CS60050: Machine Learning

Clustering

Sourangshu Bhattacharya

CLUSTERING APPLICATIONS

Supervised vs Unsupervised learning

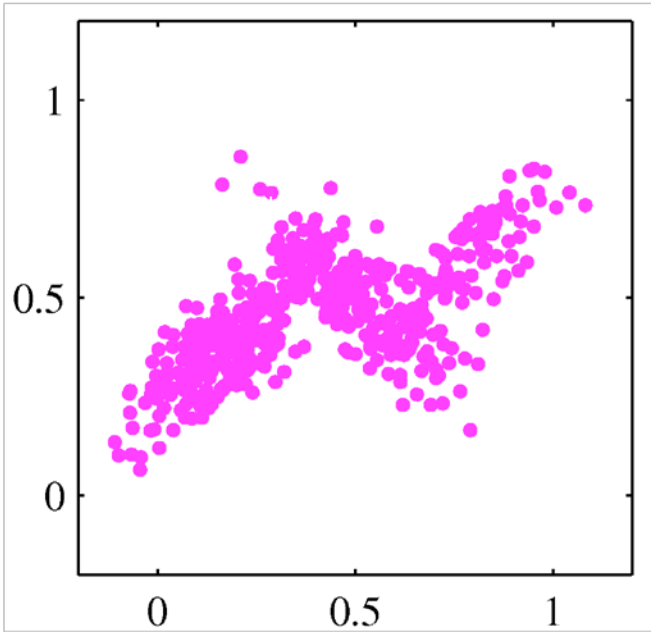
- Supervised learning: Given $(x_i, y_i), i = 1, \ldots, n$, learn a function $f : X \; ! \;\; Y$.
  - Categorical Y: classification
  - Continuous Y: regression
- Unsupervised learning: Given only $(x_i), i = 1, \ldots, n$, can we infer the underlying structure of X?

Why do unsupervised learning?

- Raw data cheap. Labeled data expensive.
- Save memory/computation.
- Reduce noise in high-dimensional data.
- Useful in exploratory data analysis.
- Often a pre-processing step for supervised learning.

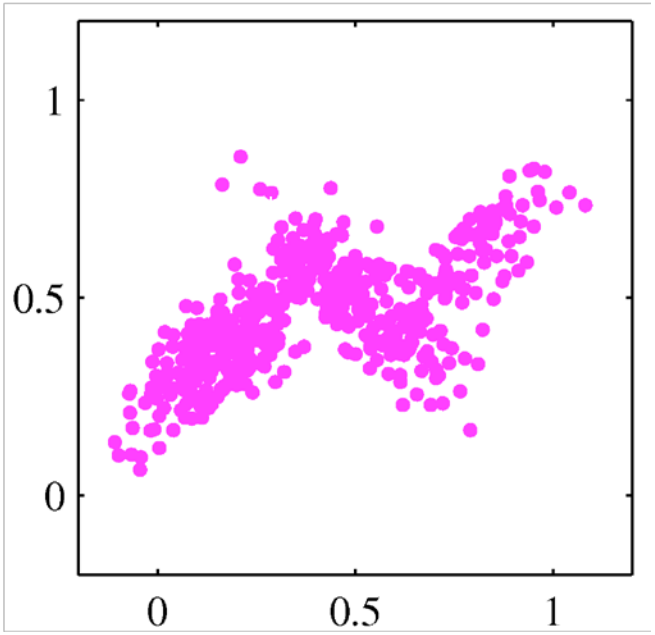
Cluster analysis

Discover groups such that samples within a group are more similar to each other than samples across groups.



Cluster analysis

Discover groups such that samples within a group are more similar to each other than samples across groups.



.

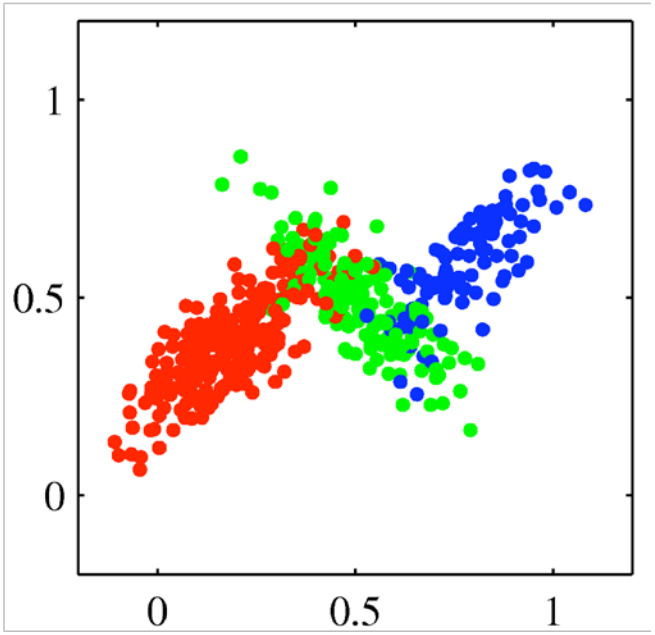
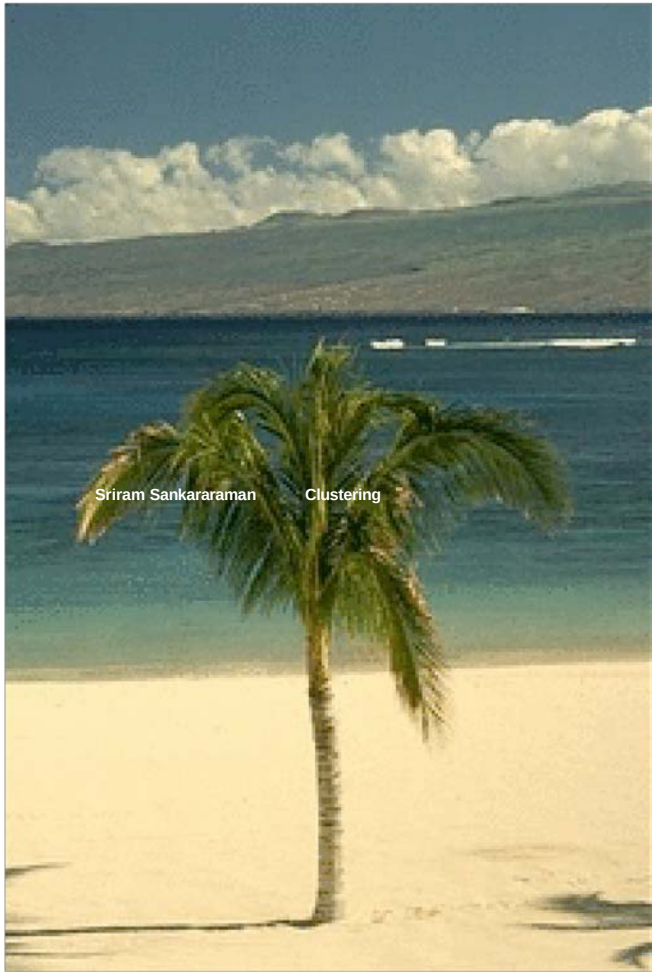
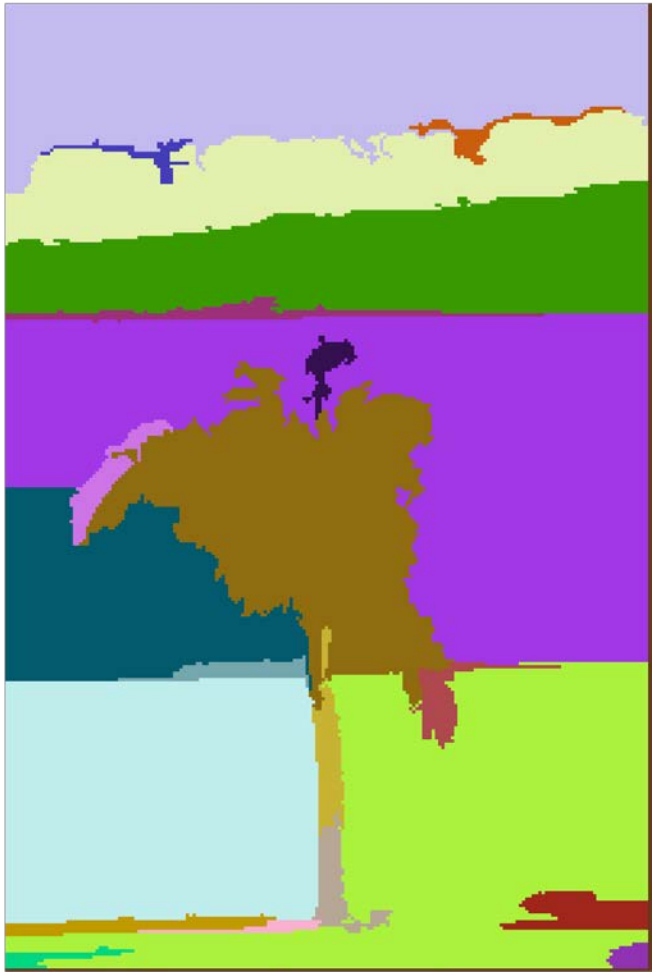


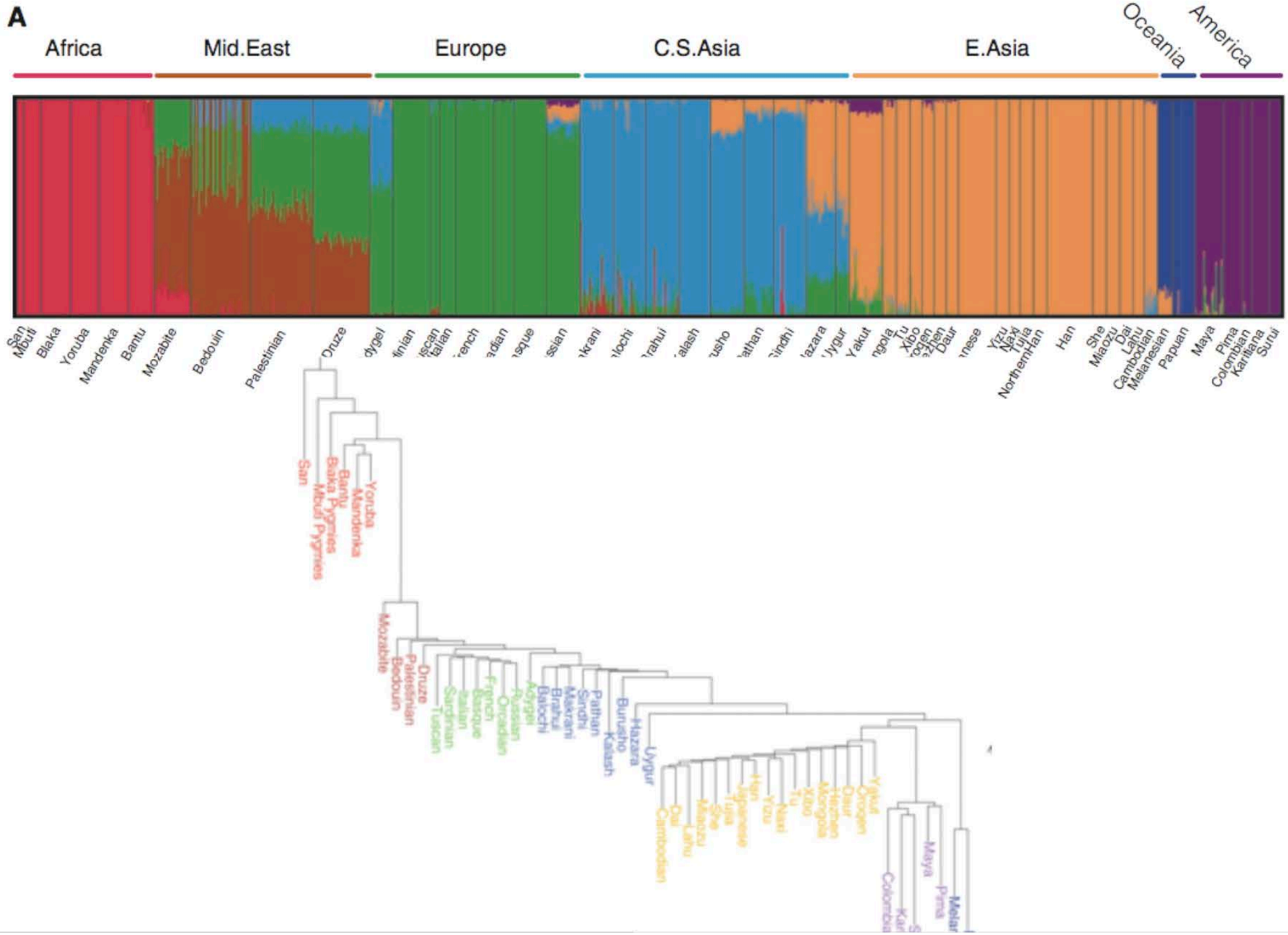
Image Segmentation



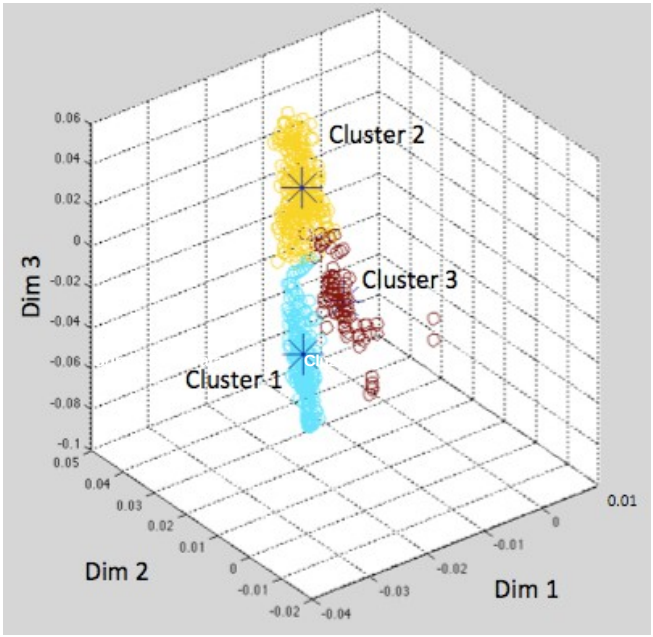


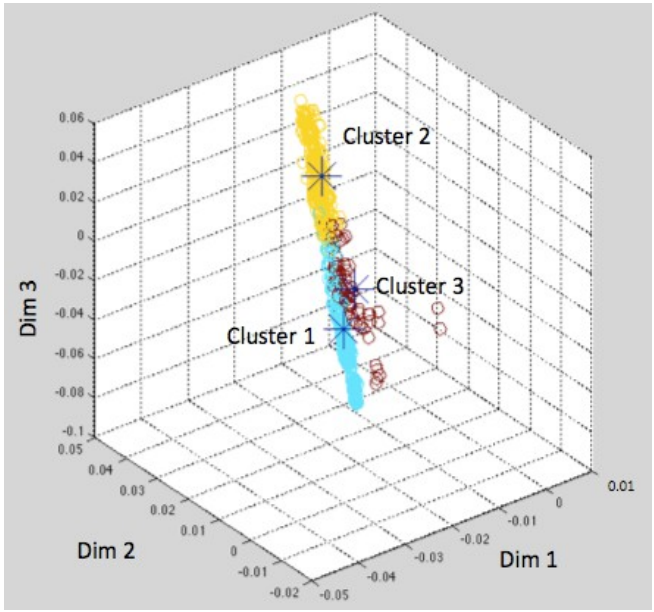
http://people.cs.uchicago.edu/ pff/segment

Human population structure

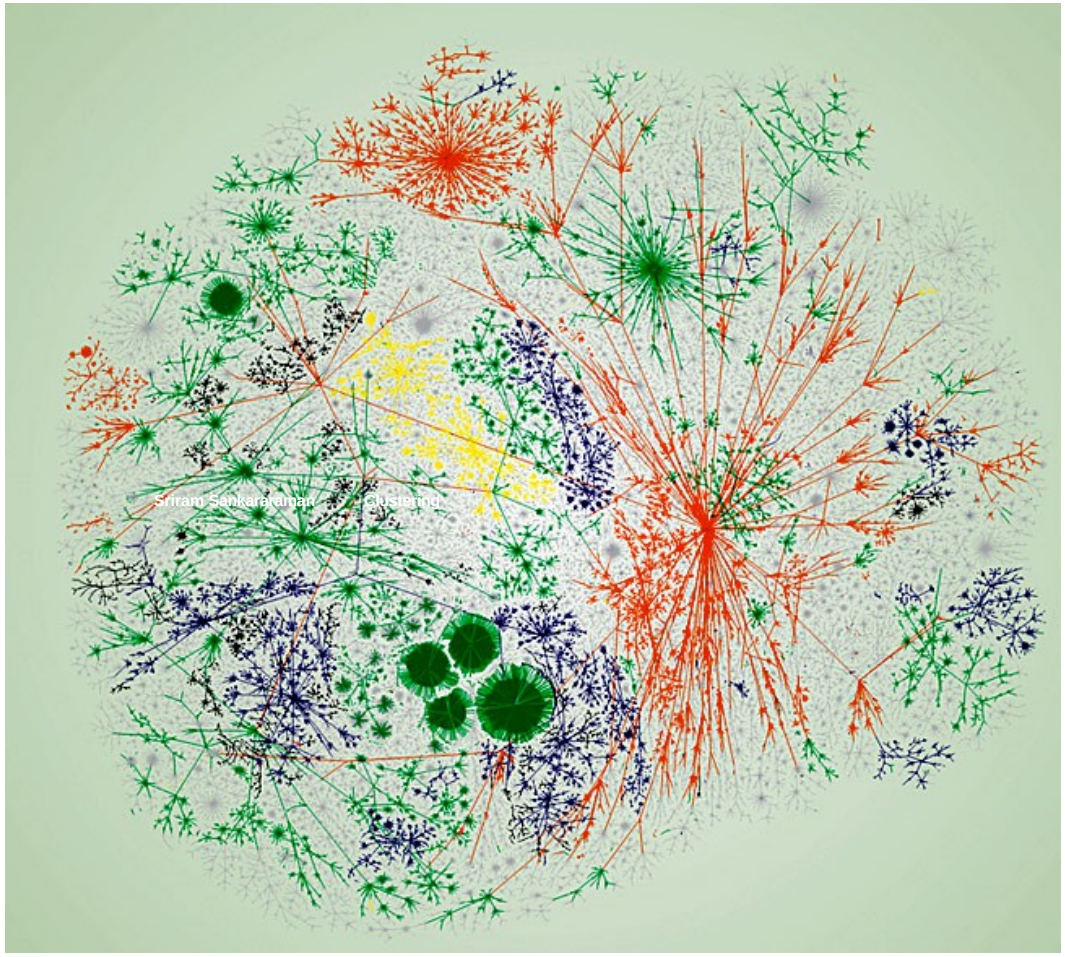


Clustering Web2.0 workloads





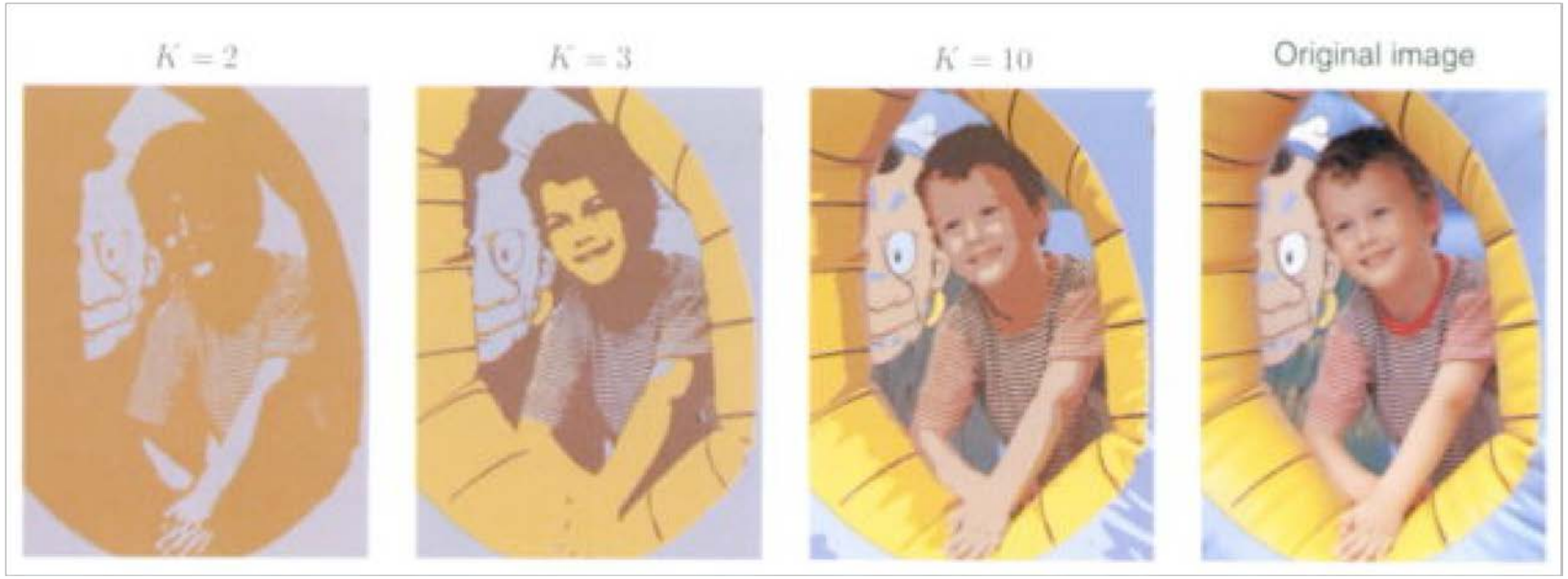
Clustering graphs



Finding communities in social networks

Newman, 2008

Vector quantization to compress images



Bishop, PRML

Ingredients of cluster analysis

- A dissimilarity function between samples.
- A loss function to evaluate clusters.
- Algorithm that optimizes this loss function.

The Dissimilarity function

- Choice of dissimilarity function is application dependent.
- Need to consider the type of features.
  - Categorical, ordinal or quantitative.
- Possible to learn dissimilarity from data (later).

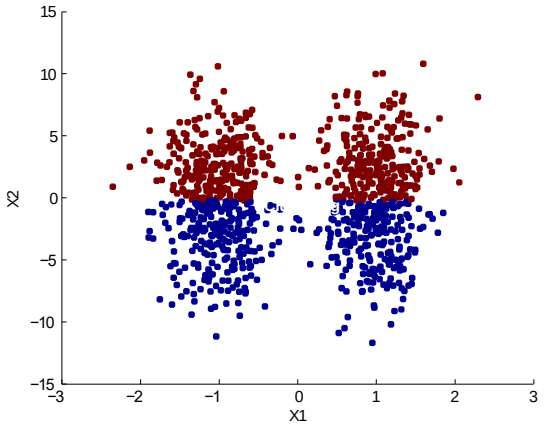
Dissimilarity based on features

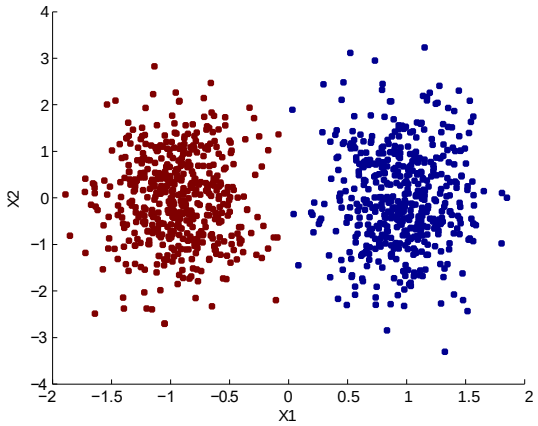
- Data point x_i has features $x_{ij}, j = 1, \ldots, p$.
- One choice of dissimilarity function is the Euclidean distance

$$D(x_i, x_{i'}) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}$$

- Resulting clusters invariant to rotation and translation of features but not to scaling.
- If the features have different scales, standardize the data.

Standardization

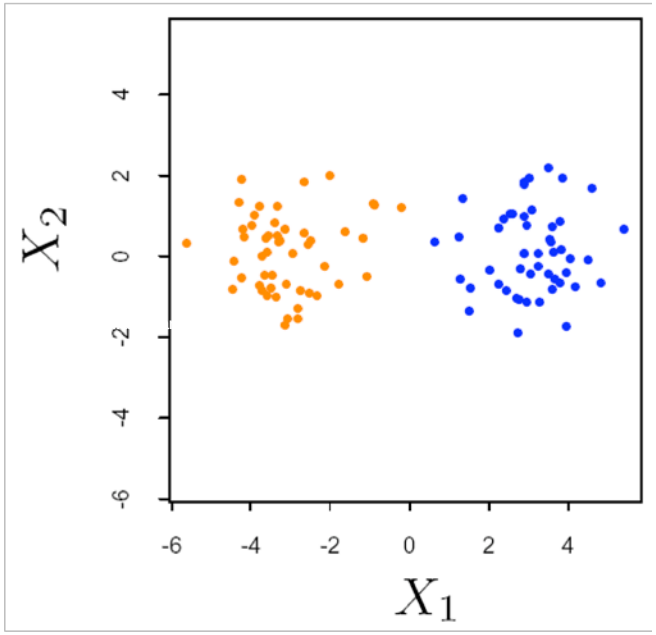




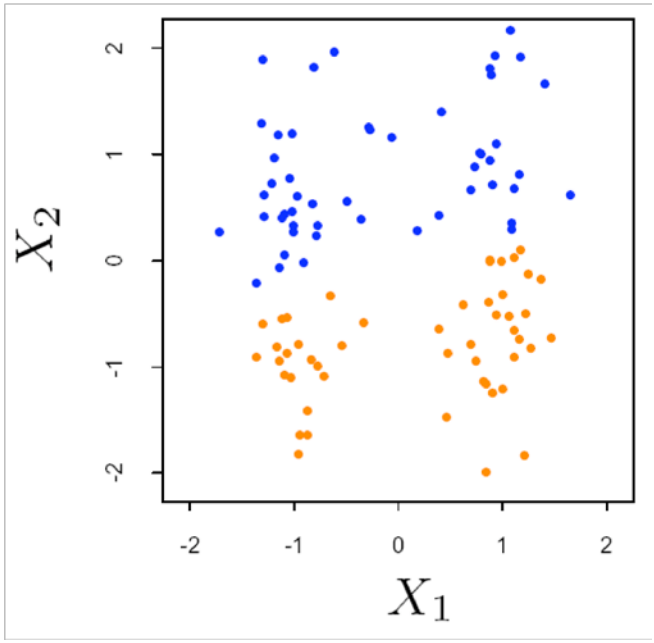
Without standardization

With standardization

Standardization not always helpful



Without standardization



With standardization

K-MEANS CLUSTERING

K-means: Idea

- K clusters each summarized by a prototype μ_k.
- Assignment of data x_i to a cluster represented by responsibilities $r_{ik} \in \{0, 1\}$ with $\sum_{k=1}^{K} r_{ik} = 1$.
- An example with 4 data points and 3 clusters.

$$(r_{ik}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Loss function $J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|_2^2$.

K-means: minimizing the loss function

- How do we minimize J w.r.t (r_{ik} , μ_k)?
- Chicken and egg problem
  - If prototypes known, can assign responsibilities.
  - If responsibilities known, can compute prototypes.
- We use an iterative procedure.

K-means: minimizing the loss function

- E-step: Fix μ_k, minimize J w.r.t. r_{ik}.
  - Assign each data point to its nearest prototype.
- M-step: Fix r_{ik}, minimize J w.r.t. μ_k.
  - Set each prototype to the mean of the points in that cluster,

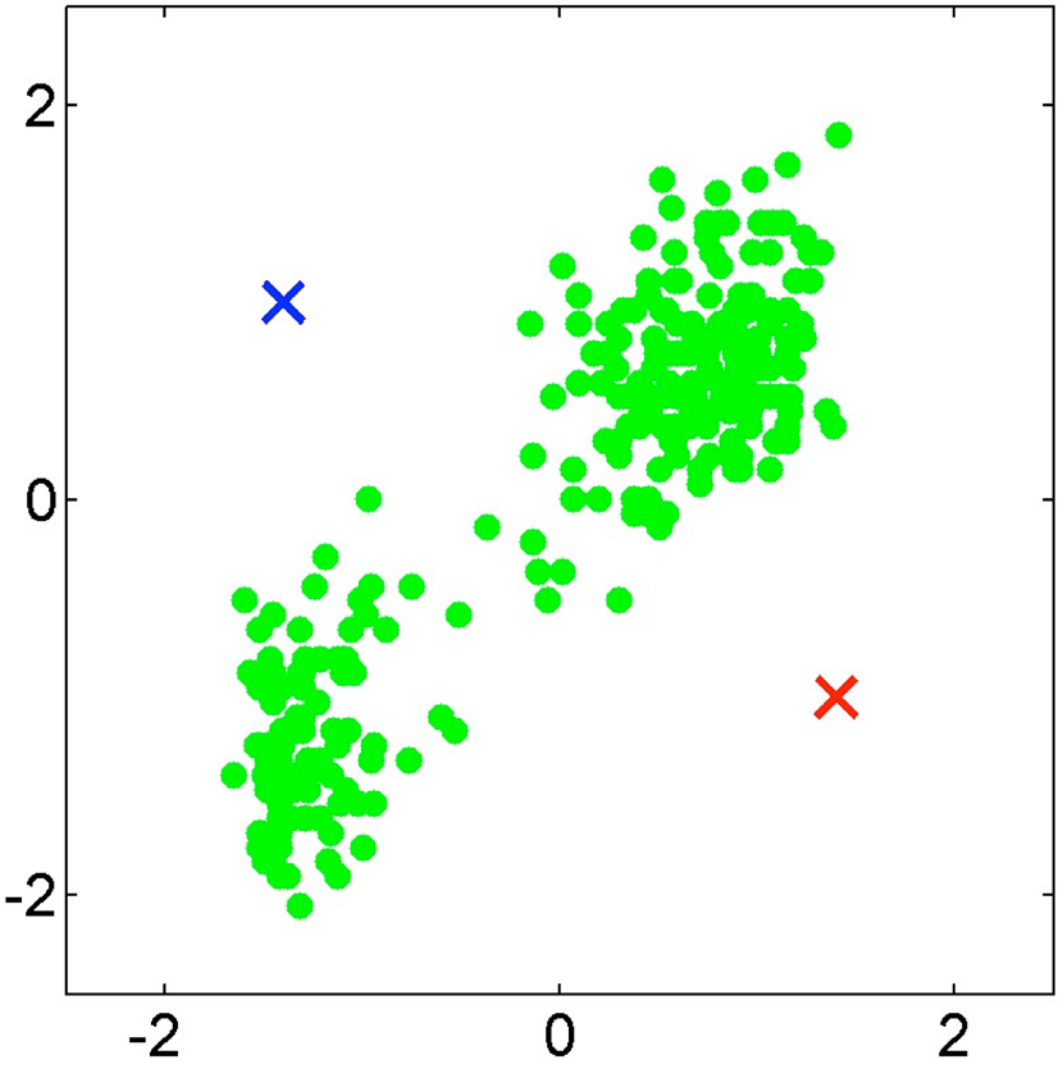
$$\text{i.e., } \mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}.$$

- This procedure is guaranteed to converge.
- Converges to a local minimum.
  - Use different initializations and pick the best solution.
  - May still be insufficient for large search spaces.
  - Other ways include a split-merge approach.

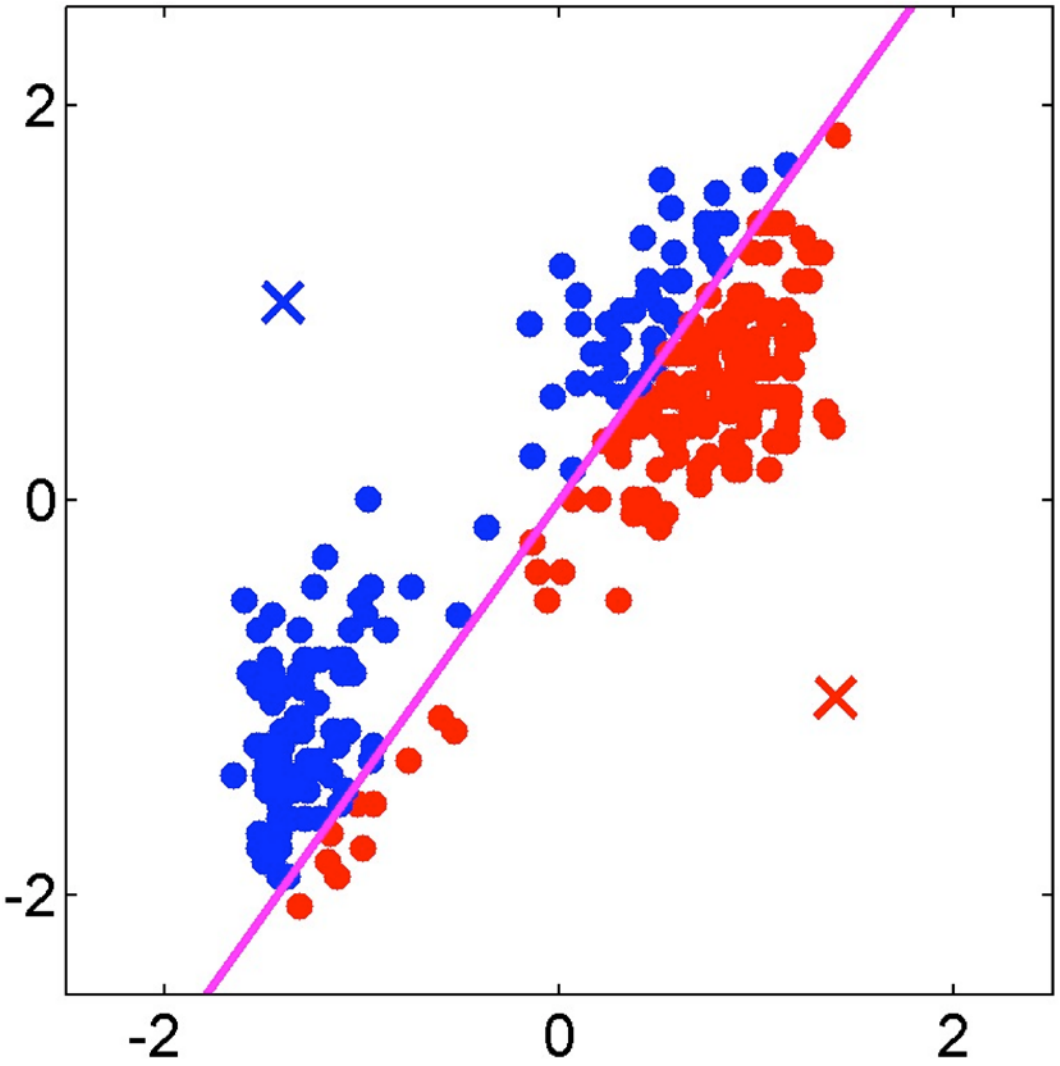
How do we initialize K-means?

- Some heuristics
  - Randomly pick K data points as prototypes.
  - Pick prototype $i + 1$ to be farthest from prototypes $\{1, \ldots, i\}$.

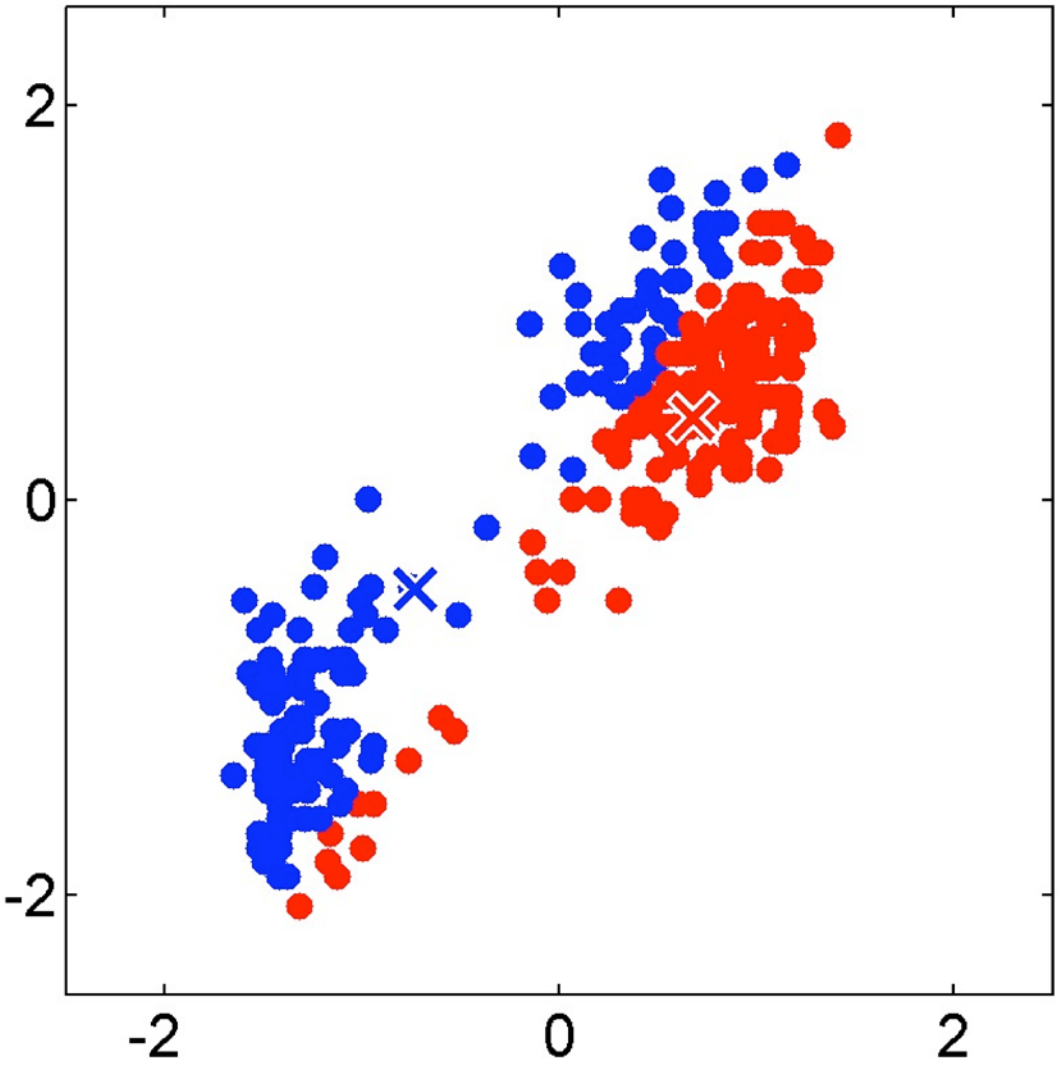
K-means Execution Example



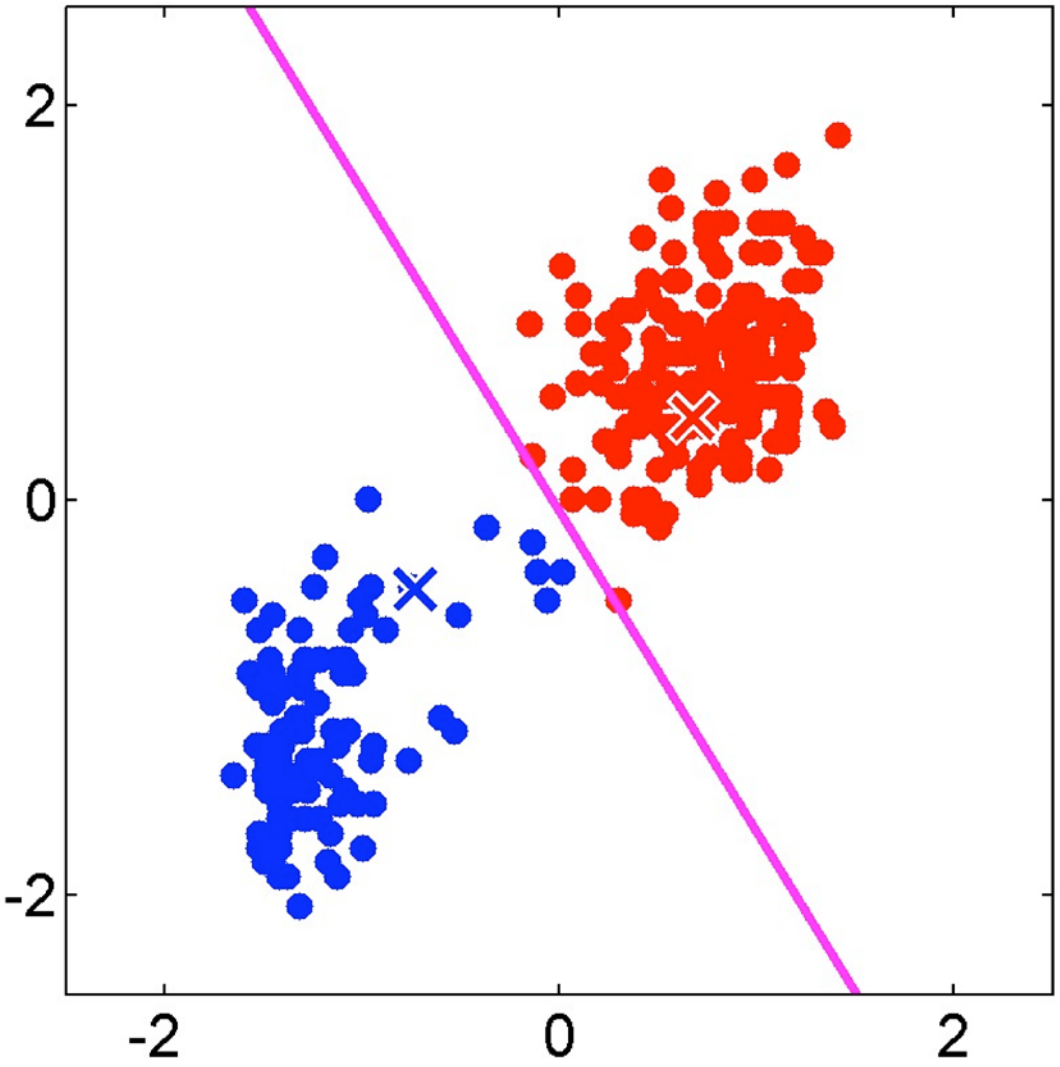
K-means Execution Example



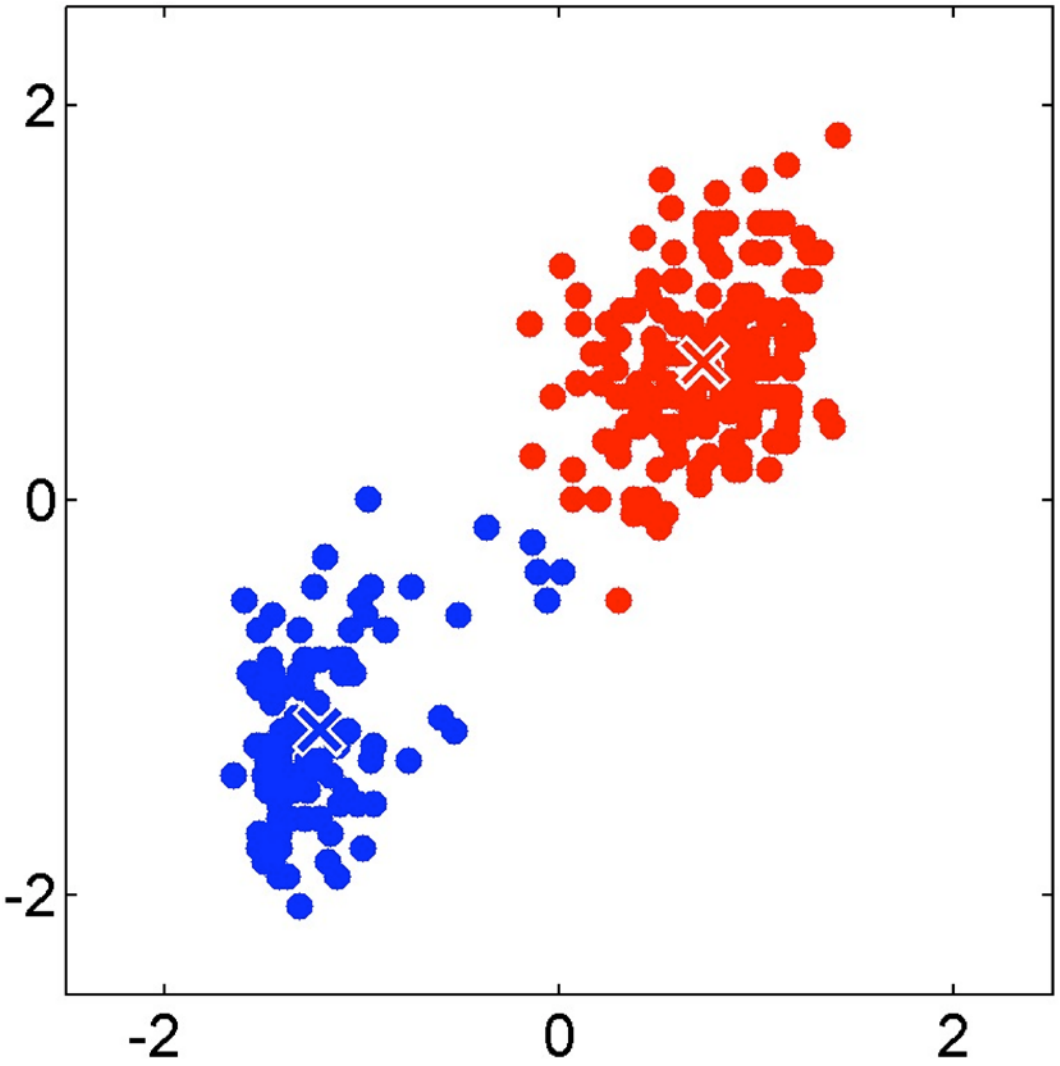
K-means Execution Example



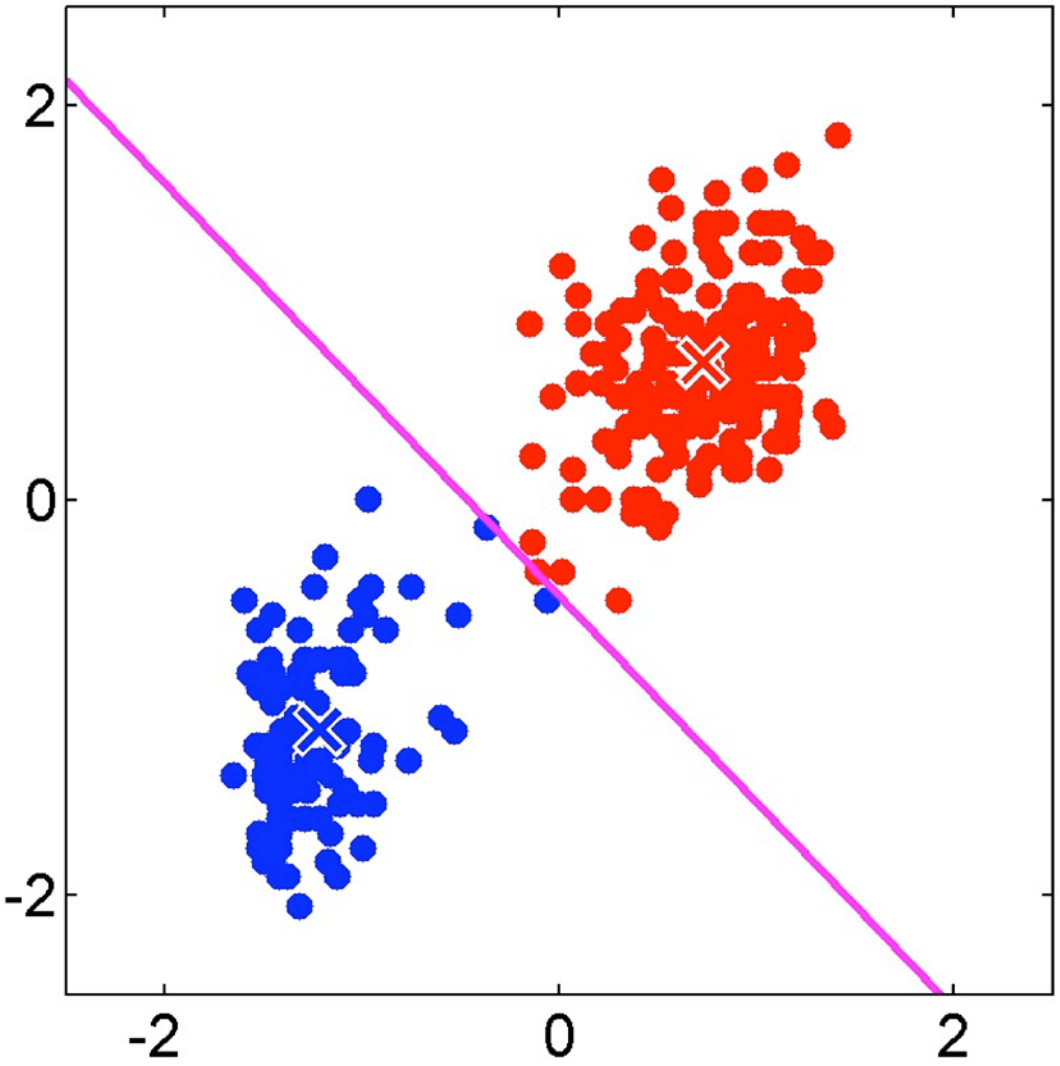
K-means Execution Example



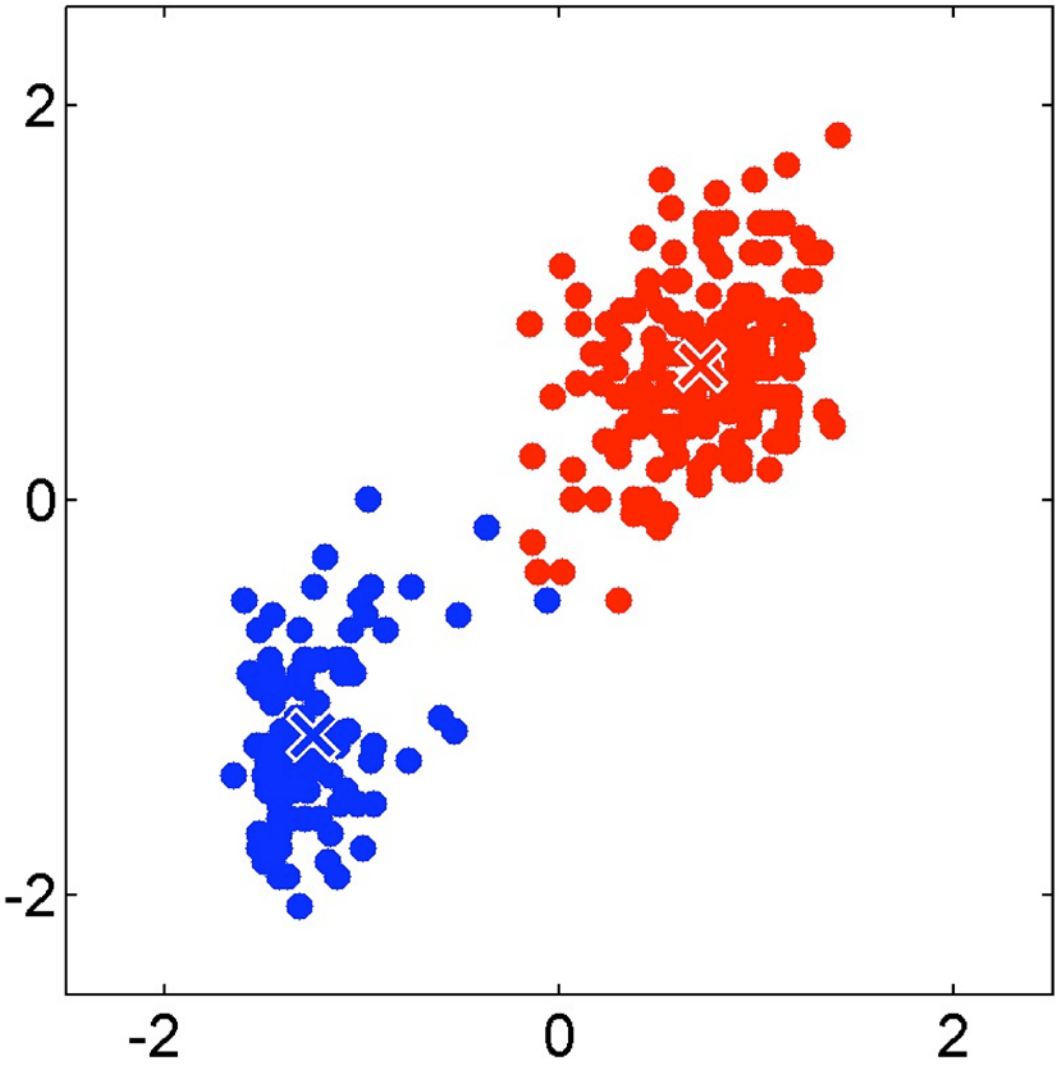
K-means Execution Example



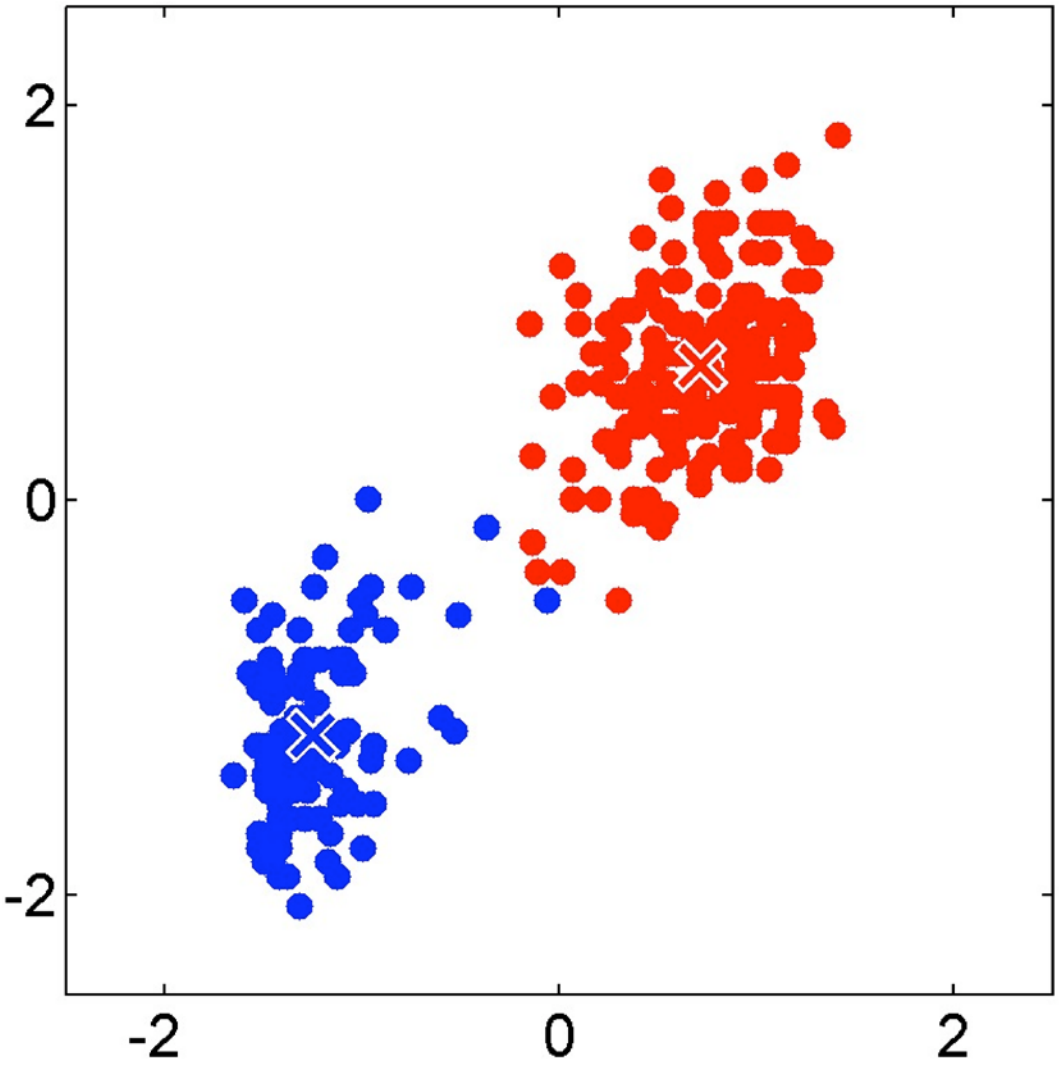
K-means Execution Example



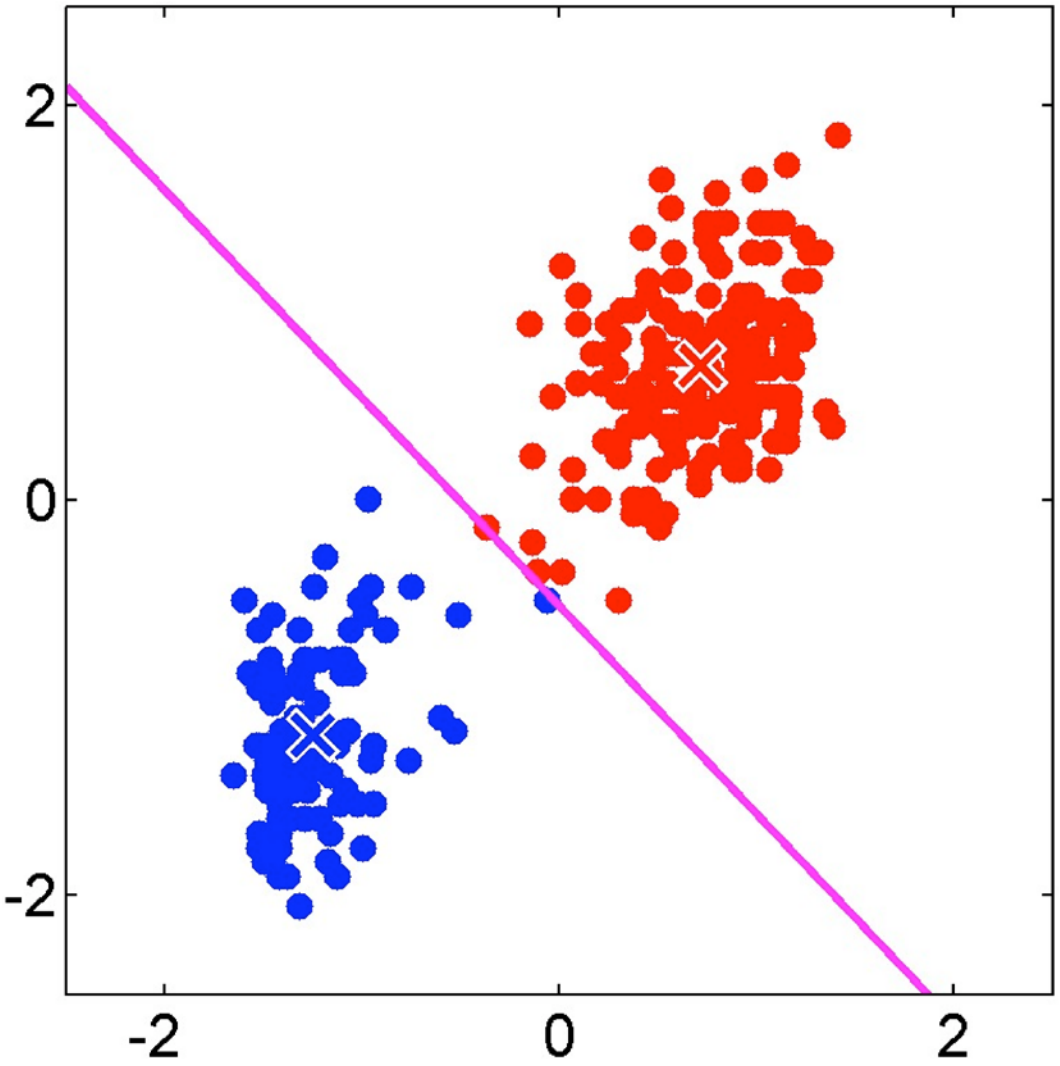
K-means Execution Example



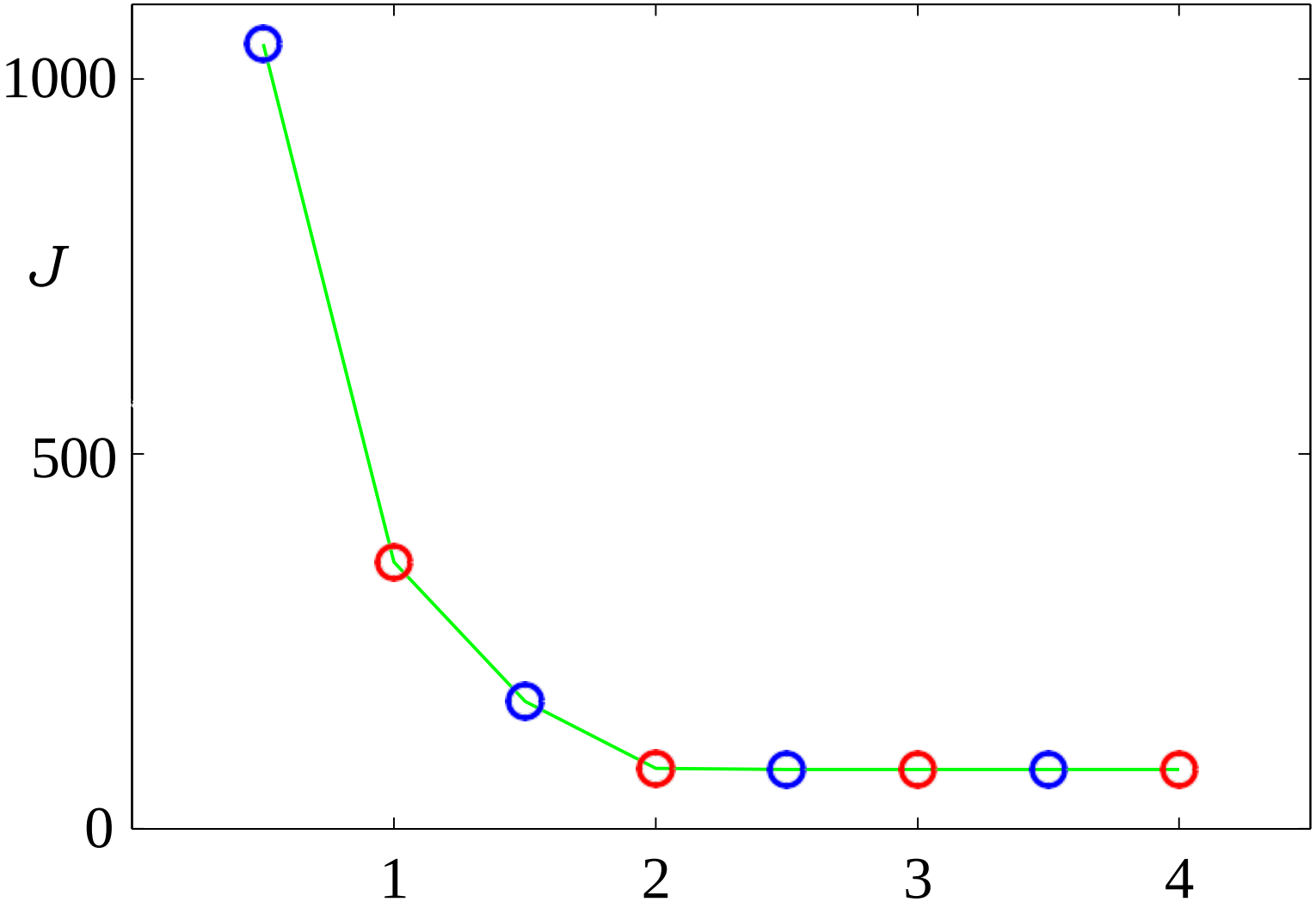
K-means Execution Example



K-means Execution Example

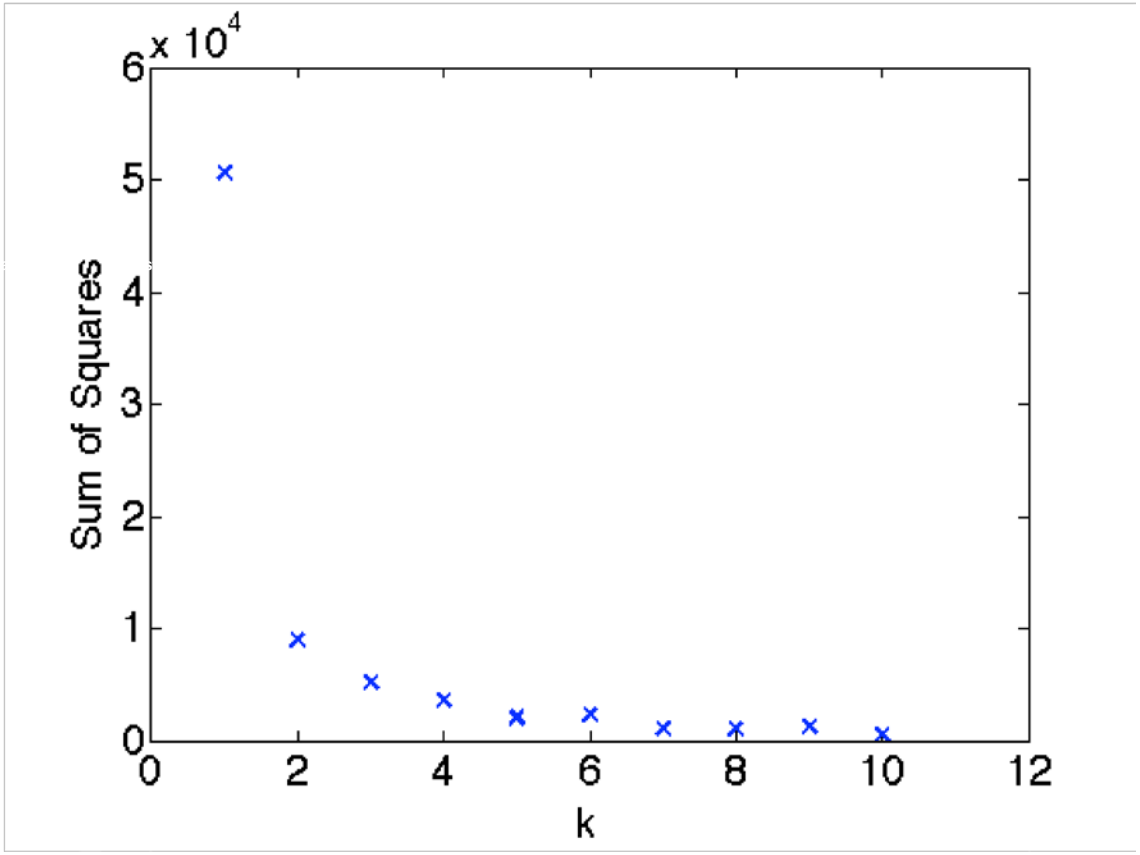


Loss function J after each iteration



How to choose K ?

- Like choosing K in kNN.
- The loss function J generally decreases with K.



How to choose K ?

- Gap statistic
- Cross-validation: Partition data into two sets. Estimate prototypes on one and use these to compute the loss function on the other.
- Stability of clusters: Measure the change in the clusters obtained by resampling or splitting the data.
- Non-parametric approach: Place a prior on K. More details in the Bayesian non-parametric lecture.

Limitations of K-means

- Hard assignments of data points to clusters can cause a small perturbation to a data point to flip it to another cluster.
  - Solution: GMM
- Assumes spherical clusters and equal probabilities for each cluster.
  - Solution: GMM
- Clusters change arbitrarily for different K.
  - Solution: Hierarchical clustering
- Sensitive to outliers.
    - Solution: Use a robust loss function.
    - Works poorly on non-convex clusters.
  - Solution: Spectral clustering.

GAUSSIAN MIXTURE MODELS

Mixture of Gaussians

- $z \in \{0,1\}^K$: be a discrete latent variable, such that $\sum_k z_k = 1$.
- z_k selects the cluster (mixture component) from which the data point is generated.
- There are K Gaussian distributions:

$$\mathcal{N}(x|\mu_1, \Sigma_1)$$

...

$$\mathcal{N}(x|\mu_K, \Sigma_K)$$

Mixture of Gaussians

- Given a data point x:

$$P(x) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x|\mu_k, \Sigma_k)$$

- Where:

$$\pi_k = P(z_k = 1)$$

Generative Procedure

- Select z from probability distr. π_k.
- Hence: $P(z) = \prod_{k=1}^{K} \pi_k^{z_k}$.
- Given z, generate x according to the conditional distr.:

$$P(x|z_k = 1) = \mathcal{N}(x|\mu_k, \Sigma_k)$$

- Hence:

$$P(x|z) = \prod_{k=1}^{K} \left(\mathcal{N}(x|\mu_k, \Sigma_k)\right)^{z_k}$$

Generative Procedure

- Joint distr.:

$$P(x, z) = p(z)p(x|z)$$

$$= \prod_{k=1}^{K} \left(\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)\right)^{z_k}$$

- Marginal:

$$p(x) = \sum_{z} p(x, z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

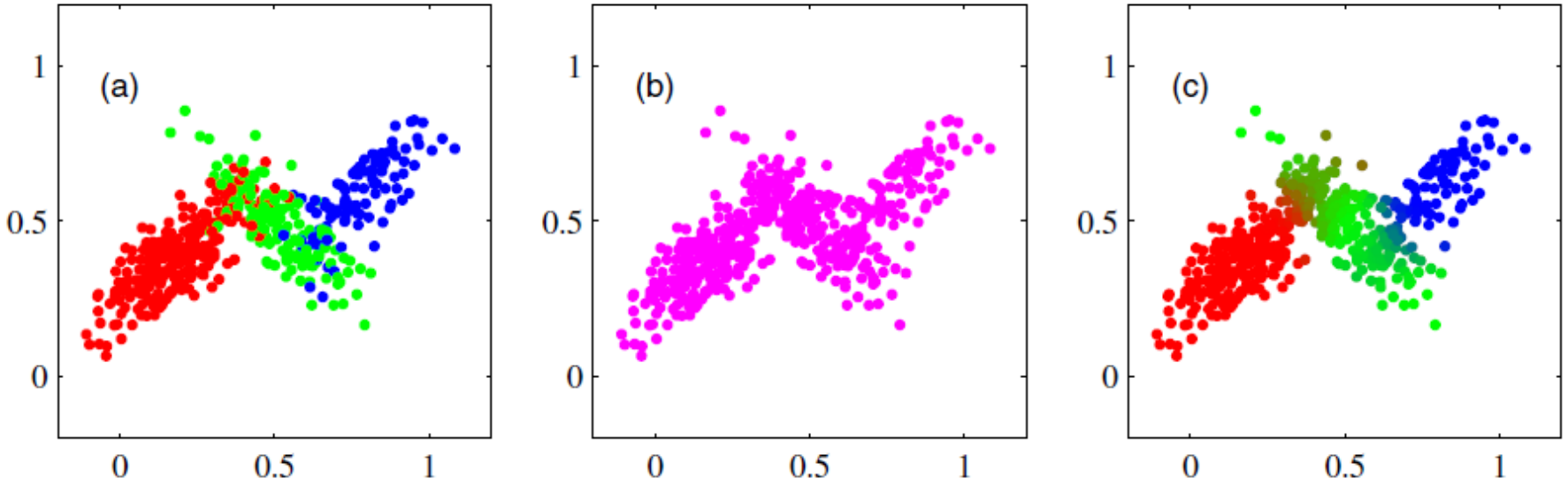
Posterior distribution

- $z_k = 1$ given x:

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

Example



Max-likelihood

- Let $D = \{x_1, \ldots, x_N\}$
- Likelihood function:

$$P(D|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

- Log likelihood:

$$\ln\big(P(D|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\big) = \sum_{n=1}^{N} \ln(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k))$$

- Maximize log-likelihood w.r.t. $\boldsymbol{\pi}$, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

KKT conditions

- Differentiating w.r.t. μ_k:

$$0 = -\sum_{n=1}^{N} \underbrace{\frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}}_{\gamma(z_{nk})} \boldsymbol{\Sigma}_k(\mathbf{x}_n - \boldsymbol{\mu}_k)$$

- Multiplying by Σ_k^{-1}:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

- Where:

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

KKT conditions

- Similarly, differentiating w.r.t. Σ_k:

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

- Lagrangian w.r.t. π_k:

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

KKT conditions

- Minimizing:

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \lambda$$

- Multiplying with π_k and adding over k: $\lambda = -N$.
- Hence: $$\pi_k = \frac{N_k}{N}$$
- Where: $$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

(EM) Algorithm

- Initialize μ_k, Σ_k and π_k.
- E-step:

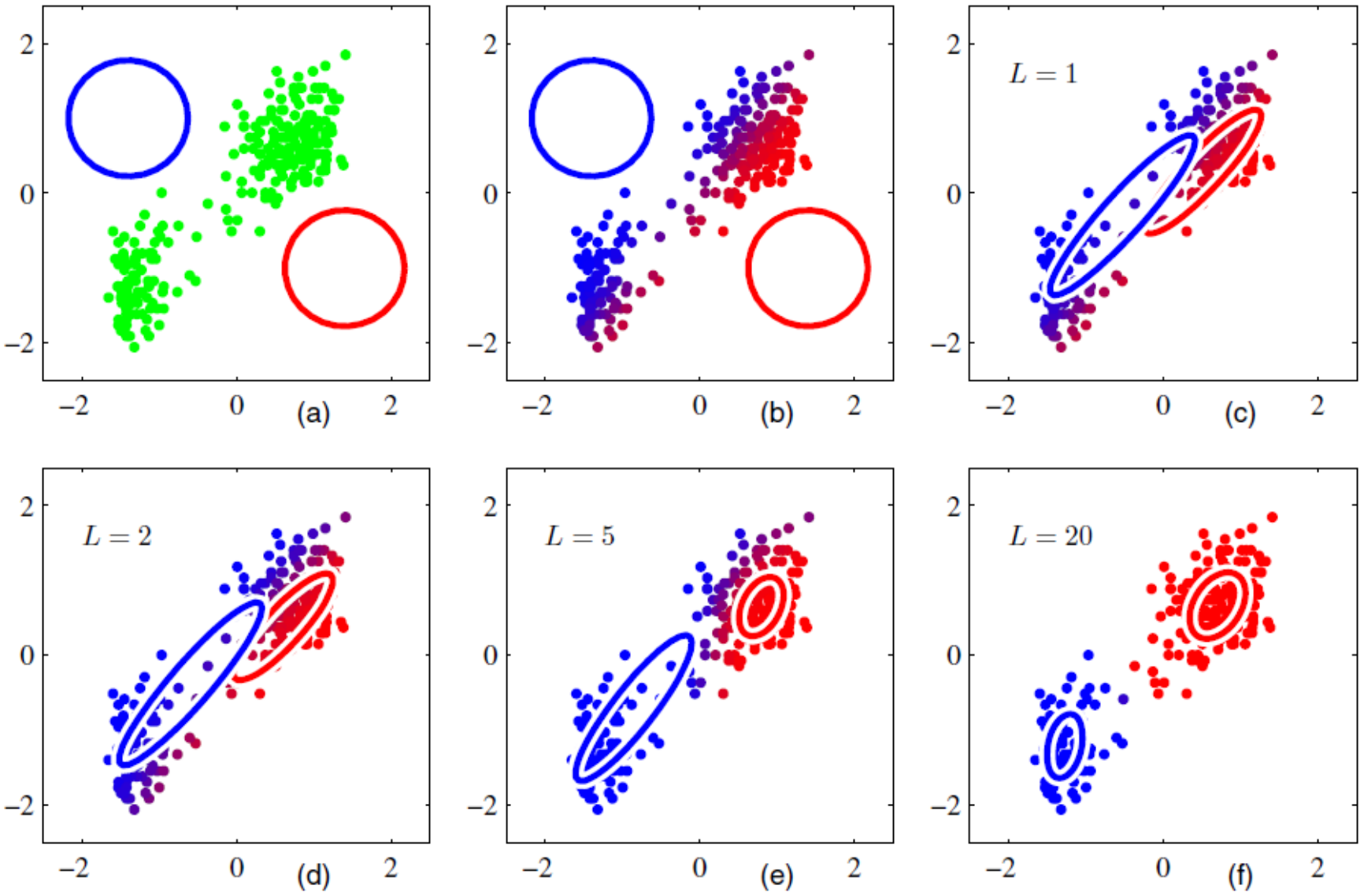
$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

- M-step:

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

- Repeat above two steps till $\ln\left(P(D|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})\right)$ converges.

Example



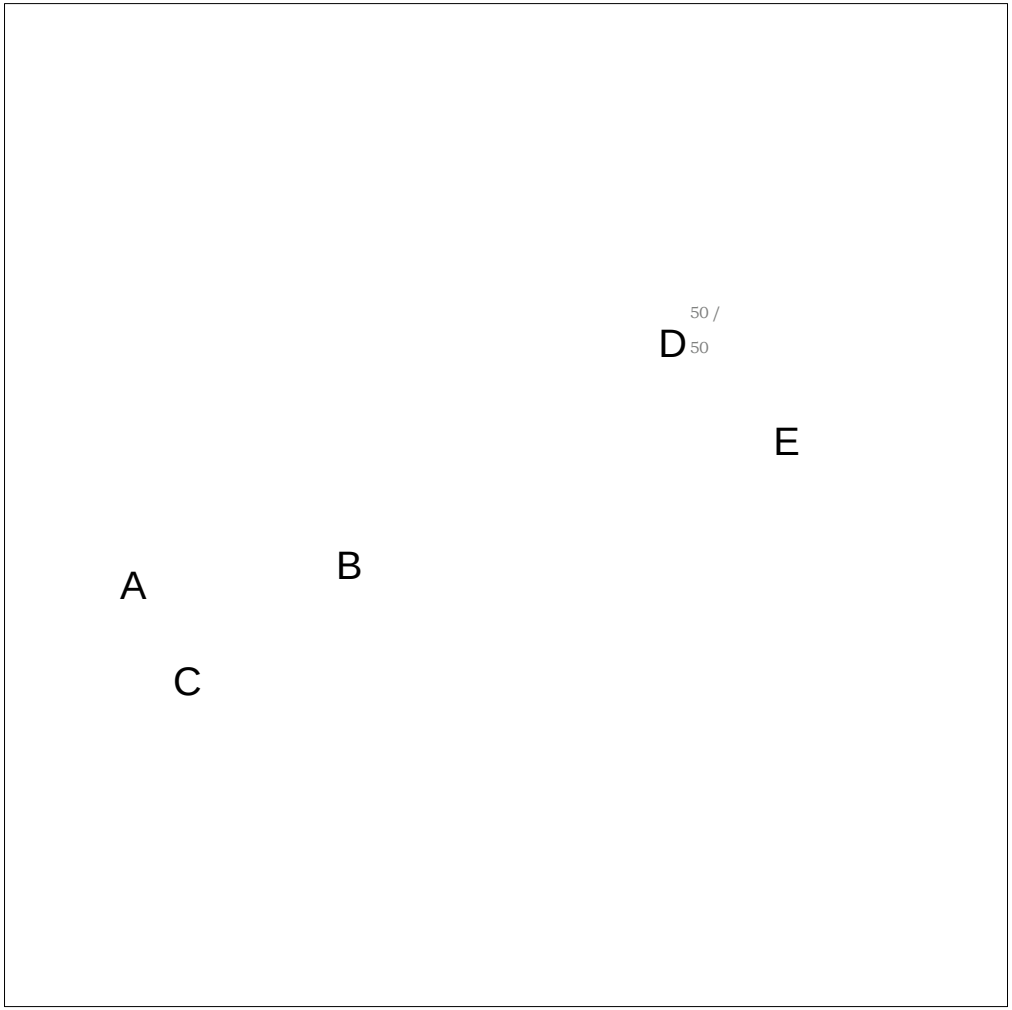
HIERARCHICAL CLUSTERING

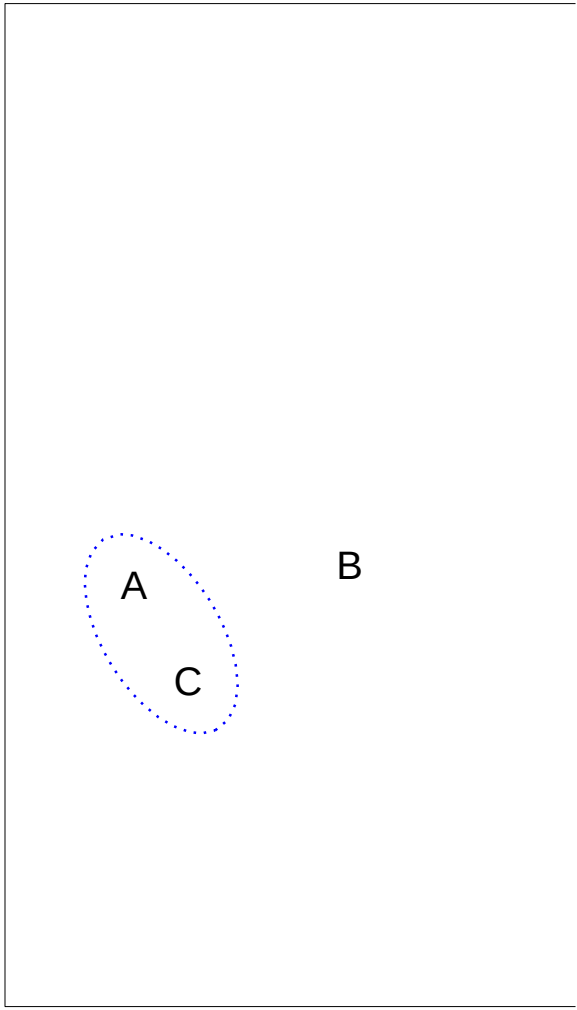
Hierarchical Clustering

- K-means clustering requires us to pre-specify the number of clusters K. This can be a disadvantage (later we discuss strategies for choosing K)
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of K.
- In this section, we describe *bottom-up* or *agglomerative* clustering. This is the most common type of hierarchical clustering, and refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

Hierarchical Clustering: the idea

Builds a hierarchy in a "bottom-up" fashion...

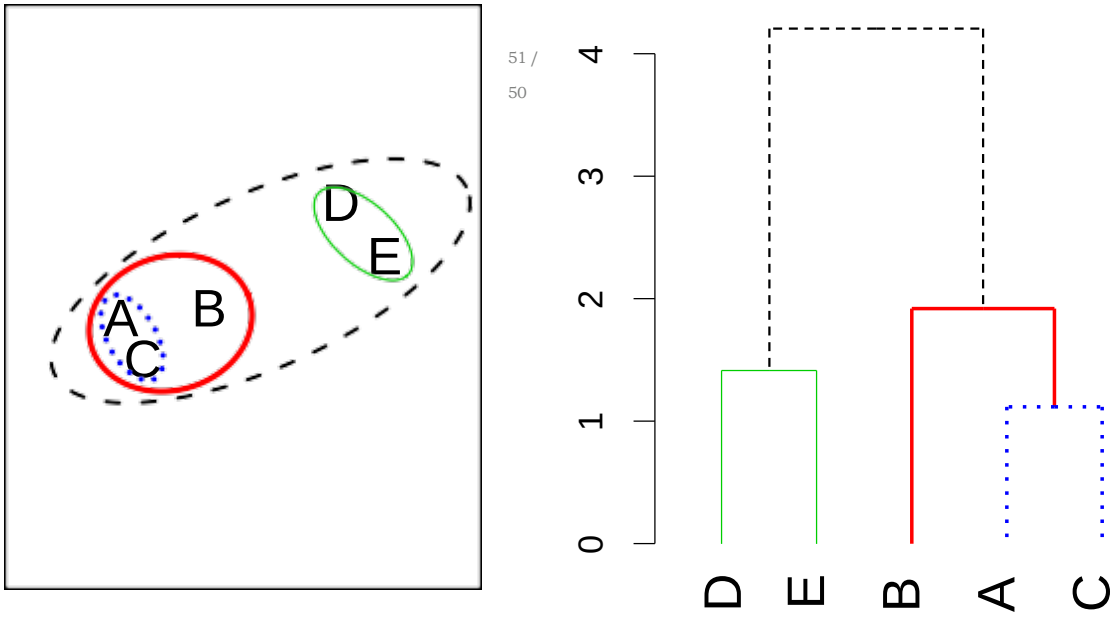




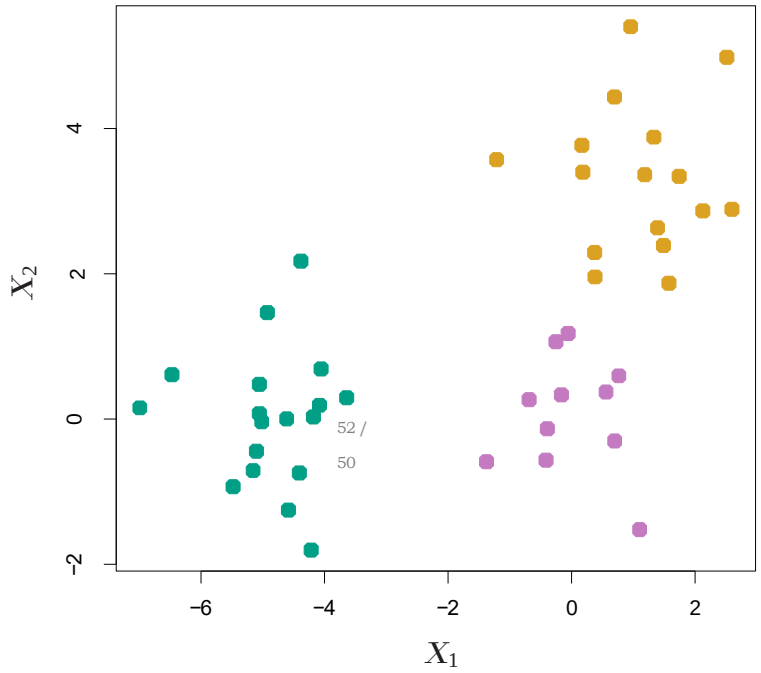
Hierarchical Clustering Algorithm

The approach in words:

- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

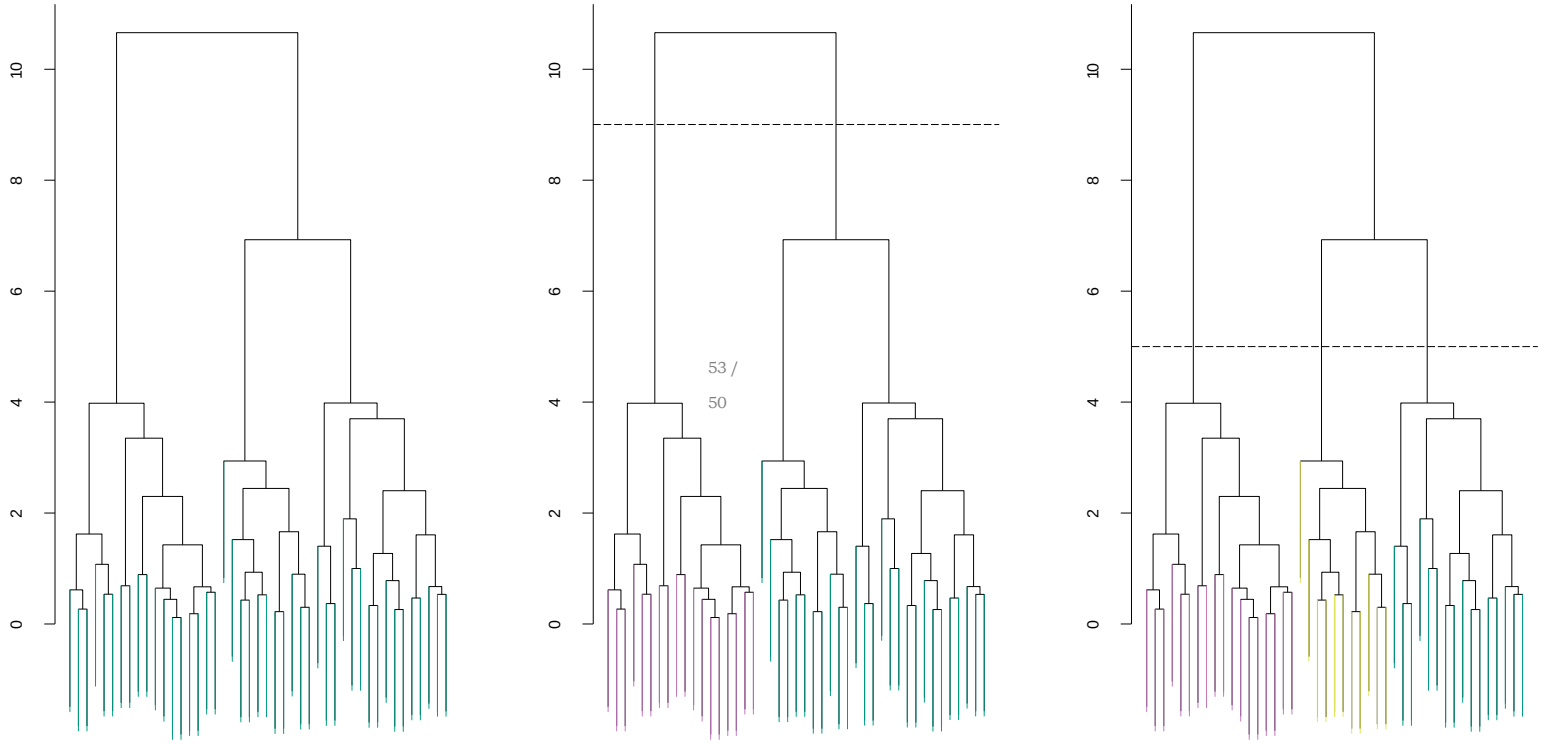


An Example



45 observations generated in 2-dimensional space. In reality there are three distinct classes, shown in separate colors. However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

Application of hierarchical clustering



Details of previous figure

- *Left:* Dendrogram obtained from hierarchically clustering the data from previous slide, with complete linkage and Euclidean distance.
- *Center:* The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in di↵erent colors.
- *Right:* The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in di↵erent colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure

Types of Linkage

Linkage	*Description*
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length *p*) and the centroid for cluster B. Cen- troid linkage can result in undesirable *inversions*.

Cluster Dissimilarity

- Dissimilarity for two disjoint groups G and H, $d(G, H)$ is computed from pairwise dissimilarities $D(i, j), i\ inG, j \in H$.
  - Single linkage: tends to yield extended clusters.

$$D_{SL}(G, H) = min_{i \in G, j \in H} D(i, j)$$

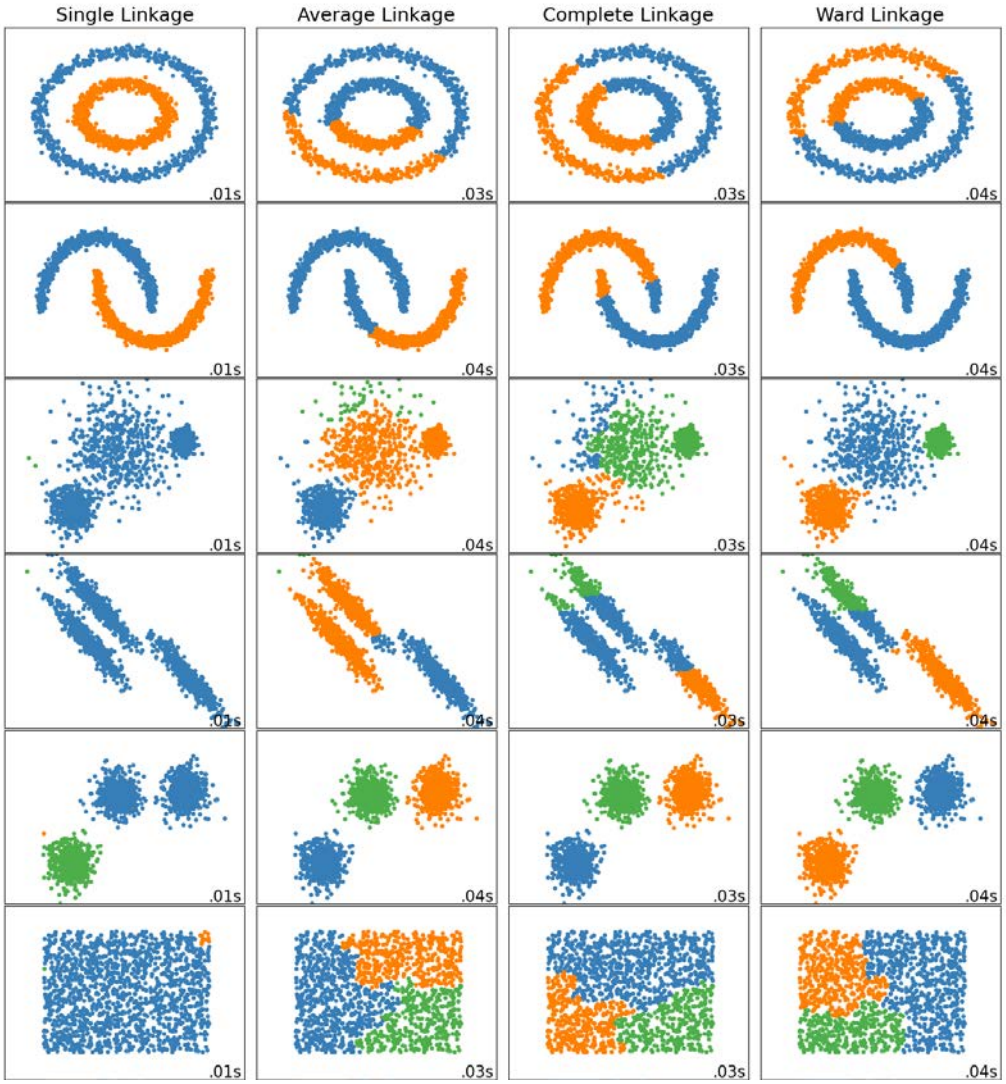
  - Complete linkage: tends to yield round clusters.

$$D_{CL}(G, H) = max_{i \in G, j \in H} D(i, j)$$

  - Group average: tradeoff between the two. Not invariant under monotone increasing transform.

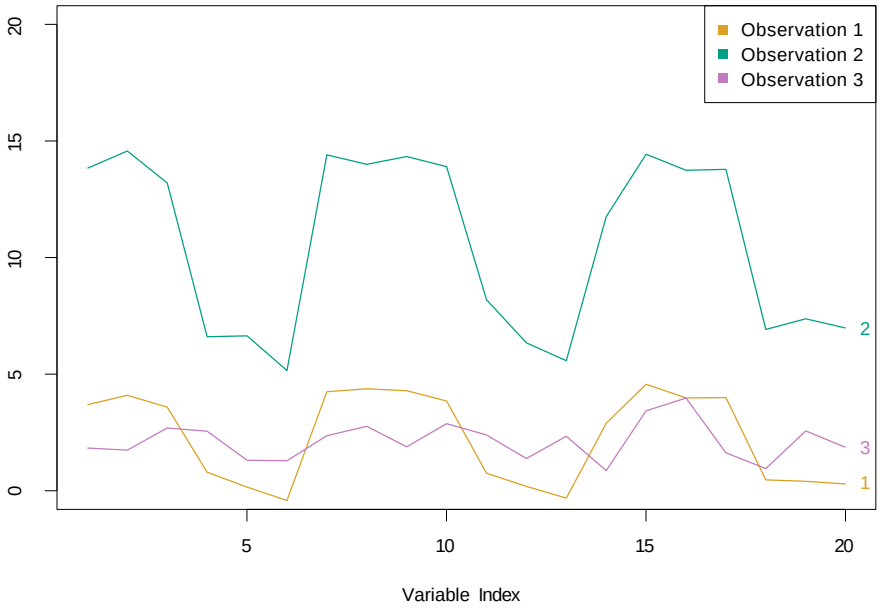
$$D_{GA}(G, H) = \frac{1}{n_G n_H} \sum_{i \in G, j \in H} D(i, j)$$

Comparison


Single Linkage
Average Linkage
Complete Linkage
Ward Linkage
.01s
.03s
.03s
.04s
.01s
.04s
.03s
.04s
.01s
.04s
.03s
.04s
.01s
.04s
.03s
.04s
.01s
.04s
.03s
.04s
.01s
.03s
.03s
.04s

Choice of Dissimilarity Measure

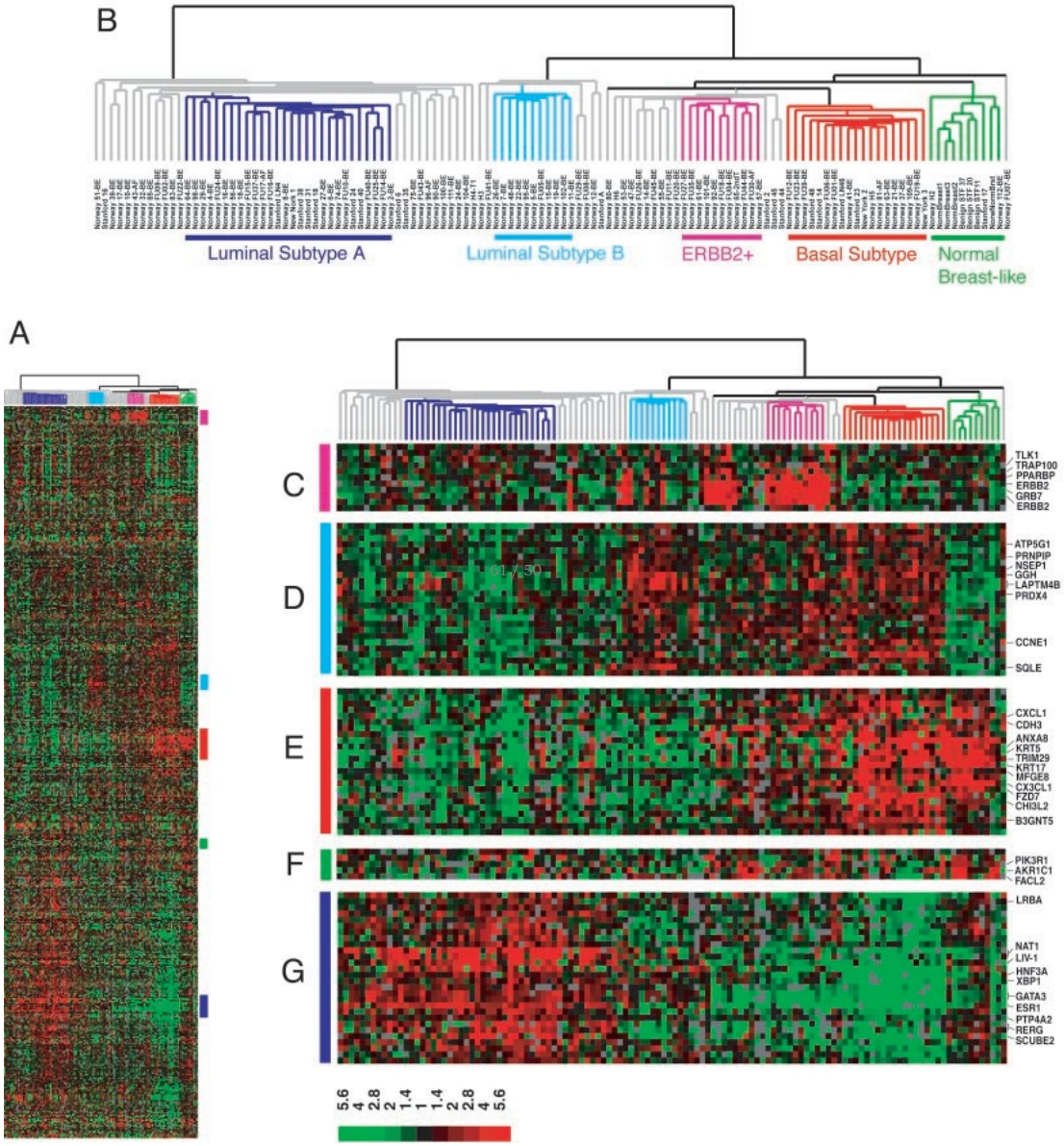
- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.



Example: breast cancer microarray study

- "Repeated observation of breast tumor subtypes in independent gene expression data sets;" Sorlie at el, PNAS 2003
- Gene expression measurements for about ← 8000 genes, for each of 88 breast cancer patients.

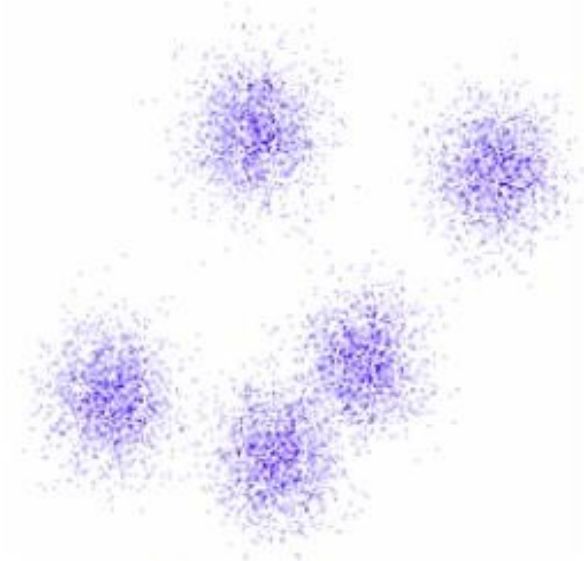
- Average linkage, correlation metric
- Clustered samples using 500 *intrinsic genes:* each woman was measured before and after chemotherapy. Intrinsic genes have smallest within/between variation.


B
Luminal Subtype A
Luminal Subtype B
ERBB2+
Basal Subtype
Normal Breast-like
A
C
TLK1
TRAP100
PPARBP
ERBB2
GRB7
ERBB2
D
ATP5G1
PRNPIP
NSEP1
GGH
LAPTM4B
PRDX4
CCNE1
SQLE
E
CXCL1
CDH3
ANXA8
KRT5
TRIM29
KRT17
MFGE8
CXCL1
FZD7
CHI3L2
B3GNT5
F
PIK3R1
AKR1C1
FACL2
G
LRBA
NAT1
LIV-1
HNF3A
XBP1
GATA3
ESR1
PTP4A2
RERG
SCUBE2
5.6
4
2.8
2
1.4
1
1.4
2
2.8
4
5.6

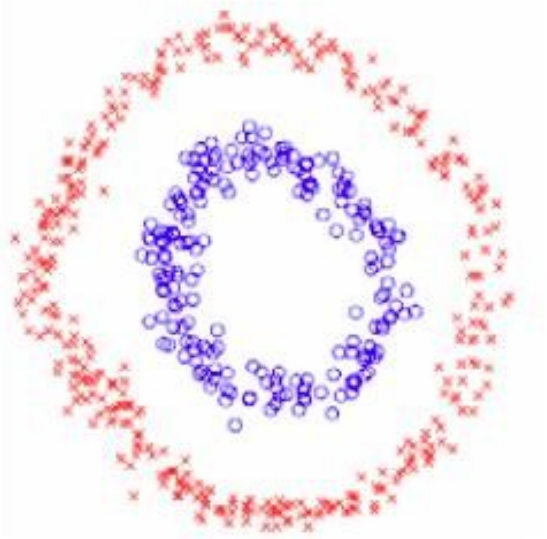
SPECTRAL CLUSTERING

Data Clustering

- Two different criteria
  - Compactness, e.g., k-means, mixture models
  - Connectivity, e.g., spectral clustering



Compactness



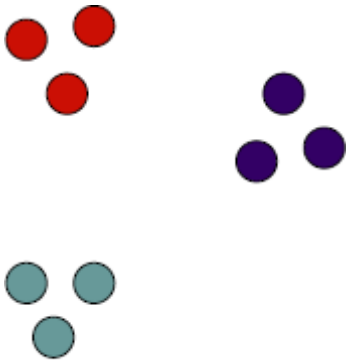
Connectivity

Graph Clustering

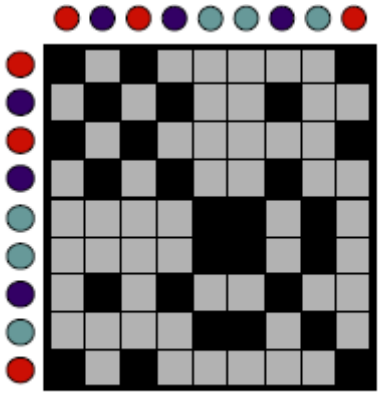
Goal: Given data points X_1, …, X_n and similarities $w(X_i,X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

Similarity Graph: G(V,E,W)

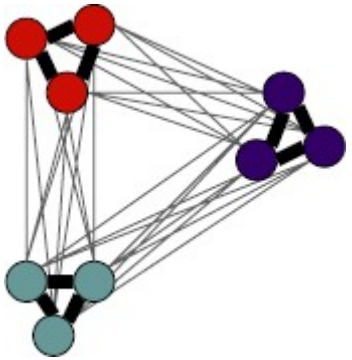
V – Vertices (Data points)
E – Edge if similarity > 0
W - Edge weights (similarities)



Data



Similarities



Similarity graph

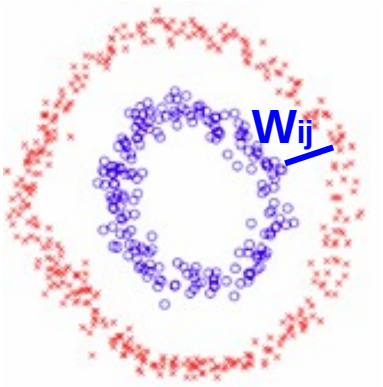
Partition the graph so that edges within a group have large weights and edges across groups have small weights.

Similarity graph construction

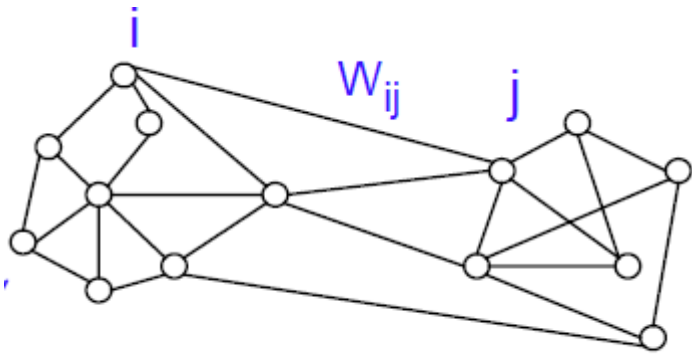
Similarity Graphs: Model local neighborhood relations between data points

E.g. Gaussian kernel similarity function

$$W_{ij} = e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$ → Controls size of neighborhood



Data clustering

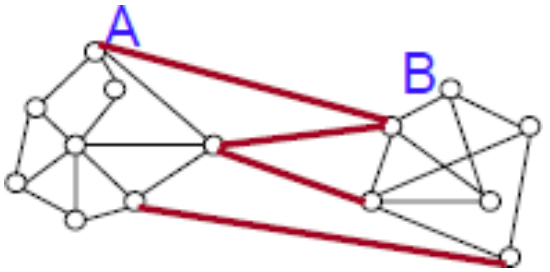


$G = \{V,E\}$

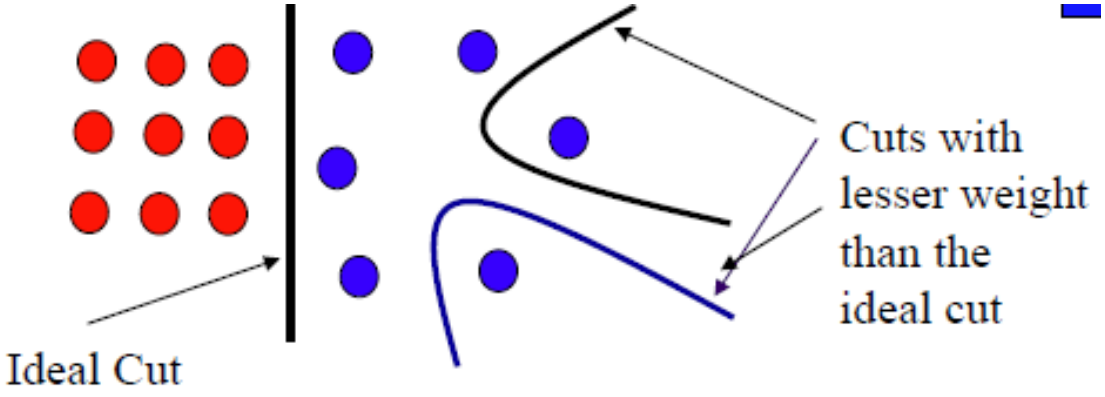
Partitioning a graph into two clusters

Min-cut: Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum.

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



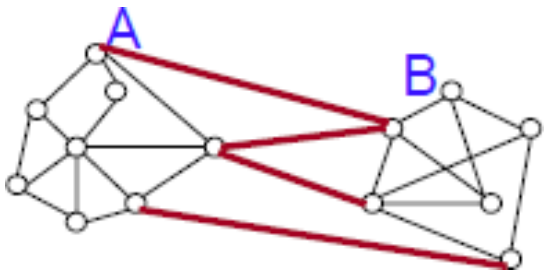
- Easy to solve O(VE) algorithm
- Not satisfactory partition – often isolates vertices



Partitioning a graph into two clusters

Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum & size of A and B are very similar.

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



Normalized cut:

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

$$\text{vol}(A) = \sum_{i \in A} d_i$$

But NP-hard to solve!!

Spectral clustering is a relaxation of these.

Normalized Cut and Graph Laplacian

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

Let $f = [f_1\ f_2\ \ldots\ f_n]^T$ with $f_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\text{vol}(B)} & \text{if } i \in B \end{cases}$

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \sum_{ij} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 = \sum_{i\in A, j\in B} w_{ij}\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)^2$$

$$\mathbf{f}^T\mathbf{D}\mathbf{f} = \sum_j d_i\mathbf{f}_i^2 = \sum_{i\in A}\frac{d_i}{\text{vol}(A)^2} + \sum_{j\in B}\frac{d_i}{\text{vol}(B)^2} = \frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}$$

$$\text{Ncut}(A, B) = \frac{\mathbf{f}^T\mathbf{L}\mathbf{f}}{\mathbf{f}^T\mathbf{D}\mathbf{f}}$$

Normalized Cut and Graph Laplacian

$$\min \text{Ncut}(A, B) = \min \frac{\mathbf{f}^T \mathbf{L}\mathbf{f}}{\mathbf{f}^T \mathbf{D}\mathbf{f}}$$

where $\mathbf{f} = [f_1\ f_2\ \ldots\ f_n]^T$ with $f_i = \begin{cases} \frac{1}{\text{vol}(A)} & \text{if } i \in A \\ -\frac{1}{\text{vol}(B)} & \text{if } i \in B \end{cases}$

Relaxation: $\min \frac{\mathbf{f}^T \mathbf{L}\mathbf{f}}{\mathbf{f}^T \mathbf{D}\mathbf{f}}$ s.t. $\mathbf{f}^T\mathbf{D}\mathbf{1} = 0$

Solution: f – second eigenvector of generalized eval problem

$$\mathbf{L}\mathbf{f} = \lambda \mathbf{D}\mathbf{f}$$

Obtain cluster assignments by thresholding f at 0

Approximation of Normalized cut

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

Let *f* be the eigenvector corresponding to the second smallest eval of the generalized eval problem.

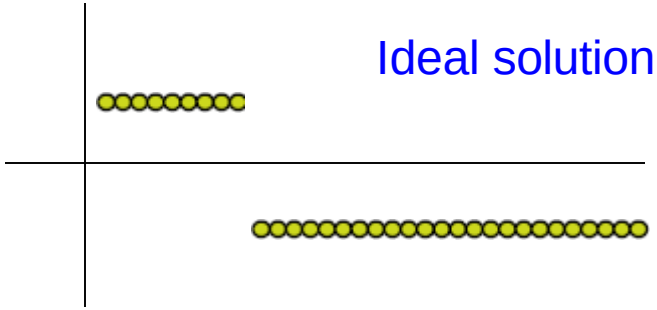
$$\mathbf{Lf} = \lambda \mathbf{Df}$$

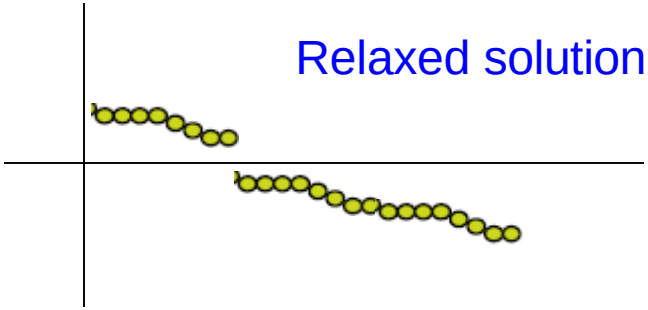
Equivalent to eigenvector corresponding to the second smallest eval of the normalized Laplacian $L' = D^{-1}L = I - D^{-1}W$

Recover binary partition as follows:

$i \in A$ *if* $f_i \geq 0$

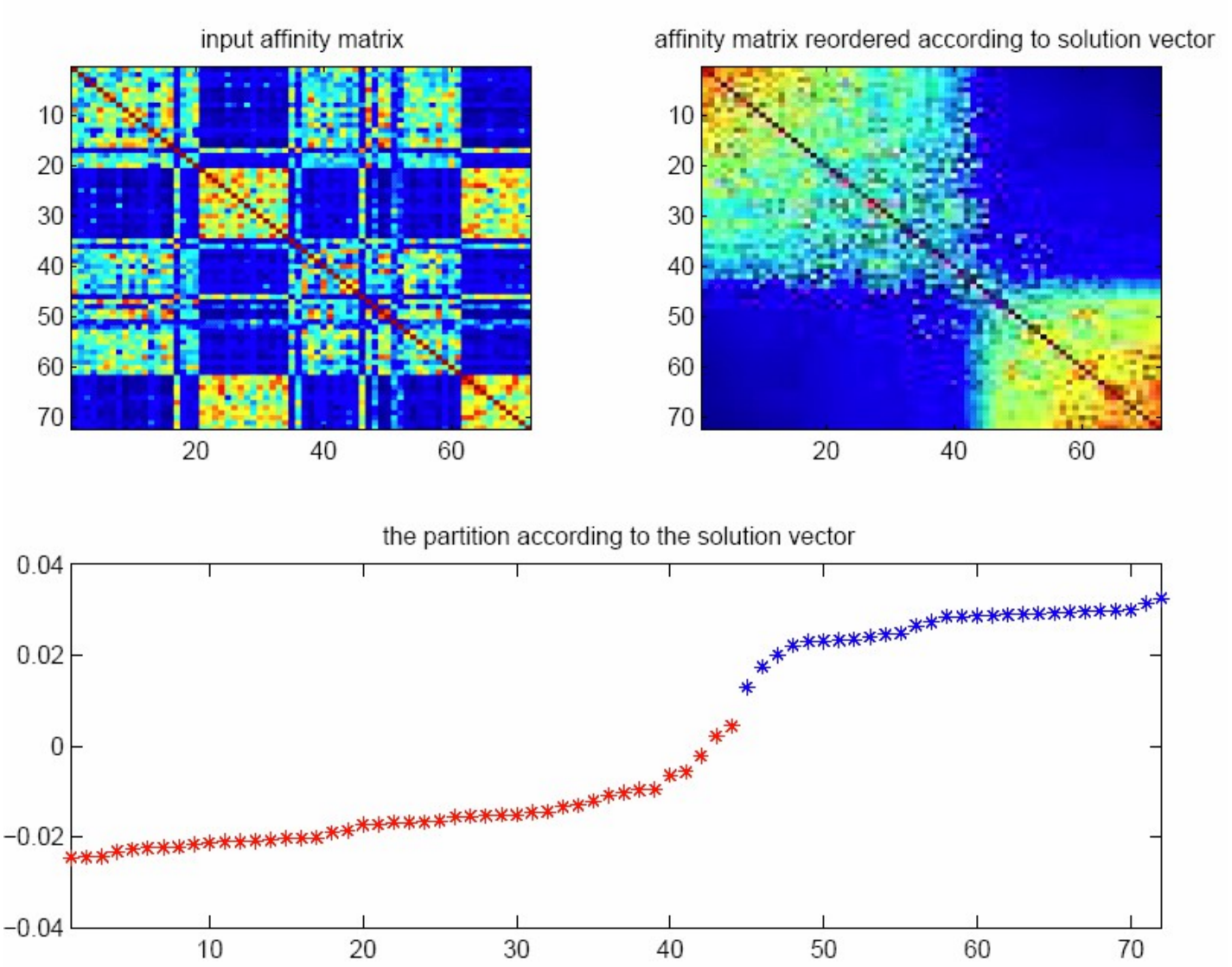
$i \in B$ *if* $f_i < 0$





Example

Xing et al 2001



How to partition a graph into k clusters?

Spectral Clustering Algorithm

Input: Similarity matrix W, number k of clusters to construct

- Build similarity graph
- Compute the first k eigenvectors $v_1, \ldots, v_k$ of the matrix

$$\begin{cases} L & \text{for unnormalized spectral clustering} \\ L' & \text{for normalized spectral clustering} \end{cases}$$

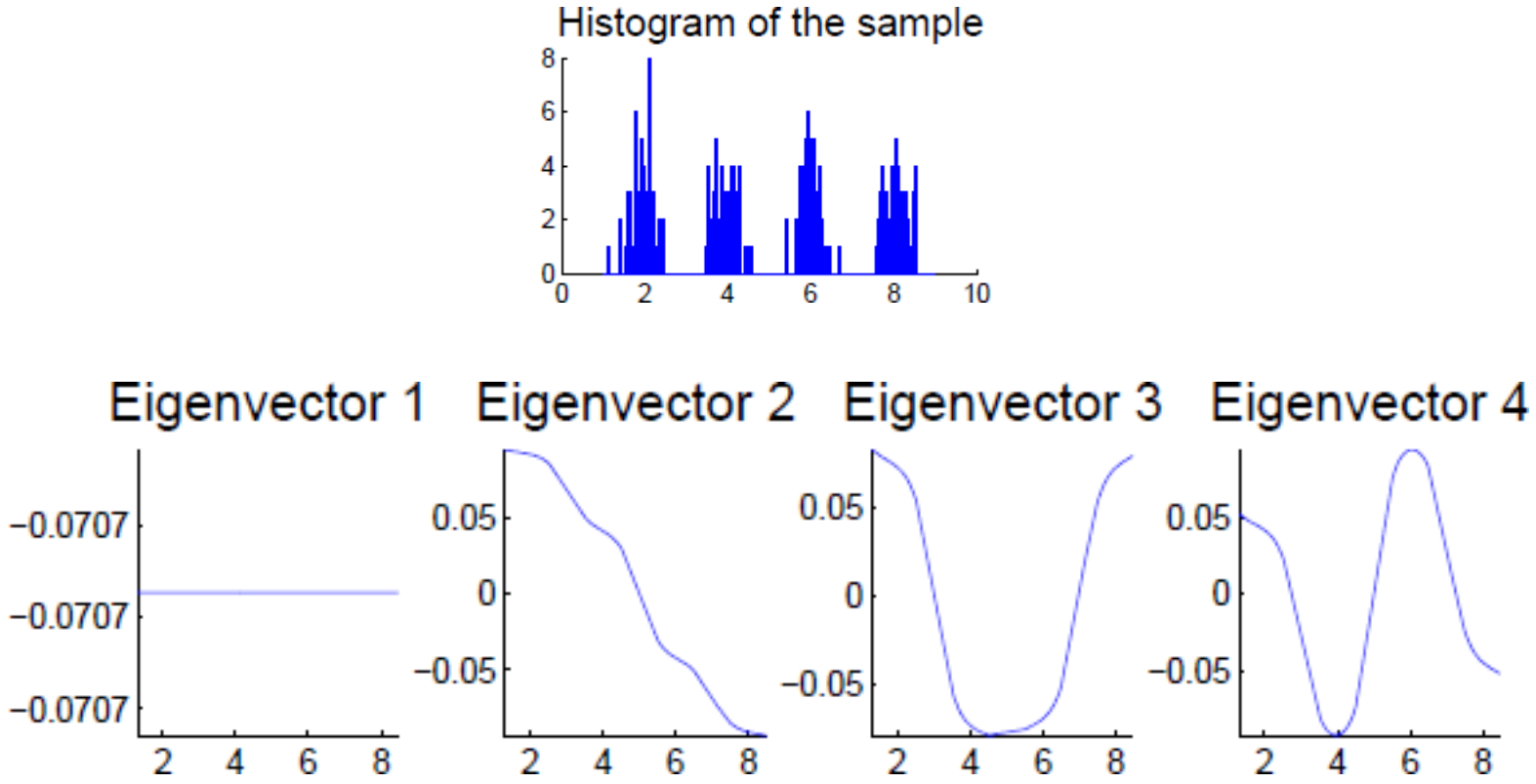
- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$

	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Z_n	v_{n1}	v_{n2}	v_{n3}

Dimensionality Reduction
n x n → n x k

- Cluster the points Z_i with the k-means algorithm in $\mathbb{R}^k$.

Eigenvectors of Graph Laplacian

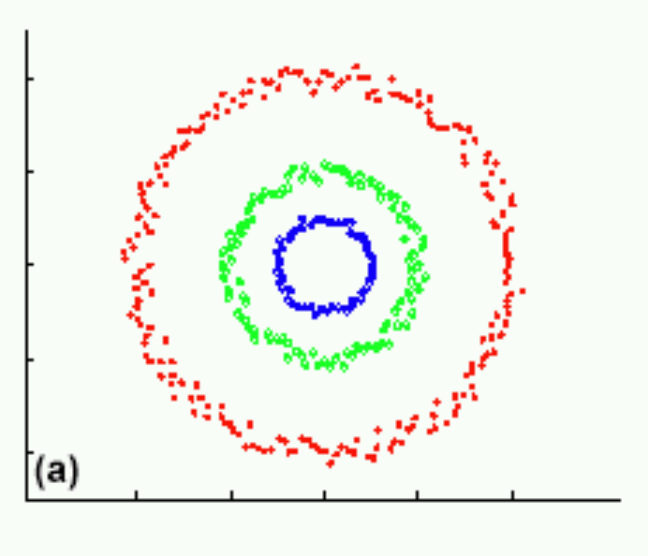


- 1st Eigenvector is the all ones vector ***1*** (if graph is connected)
- 2nd Eigenvector thresholded at 0 separates first two clusters from last two
- k-means clustering of the 4 eigenvectors identifies all clusters

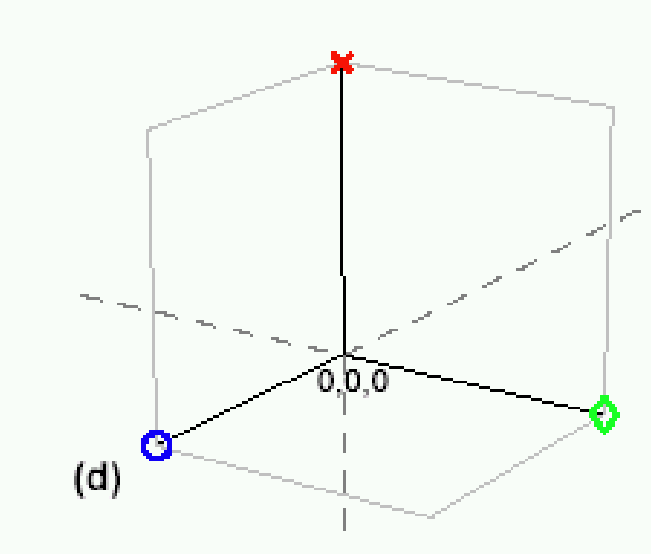
Why does it work?

Data are projected into a lower-dimensional space (the spectral/eigenvector domain) where they are easily separable, say using k-means.

Original data



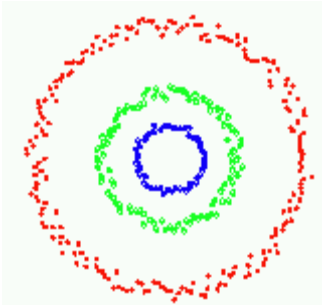
Projected data



Graph has 3 connected components – first three eigenvectors are constant (all ones) on each component.

Understanding Spectral Clustering

- If graph is connected, first Laplacian evec is constant (all 1s)
- If graph is disconnected (k connected components), Laplacian is block diagonal and first k Laplacian evecs are:



OR

$$L = \begin{bmatrix} L_1 & & 0 \\ & L_2 & \\ 0 & & L_3 \end{bmatrix}$$

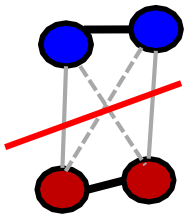
$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

First three eigenvectors

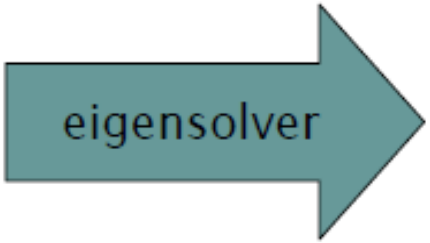
Understanding Spectral Clustering

- Is all hope lost if clusters don't correspond to connected components of graph? No!
- If clusters are connected loosely (small off-block diagonal enteries), then 1st Laplacian even is all 1s, but second evec gets first cut (min normalized cut)

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$



1	1	.2	0
1	1	0	.1
.2	0	1	1
0	.1	1	1



.50
.50
.50
.50

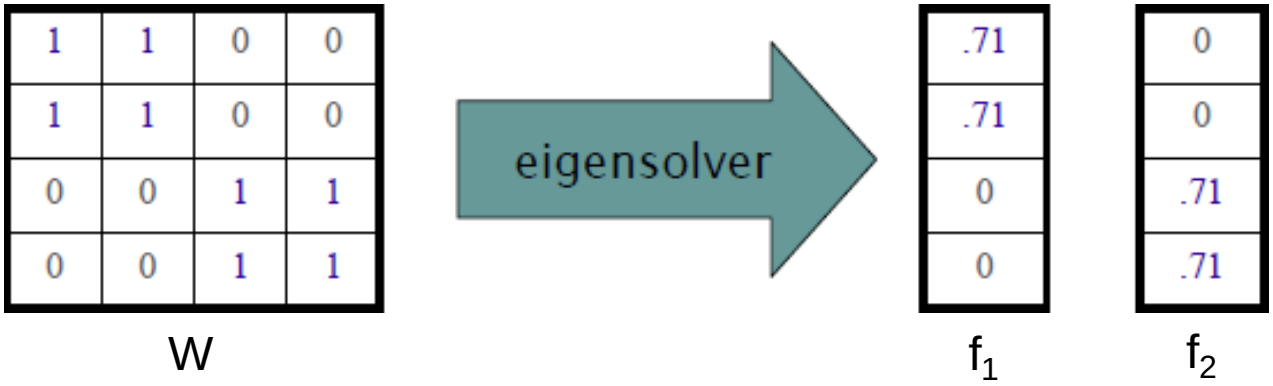
.47
.52
-.47
-.52

1st evec is constant since graph is connected

Sign of 2nd evec indicates blocks

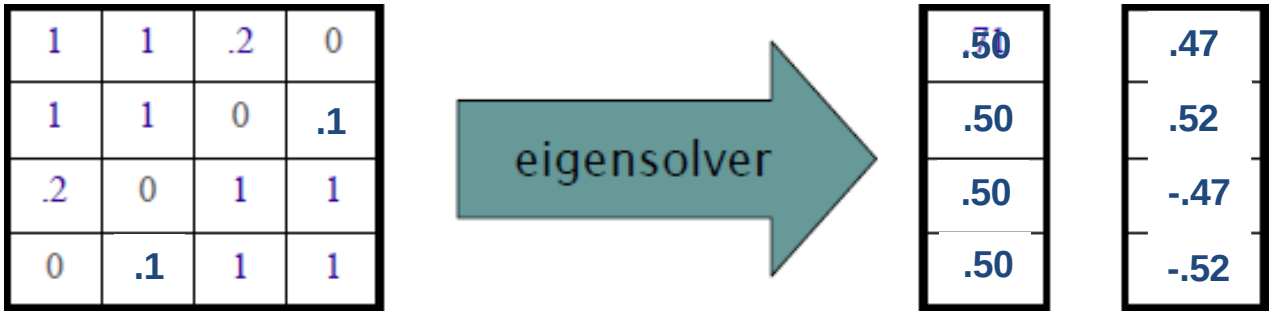
Why does it work?

Block weight matrix (disconnected graph) results in block eigenvectors:



Normalized to have unit norm

Slight perturbation does not change span of eigenvectors significantly:



1st evec is constant since graph is connected

Sign of 2nd evec indicates blocks

Why does it work?

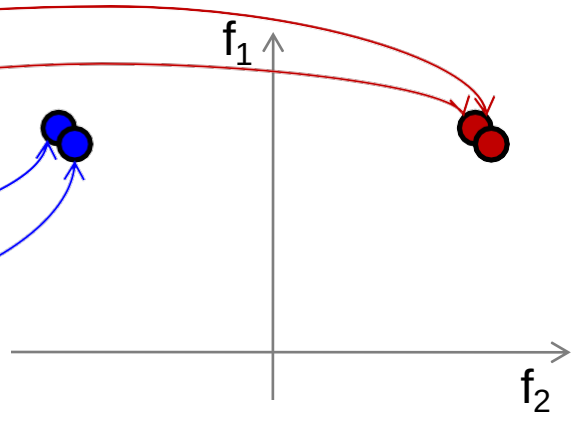
Can put data points into blocks using eigenvectors:

1	1	.2	0
1	1	0	.1
.2	0	1	1
0	.1	1	1

W

.50	.47
.50	.52
.50	-.47
.50	-.52

f_1 f_2



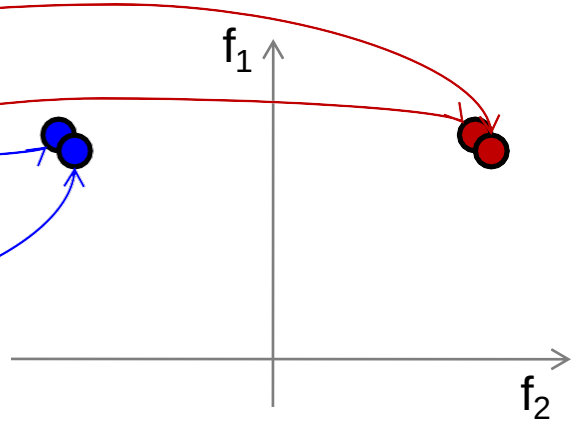
Embedding is same regardless of data ordering:

1	.2	1	0
.2	0	1	1
1	1	0	.1
0	1	.1	1

W

.50	.47
.50	-.47
.50	.52
.50	-.52

f_1 f_2



Understanding Spectral Clustering

- Is all hope lost if clusters don't correspond to connected components of graph? No!
- If clusters are connected loosely (small off-block diagonal enteries), then 1st Laplacian even is all 1s, but second evec gets first cut (min normalized cut)

$$\text{Ncut}(A, B) := \text{cut}(A, B)\left(\frac{1}{\text{vol}(A)} + \frac{1}{\text{vol}(B)}\right)$$

- What about more than two clusters?

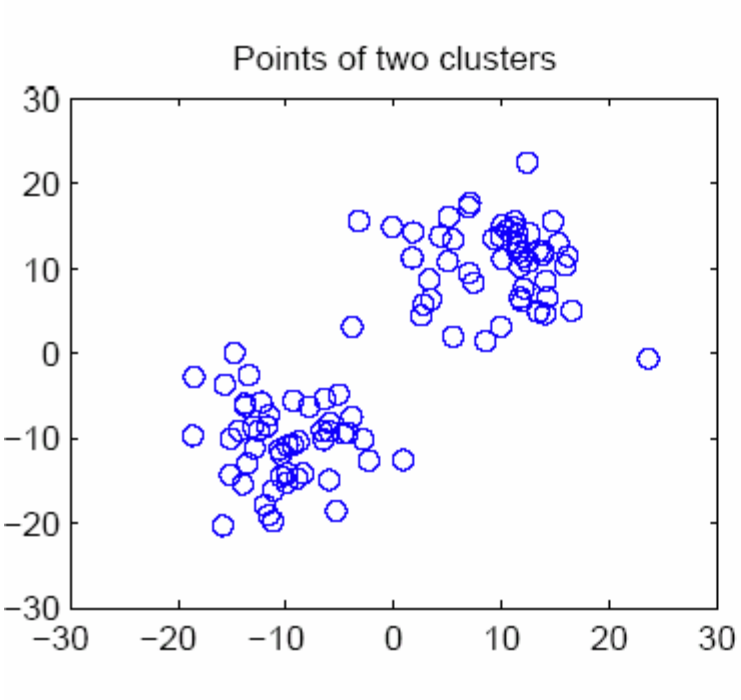
  eigenvectors f_2, …, f_{k+1} are solutions of following normalized cut:

$$\text{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\text{cut}(A_i, \overline{A}_i)}{\text{vol}(A_i)}$$

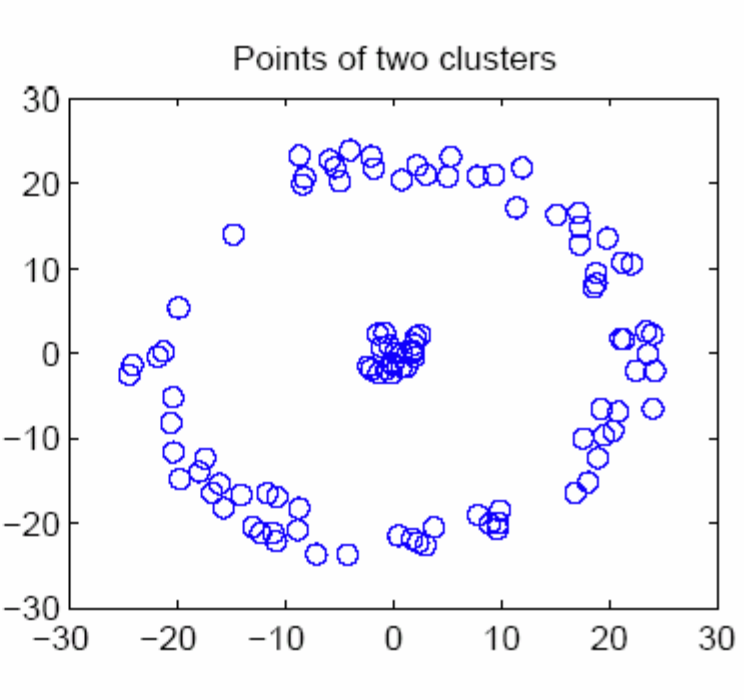
Demo: http://www.ml.uni-saarland.de/GraphDemo/DemoSpectralClustering.html

k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



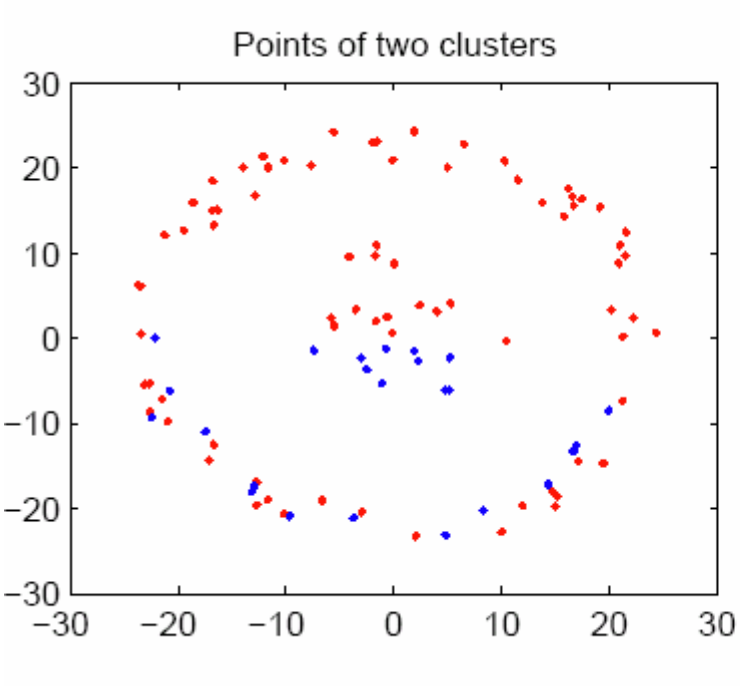
Both perform same



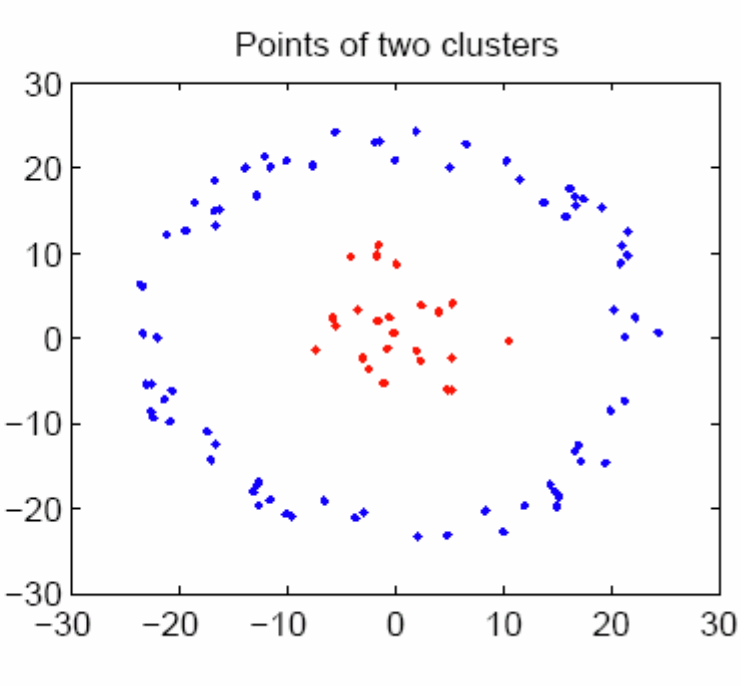
Spectral clustering is superior

k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



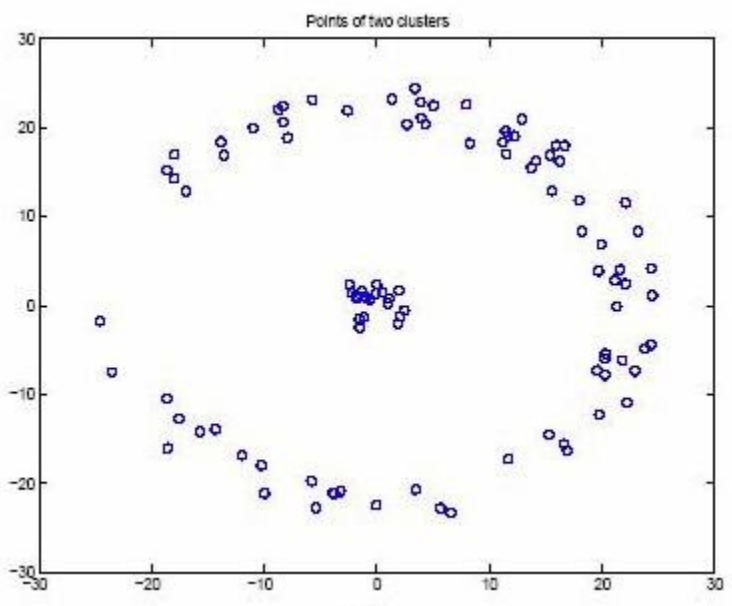
k-means output



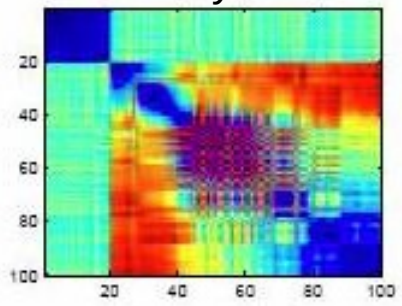
Spectral clustering output

k-means vs Spectral clustering

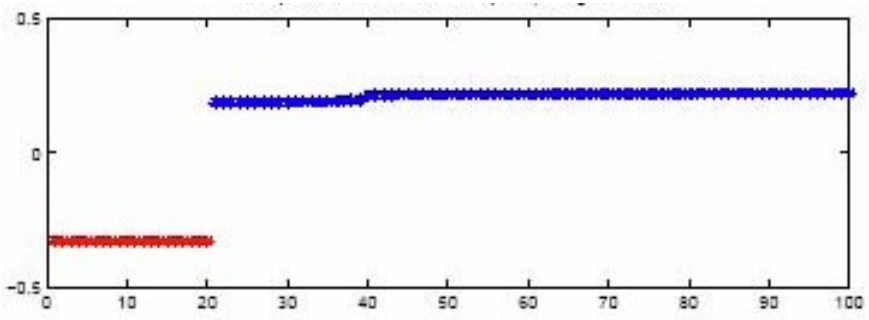
Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



Similarity matrix

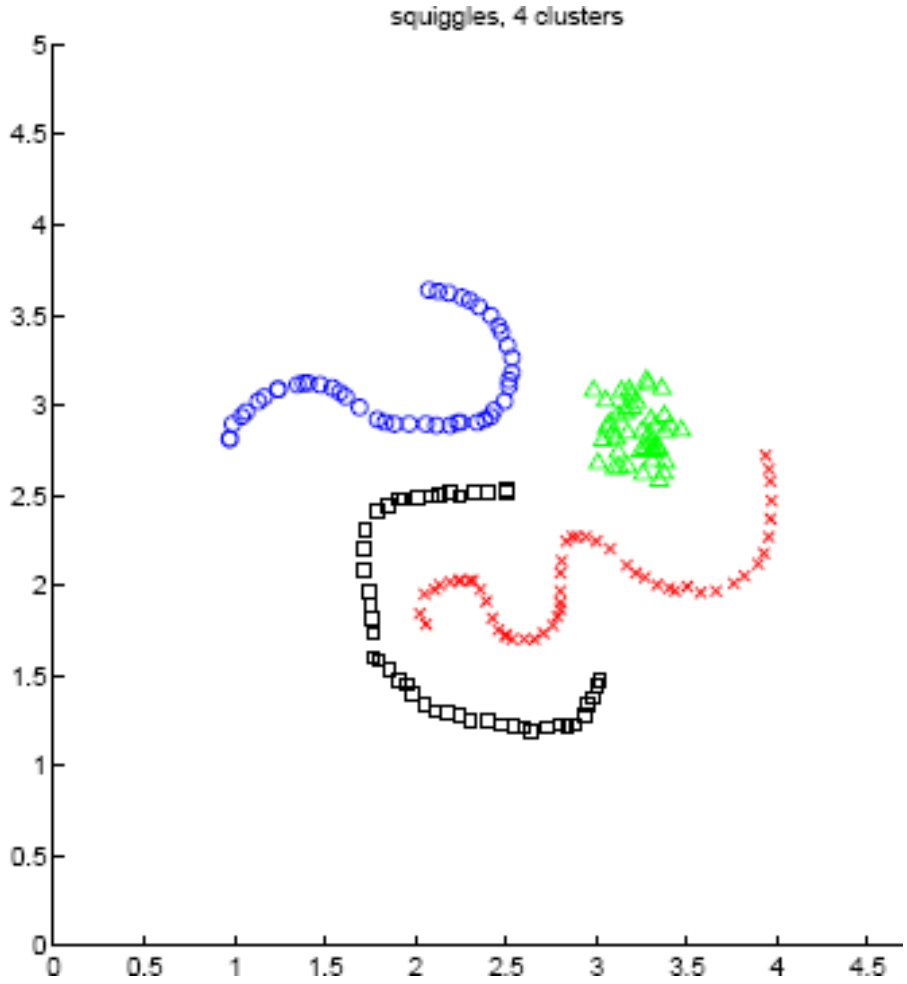


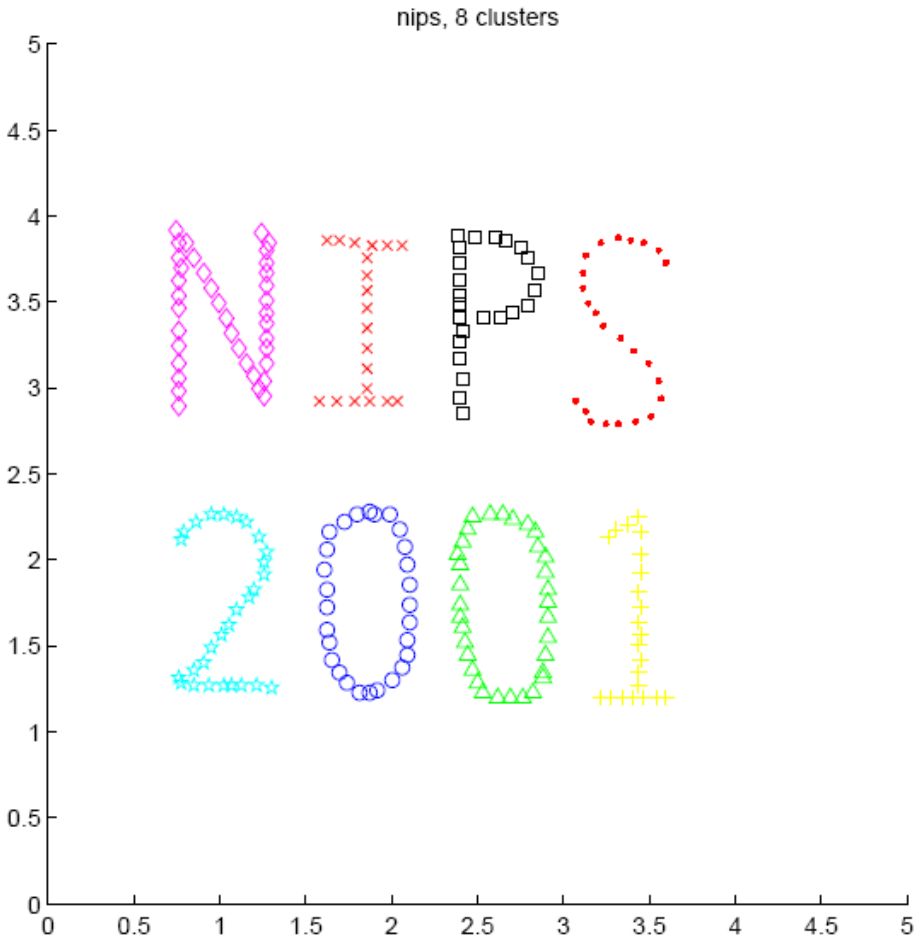
Second eigenvector of graph Laplacian



Examples

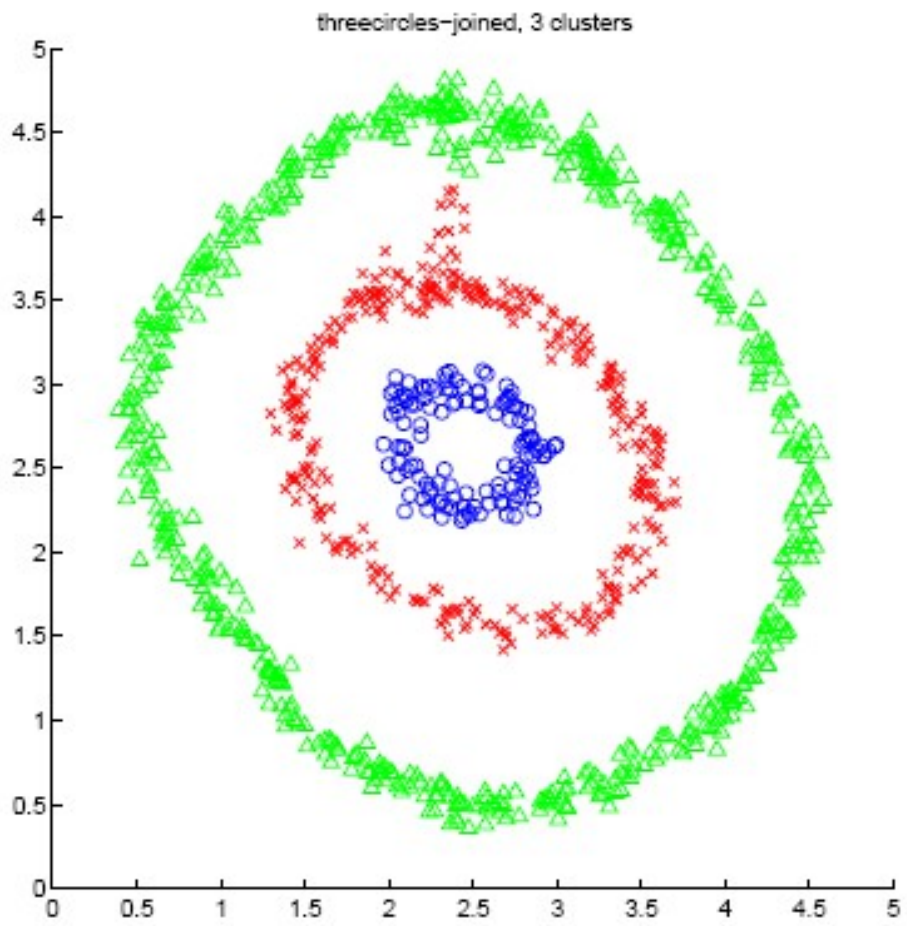
Ng et al 2001

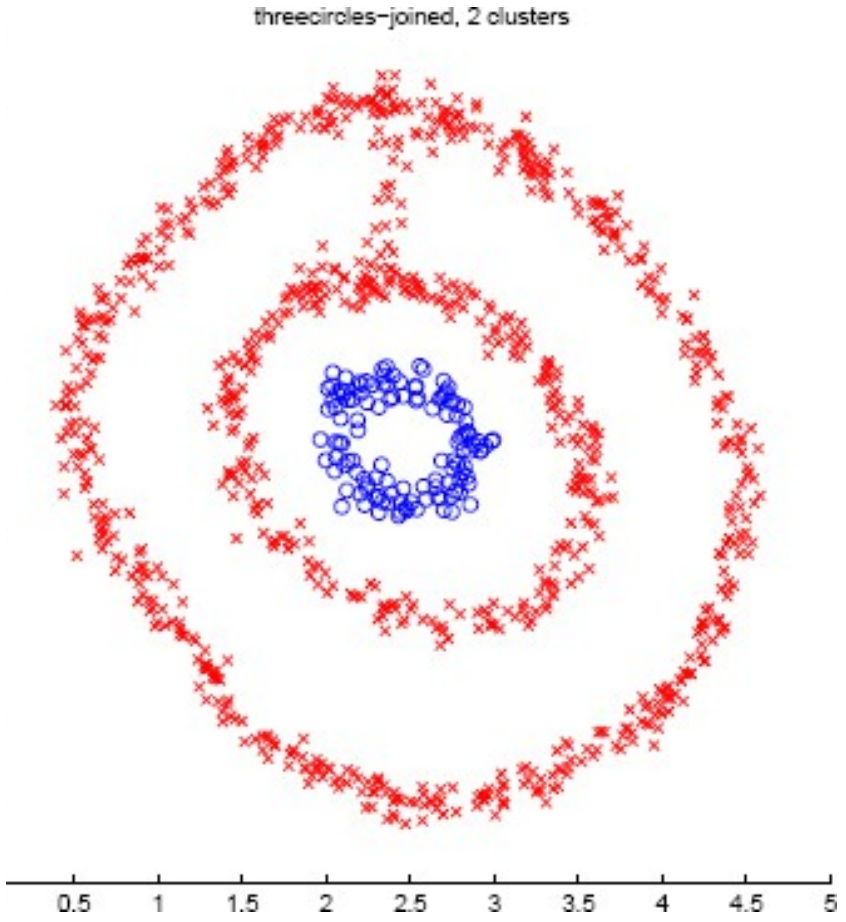




Examples (Choice of k)

Ng et al 2001



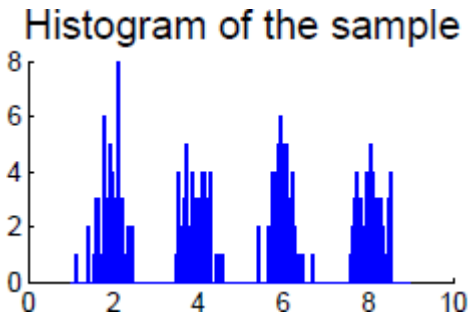


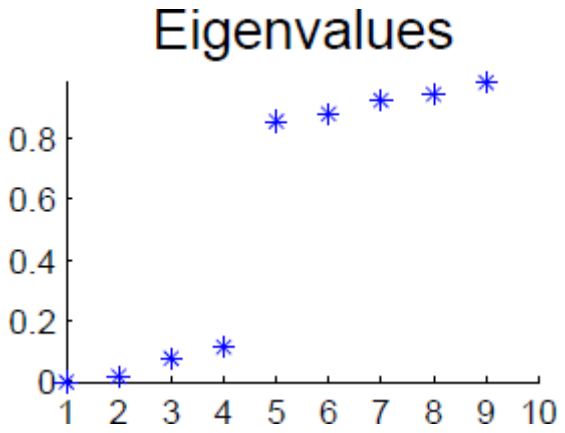
Some Issues

- Choice of number of clusters k

  Most stable clustering is usually given by the value of k that maximizes the eigengap (difference between consecutive eigenvalues)

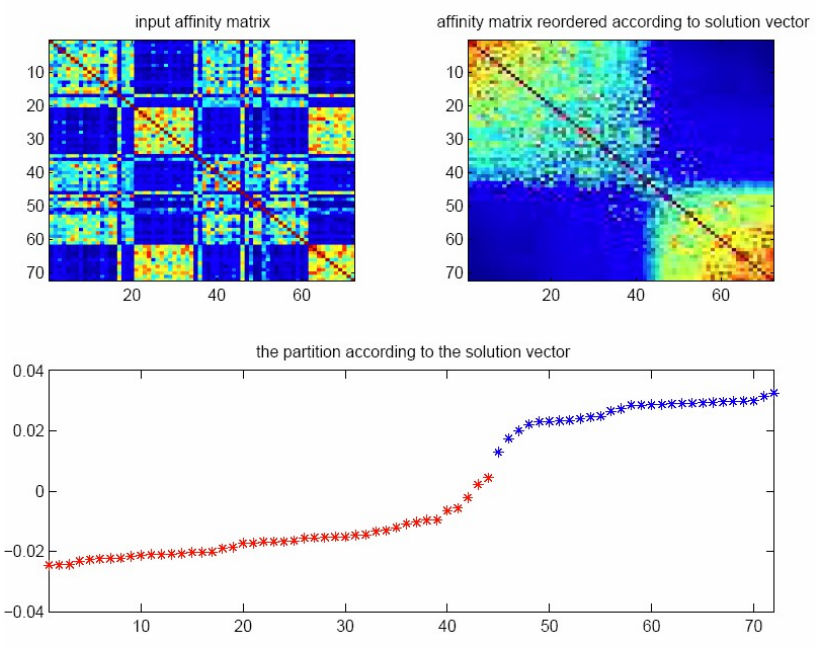
$$\Delta_k = \left|\lambda_k - \lambda_{k-1}\right|$$



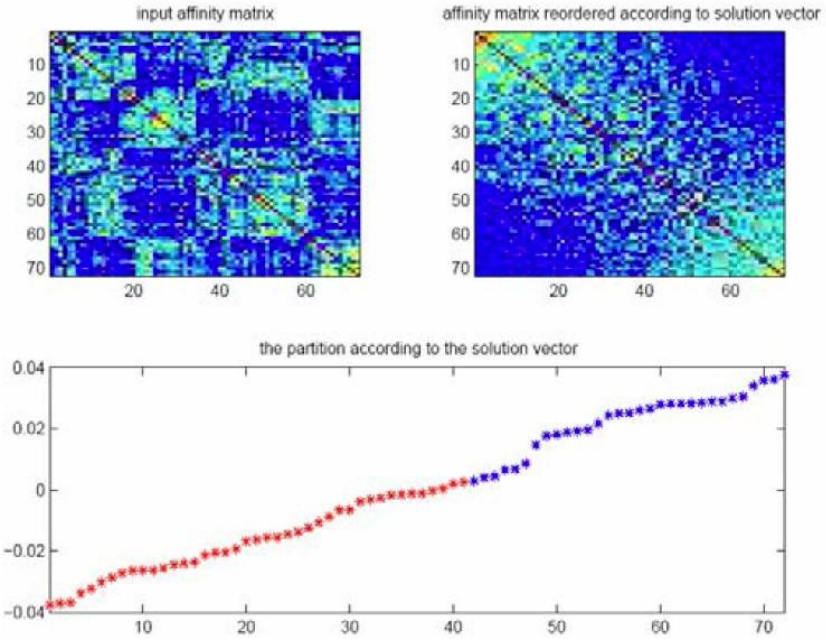


Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of σ



Good similarity measure



Poor similarity measure

Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of σ
- Choice of clustering method – k-way vs. recursive bipartite

Spectral clustering summary

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Useful in hard non-convex clustering problems
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use eigenvectors of unnormalized or normalized Laplacian, differ in how to derive clusters from eigenvectors, k-way vs repeated 2-way
- Empirically very successful

References

- Pattern Recognition and Machine Learning. Christopher Bishop.
- Elements of Statistical Learning. Hastie and Tibshirani.
- A Tutorial on Spectral Clustering. Ulrike von Luxburg.