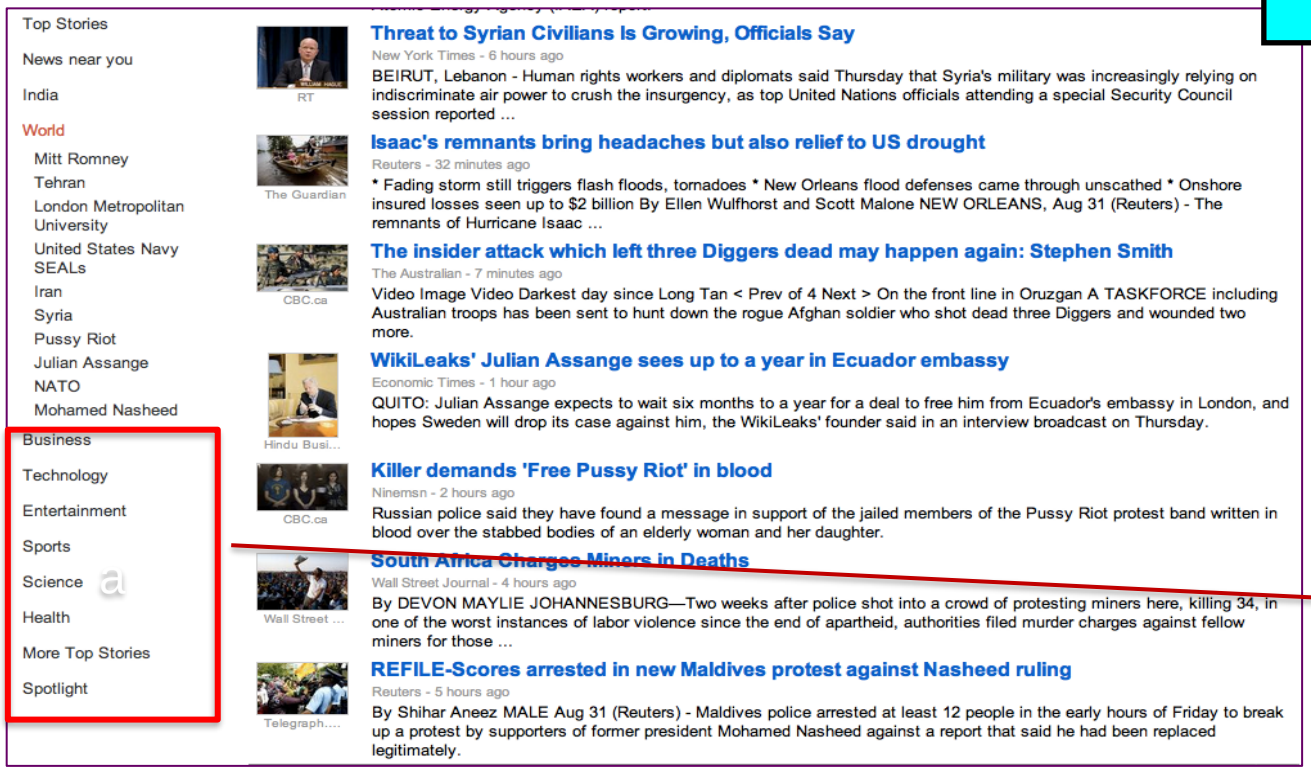
What is Unsupervised Learning?

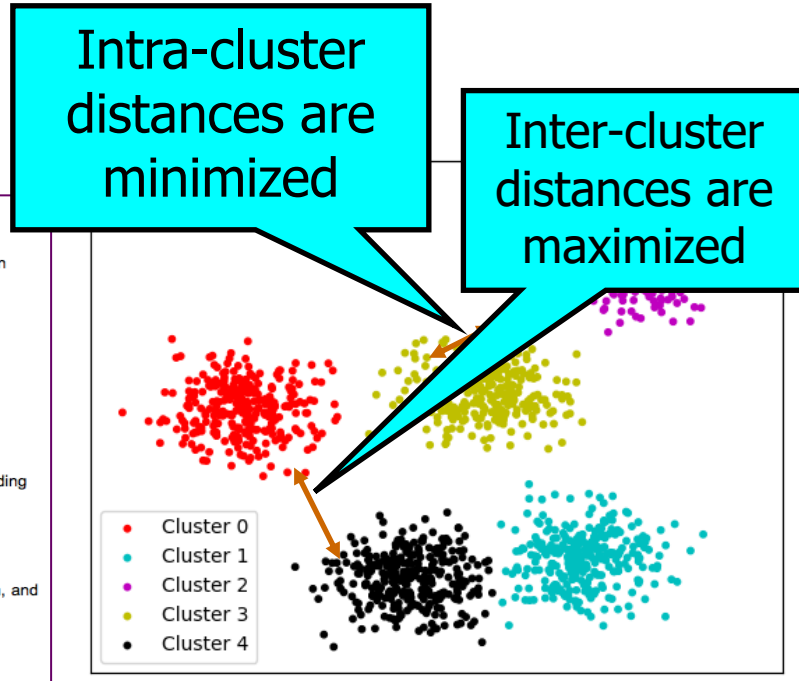
Jie Tang
Department of Computer Science & Technology
Tsinghua University

Clustering

Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (or related) to each other than to those in other clusters.

1. News Services like Google News(below figure)





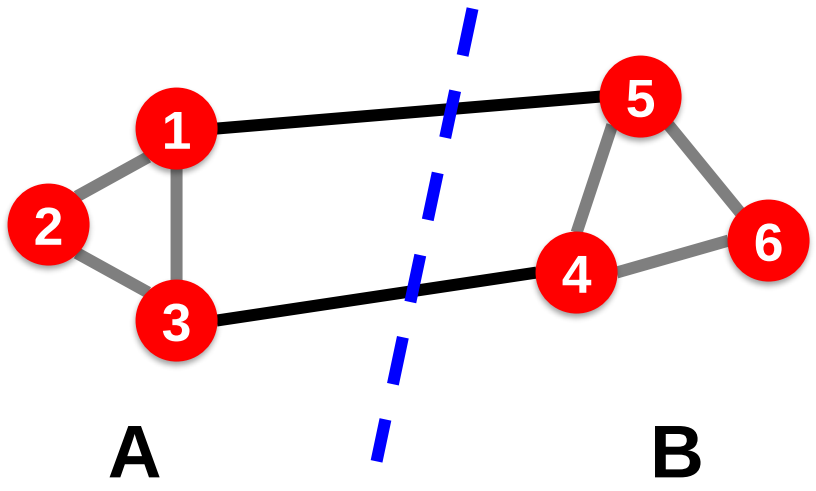
Similar news are grouped together.

Graph Clustering

2. Social network analysis

In the study of social networks, clustering can be used to recognize communities within large groups of people.

- Maximize the number of within-group connections
- Minimize the number of between-group connections



What Is Good Clustering?

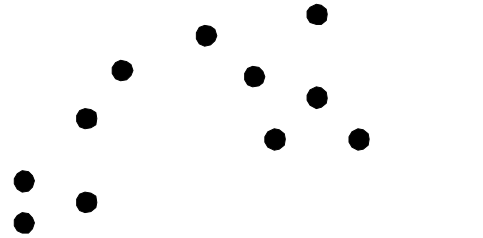
- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.

Some of the following slides are from Chris Clifton's course slides

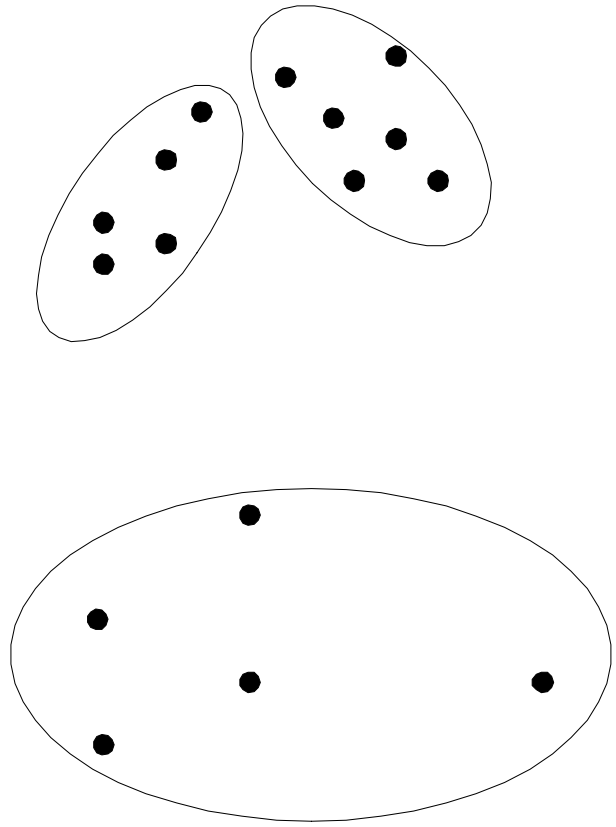
Types of Clusterings

- A clustering is a set of clusters
- Important distinction between hierarchical and partitional sets of clusters
- 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

Partitional Clustering

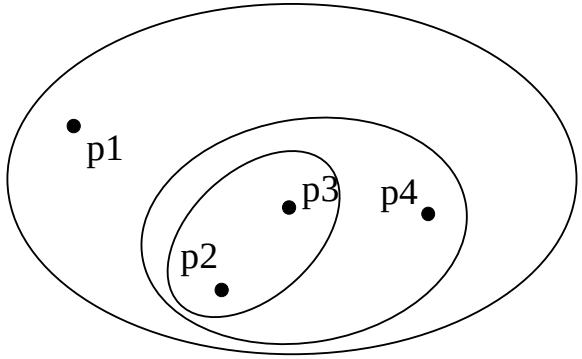


Original Points

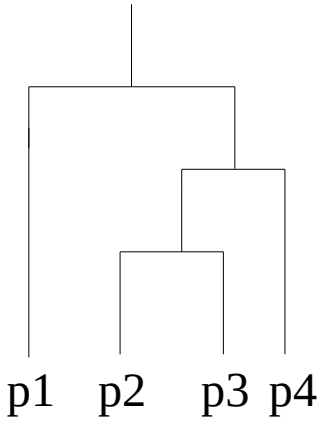


A Partitional Clustering

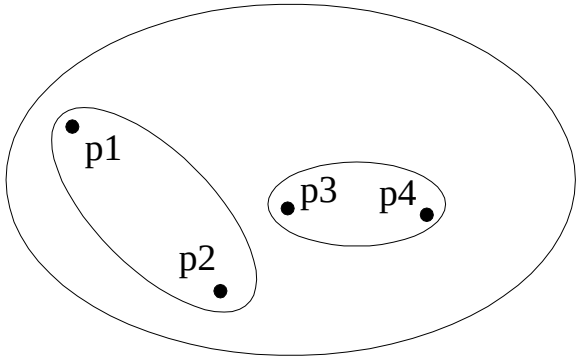
Hierarchical Clustering



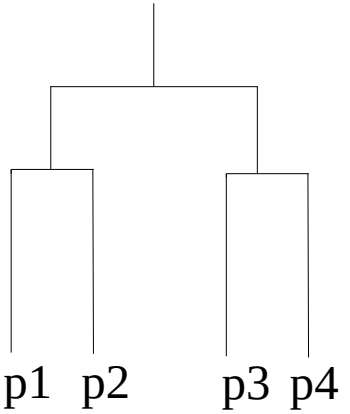
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Data Structures

- An example with *m* features

$$\boldsymbol{x}_i=[x_{i1}, x_{i2},\ldots, x_{im}]$$

- Data (set) matrix

$$\begin{bmatrix} x_{11} & \ldots & x_{1k} & \ldots & x_{1m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{i1} & \ldots & x_{ik} & \ldots & x_{im} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{N1} & \ldots & x_{Nk} & \ldots & x_{Nm} \end{bmatrix}$$

Partitioning Algorithms: Basic Concept

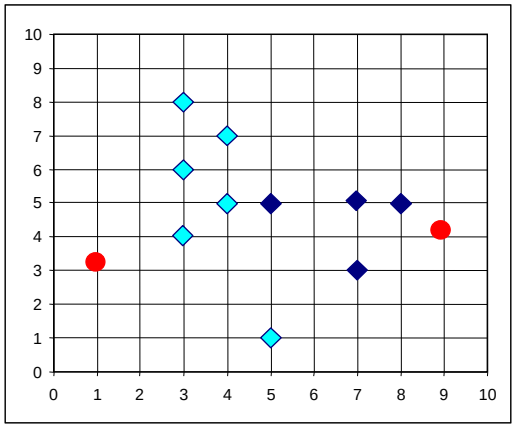
- Partitioning method: Construct a partition of a data ***D*** of ***N*** objects into a set of ***k*** clusters
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion
  - Global optimal: exhaustively enumerate all partitions
  - Heuristic methods: *k-means* and *k-medoids* algorithms
  - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
  - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

The *K-Means* Clustering Method

- The *k-means* algorithm is implemented in four steps:
  - Compute seed k
  - Partition objects points as the centroids of the clusters of the current partition (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when no more new assignment

The *K-Means* Clustering Method

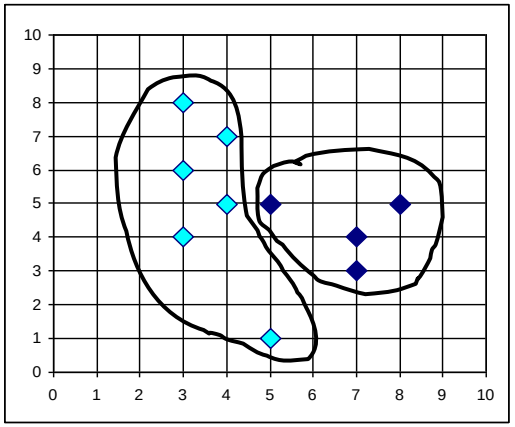
- Example



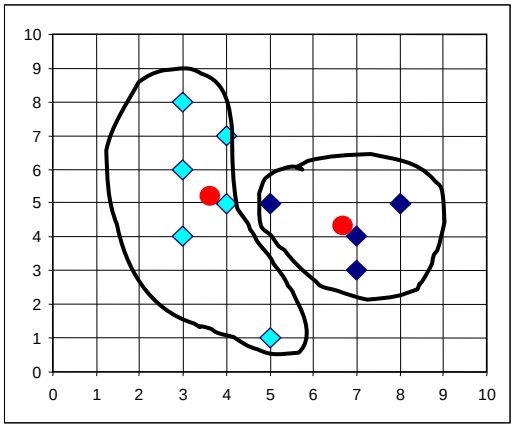
K=2

Arbitrarily choose K object as initial cluster center

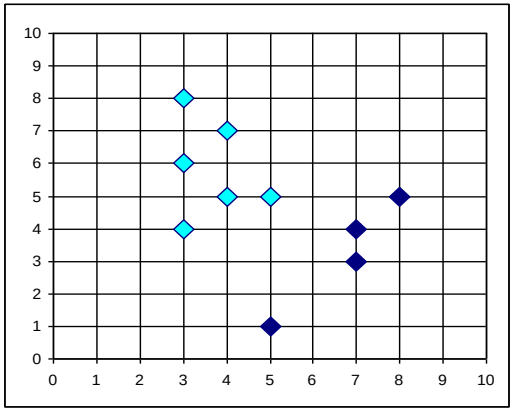
Assign each objects to most similar center



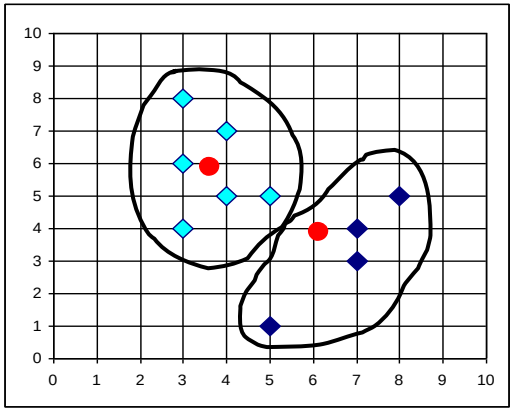
Update the cluster means



reassign



Update the cluster means



reassign

Clustering

- Similarity Metric
  - distance function: $d(i,j)$
  - Minkowski distance

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

$q=1$ for Manhattan distance, $q=2$ for Euclidean

  - Asymmetric binary distance

$$d(i,j)=\frac{b+c}{a+b+c}$$

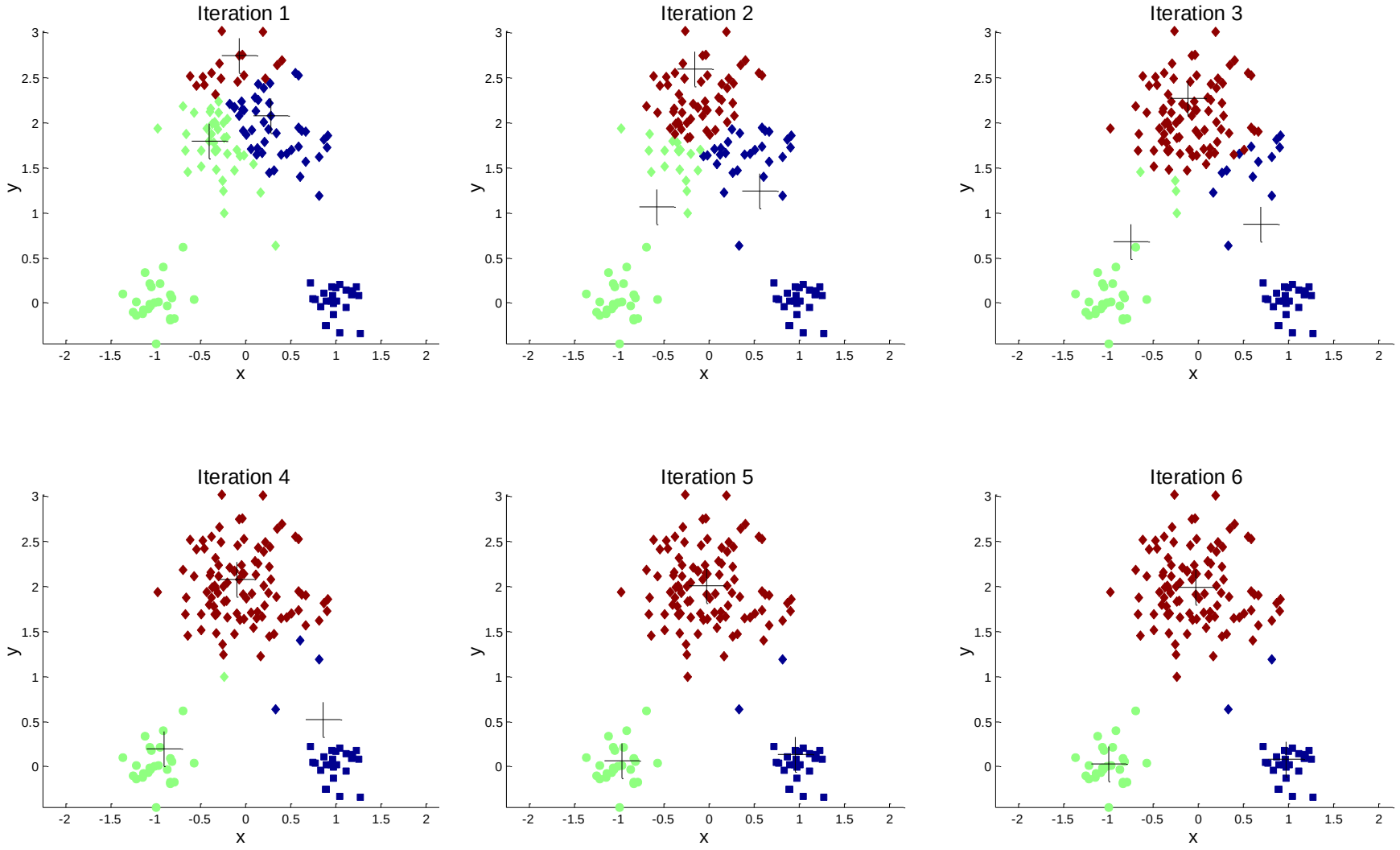
		Object j 1	Object j 0
Object i	1	a	b
	0	c	d

  - Cosine distance $\frac{\vec{X}^t \cdot \vec{Y}}{|\vec{X}||\vec{Y}|}$

Comments on the *K-Means* Method

- Strength: *Relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - Comparing: PAM: $O(k(n-k)^2)$, CLARA: $O(ks^2 + k(n-k))$
- Comment: Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

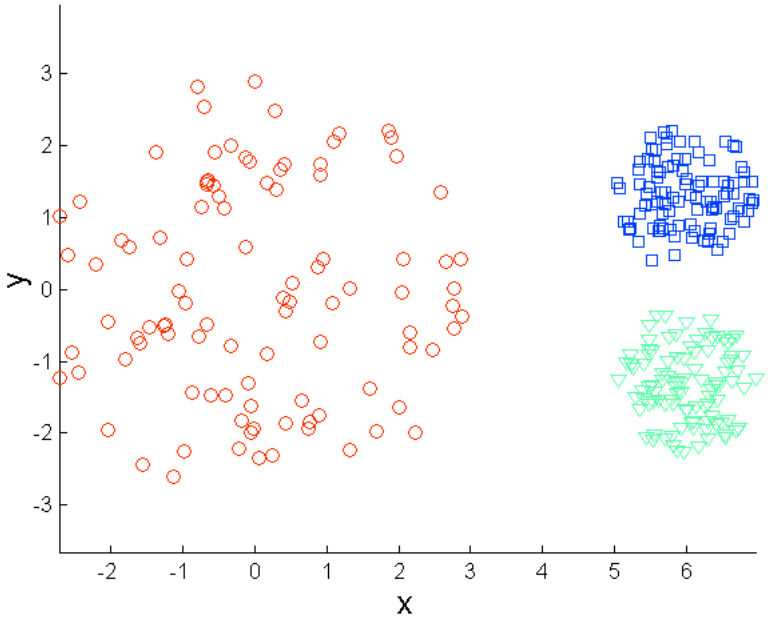
Initial Centroids Selection



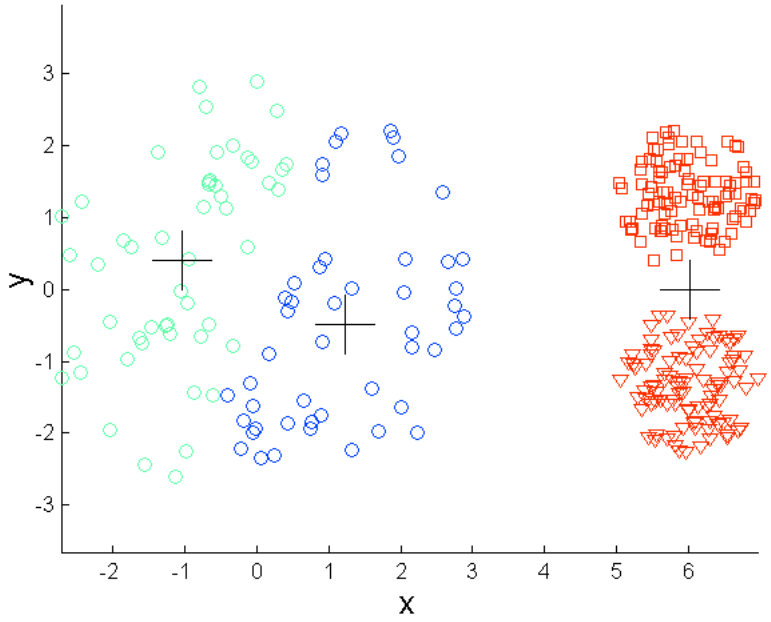
Initial Centroids Selection

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Postprocessing

Limitations of *K*-means: Density

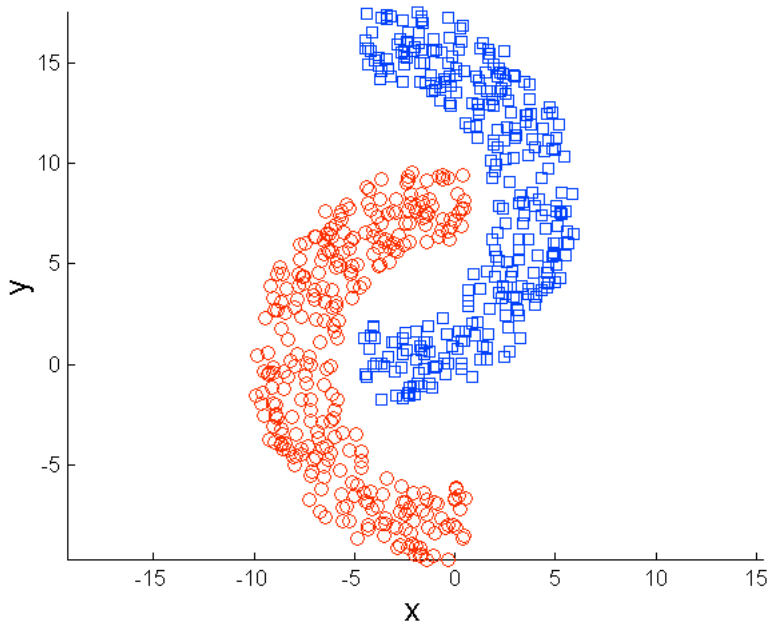


Original Points

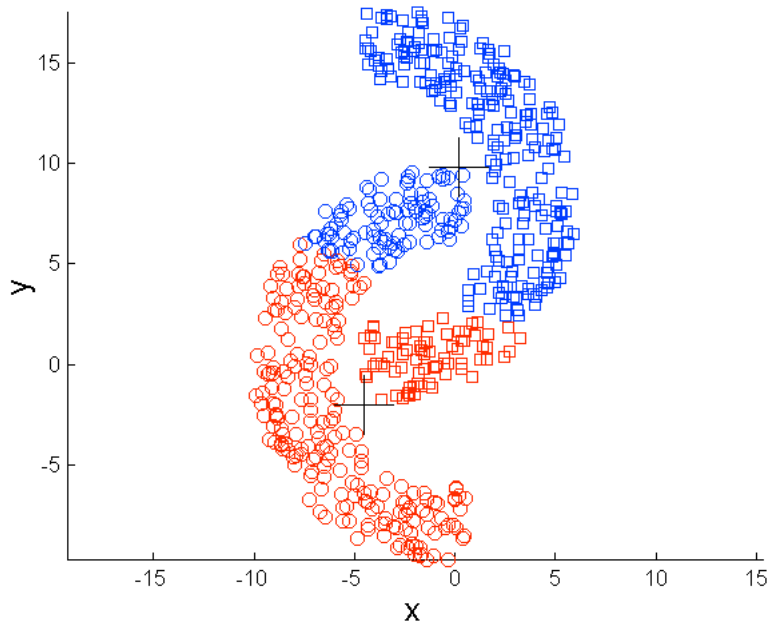


K-means (3 Clusters)

Limitations of *K*-means: Non-global Shape



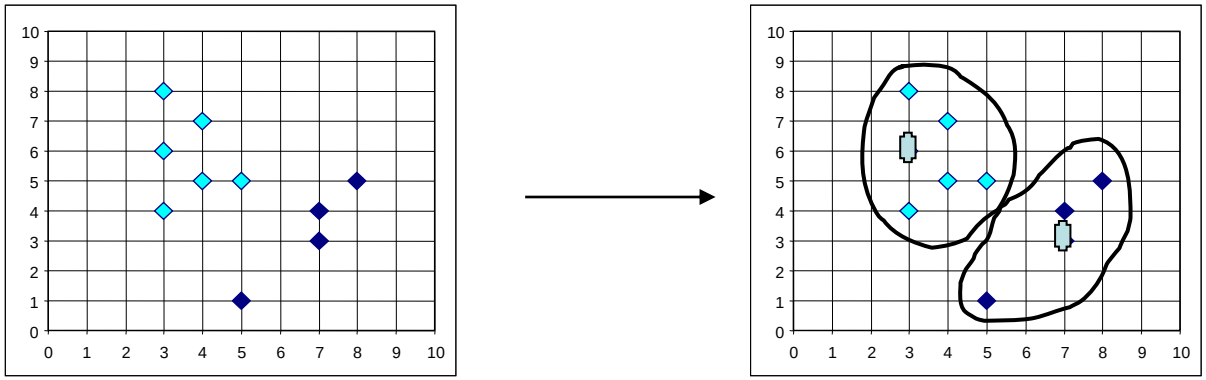
Original Points



K-means (2 Clusters)

The *K-Medoids* Clustering Method

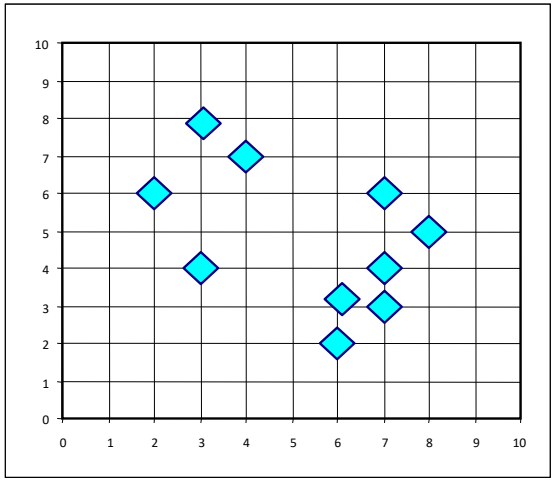
- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.



The *K-Medoids* Clustering Method

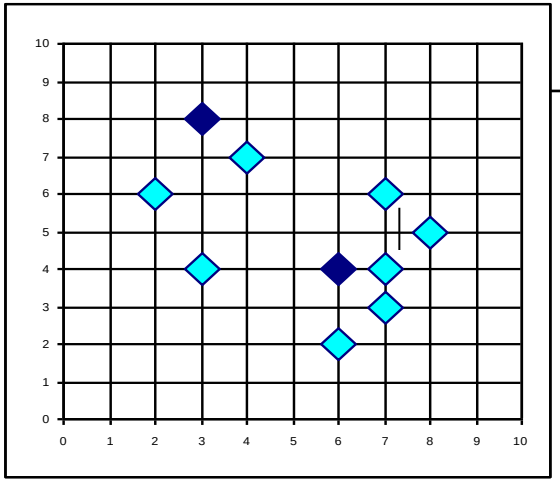
- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling
- Focusing + spatial data structure (Ester et al., 1995)

Typical *K*-medoids algorithm



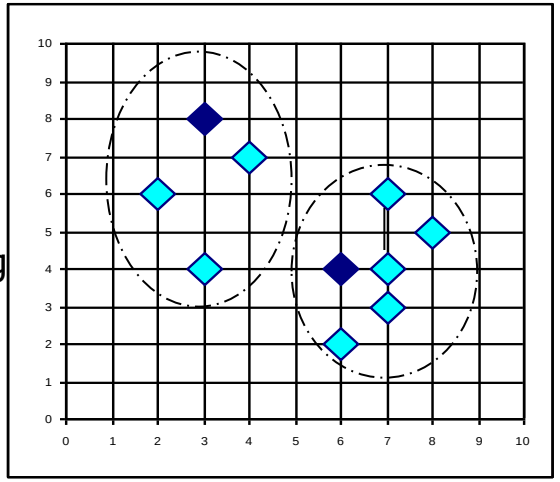
K=2

Arbitrary choose k object as initial medoids

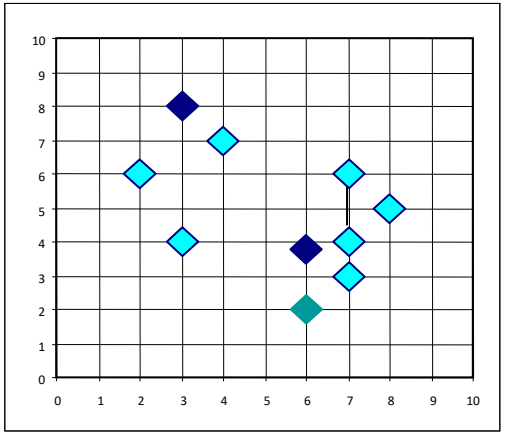


Assign each remaining object to nearest medoids

Total Cost = 20

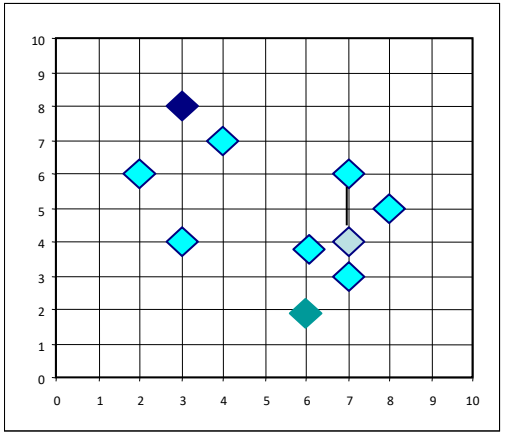


Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping

Total Cost = 26



Swapping O and O_{ramdom}

If quality is improved.

Do loop

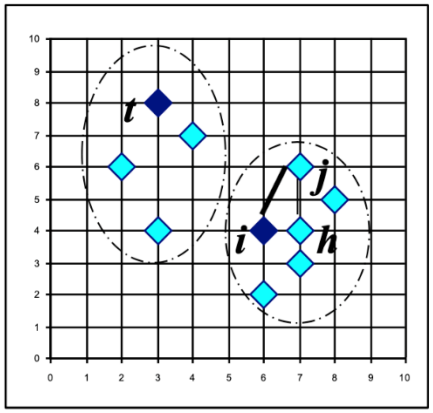
Until no change

PAM (Partitioning Around Medoids) (1987)

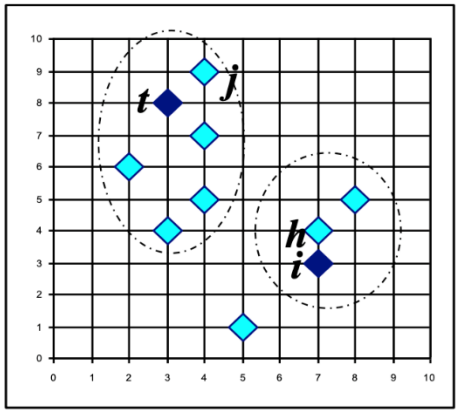
- PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search.
- Initialize: greedily select k of the n data points as the medoids to minimize the cost.
- Associate each data point to the closest medoid.
- Repeat
  - For each medoid i, for each non-medoid data point h, compute the total swapping cost TC_{ih};
  - Select a pair (i, h) that minimizes TC_{ih};
  - If min $TC_{ih} < 0$ then swap i with h to form the new set of k representative objects;
- Until min $TC_{ih} > 0$

PAM(Partitioning Around Medoids)(1987)

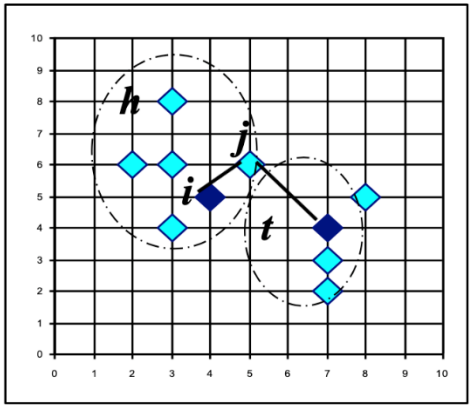
- Runtime complexity:
  - $O(k(n-k)^2)$ per iteration, by only computing the change in cost.
  - Cost of swapping i and h: $TC_{ih} = \sum_j C_{jih}$



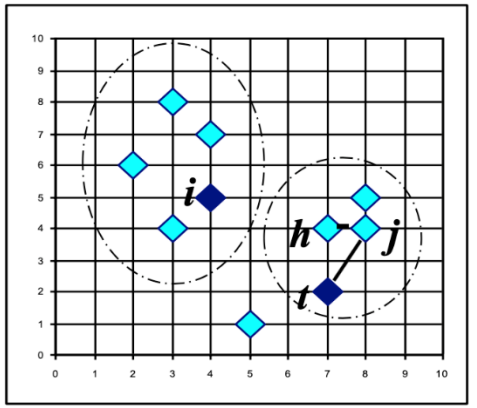
$C_{jih} = d(j, h) - d(j, i)$



$C_{jih} = 0$



$C_{jih} = d(j, t) - d(j, i)$



$C_{jih} = d(j, h) - d(j, t)$

CLARA: Clustering LARge Applications(1990)

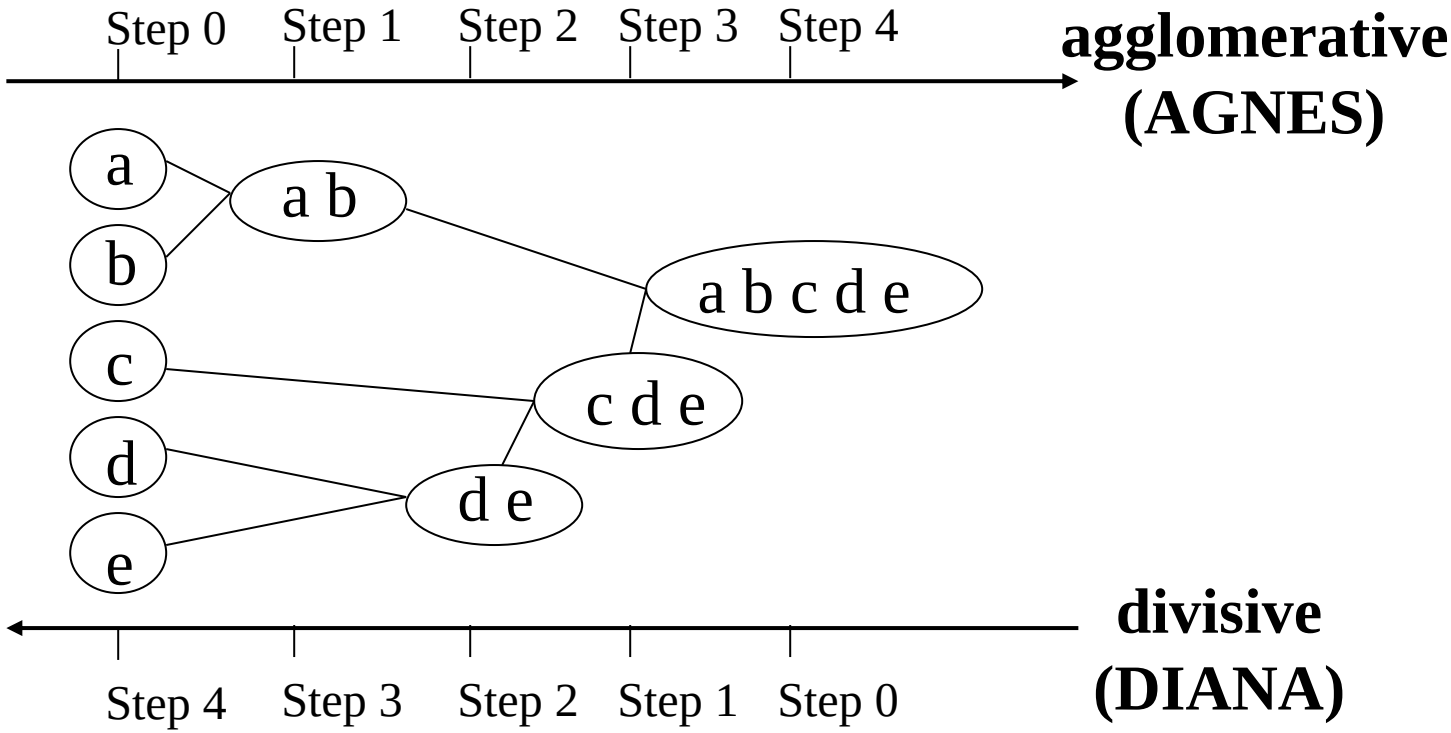
- Sampling based method:
  - It draws multiple samples of the data set, applies PAM on each sample, and gives the best clustering as the output.
- Strength: deals with larger data sets than PAM.
- Weakness:
  - Efficiency depends on the sample size. ($O(ks^2+k(n-k))$, where s is the size of the sample)
  - A good clustering based on samples will not necessarily represent a good clustering of the whole data set if the sample is biased.

CLARANS: Clustering LARge Applications Based Upon Randomized Search (1994)

- Works on the entire data set
- Only explores a subset of the possible swaps of medoids and non-medoids using sampling
- Time complexity: $O(n^2)$

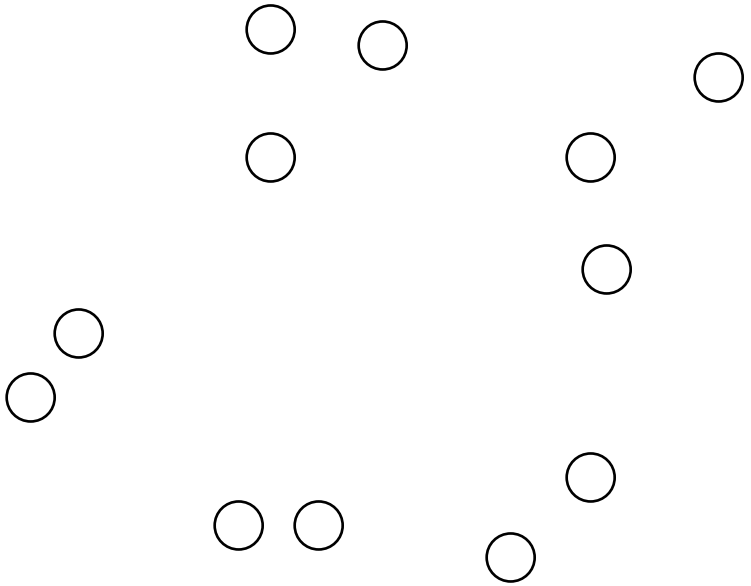
Hierarchical Clustering

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



Hierarchical Clustering

- Start with clusters of individual points and a proximity matrix



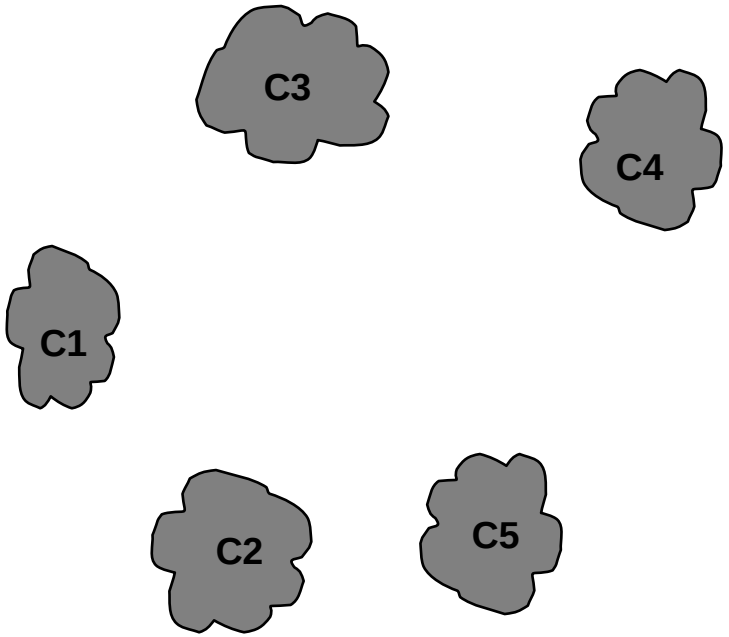
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



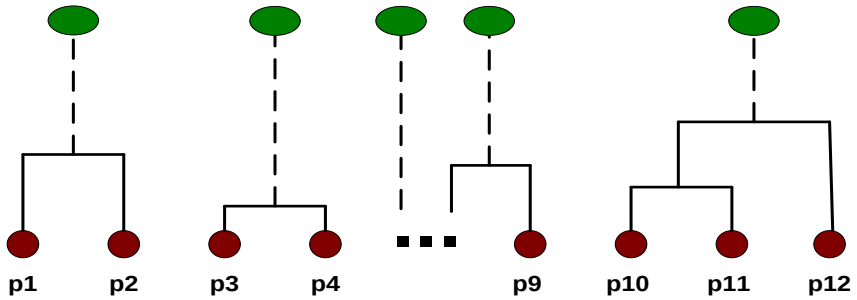
Hierarchical Clustering

- After some merging steps, we have some clusters



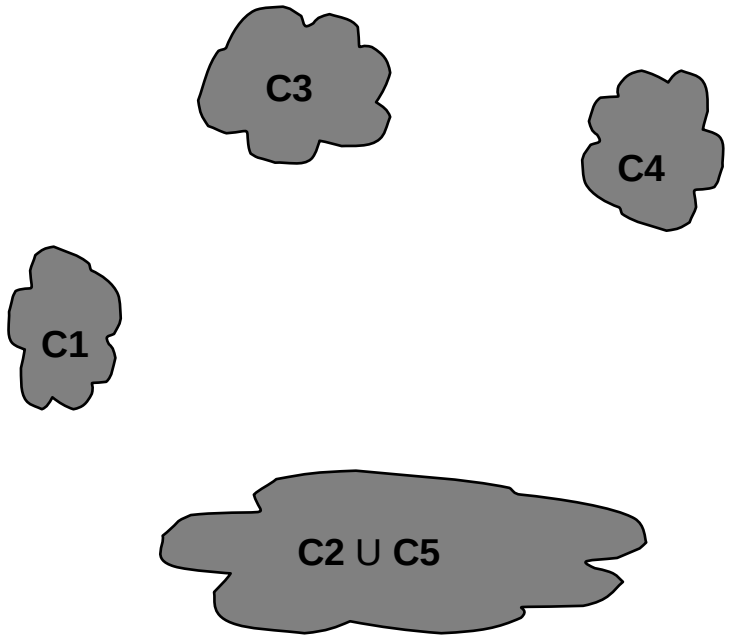
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



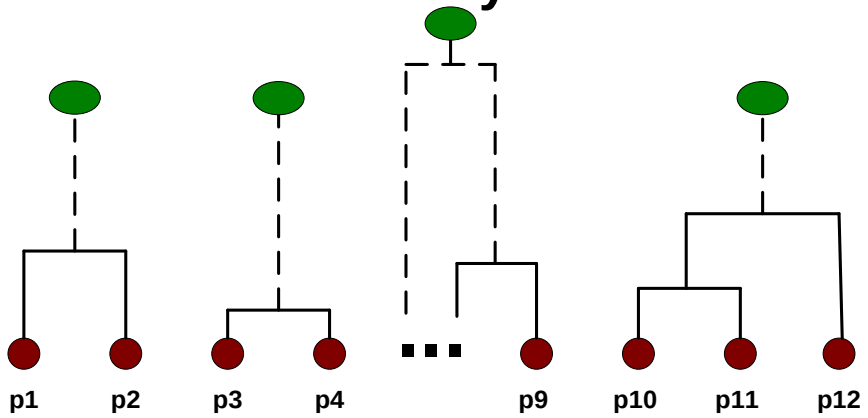
Hierarchical Clustering

- The question is “How do we update the proximity matrix?”

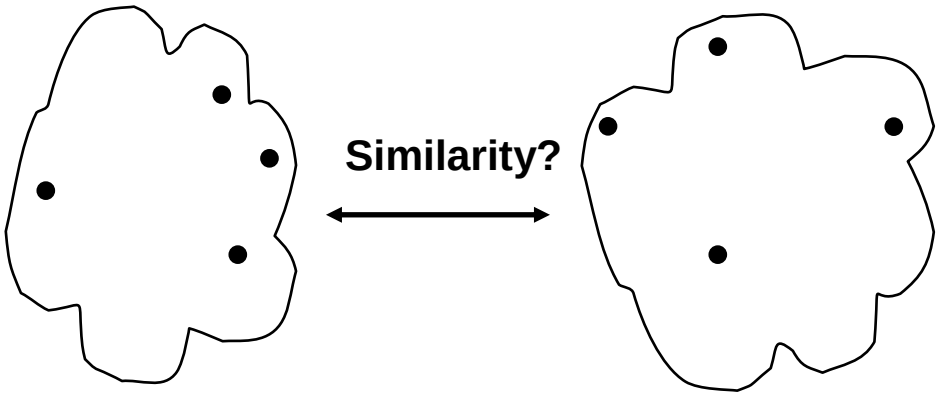


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

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

Proximity Matrix

Hierarchical Clustering: Time and Space

- 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

Hierarchical Clustering: Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

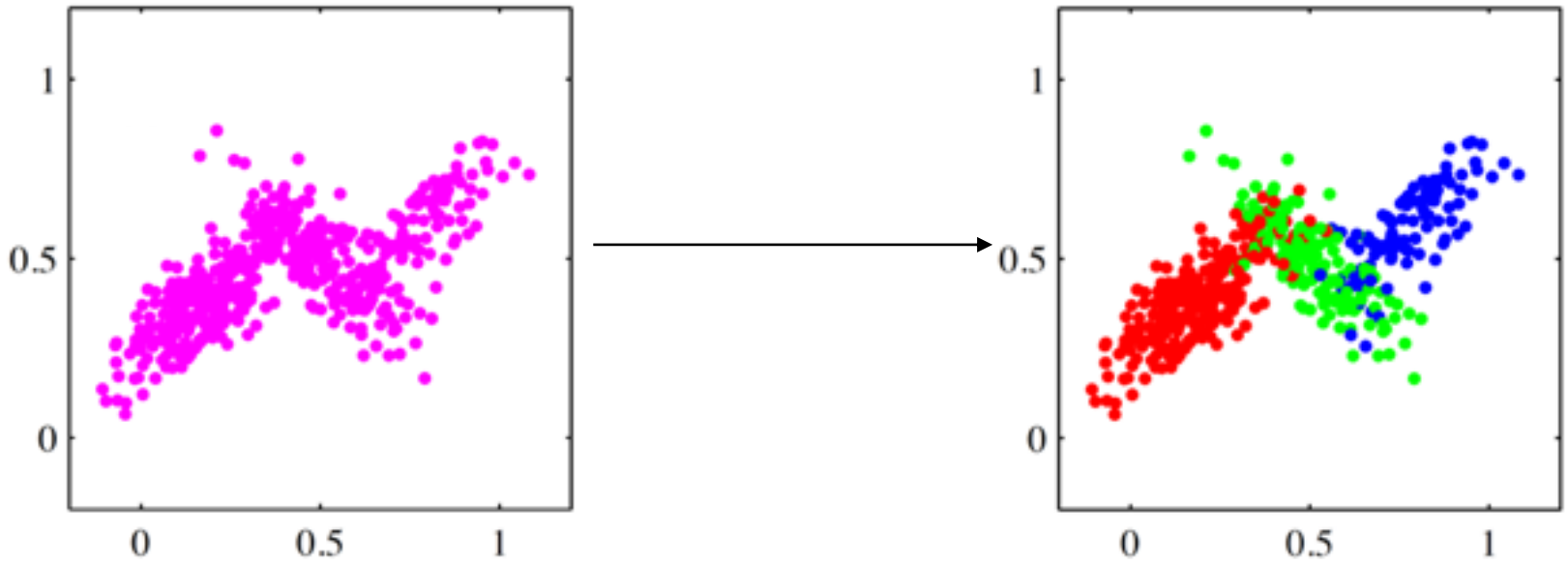
Unsupervised Learning

- Clustering
  - K-means, K-medoids, Hierarchical clustering
  - Mixture Models
- Dimension Reduction
  - Principal Component Analysis

Mixture Models

Mixture Models

- Problems with K-means
  - Hard assignment: Data point is deterministically assigned to one and only one cluster.
  - In reality, clusters may overlap.



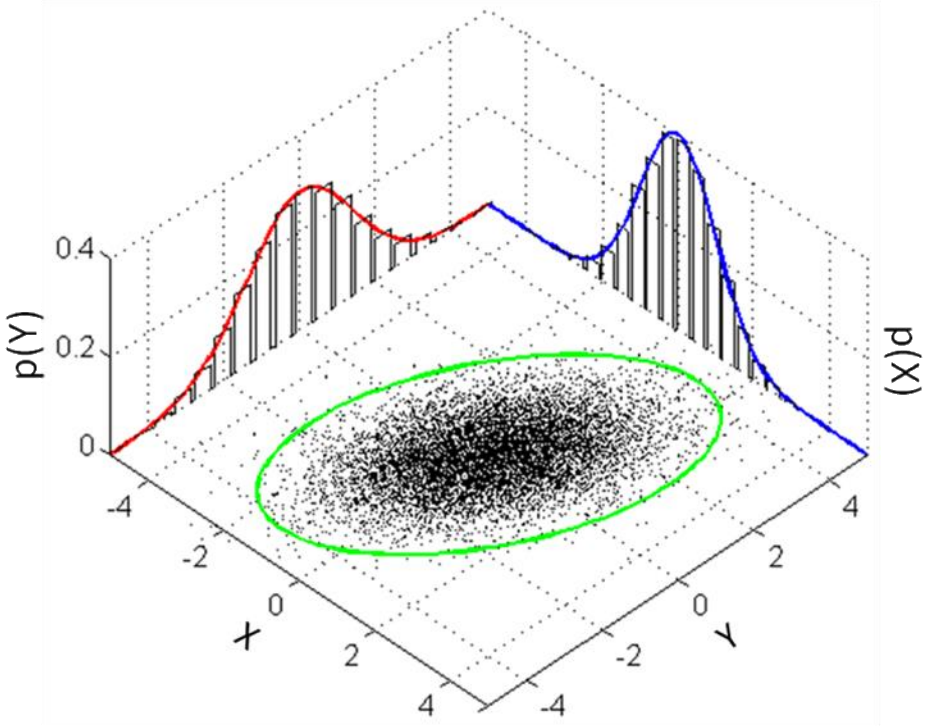
Mixture Models

- *N* random variables distributed according to a mixture of *K* components, with the components belonging to the same parametric family of distributions (e.g. all normal, etc.) but with different parameters.
- Latent variables:
  - specifying the identity of the mixture component of each observation. (Each follows a *K*-dimensional distribution)
  - z_i: component of observation i.
- Mixture weights:
  - probabilities that sum to 1.
  - ϕ_i: prior probability of a particular component i.
  - $z_i \sim \text{Multinomial}(\phi)$

Mixtures of Gaussians

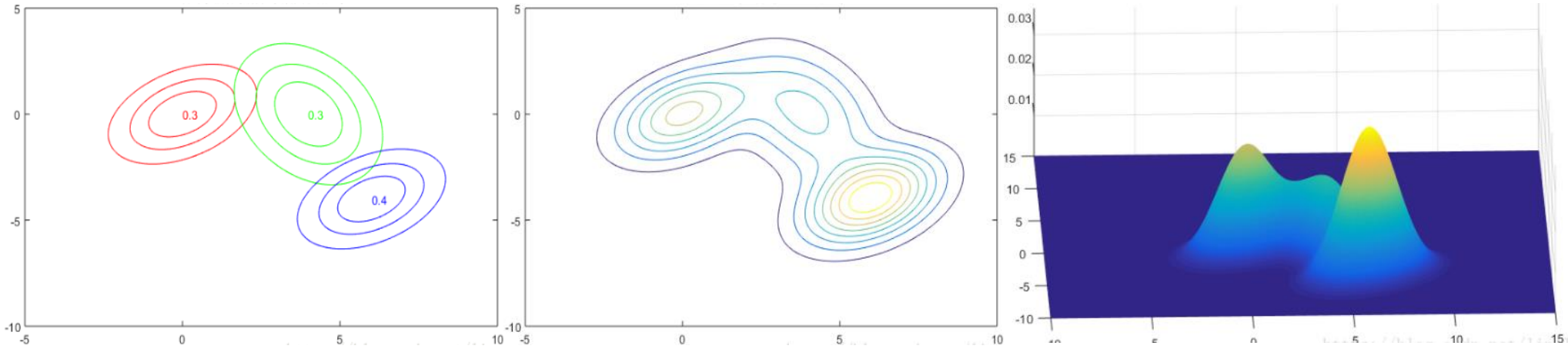
- Recall the Gaussian distribution

$$P(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)$$



Mixtures of Gaussians

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} p(k)p(\boldsymbol{x}|k) = \sum_{k=1}^{K} \phi_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$



Mixtures of Gaussians

- Given a training set $\{x_1, x_2, \ldots, x_N\}$, we wish to model the data by specifying a joint probability $p(x_i, z_i) = p(x_i|z_i)p(z_i)$.
  - Here, $z_i \sim \text{Multinomial}(\phi)$, ($\sum_{j=1}^{k} \phi_j = 1$, and the parameter ϕ_j gives $p(z_i = j)$), and $x_i|z_i = j \sim N(\mu_j, \Sigma_j)$.
  - Our model posits that each x_i was generated by randomly choosing z_i from $\{1, \ldots, k\}$, and then x_i was draw from one of k Gaussian depending on z_i.

Mixtures of Gaussians

- Parameters of our model: ϕ, μ and Σ.
- How to estimator?
  - Likelihood of our data:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{N} \log p(x_i; \phi, \mu, \Sigma)$$

$$= \sum_{i=1}^{N} \log \sum_{j=1}^{k} p(x_i; \mu, \Sigma) p(z_i; \phi)$$

  - Maximum Likelihood Estimation

However, if we calculate derivatives w.r.t. the parameters and try to solve, we will find that it is not possible to find the maximum likelihood estimates of the parameters in closed form.

Mixtures of Gaussians

- Note that if we knew the z_i's, the maximum likelihood problem would be much easier. Specifically, we could then write down the likelihood as:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{N} \log p(x_i \mid z_i; \mu, \Sigma) + \log(z_i; \phi)$$

- Maximizing this with respect to ϕ, μ and Σ:

$$\phi_j = \frac{1}{N} \sum_{i=1}^{N} 1\{z_i = j\},\ \mu_j = \frac{\sum_{i=1}^{N} 1\{z_i = j\} x_i}{\sum_{i=1}^{N} 1\{z_i = j\}},\ \Sigma_j = \frac{\sum_{i=1}^{N} 1\{z_i = j\}(x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{N} 1\{z_i = j\}}$$

However, the z_i's are unknown, what can we do?
Solution: the EM algorithm

EM Algorithm

- The EM (Expectation Maximalization) algorithm is an iterative algorithm solving MLE for models depending on unobserved latent variables. It has two steps
- E-Step:
  - Given current estimate of parameters
  - Output “membership probabilities” - z_{ij}
  - Set up $Q(\theta|\theta^{(t)})$: the expectation of log-likelihood given current estimates.
- M-Step:
  - Maximize the *Q* function

EM Algorithm

$$
\begin{aligned}
Q(\theta \mid \theta^{(t)}) &= \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta^{(t)}}\left[\log L(\theta; \mathbf{x}, \mathbf{Z})\right] \\
&= \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta^{(t)}}\left[\log \prod_{i=1}^{n} L(\theta; \mathbf{x}_i, Z_i)\right] \\
&= \mathrm{E}_{\mathbf{Z}|\mathbf{X},\theta^{(t)}}\left[\sum_{i=1}^{n} \log L(\theta; \mathbf{x}_i, Z_i)\right] \\
&= \sum_{i=1}^{n} \mathrm{E}_{Z_i|\mathbf{X};\theta^{(t)}}\left[\log L(\theta; \mathbf{x}_i, Z_i)\right] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} P(Z_i = j \mid X_i = \mathbf{x}_i; \theta^{(t)}) \log L(\theta_j; \mathbf{x}_i, j)
\end{aligned}
$$

EM Algorithm

EM Algorithm description

Repeat until convergence: {

(E-Step) For each i, j, set:

$$Z_{ij} = P(z_i = j | x_i; \phi, \mu, \Sigma) = \frac{P(x_i | z_i = j; \mu, \Sigma) P(zi=j|\phi)}{\sum_{l=1}^{k} P(x_i | z_i = l; \mu, \Sigma) P(zi=l|\phi)}$$

soft cluster assignment

(M-Step) Update the parameters:

$$\phi_j = \frac{1}{N} \sum_{i=1}^{N} z_{ij}, \mu_j = \frac{\sum_{i=1}^{N} x_i z_{ij}}{\sum_{i=1}^{N} z_{ij}}, \Sigma_j = \frac{\sum_{i=1}^{N} z_{ij} (x_i - \mu_i)(x_i - \mu_i)}{\sum_{i=1}^{N} z_{ij}}$$

}

- Q: What's the relationship between *EM* and *K-means*?

E-Step = Cluster
M-Step = Update the cluster center

Unsupervised Learning

- Clustering
  - K-means, K-medoids, Hierarchical clustering
  - Mixture Models
- Dimension Reduction
  - Principal Component Analysis

Dimension Reduction

- Dimension Reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables.
  - Given some data $X \in R^{N \times m}$ (m might be huge), we would to find a new representation $\tilde{X} \in R^{N \times K}, K \ll$m.
  - Application: data visualization, data compression.



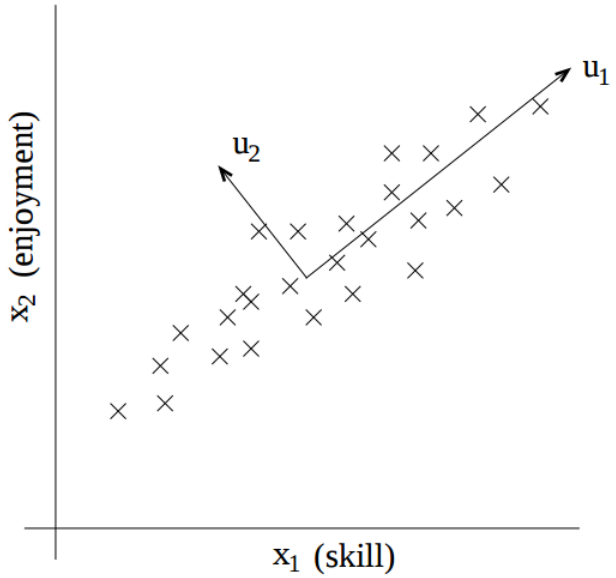


Projection of the MNIST dataset via PCA

PCA: Principal Components Analysis

- General Idea:
  - Given data points in *m*-dimensional space
  - Project into lower dimensional space while preserving as much information as possible --- minimize the square error for representing the original data

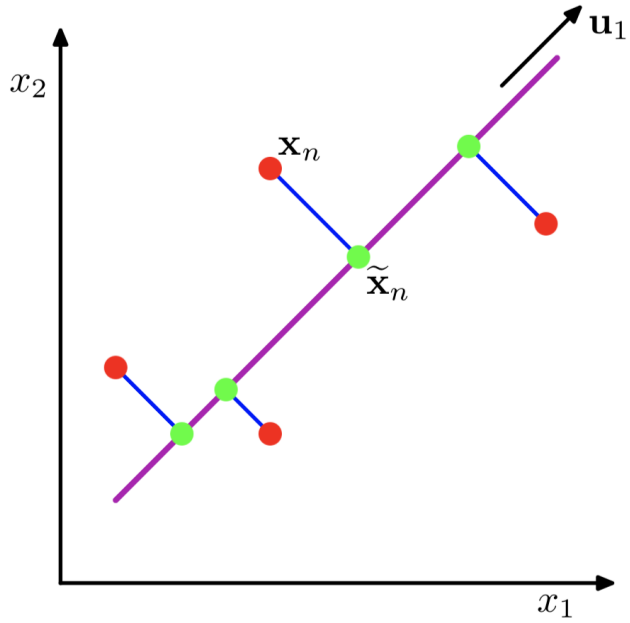
For example, given *N* data point $\{x_i; i=1,\ldots, N\}$, x_i \in $\mathbf{R}^m$, represent all data in two dimensions.



Principal Component Analysis (PCA)

- Principal Component Analysis is defined as an orthogonal linear transformation.
  - Linear transformation

$$\tilde{x}_n = x_n^T u_1, \ \left\| u_1 \right\|_2 = 1$$



- What is a good transformation?
  - Find directions of maximum variance.
    - The greatest variance of the data comes to lie on the first direction (first principal component), the second greatest variance on the second direction, and so on.

Principal Component Analysis (PCA)

- Find the first principal component:
  - Maximize the variance:

$$\max \frac{1}{2N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T \bar{x}) = \max u_1^T S u_1, \ \left\|u_1\right\|_2 = 1$$

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n, \ S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$$ ← Covariance matrix of X

  - Recall the Rayleigh quotient of S and u_1:

$$R(S,u_1) = \frac{u_1^T S u_1}{u_1^T u_1} = u_1^T S u_1 \leq l_{\max}, \text{ and } R(S,v_{\max}) = l_{\max} \Rightarrow u_1 = v_{max}$$

λ_{max}: the largest eigenvalue of S
v_{max}: the eigenvector corresponding to λ_{max}

https://en.wikipedia.org/wiki/Rayleigh_quotient

First Principle Component

Maximize Variance of Projection — First Principle Component Case

$$\begin{aligned}
\frac{1}{N}\sum_{i}^{N}\left(\mu^{\top}x^{(i)}\right)^{2} &= \frac{1}{N}\sum_{i=1}^{N}\mu^{\top}x_i x_i{}^{\top}\mu \\
&= \mu^{\top}\left(\frac{1}{N}\sum_{i=1}^{N}x_i x_i{}^{\top}\right)\mu \qquad (1) \\
&= \mu^{\top}\Sigma\mu \\
\mu^{*} &= \arg\max_{u,||u||_2=1}\mu^{\top}\Sigma\mu
\end{aligned}$$

Claim

μ^{*} is the eigenvector of Σ corresponding to the largest eigenvalue.

First Principle Component

Proof

A is a real symmetric and positive semi-definite matrix $\Rightarrow A = V\Lambda V^\top$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$, $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ and V is a matrix of eigenvectors of A arranged as columns (orthonormal): $V = [v_1, v_2, \cdots, v_n]$.
Suppose $\mu = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$, $\sum_{i=1}^n a_i^2 = 1$, then:

$$\mu^\top A \mu = \sum_{i=1}^n a_i^2 \lambda_i \tag{2}$$

$$\mu^* = \arg \max_{u, ||u||_2 = 1} \mu^\top A \mu = \arg \max_{a_i, \sum_{i=1}^n a_i^2 = 1} \sum_{i=1}^n a_i^2 \lambda_i = v_1$$

Principal Component Analysis (PCA)

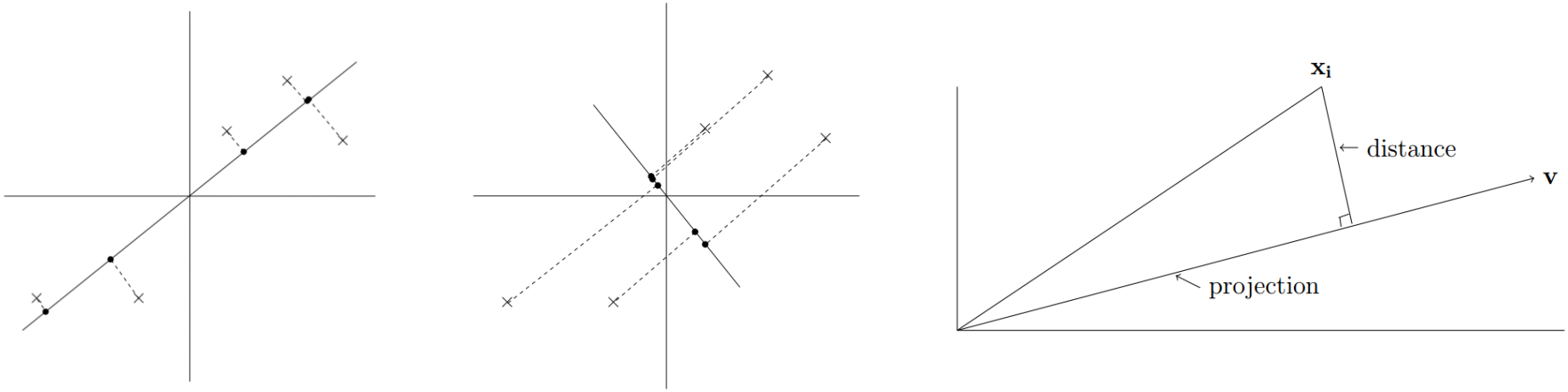
- Find the second principal component:

$$\max \frac{1}{2N} \sum_{n=1}^{N} (u_2^T x_n - u_2^T \bar{x}) = \max u_2^T S u_2$$

$$subject\ to\ \left\| u_2 \right\|_2 = 1,\ u_1^T u_2 = 0$$

  - Solution: u_2 is the eigenvector corresponding to the second largest eigenvalue of S.
- We can compute the entire PCA solution by just computing the eigenvectors with the top-*K* eigenvectors of the covariance matrix.
  - We can compute PCA using Singular Value Decomposition (SVD).

PCA



a b c

Figure: (a)(b) Projection to different axes u, where $\|u\|_2 = 1$;
(c) $\|x_i\|_2^2$= distance^2+ projection^2

Minimizing the perpendicular distance to the axis is equivalent to maximizing the variance of projection (in sense of 2-norm).

PCA Algorithm

- PCA algorithm:
  - X: N x m data matrix, with each row x_i representing a data point
  - X subtract mean x from each row vector x_i in X
  - covariance matrix $\boldsymbol{\Sigma}$ of X
  - Find eigenvectors and eigenvalues of $\boldsymbol{\Sigma}$
  - PC's ← the M eigenvectors with largest eigenvalues

- Zero mean and unit variable normalization
  - Let $\boldsymbol{\mu} = 1/N \sum x_i$
  - Replace each x_i with $x_i - \boldsymbol{\mu}$
  - Let $\boldsymbol{\sigma}_j^2 = 1/N \sum x_{ij}^2$
  - Replace each x_{ij} with $x_{ij}/\boldsymbol{\sigma}_j$

How many components?

- More generally, if we wish to project our data onto a k-dimensional subspace, we should choose $v_1,\cdots,v_k$, but how many should be enough?

- For each singular vector, check how much data variance is represented by this singular vector.
- Take enough many eigenvectors to cover e.g., 80-90% of the variance

$$\frac{\sum_i^k \sigma_i}{\sum_i^m \sigma_i} = 80\%$$

PCA Limitations

Problem

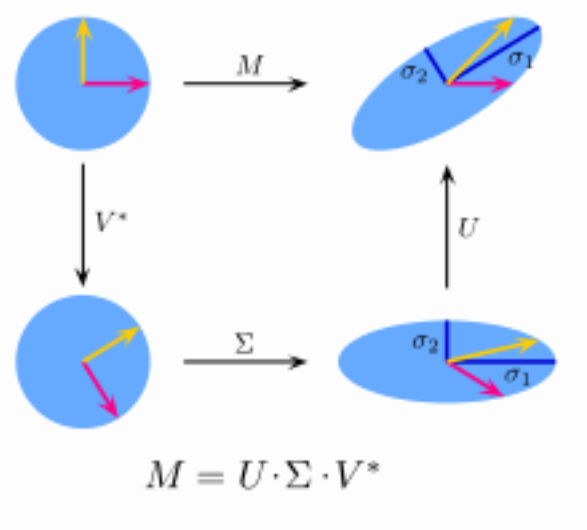
- The covariance matrix Σ is huge —$O(m^2)$, if the dimension of the original data is large

Singular Value Decomposition

- Definition: Singular Value Decomposition: For $X \in R^{n \times m}$, there exists a factorization of the form

$$X = UDV^{\mathsf{T}}$$

with U an $n \times k$ orthogonal matrix, D an $r \times k$ diagonal matrix of singular values, and V an $m \times k$ orthogonal matrix.



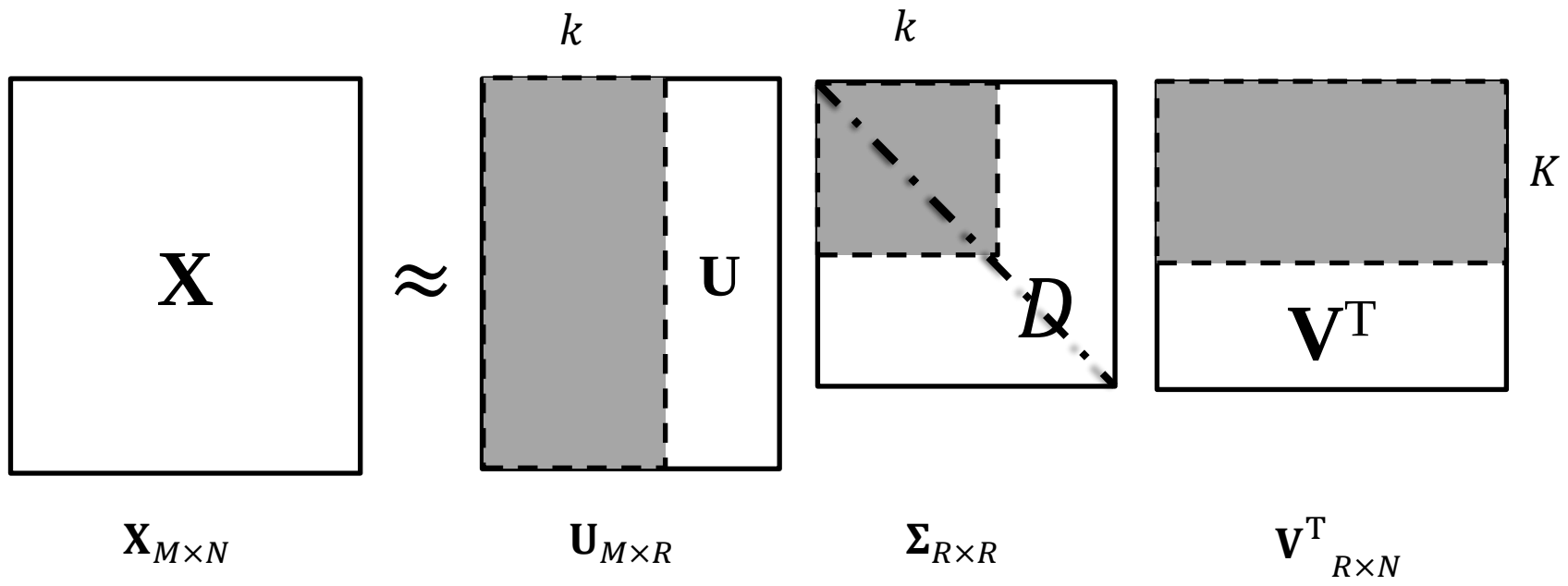
Singular Value Decomposition

- Singular value decomposition (SVD) provides another way to factorize a matrix, into singular vectors and singular values.
- Every real matrix has a singular value decomposition.
- Interpretation:
  - the left-singular vectors of i (i.e., columns in U) are the eigenvectors of XX^T.
  - The right-singular vectors of X (i.e., columns in V) are the eigenvectors of X^TX.
  - The nonzero singular values of X (i.e., D) are the square roots of the eigenvalues of X^TX.
- One useful feature of the SVD is to partially generalize matrix inversion to nonsquare matrices.

Truncated SVD

- TSVD factorizes the data matrix into the three matrices U, D, and V with the *k*-dimension.

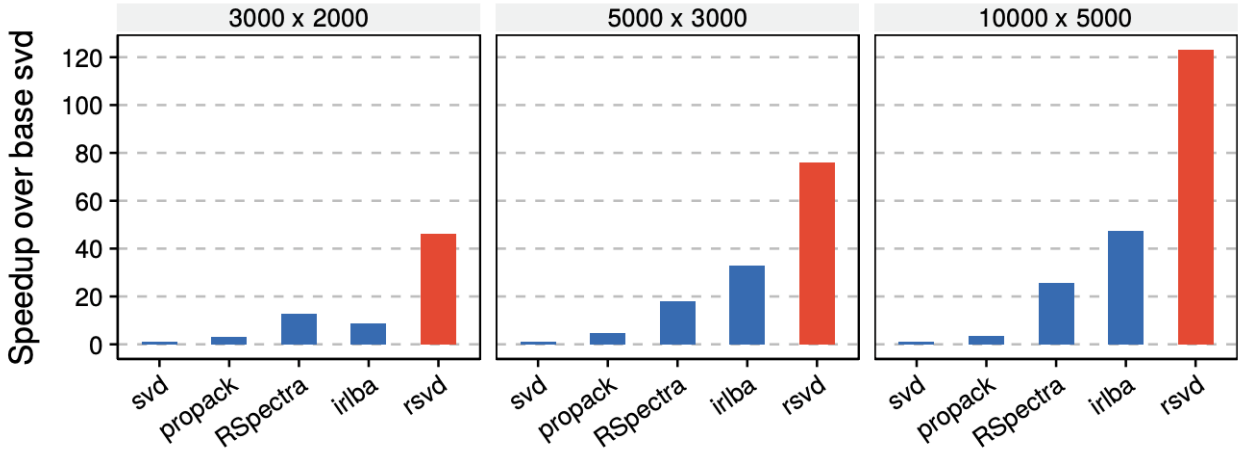
$$X \approx X_k = U_k D_k V_k^{\mathrm{T}}$$



The complexity is $O(\min(m^2n, mn^2))$, how to speed up?

Fast Random tSVD

- Randomness as a computational strategy
- 10^2 speedup!
- Brief Algorithm:
  - Stage A: Construct a low dimensional subspace to approximate the column space of X, with $X \approx QQ^TX$, Q is mxk, k<<min(m, n)
    - $Y=A\Omega$, Ω is randomly generated by gaussian;
    - $Y=QR$
  - Stage B: $B=Q^TX$, do tSVD for $B=SDV^T$, $QS=U$



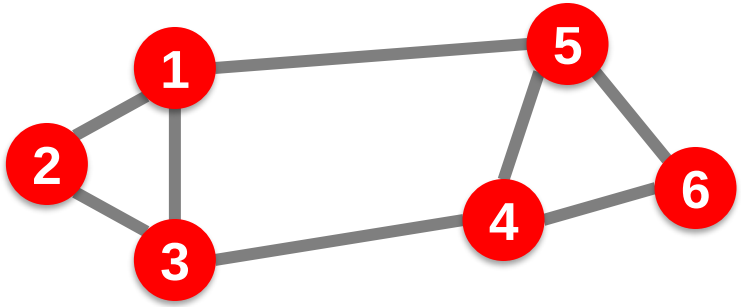
$$X \approx Q \quad B$$
$$nxm \quad mxk \quad kxn$$

- Let us talk a bit about graph clustering
  - Spectral clustering
    [Donath & Hoffman, 1973; Fiedler, 1973]

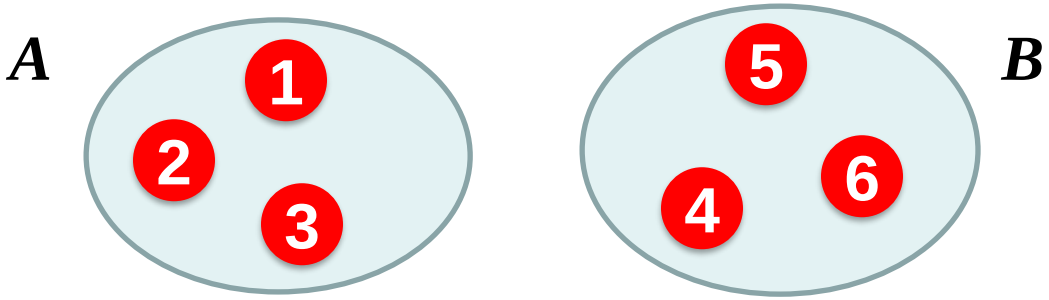


Graph Partitioning

- Undirected graph $G(V, E)$:



- Bi-partitioning task:
  - Divide vertices into two disjoint groups A, B

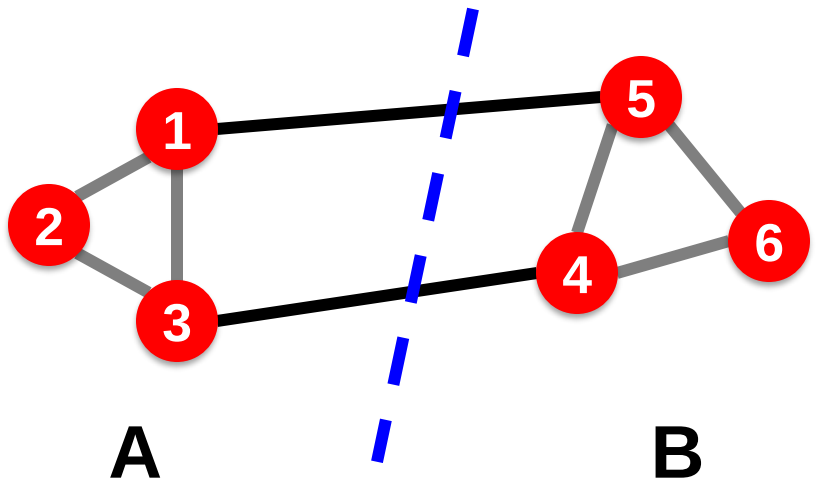


- Questions:
  - How can we define a "good" partition of G?
  - How can we efficiently identify such a partition?

Some of the following slides are from Jure Leskovec's course slides

Graph Partitioning

- What makes a good partition?
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections

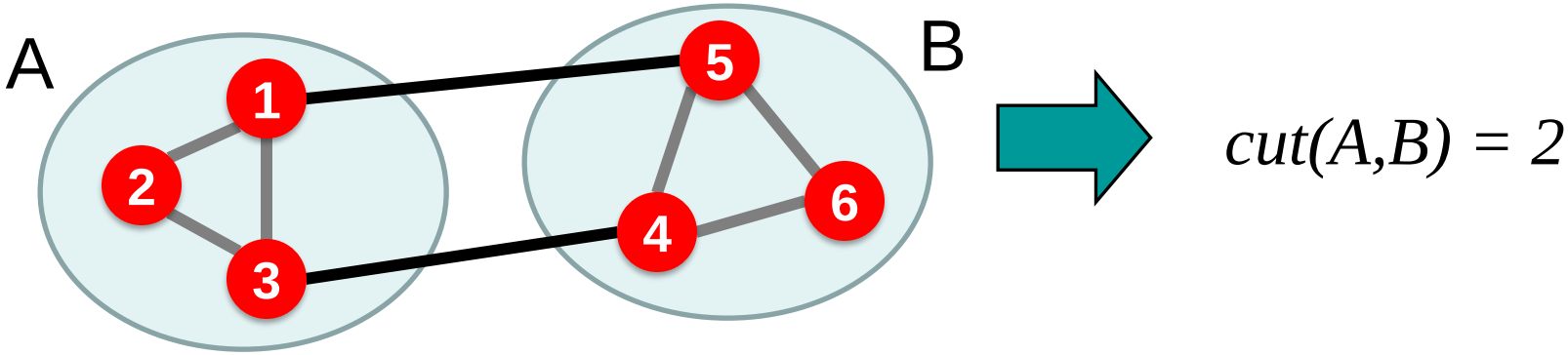


Graph Cuts

- Express partitioning objectives as a function of the “edge cut” of the partition
- Cut: Set of edges with one endpoint in each group:

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$

If the graph is weighted w_{ij} is the weight, otherwise, all $w_{ij} \in \{0,1\}$

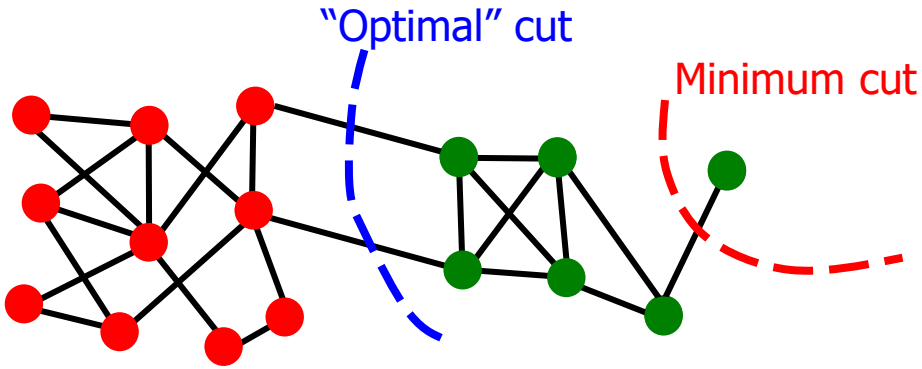


Graph Cut Criterion

- Criterion: Minimum-cut
  - Minimize weight of connections between groups

$$\arg\min_{A,B} cut(A,B)$$

- Degenerate case:



- Problem:
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

Graph Cut Criterion

- RatioCut
  - Connectivity between groups relative to the size of each group

$$RatioCut(A, B) = \text{cut}(A, B) / |A|$$

$|A|$: the number of nodes in A;

- NormalizedCut
  - Connectivity between groups relative to the density of each group

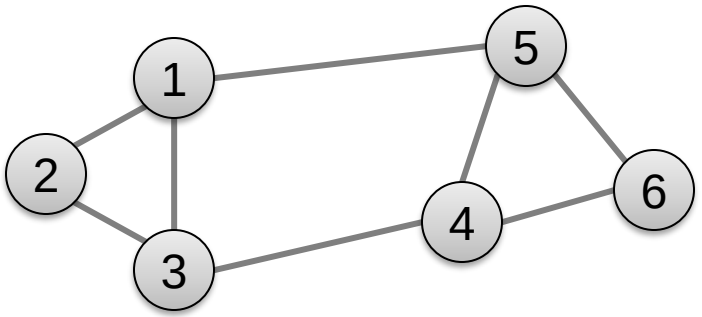
$$NCut(A, B) = \text{cut}(A, B) / \min(\text{vol}(A), \text{vol}(B))$$

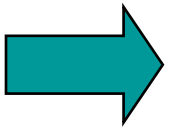
$vol(A)$: total weighted degree of the nodes in A:
$vol(A) = \sum_{i \in A} k_i$ (number of edge end points in A)

- Why use this criterion?
  - Produces more balanced partitions
- How do we efficiently find a good partition?
  - Problem: Computing best conductance cut is NP-hard, $O(k2^n)$

Adjacency Matrix

- Adjacency matrix (A):
  - $n \times n$ matrix
  - $A=[a_{ij}]$, $a_{ij}=1$ if edge between node i and j



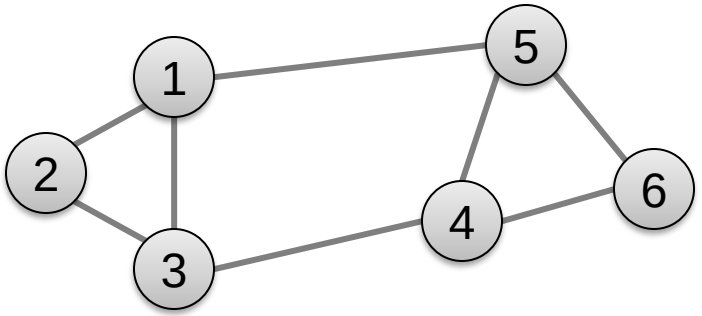


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- Important properties:
  - Symmetric matrix
  - Has n real eigenvalues
  - Eigenvectors are real-valued and orthogonal

Degree Matrix

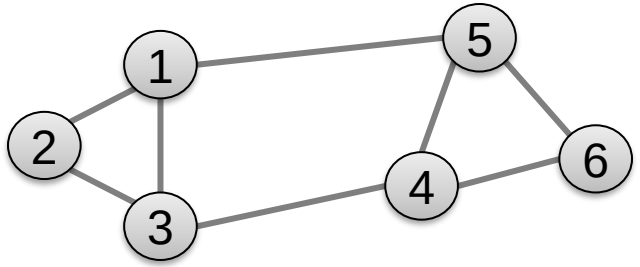
- Degree matrix (D):
  - $n \times n$ diagonal matrix
  - $D=[d_{ii}]$, d_{ii} = degree of node i



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Laplacian Matrix

- Laplacian matrix (L):
  - $n \times n$ symmetric matrix



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$$L = D - A$$

- What is trivial eigenpair?
  - $x = (1, \ldots, 1)$ then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$
  - $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

Important Properties Laplacian L

(a) All eigenvalues are ≥ 0

(b) $x^T L x = \sum_{ij} L_{ij} x_i x_j \geq 0$ for every x

(c) L can be written as $L = N^T \cdot N$

– That is, L is positive semi-definite

	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- Proof: (the 3 facts are saying the same thing)
  - (c)⇒(b): $x^T L x = x^T N^T N x = (Nx)^T (Nx) \geq 0$
    - As it is just the square of length of Nx
  - (b)⇒(a): Let λ be an eigenvalue of L. Then by (b) $x^T L x \geq 0$ so $x^T L x = x^T \lambda x = \lambda x^T x \Rightarrow \lambda \geq 0$
  - (a)⇒(c): is also easy!

Important Properties Laplacian *L*

- Important properties of L:
  - Symmetric
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real (and always orthogonal)
  - For any vector *f*, we have

	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$$L = D - A$$

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

  - which can be proved:

$$f^T L f = fTDf - f^T W f = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} w_{ij} f_i \ \ f_j$$

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

Example: RatioCut for *k*-spectral clustering

- Let us introduce $h_j=\{h_1, \ldots, h_k\}$, and each h_j is a n-dimensional vector

$$h_{ji} = \begin{cases} 0, & vi \notin A_j \\ \dfrac{1}{\sqrt{|A_j|}}, & vi \in A_j \end{cases}$$

Then we have

$$\begin{aligned} h_i^T L h_i &= \frac{1}{2}\sum_{m=1}\sum_{n=1} w_{mn}(h_{im}-h_{in})^2 \\ &= \frac{1}{2}\Big(\sum_{m\in A_i, n\notin A_i} w_{mn}\Big(\frac{1}{\sqrt{|A_i|}}-0\Big)^2 + \sum_{m\notin A_i, n\in A_i} w_{mn}\Big(0-\frac{1}{\sqrt{|A_i|}}\Big)^2 \\ &= \frac{1}{2}\Big(\sum_{m\in A_i, n\notin A_i} w_{mn}\frac{1}{|A_i|} + \sum_{m\notin A_i, n\in A_i} w_{mn}\frac{1}{|A_i|} \\ &= \frac{1}{2}\Big(cut(A_i,\overline{A}_i)\frac{1}{|A_i|} + cut(\overline{A}_i, A_i)\frac{1}{|A_i|}\Big) \\ &= \frac{cut(A_i,\overline{A}_i)}{|A_i|} \\ &= RatioCut(A_i,\overline{A}_i) \end{aligned}$$

Now we have

$$RatioCut(A_1, A_2, \ldots A_k) = \sum_{i=1}^{k} h_i^T L h_i = \sum_{i=1}^{k} (H^T L H)_{ii} = tr(H^T L H)$$

Spectral Clustering

- A: adjacency matrix of undirected G
  - A_{ij} =1 if (i, j) is an edge, else 0
- x is a vector in $\Re^n$ with components $(x_1, \ldots, x_n)$
  - Think of it as a label/value of each node of G
- What is the meaning of $A \cdot x$?

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$y_i = \sum_{j=1}^{n} A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

- Entry y_i is a sum of labels x_j of neighbors of i

What is the Meaning of *Ax*?

- j^{th} coordinate of $A \cdot x$:
  - Sum of the x-values of neighbors of j
  - Make this a new x-value at node j

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{A \cdot x = \lambda \cdot x}$$

- Spectral Graph Theory:
  - Analyze the "spectrum" of matrix representing G
  - Spectrum: Eigenvectors $x^{(i)}$ of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i: $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$

$$\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$$

Note: We sort $\boldsymbol{\lambda_i}$ in ascending (not descending) order!

Find Optimal Cut [Fiedler'73]

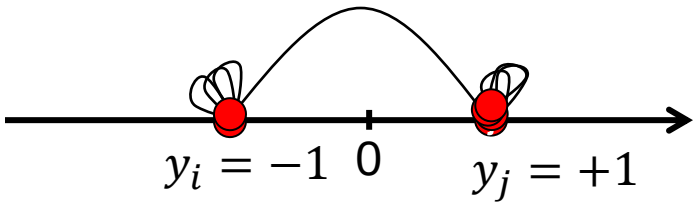
- Back to finding the optimal cut
- Express partition (A,B) as a vector

$$y_i = \begin{cases} +1 & if\ i \in A \\ -1 & if\ i \in B \end{cases}$$

- Enforce that |A| = |B| → $\Sigma_i y_i = 0$
  - Equivalent to being orthogonal to the trivial eigenvector $(1, \ldots, 1)$
- We can minimize the cut of the partition by finding a vector y that minimizes:

$$\arg \min_{y \in \{-1,+1\}^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2$$

Can't solve exactly. Let's relax y and allow it to take any real value.



λ_2 as an Optimization Problem

- Fact: For symmetric matrix M:

$$\lambda_2 = \min_{x \,:\, x^T w_1 = 0} \frac{x^T M x}{x^T x}$$

($\mathbf{w_1}$ is eigenvector corresponding to λ_1)

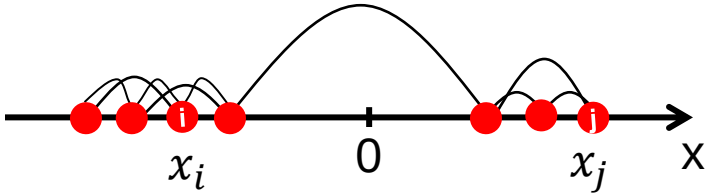
See next slide for the proof. Deriving this is a HW problem.

- What is the meaning of min $x^T L\, x$ on G?
  - $x^T L\, x = \sum_{i,j=1}^{n} L_{ij}\, x_i x_j = \sum_{i,j=1}^{n} \left(D_{ij} - A_{ij}\right) x_i x_j$
  - $= \sum_i D_{ii} x_i^2 - \sum_{(i,j)\in E} 2 x_i x_j$
  - $= \sum_{(i,j)\in E} (x_i^2 + x_j^2 - 2 x_i x_j) = \sum_{(i,j)\in E} \left(x_i - x_j\right)^2$

Node i has degree d_i. So, value x_i^2 needs to be summed up d_i times.
But each edge (i,j) has two endpoints so we need $x_i^2 + x_j^2$

Rayleigh Theorem $\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$

$$\min_{y \in \mathbb{R}^n \,:\, \sum_i y_i = 0,\ \sum_i y_i^2 = 1} f(y) = \sum_{(i,j)\in E} (y_i - y_j)^2 = y^T L y$$



- $\lambda_2 = \min_y f(y)$: The minimum value of $f(y)$ is given by the 2nd smallest eigenvalue λ_2 of the Laplacian matrix L
- $x = \arg\min_y f(y)$: The optimal solution for y is given by the eigenvector x corresponding to λ_2, referred to as the Fiedler vector
- Can use sign of x_i to determine cluster assignment of node i

Approx. Guarantee of Spectral

- Suppose there is a partition of G into A and B where $|A| \leq |B|$, s.t. "conductance" of the cut (A,B) is $\beta = \frac{(\#\ edges\ from\ A\ to\ B)}{|A|}$ then $\lambda_2 \leq 2\beta$

  Note: |A|<|B|

  - This is the approximation guarantee of the spectral clustering: Spectral finds a cut that has at most twice the conductance as the optimal one of conductance β.
- Proof:
  - Let: $a = |A|, b = |B|$ and $e = \#$ edges from A to B
  - Enough to choose some x_i based on A and B such that:

    $\lambda_2 \leq \underbrace{\frac{\sum(x_i - x_j)^2}{\sum_i x_i^2}}_{\lambda_2 \text{ is only smaller}} \leq 2\beta$ (while also $\sum_i x_i = 0$)

Approx. Guarantee of Spectral

- Proof (continued):
  - 1) Let's set: $x_i = \begin{cases} -\frac{1}{a} & if\ i \in A \\ +\frac{1}{b} & if\ i \in B \end{cases}$

    Note: |A|<|B|
    - Let's quickly verify that $\sum_i x_i = 0$: $a\left(-\frac{1}{a}\right) + b\left(\frac{1}{b}\right) = 0$
  - 2) Then: $\frac{\sum(x_i - x_j)^2}{\sum_i x_i^2} = \frac{\sum_{i\in A, j\in B}\left(\frac{1}{b}+\frac{1}{a}\right)^2}{a\left(-\frac{1}{a}\right)^2 + b\left(\frac{1}{b}\right)^2} = \frac{e\cdot\left(\frac{1}{a}+\frac{1}{b}\right)^2}{\frac{1}{a}+\frac{1}{b}} =$

    $e\left(\frac{1}{a}+\frac{1}{b}\right) \le e\left(\frac{1}{a}+\frac{1}{a}\right) = e\frac{2}{a} \le 2\beta$

    Which proves that the cost achieved by spectral is better than twice the OPT cost

    e … number of edges between A and B

Approx. Guarantee of Spectral

- Putting it all together: The Cheeger inequality

$$\frac{\beta^2}{2k_{max}} \leq \lambda_2 \leq 2\beta$$

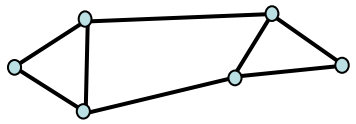
  - where k_{max} is the maximum node degree in the graph
    - Note we only provide the 1st part: $\lambda_2 \leq 2\beta$
    - We did not prove $\frac{\beta^2}{2k_{max}} \leq \lambda_2$
  - Overall this always certifies that λ_2 always gives a useful bound

Spectral Clustering Algorithm

- Three basic stages:
  - 1) Pre-processing
    - Construct a matrix representation of the graph
  - 2) Decomposition
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - 3) Grouping
    - Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

- 1) Pre-processing:
  - Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- 2) Decomposition:
  - Find eigenvalues λ and eigenvectors x of the matrix L

$\lambda =$

0.0
1.0
3.0
3.0
4.0
5.0

$X =$

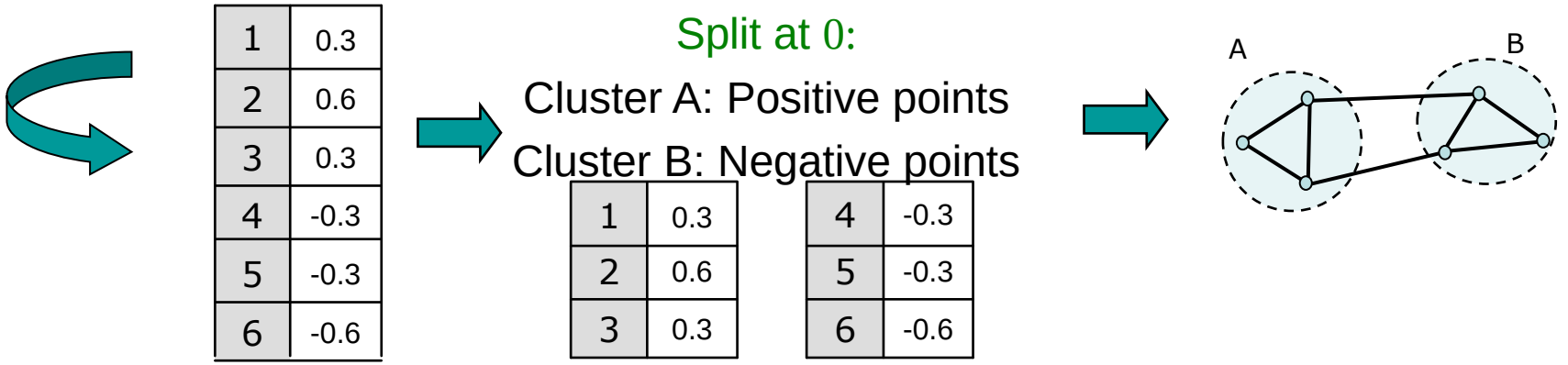
0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

  - Map vertices to corresponding components of x_2

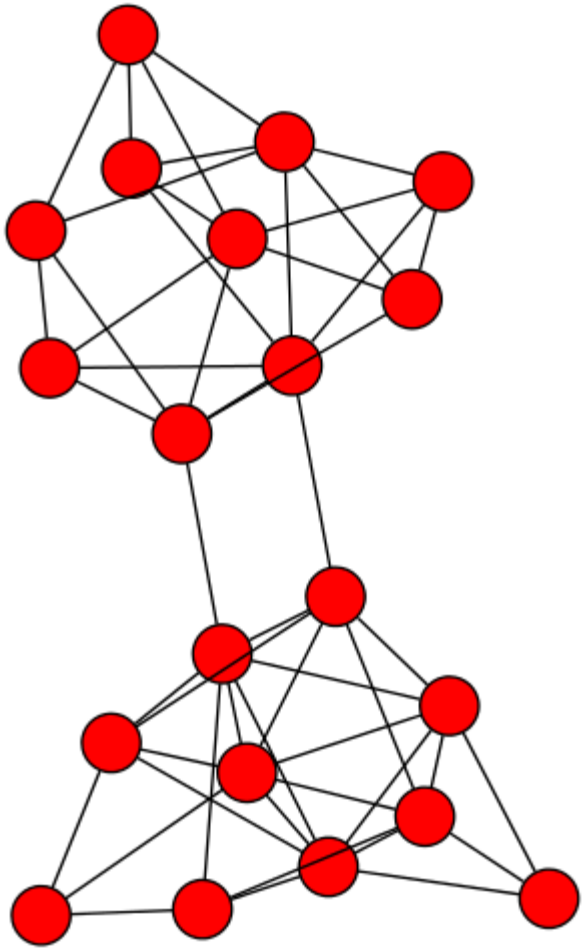
1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

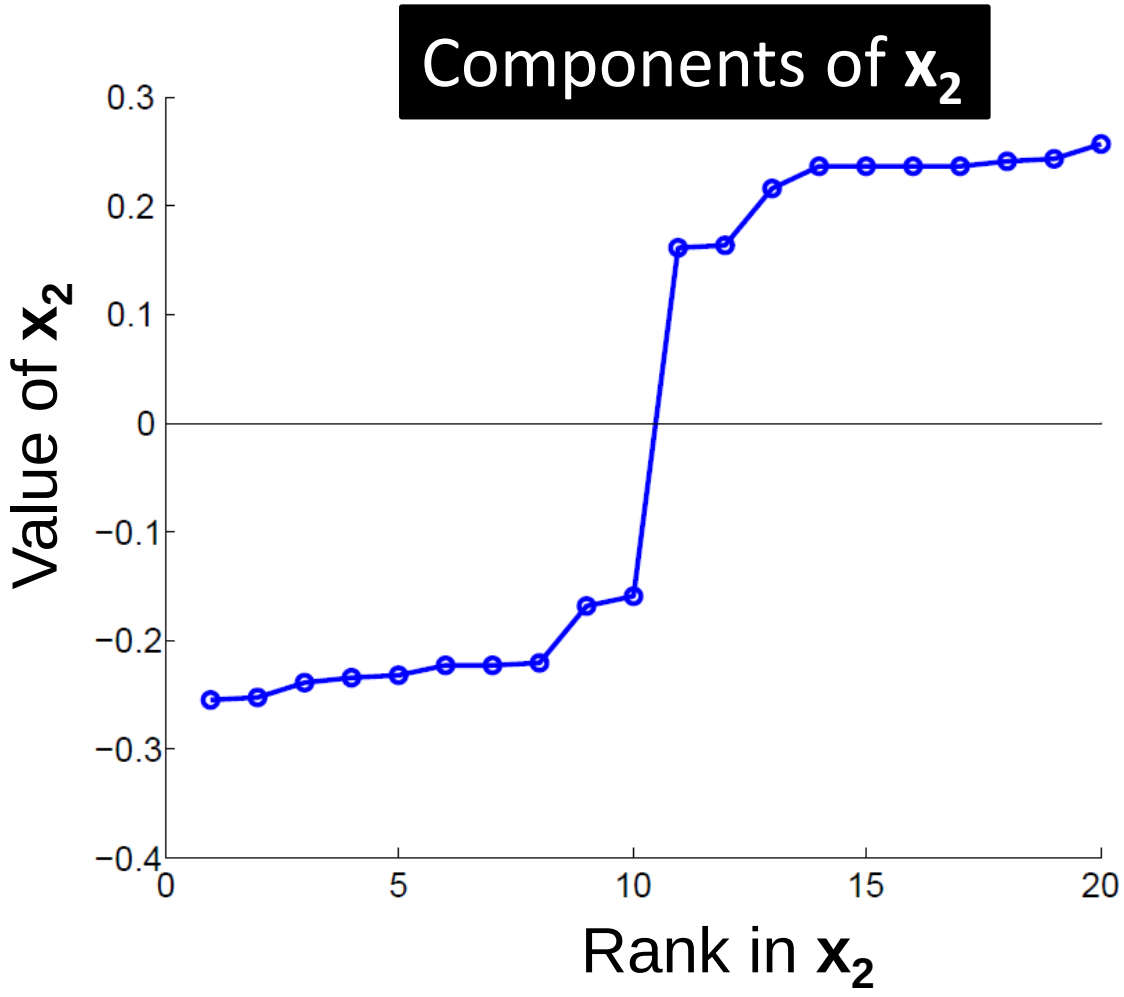
Spectral Partitioning

- 3) Grouping:
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two
- How to choose a splitting point?
  - Naïve approaches:
    - Split at 0 or median value
  - More expensive approaches:
    - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

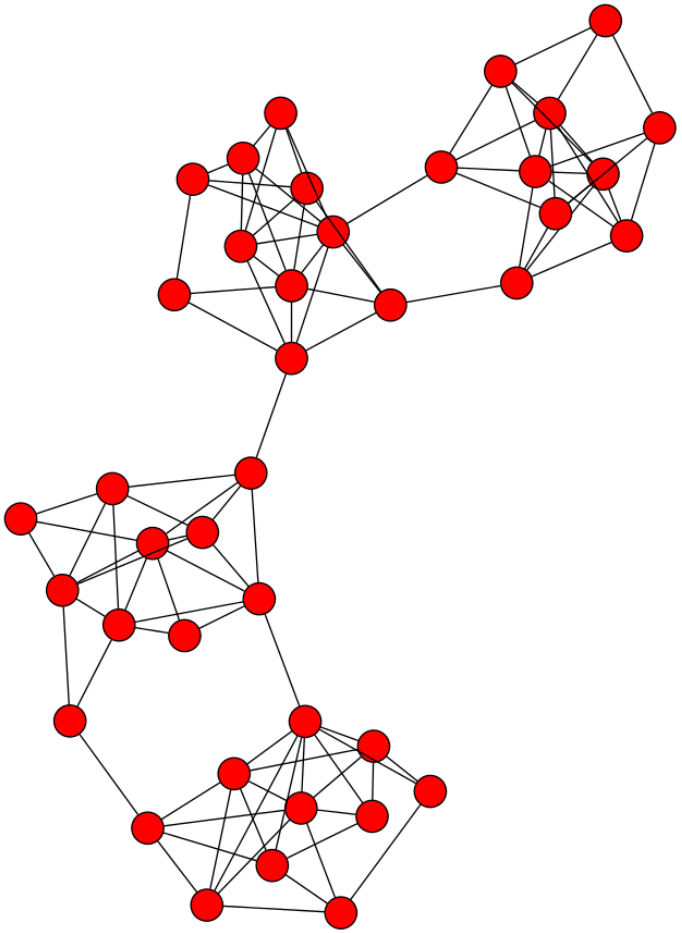


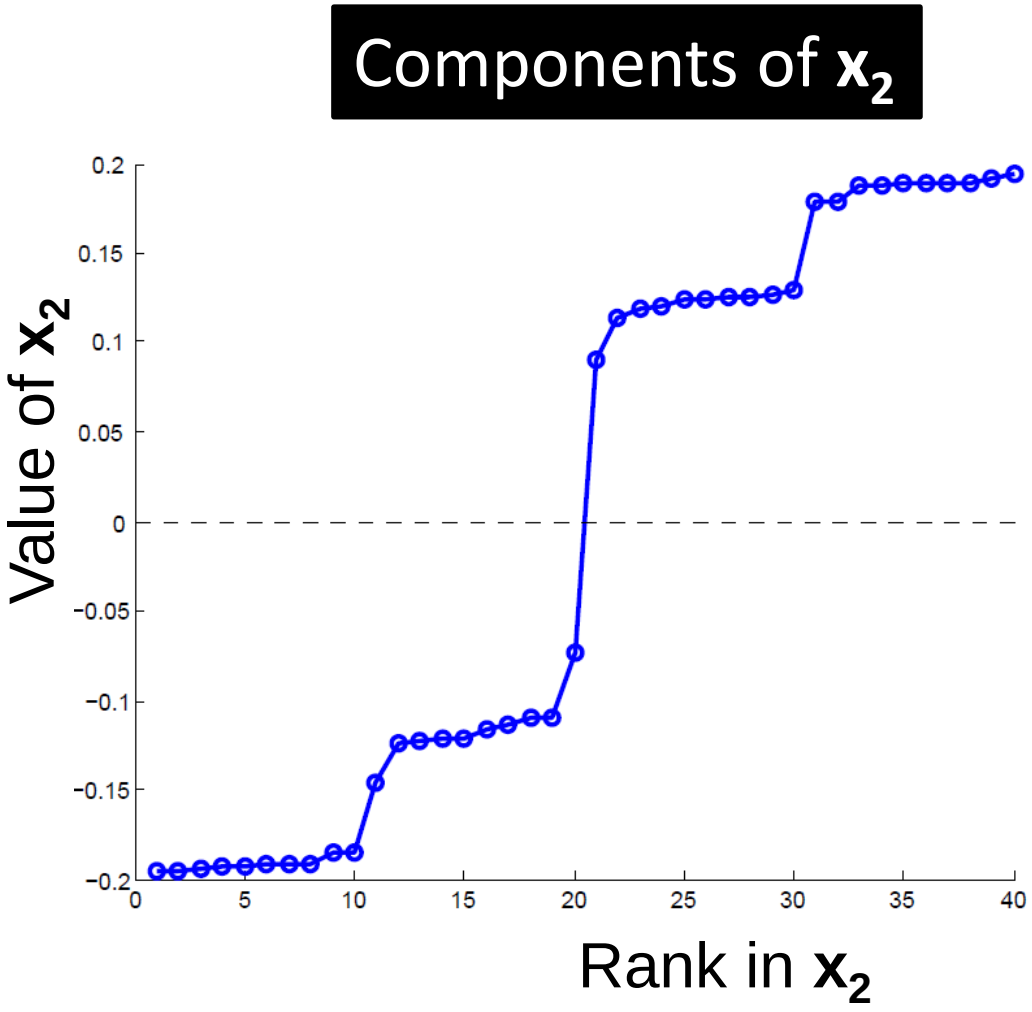
Example: Spectral Partitioning



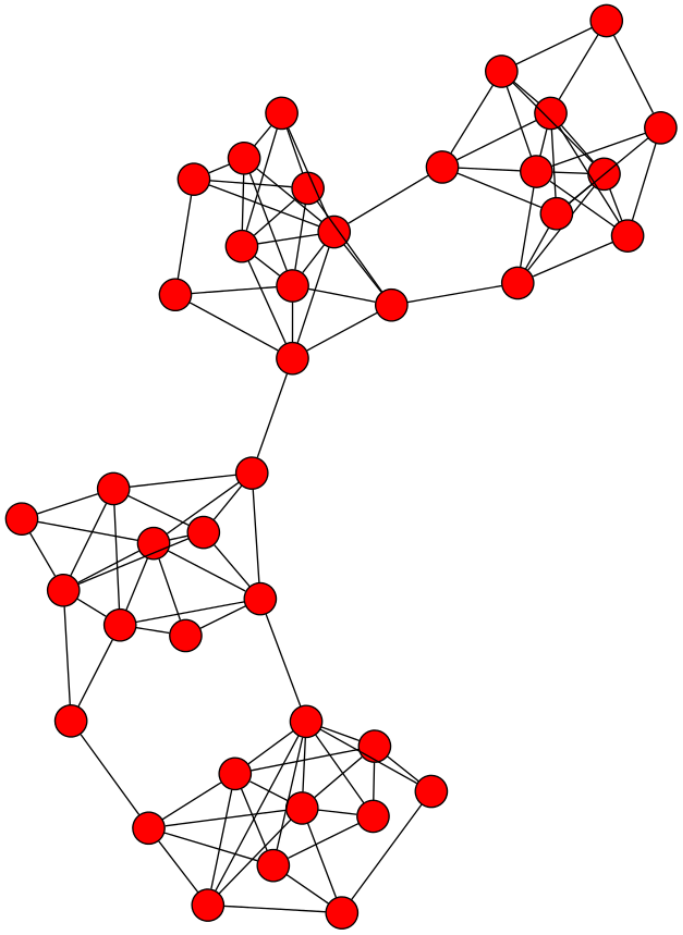


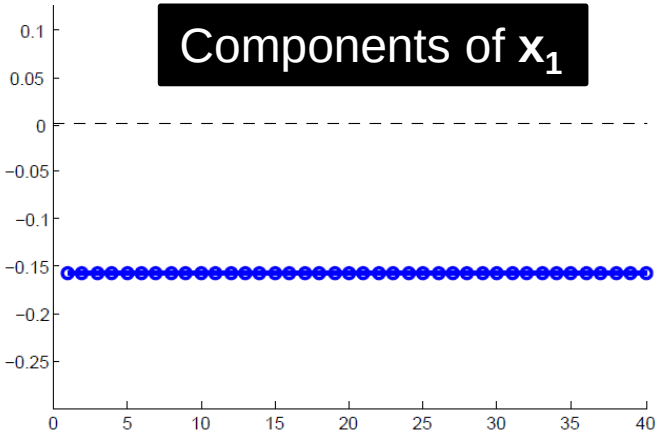
Example: Spectral Partitioning

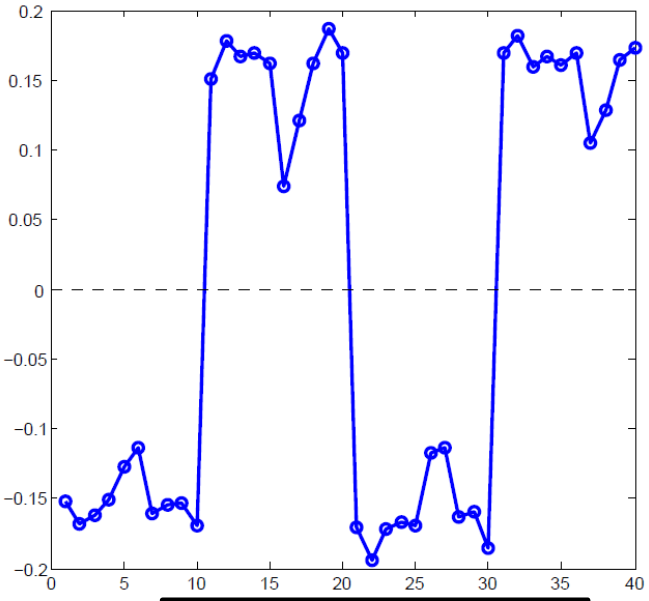




Example: Spectral Partitioning







k-Way Spectral Clustering

- How do we partition a graph into k clusters?
- Two basic approaches:
  - Recursive bi-partitioning [Hagen et al., '92]
    - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    - Disadvantages: Inefficient, unstable
  - Cluster multiple eigenvectors [Shi-Malik, '00]
    - Build a reduced space from multiple eigenvectors
      - Each node is now represented by k numbers
      - We then cluster (apply k-means) the nodes based on their k-dim representation
    - Commonly used in recent papers
    - A preferable approach…

Why Use Multiple Eigenvectors?

- Approximates the optimal cut [Shi-Malik, ’00]
  - Can be used to approximate optimal k-way normalized cut
- Emphasizes cohesive clusters
  - Increases the unevenness in the distribution of the data
  - Associations between similar points are amplified, associations between dissimilar points are attenuated
  - The data begins to “approximate a clustering”
- Well-separated space
  - Transforms data to a new “embedded space”, consisting of k orthogonal basis vectors
- Multiple eigenvectors prevent instability due to information loss

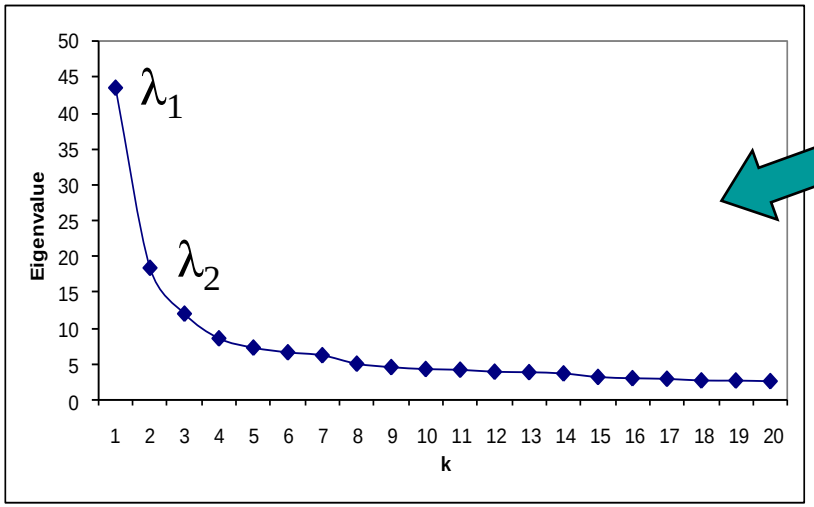
How to Select *k*?

- Eigengap:
  - The difference between two consecutive eigenvalues
- Most stable clustering is generally given by the value k that maximizes eigengap Δ_k:

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

- Example:

Note eigenvalues are sorted in descending order



$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

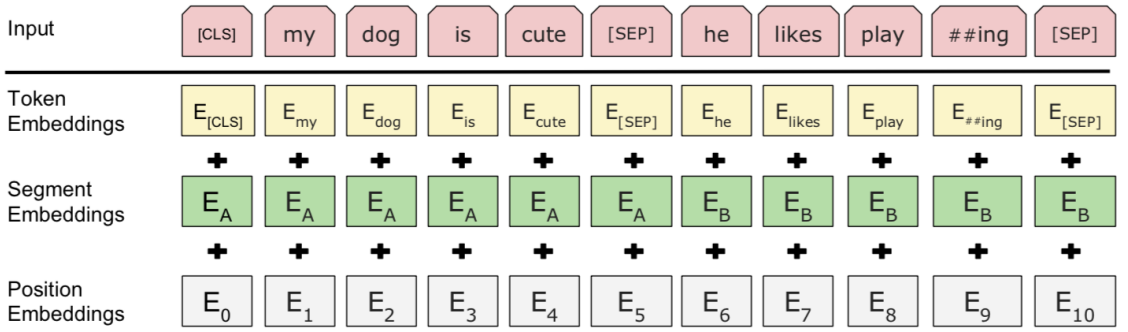
$\Rightarrow$ Choose $k = 2$

Unsupervised Learning

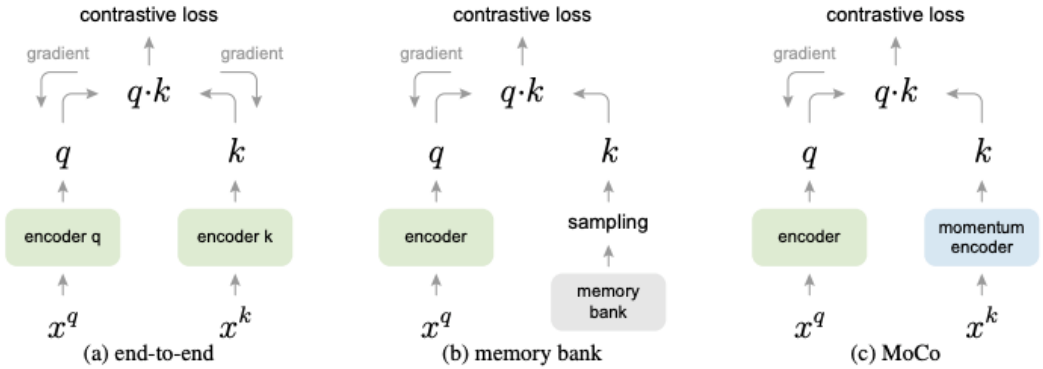
- Clustering
  - K-means, K-medoids, Hierarchical clustering
  - Mixture Models
- Dimension Reduction
  - Principal Component Analysis
  - Spectral Clustering

Still fundamentally challenging

- How to select the best *k*, rather than ad-hoc methods?
- Direction: Pre-training
  [BERT, Jacob'18]



- Self-learning
  [MOCO, He'19]



Thanks!

Jie Tang, DCST
http://keg.cs.tsinghua.edu.cn/jietang/
http://aminer.org
Email: jietang@tsinghua.edu.cn

Appendix: Several useful slides

Proof: $\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$

- Write x in basis of eigenvecs $w_1, w_2, \ldots, w_n$ of $\boldsymbol{M}$ and λ_i are corresponding eigenvalues. So, $x = \sum_i^n \alpha_i w_i$
- Then we get: $Mx = \sum_i \alpha_i M w_i = \sum_i \alpha_i \lambda_i w_i$ ($Mw_i = \lambda_i w_i$)
- So, what is $\boldsymbol{x^T M x}$? (x^T, Mx)
  - $x^T Mx = (\sum_i \alpha_i w_i)^T (\sum_i \alpha_i \lambda_i w_i) = \sum_{ij} \alpha_i \lambda_j \alpha_j$
    $= \sum_i \alpha_i^2 \lambda_i w_i^T w_i = \sum_i \boldsymbol{\lambda_i \alpha_i^2}$
    ($w_i^T w_i$ = 0 if $i \neq j$, 1 otherwise)
  - Want minimize this over all unit vectors $\boldsymbol{w}$:
    $\boldsymbol{w}$ = min over choices of $(\alpha_1, \ldots \alpha_n)$ so that:
    - $x^T w_1 = 0$, rewrite it as $(\sum_i \alpha_i w_i) \cdot w_1 = 0$ and remember that $w_i^T w_j = 0$ (because w are eigenvectors). Then $\alpha_1 = 0$
    - $\sum \alpha_i^2 = 1$ (unit length)
  - So, to minimize this, set $\boldsymbol{\alpha_2 = 1}$ and the rest to 0 $\sum_i \boldsymbol{\lambda_i \alpha_i^2 = \lambda_2}$

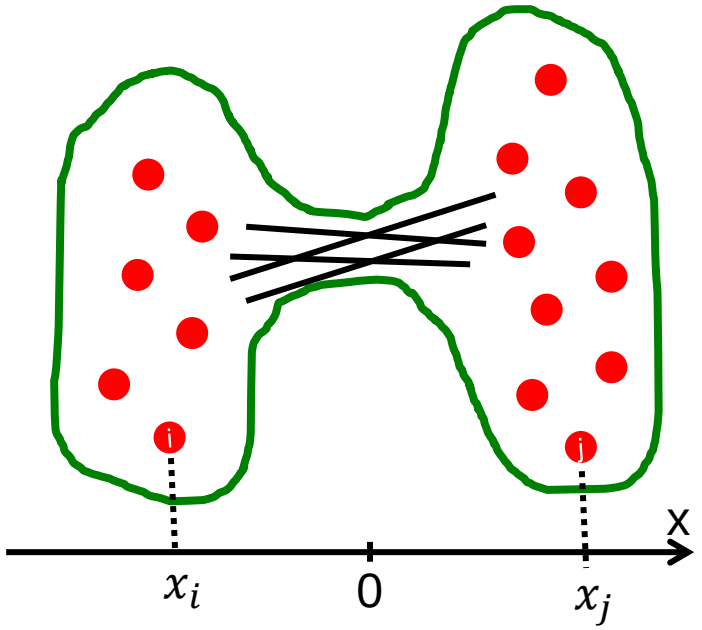
Finding x that Solves $\lambda_2 = \min_{x\,:\,x^T w_1 = 0} \frac{x^T M x}{x^T x}$

- What else do we know about x?
  - x is unit vector: $\sum_i x_i^2 = 1$
  - x is orthogonal to 1st eigenvector $(1, \ldots, 1)$ thus: $\sum_i x_i \cdot 1 = \sum_i x_i = 0$

- Remember:

$$\lambda_2 = \min_{\substack{\text{All labelings}\\ \text{of nodes } i \text{ so}\\ \text{that } \sum x_i = 0}} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_i x_i^2}$$

We want to assign values x_i to nodes i such that few edges cross 0.
(we want x_i and x_j to subtract each other)



Balance to minimize