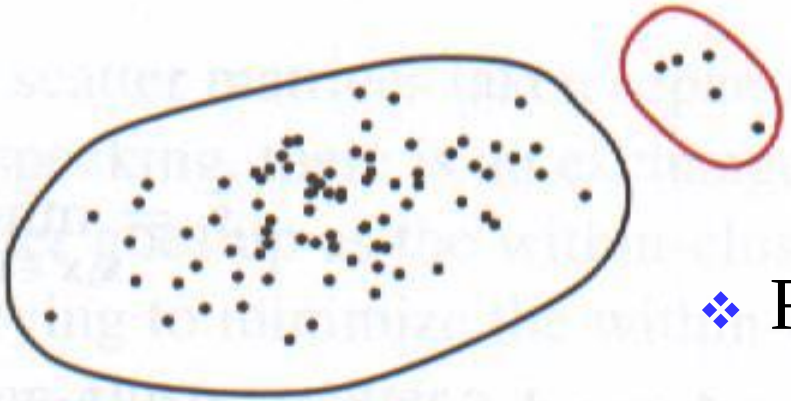
Criteria for Clustering (cont.)

- *Within* and *Between* group variance
  - *minimize* and *maximize* the trace and determinant of the appropriate scatter matrix
  - ***trace***: square of the scattering radius
  - ***determinant***: square of the scattering volume

$$\mathbf{S}_W = \sum_{i=1}^{c} \sum_{x \in \aleph_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t \qquad \mathbf{m}_i = \frac{1}{n_i} \sum_{x \in \aleph_i} \mathbf{x}$$

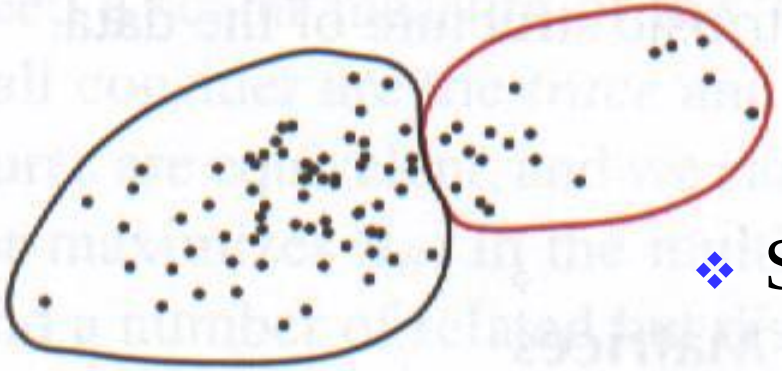
$$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t \qquad \mathbf{m} = \frac{1}{n} \sum_{x \in \aleph} \mathbf{x}$$

Pathology: when class sizes differ



$J_e = large$

- Relaxed constraint
  - Allow the large class to grow into smaller class



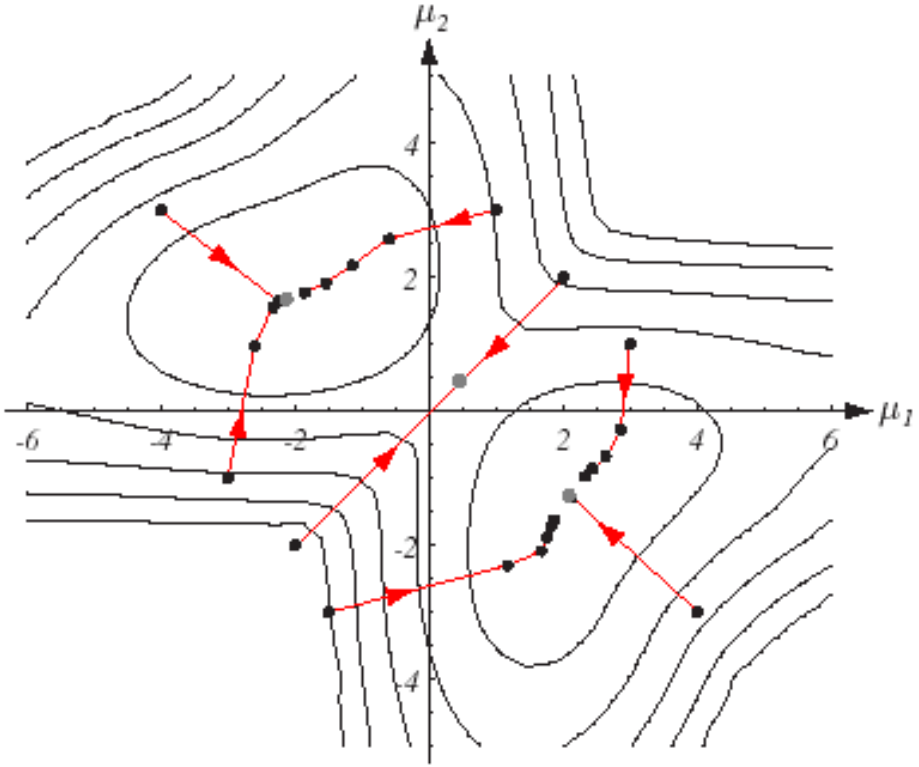
$J_e = small$

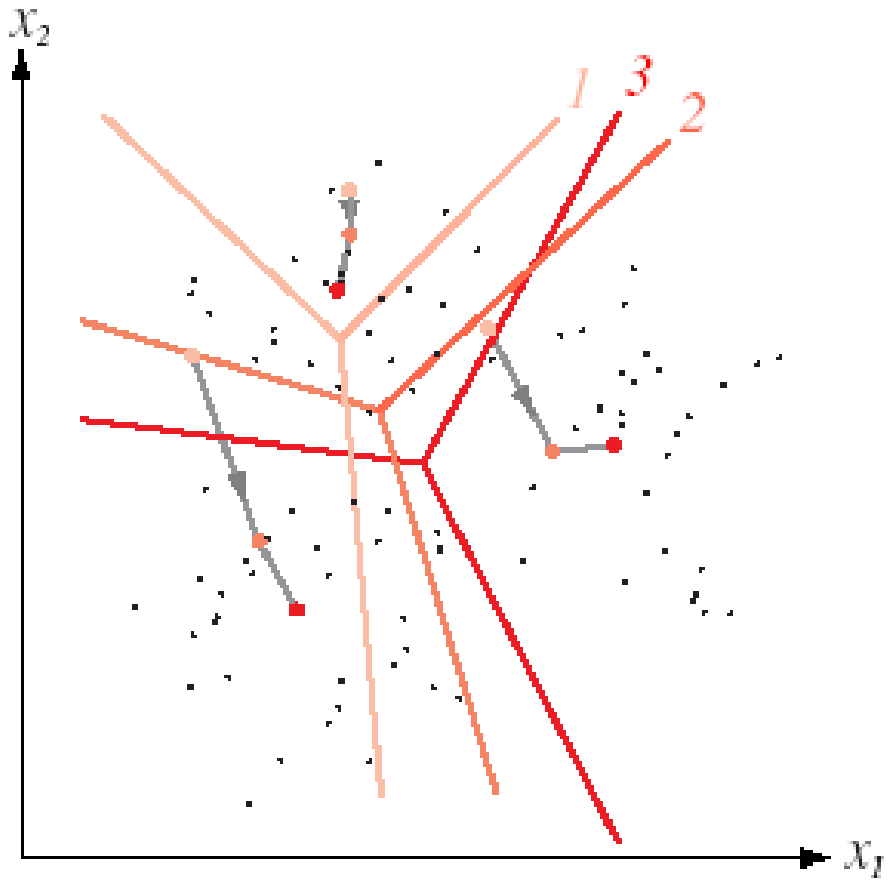
- Stringent constraint
  - Disallow large class and split it

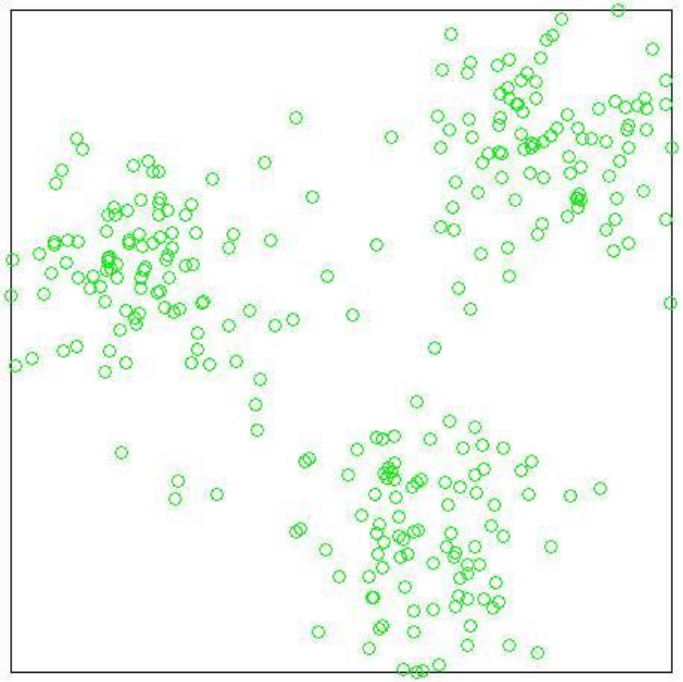
$$J_e = \sum_{i=1}^{c} \sum_{x \in \aleph_i} |\mathbf{x} - \mathbf{m}_i|^2 \qquad \mathbf{m}_i = \frac{1}{n_i} \sum_{x \in \aleph_i} \mathbf{x}$$

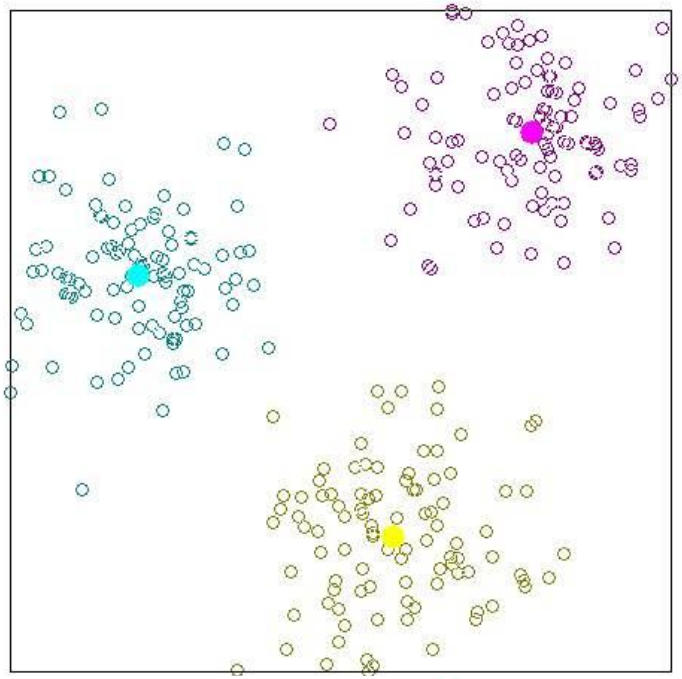
K-Means Algorithm (fixed # of clusters)

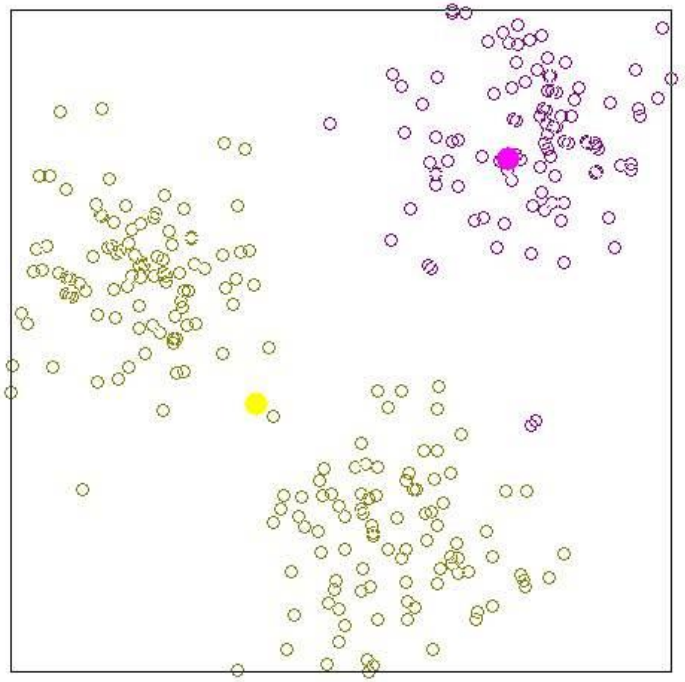
- Arbitrarily pick *N* cluster centers, assign samples to nearest center
- Compute sample mean of each cluster
- Reassign samples to clusters with the nearest mean (for all samples)
- Repeat if there are changes, otherwise stop

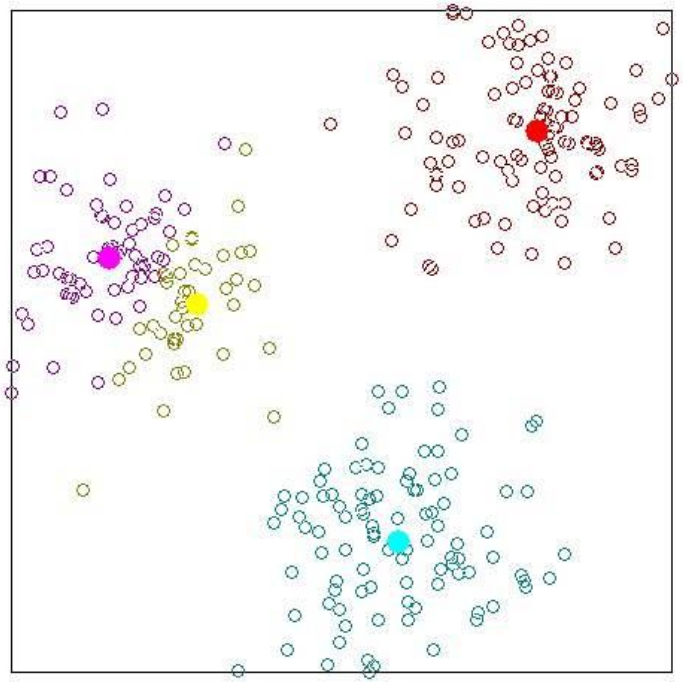


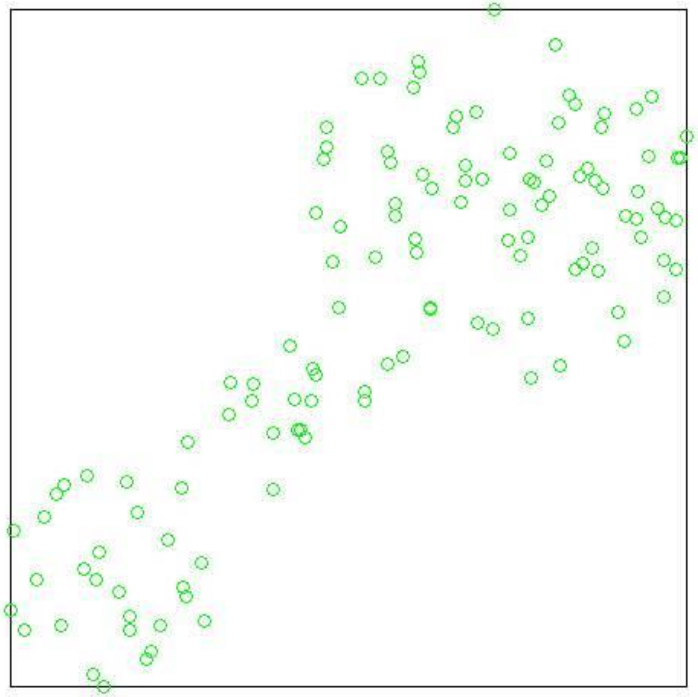


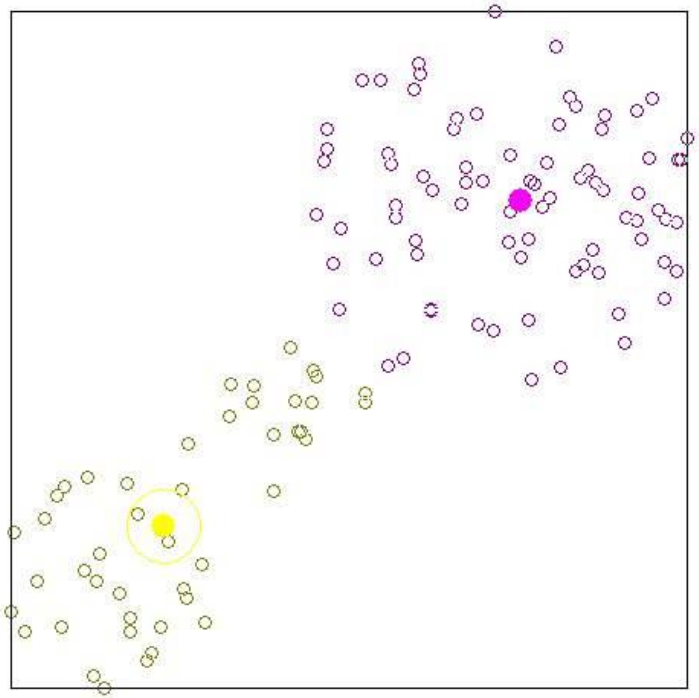


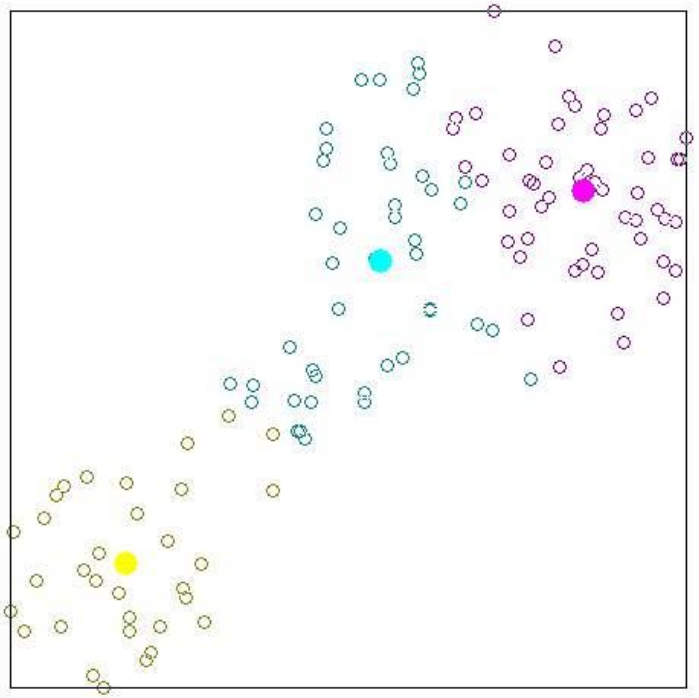












K-Means

- In some sense, it is a simplified version of the case II of learning parametric form

$$\hat{P}(w_i \mid x_k, \hat{\theta}) = \frac{p(x_k \mid w_i, \hat{\theta}_i)\hat{P}(w_i)}{\sum_{j=1}^{c} p(x_k \mid w_j, \hat{\theta}_j)\hat{P}(w_j)}$$

- In stead of multiple membership, give the sample to whichever class whose mean is closest to it

$$\hat{P}(w_i \mid x_k, \hat{\theta}) = \begin{cases} 1 & \text{class with closest class mean} \\ 0 & \textit{otherwise} \end{cases}$$

Fuzzy K-Means

- In Matlab, you can find k-mean (it is called c-mean) under fuzzy logic toolbox
- It implements fuzzy k-mean
  - Where membership function is not 1 or 0, but can be fractional

Iterative K-means

- Iterative refinement of clusters to minimize

$$J = \sum_{i=1}^{c} \sum_{x \in \aleph_i} |\mathbf{x} - \mathbf{m}_i|^2 \qquad \mathbf{m}_i = \frac{1}{n_i} \sum_{x \in \aleph_i} \mathbf{x}$$

- Each step: move a sample from one to another

$\hat{\mathrm{x}}$ from $\aleph_i$ to $\aleph_j$

$$\mathbf{m}_i^* = \mathbf{m}_i - \frac{\hat{\mathbf{x}} - \mathbf{m}_i}{n_i - 1} \qquad J_i^* = J_i - \frac{n_i}{n_i - 1} |\hat{\mathbf{x}} - \mathbf{m}_i|^2$$

$$\mathbf{m}_j^* = \mathbf{m}_j + \frac{\hat{\mathbf{x}} - \mathbf{m}_j}{n_j + 1} \qquad J_j^* = J_j + \frac{n_j}{n_j + 1} |\hat{\mathbf{x}} - \mathbf{m}_j|^2$$

$$\frac{n_i}{n_i - 1} |\hat{\mathbf{x}} - \mathbf{m}_i|^2 > \frac{n_j}{n_j + 1} |\hat{\mathbf{x}} - \mathbf{m}_j|^2$$

1. Select an initial partition of the *n* samples into clusters and compute $J, m_1, \ldots, m_c$
2. Select the next candidate sample $\hat{x}$
3. If the current cluster is larger than 1, then

$$\rho_j = \begin{cases} \dfrac{n_j}{n_j - 1} \| \hat{\mathbf{x}} - \mathbf{m}_j \|^2 & j = i \\ \dfrac{n_j}{n_j + 1} \| \hat{\mathbf{x}} - \mathbf{m}_j \|^2 & j \neq i \end{cases}$$

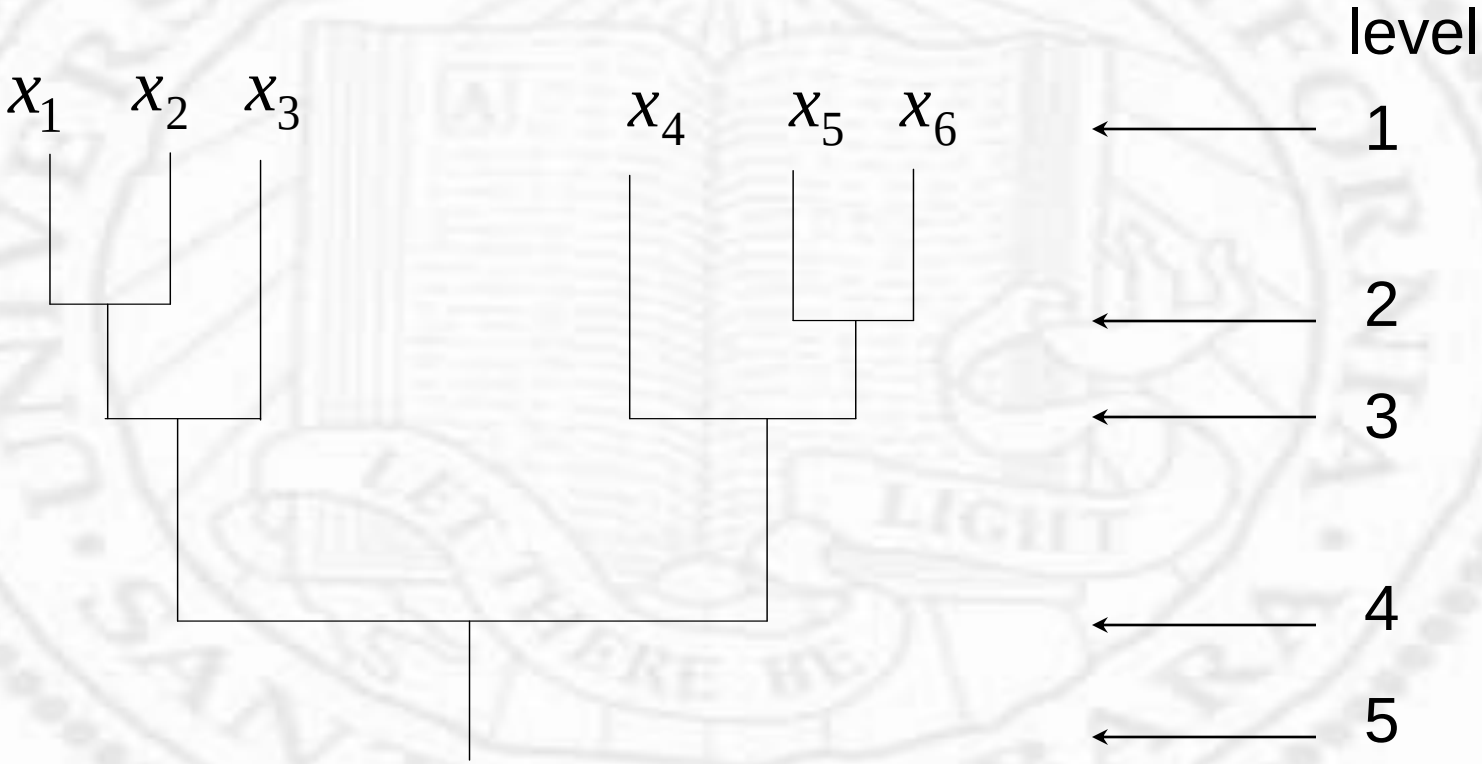
4. Transfer $\hat{x}$ to $\aleph_k$ if $\rho_k \leq \rho_j$ for all j
5. Update J, m_i, m_k
6. If *J* has not changed in *n* attempts, stop, otherwise go back to 2.

Hierarchical Clustering

- K-means assume a “flat” data description
- Hierarchical descriptions are more frequently

Hierarchical clustering (cont.)

- Samples in the same cluster will remain so at a higher level



Hierarchical clustering Options

- Bottom-up: agglomerate
- Top-down: divisive

1. *Let* $\hat{c} = n$ *and* $\aleph = \{x_1,\ldots,x_n\}$
2. *If* $\hat{c} \leq c$, *stop*
3. *Find the nearest pair of distinct clusters,* $\aleph_i$ *and* $\aleph_j$
4. *Merge* $\aleph_i$ *and* $\aleph_j$, *delete* $\aleph_j$, *and decrement* $\hat{c}$ *by* 1
5. *Go to* 2.

Hierarchical clustering Procedure

- At a certain step, we have *c* classes

$$|\aleph_i| = n_i, \mathbf{m}_i = \frac{1}{n_i}\sum_{x\in\aleph_i}\mathbf{x}$$

Merge two clusters *i* and *j*

$$\aleph_{ij} = \aleph_i \cup \aleph_j$$

$$|\aleph_{ij}| = n_i + n_j, m_{ij} = \frac{1}{n_i + n_j}\sum_{x\in\aleph_{ij}} x = \frac{1}{n_i + n_j}(n_i\mathbf{m}_i + n_j\mathbf{m}_j)$$

Hierarchical clustering Procedure (cont.)

- Then certain criteria function will increase

$$\Delta E_{ij} = \sum_{x \in \aleph_{ij}} |\mathbf{x} - \mathbf{m}_{ij}|^2 - \sum_{x \in \aleph_i} |\mathbf{x} - \mathbf{m}_i|^2 - \sum_{x \in \aleph_j} |\mathbf{x} - \mathbf{m}_j|^2$$

$$= n_i \mathbf{m}_i^2 + n_j \mathbf{m}_j^2 - (n_i + n_j)\mathbf{m}_{ij}^2$$

$$= \frac{n_i n_j}{n_i + n_j} |\mathbf{m}_i - \mathbf{m}_j|^2$$

I,j** chosen in such a way

$$< i^*, j^* > = \arg\min_{i,j} \frac{n_i n_j}{n_i + n_j} |\mathbf{m}_i - \mathbf{m}_j|^2 = \arg\min_{i,j} d(\aleph_i, \aleph_j)$$

Until no such pair exist that cause an increase in the criteria function less than certain preset threshold

Criteria Function

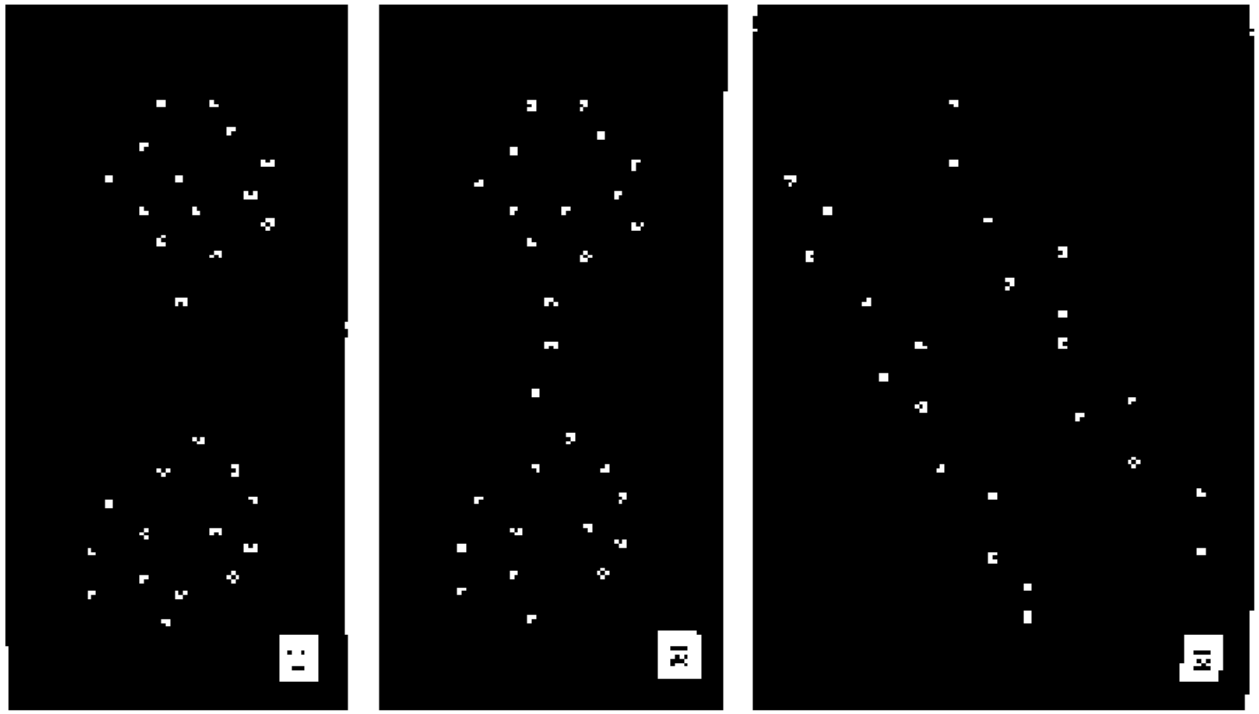
- In fact, the criteria function can be chosen in many different ways, resulting in different clustering behaviors

Nearest – neighbor (elongated classes)

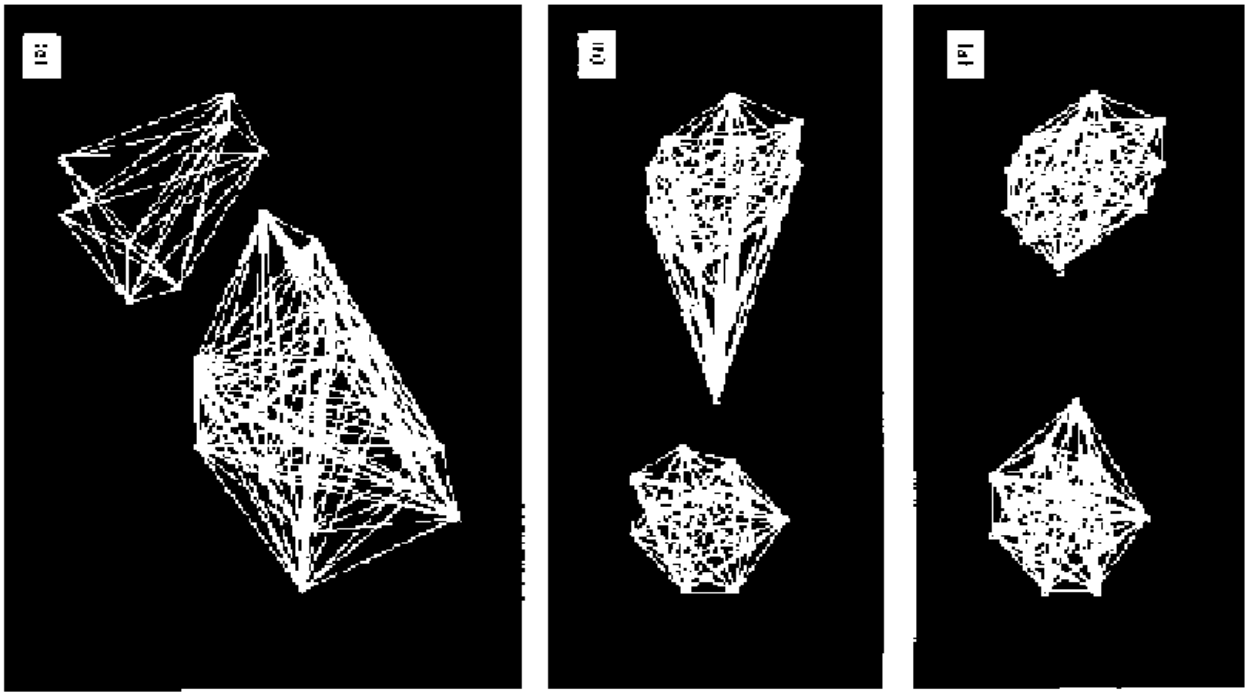
$$d_{\min}(\aleph_i, \aleph_j) = \min_{x \in \aleph_i, y \in \aleph_j} (\mathbf{x}, \mathbf{y})$$

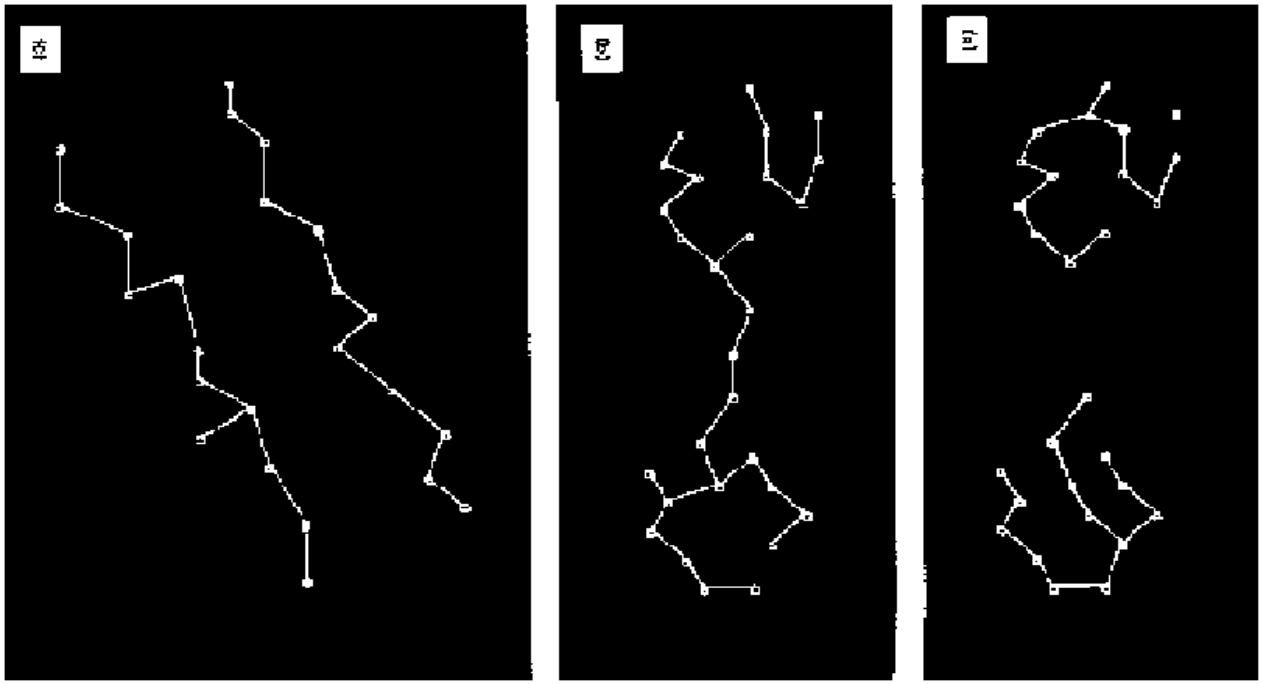
Furthest – neighbor (compact classes)

$$d_{\max}(\aleph_i, \aleph_j) = \max_{x \in \aleph_i, y \in \aleph_j} (\mathbf{x}, \mathbf{y})$$



- Starting from every sample in a cluster
- Merge them according to some criteria function
- Until two clusters exist



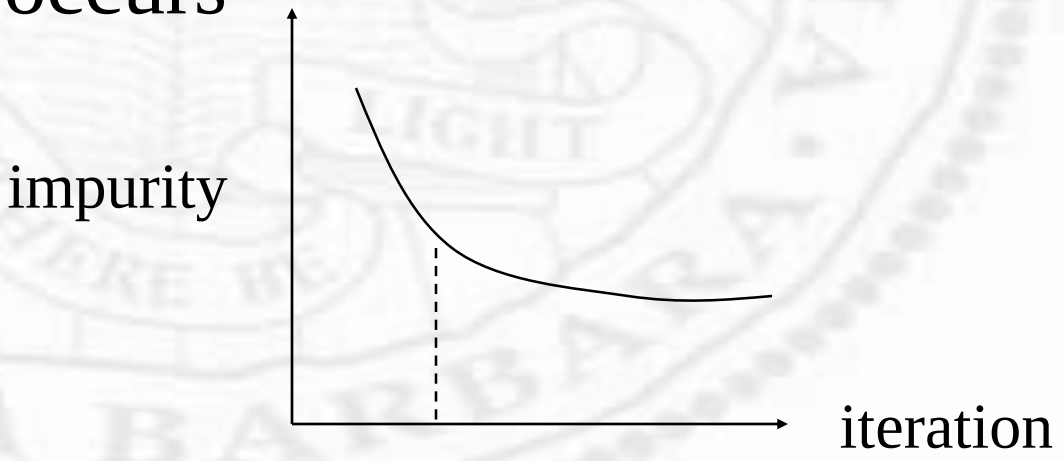


In Reality

- With *n* objects, the distance matrix is *n x n*
- For large databases, it is computationally expensive to compute and store the matrix
- Solution: storing only k nearest clusters in the distance matrix: complexity is *k x n*

Divisive Clustering

- Less often used
- Have to be careful that criterion function usually decreases monotonically (the samples become purer each time)
- Natural grouping is the one where a large drop in impurity occurs



ISODATA Algorithm

- Iterative self-organizing data analysis technique
- A tried-and-true clustering algorithm
- Dynamically update # of clusters
- Can go both top-down (split) and bottom-up (merge) directions

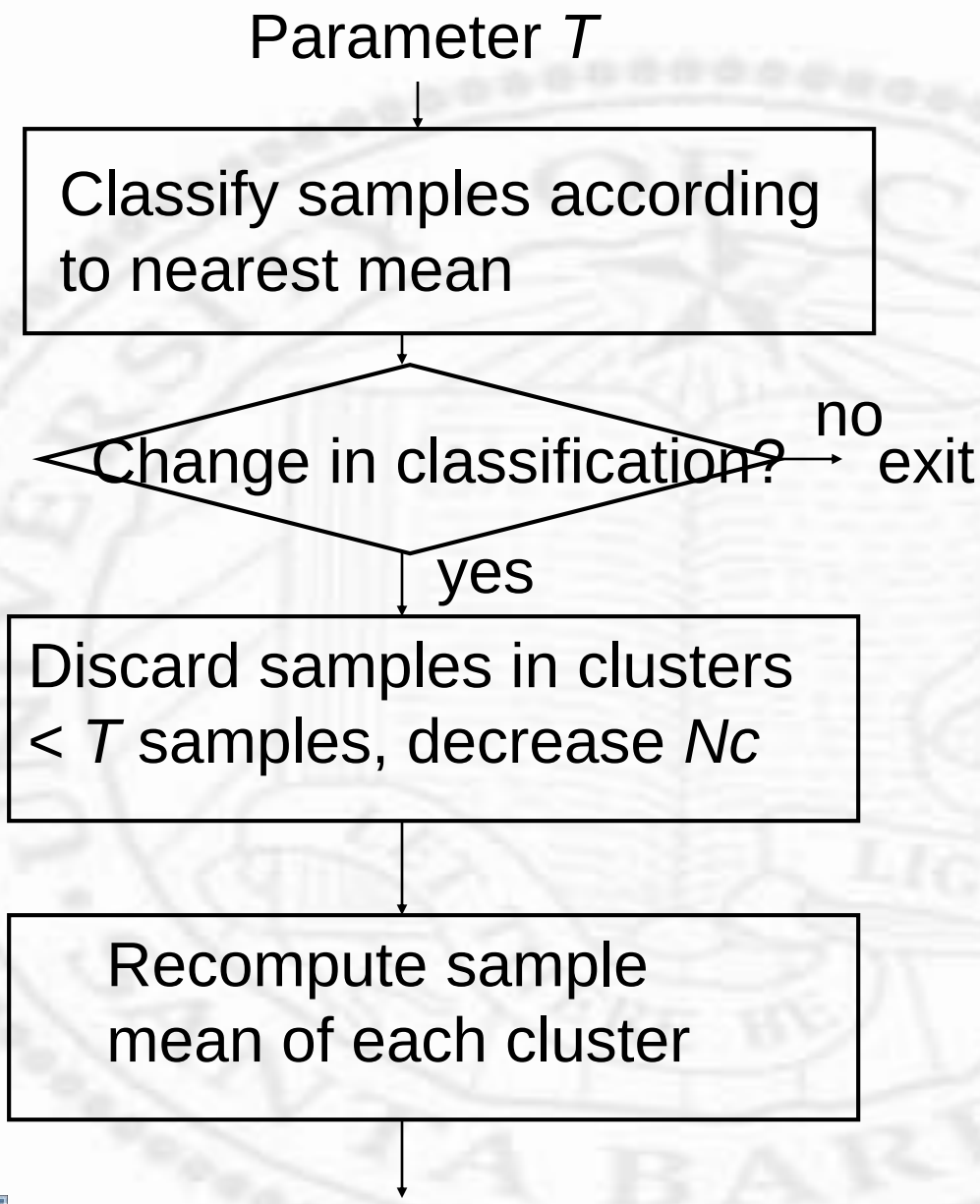
Notation:

T	threshold on number of samples in a cluster
N_D	approximat e (desired) number of clusters
σ_s	maximum spread for splitting
D_m	maximum distance for merging
$N_{\max}$	maximum number of clusters that can be merged

Algorithm

- 1. Cluster the existing data into Nc clusters but eliminate any data and classes with fewer than T members, decrease Nc. Exit when classification of samples has not changed
- 2. If iteration odd and $N_c \leq \frac{N_D}{2}$ *or* $N_c \leq 2N_D$
  - Split clusters whose samples are sufficiently disjoint, increase Nc
  - If any clusters have been split, go to 1
- 3. Merge any pair of clusters whose samples are sufficiently close
- 4. Go to step 1

Step 1


Parameter T
Classify samples according to nearest mean
Change in classification?
no
exit
yes
Discard samples in clusters < T samples, decrease Nc
Recompute sample mean of each cluster

Step 2

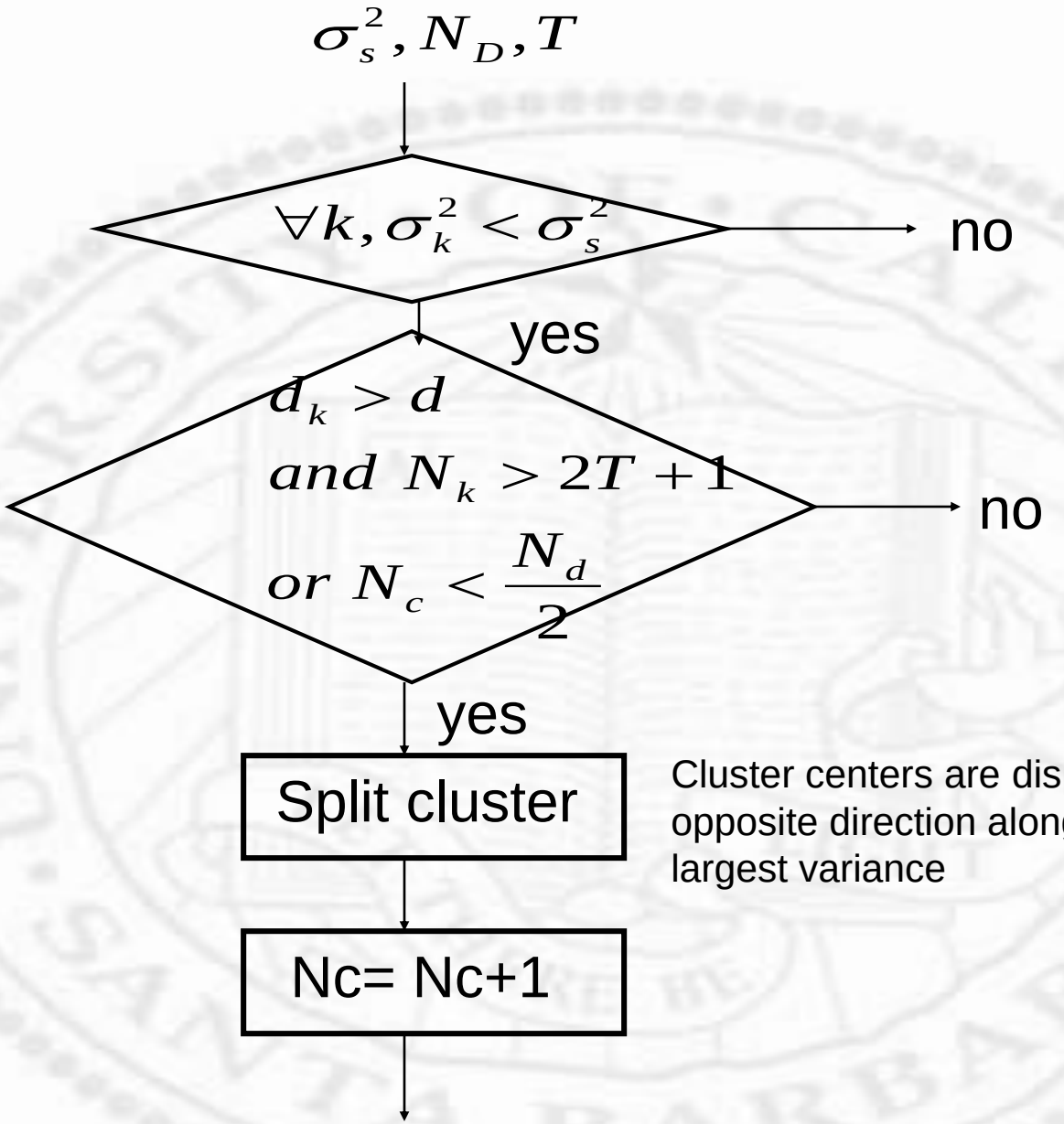
- Compute for each cluster

$$d_k = \frac{1}{N_k} \sum_{\mathbf{y}^{(i)} \in \varpi_k} | \mathbf{y}^{(i)} - \boldsymbol{\mu}_k |$$

$$\sigma_k^2 = \max_j \frac{1}{N_k} \sum_{\mathbf{y}^{(i)} \in \varpi_k} | \mathbf{y}^{(i)}{}_j - \boldsymbol{\mu}_{k\,j} |$$

Compute globally

$$d = \frac{1}{N} \sum_{k=1}^{N_c} N_k d_k$$



Cluster centers are displaced in opposite direction along the axis of largest variance

Step 3

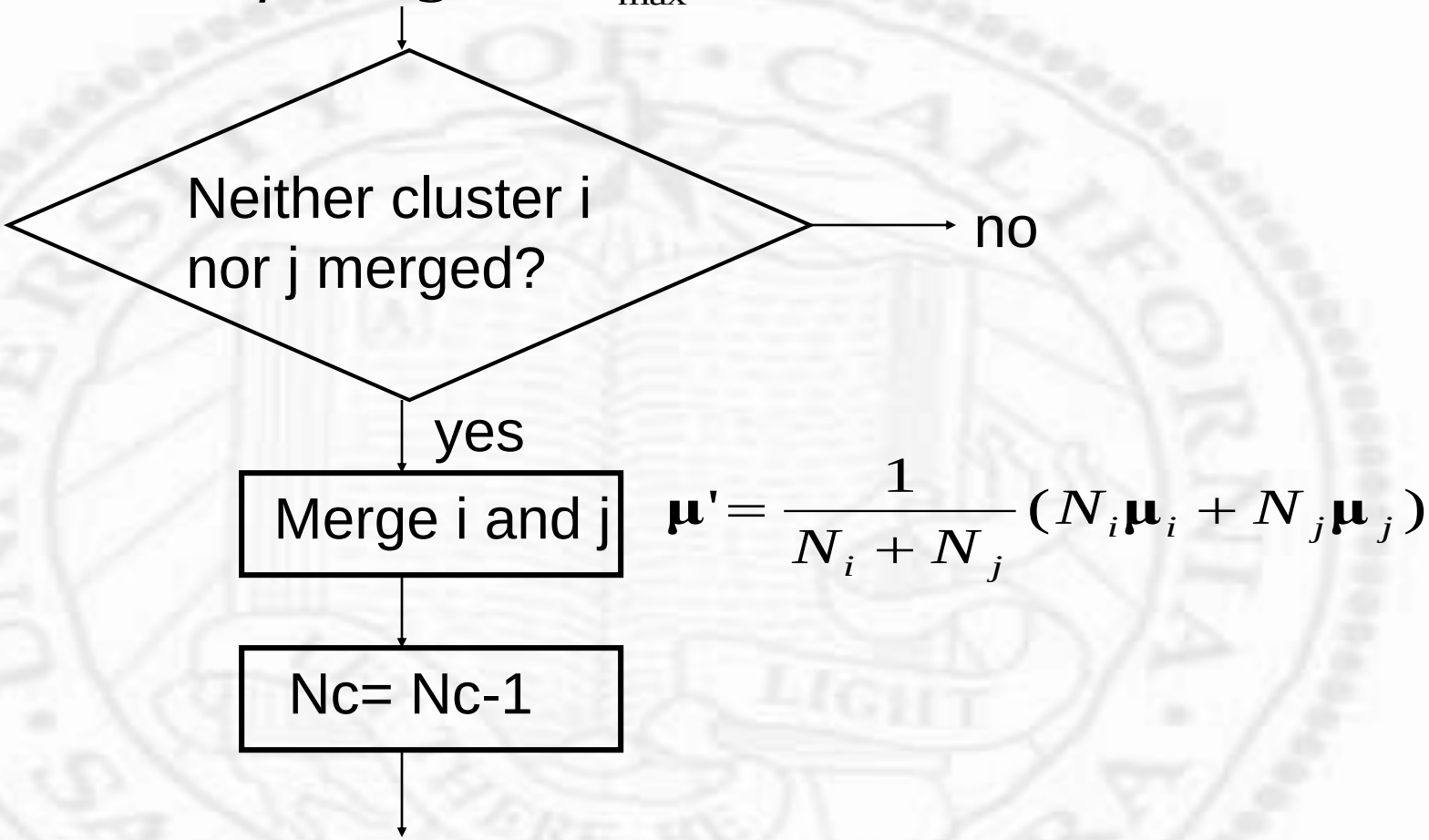
- Compute for each pairs of clusters

$$d_{ij} = | \boldsymbol{\mu}_i - \boldsymbol{\mu}_j |$$

Sort distance from smallest to largest less than *Dm*

$\forall d_{ij} < D_m$

$\#of\ \ merge < N_{\max}$



$$\boldsymbol{\mu}' = \frac{1}{N_i + N_j}(N_i\boldsymbol{\mu}_i + N_j\boldsymbol{\mu}_j)$$

Spectral Clustering

- Graph theoretical approach
- (Advanced) linear algebra is really important
- A field by itself
- Cover the basics here

Graph notations

- Undirected graph G = (V, E)
  - V = $\{v_1, \ldots, v_n\}$ are nodes (samples)
  - E = $\{e_{i,j} | 0<=i,j<n\}$ are edges, with weights (similarity) $w_{i,j}$
  - W: adjacency matrix with entries $w_{i,j}$
  - D: degree matrix (diagonal) with entries $d_i = \sum_{j=1}^{n} w_{ij}.$
  - A: subset of vertices, \bar{A}: complement V\A
  - $1_A = (f_1, \ldots f_n)^T$, fi=1 if i in A and 0 otherwise

More graph notations

- Size of subset A in V
  - Based on # of vertices $|A| :=$ the number of vertices in A
  - Based on its connections $\mathrm{vol}(A) := \sum_{i \in A} d_i.$
- Connected component A
  - Any two vertices in A can be joined by a path where all intermediate points lie in A
  - No connection between A and \bar{A}
- Partition with $A_1, \ldots, A_k$, if

$$A_i \cap A_j = \emptyset$$

$$A_1 \cup \ldots \cup A_k = V$$

Similarity Definitions

- RBF (fully connected or thresholded):
  - E.g, Gaussian kernel gives $w_{i,j}$ $\exp(-\|x_i - x_j\|^2/(2\sigma^2))$
- ε-neighborhood graph:
  - Connect all points whose pairwise distances less than ε
- K-nearest neighbor:
  - v_i and v_j are neighbors if either one is a kNN of the other
- Mutual k-nearest neighbor:
  - v_i and v_j are neighbors if both are a kNN of the other

Graph Laplacian: L = D - W

Proposition 1 (Properties of L) *The matrix L satisfies the following properties:*

1. *For every vector $f \in \mathbb{R}^n$ we have*

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

2. *L is symmetric and positive semi-definite.*
3. *The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$.*
4. *L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.*

Proof.
Part (1): By the definition of d_i,

$$f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

Part (2): The symmetry of L follows directly from the symmetry of W and D. The positive semi-definiteness is a direct consequence of Part (1), which shows that $f'Lf \geq 0$ for all $f \in \mathbb{R}^n$.
Part (3): Obvious.
Part (4) is a direct consequence of Parts (1) - (3). □

One connected component

Proposition 2 (Number of connected components and the spectrum of L) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.*

Proof. We start with the case $k = 1$, that is the graph is connected. Assume that f is an eigenvector with eigenvalue 0. Then we know that

$$0 = f'Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

As the weights w_{ij} are non-negative, this sum can only vanish if all terms $w_{ij}(f_i - f_j)^2$ vanish. Thus, if two vertices v_i and v_j are connected (i.e., $w_{ij} > 0$), then f_i needs to equal f_j. With this argument we can see that f needs to be constant for all vertices which can be connected by a path in the graph. Moreover, as all vertices of a connected component in an undirected graph can be connected by a path, f needs to be constant on the whole connected component. In a graph consisting of only one connected component we thus only have the constant one vector $\mathbb{1}$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.

Multiple connected components

Proposition 2 (Number of connected components and the spectrum of L) *Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_k}$ of those components.*

Now consider the case of k connected components. Without loss of generality we assume that the vertices are ordered according to the connected components they belong to. In this case, the adjacency matrix W has a block diagonal form, and the same is true for the matrix L:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}$$

Note that each of the blocks L_i is a proper graph Laplacian on its own, namely the Laplacian corresponding to the subgraph of the i-th connected component. As it is the case for all block diagonal matrices, we know that the spectrum of L is given by the union of the spectra of L_i, and the corresponding eigenvectors of L are the eigenvectors of L_i, filled with 0 at the positions of the other blocks. As each L_i is a graph Laplacian of a connected graph, we know that every L_i has eigenvalue 0 with multiplicity 1, and the corresponding eigenvector is the constant one vector on the i-th connected component. Thus, the matrix L has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components. □

Normalized Laplacian

$$L_{\text{sym}} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$
$$L_{\text{rw}} := D^{-1} L = I - D^{-1} W.$$

Proposition 3 (Properties of L_{sym} and L_{rw}) *The normalized Laplacians satisfy the following properties:*

1. *For every $f \in \mathbb{R}^n$ we have*

$$f' L_{sym} f = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2 .$$

2. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ is an eigenvalue of L_{sym} with eigenvector $w = D^{1/2} u$.*

3. *λ is an eigenvalue of L_{rw} with eigenvector u if and only if λ and u solve the generalized eigenproblem $Lu = \lambda D u$.*

4. *0 is an eigenvalue of L_{rw} with the constant one vector $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of L_{sym} with eigenvector $D^{1/2}\mathbb{1}$.*

5. *L_{sym} and L_{rw} are positive semi-definite and have n non-negative real-valued eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$.*

Unnormalized Spectral Clustering

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k eigenvectors $u_1, \ldots, u_k$ of L.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

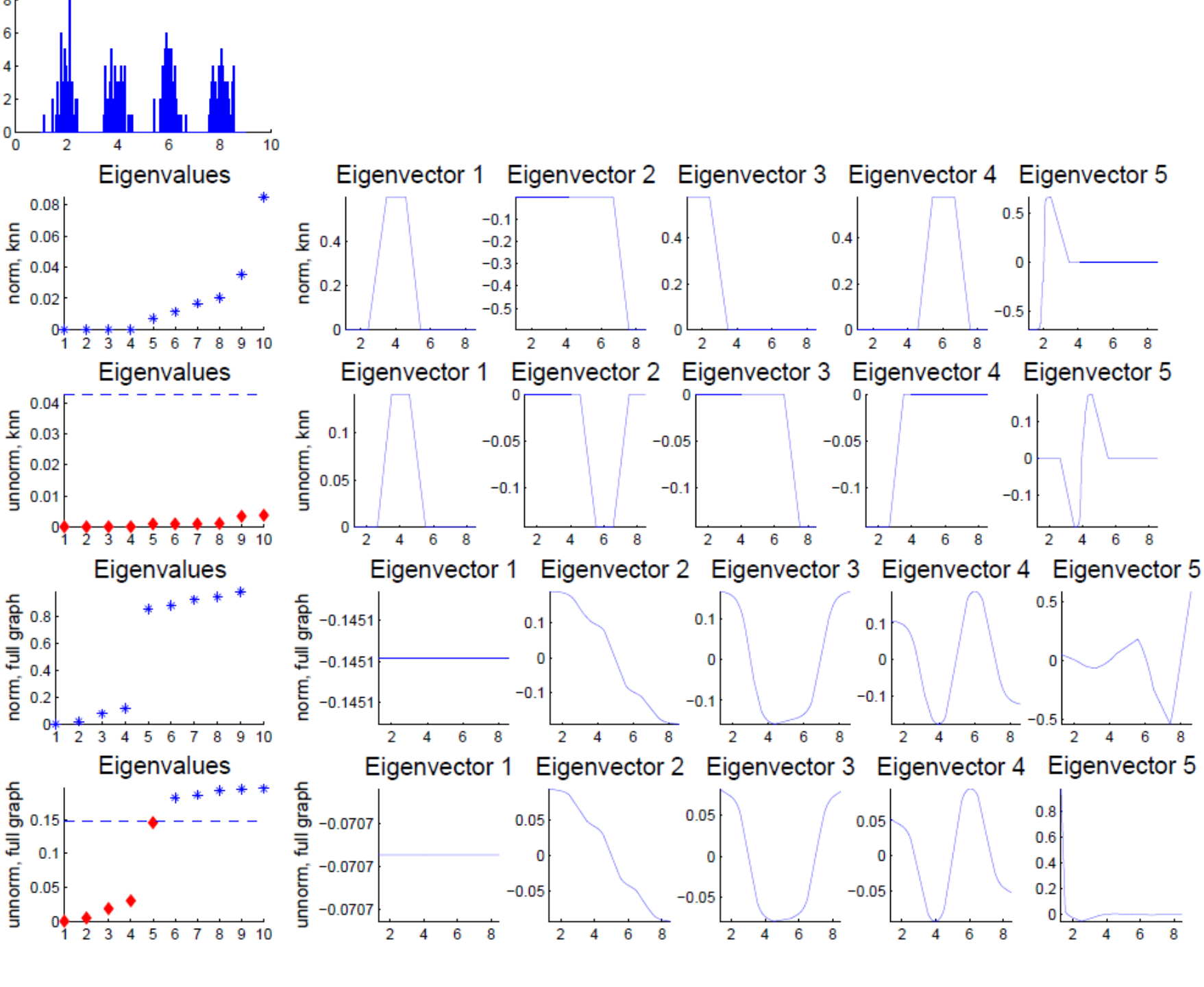
Normalized Spectral Clustering

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

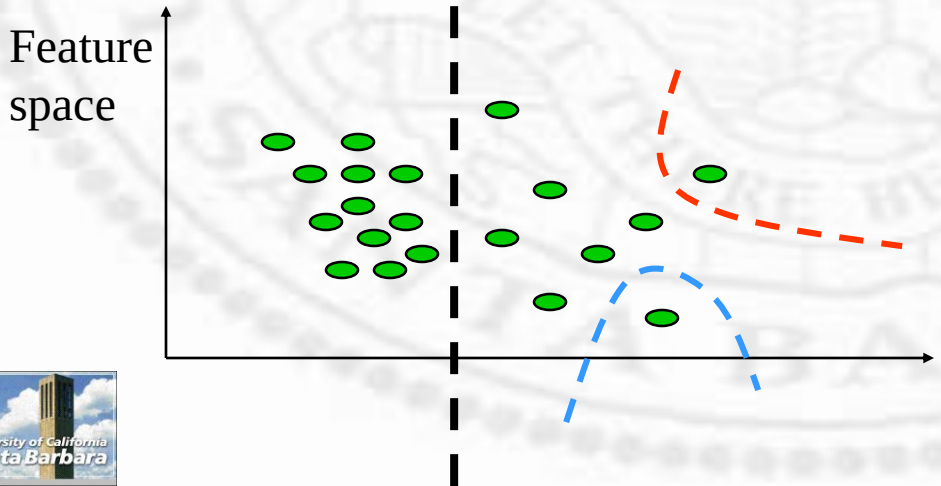

Eigenvalues
Eigenvector 1
Eigenvector 2
Eigenvector 3
Eigenvector 4
Eigenvector 5
norm, knn
unnorm, knn
norm, full graph
unnorm, full graph

Toy Example

- Random sample of 200 points drawn from 4 Gaussians
- Similarity based on $s(x_i, x_j) = \exp(-|x_i - x_j|^2/(2\sigma^2))$ with $\sigma = 1$.
- Graph
  - Fully connected or
  - 10 nearest neighbors

Min-Cut Formulation

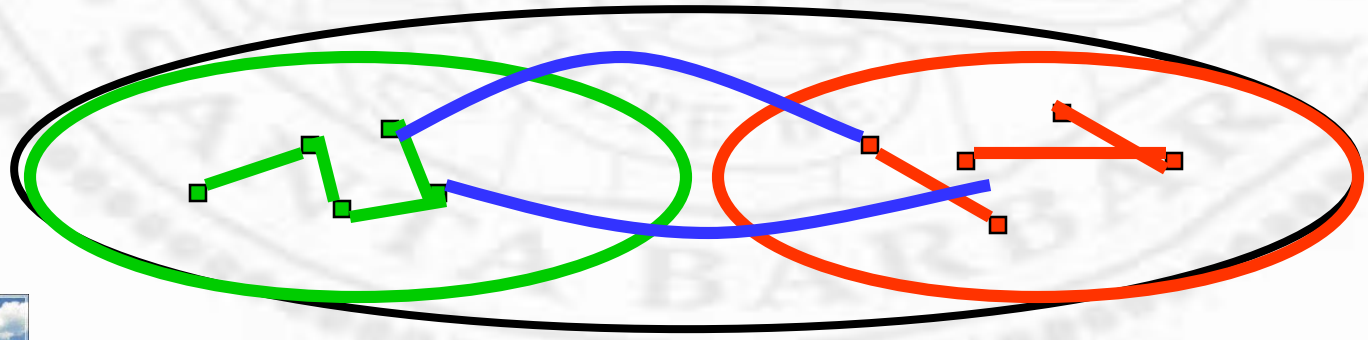
- If the edge weight represents degree of similarity, optimal bi-partitioning of a graph is to minimize the cut (so called min-cut problem)
- Intuition: Cut the connection between dis-similar samples, hence, edge weight (similarity) should be small



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

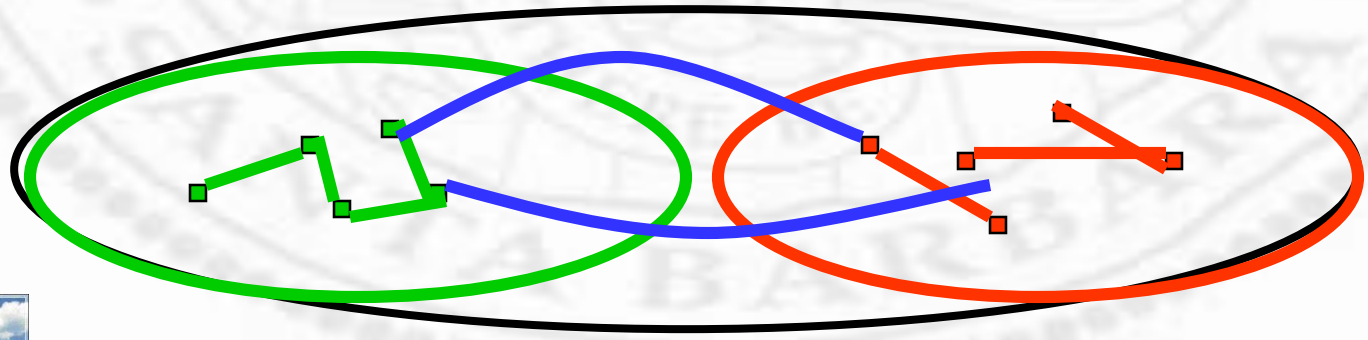
Problem with Min-cut

- Tends to cut out small regions
  - Sum weight (green + red + blue) = constant
  - Min-cut minimizes sum of weights of blue edges, with no regard to green and red (half of the picture)



Remedy

- Distribute the total weight such that
  - Sum of weights of the blue edges are minimized
  - Max *between* group variance
  - Sum of weights of the red (green) edges are maximized
  - Min *within* group variance



Normalized Cut

- Penalize cutting out small, isolated clusters

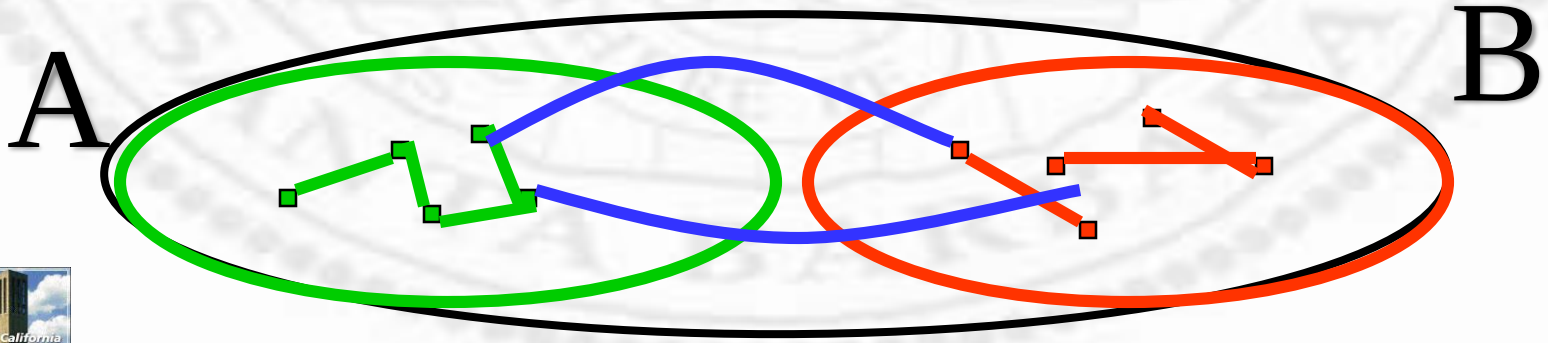
$$Ncut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)} = \frac{blue}{total - red} + \frac{blue}{total - green}$$

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

$$asso(A,V) = \sum_{u \in A, v \in V} w(u,v) \ (\text{green} + \text{blue} = \text{total} - \text{red})$$

$$asso(B,V) = \sum_{u \in A, v \in V} w(u,v) \ (\text{red} + \text{blue} = \text{total} - \text{green})$$

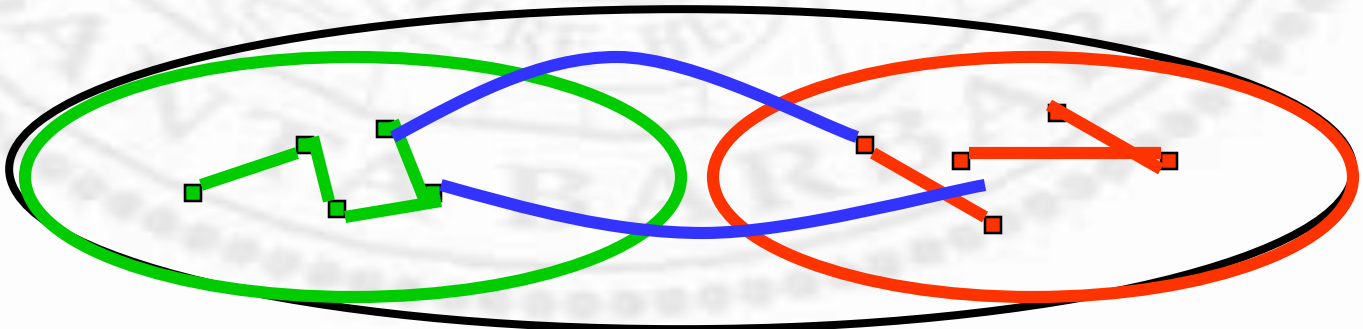
V : full vertex set



Normalized Cut (cont.)

- Penalize cutting out small, isolated clusters
  - Small blue
  - Small red (or green)

$$Ncut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)} = \frac{blue}{total - red} + \frac{blue}{total - green}$$



Intuition

- Assoc reflects *intra*-class connection which should be maximized
- Ncut represents *inter*-class connection which should be minimized

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

$$\frac{\sum_{u\in A, v\in A} w(u,v)}{asso(A,V)} + \frac{cut(A,B)}{asso(A,V)} = 1$$

$$\frac{assoc(A,A)}{asso(A,V)} + \frac{cut(A,B)}{asso(A,V)} = 1 \qquad \frac{assoc(B,B)}{asso(B,V)} + \frac{cut(A,B)}{asso(B,V)} = 1$$

$$\frac{assoc(A,A)}{asso(A,V)} + \frac{assoc(B,B)}{asso(B,V)} + \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)} = 2$$

$$Nassoc(A,B) + Ncut(A,B) = 2$$

$$\frac{green}{green+blue} + \frac{red}{red+blue} + \frac{blue}{green+blue} + \frac{blue}{red+blue} = 2$$

Solution

- How do you define similarity?

$$d(i, j) = \sum_k w_k \mid v_k^i - v_k^j \mid$$

  - Multiple measurements (i.e., a feature vector) can be used
- How do you find the minimal normalized cut?
  - Solution turns out to be a generalized eigen value problem!

$$\mathbf{x} = \begin{bmatrix} 1 \\ -1 \\ \vdots \\ \\ \vdots \\ 1 \end{bmatrix}_{N\times 1} \qquad x_i = \begin{cases} 1 & i \in A \\ -1 & i \in B \end{cases}$$

$$\mathbf{D} = \begin{bmatrix} d_1 & & & \\ & d_1 & 0 & \\ & & \ddots & \\ & 0 & & d_N \end{bmatrix}_{N\times N} \qquad d_i = \sum_j w(i,j)$$

$$\mathbf{W} = \begin{bmatrix} w(1,1) & w(1,2) & \cdots & w(1,N) \\ w(2,1) & w(2,2) & \cdots & w(2,N) \\ & & \ddots & \\ w(N,1) & w(N,2) & & w(N,N) \end{bmatrix}_{N\times N}$$

$$k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$$

$$N = |V|$$

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

$$= \frac{\sum_{x_i>0, x_j<0} -w_{ij}x_i x_j}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0, x_j>0} -w_{ij}x_i x_j}{\sum_{x_i<0} d_i}$$

$$= \frac{(\mathbf{1}+\mathbf{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}+\mathbf{x})}{k\mathbf{1}^T\mathbf{D}\mathbf{1}} + \frac{(\mathbf{1}-\mathbf{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}-\mathbf{x})}{(1-k)\mathbf{1}^T\mathbf{D}\mathbf{1}}$$

$$\frac{\mathbf{1}+\mathbf{x}}{2} = \begin{cases} 1 & x_i \in A \\ 0 & x_i \in B \end{cases} \quad \frac{\mathbf{1}-\mathbf{x}}{2} = \begin{cases} 0 & x_i \in A \\ 1 & x_i \in B \end{cases}$$

$$d_i = \sum_{x_j<0} w_{ij}x_j + \sum_{x_k>0} w_{ik}x_k \Rightarrow \sum_{x_j<0} w_{ij}x_j = d_i - \sum_{x_k>0} w_{ik}x_k$$

$$\min_x Ncut(\mathbf{x}) = \min_y \frac{\mathbf{y}^T(\mathbf{D}-\mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

$$\mathbf{y} = (\mathbf{1}+\mathbf{x}) - \mathbf{b}(\mathbf{1}-\mathbf{x})$$

$$b = \frac{k}{1-k}$$

$$(\mathbf{D}-\mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$$

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\mathbf{z} = \lambda\mathbf{z} \qquad \mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y}$$

- A symmetric semi-positive-definite matrix
  - Real, >=0 eigen values
  - Orthogonal eigen vectors

$$\text{eigenvector}: \mathbf{z}_o = \mathbf{D}^{\frac{1}{2}}\mathbf{1}$$
$$\text{eigenvalue}: 0$$

- Hence, the second smallest eigen vector contains the minimal cut solution (in floating point format)
- Even though computing all eigen vectors/values are expensive O(n^3), computing a small number of those are not that expensive (Lanczos method)
- Recursive applications of the procedure

Results