Clustering in Computer Vision

12 October 2018

CSE473/573 – Autumn 2018

Clustering

- What is clustering?
  - Grouping of "objects" into meaningful categories
  - Given a representation of N objects, find k clusters based on a suitable measure of similarity.
- Data Clustering is useful in and beyond Computer Vision
  - Segmentation as clustering (today)
  - Texture modeling
  - Quantization
  - Beyond
    - Data exploration
    - Compression
    - Natural classification

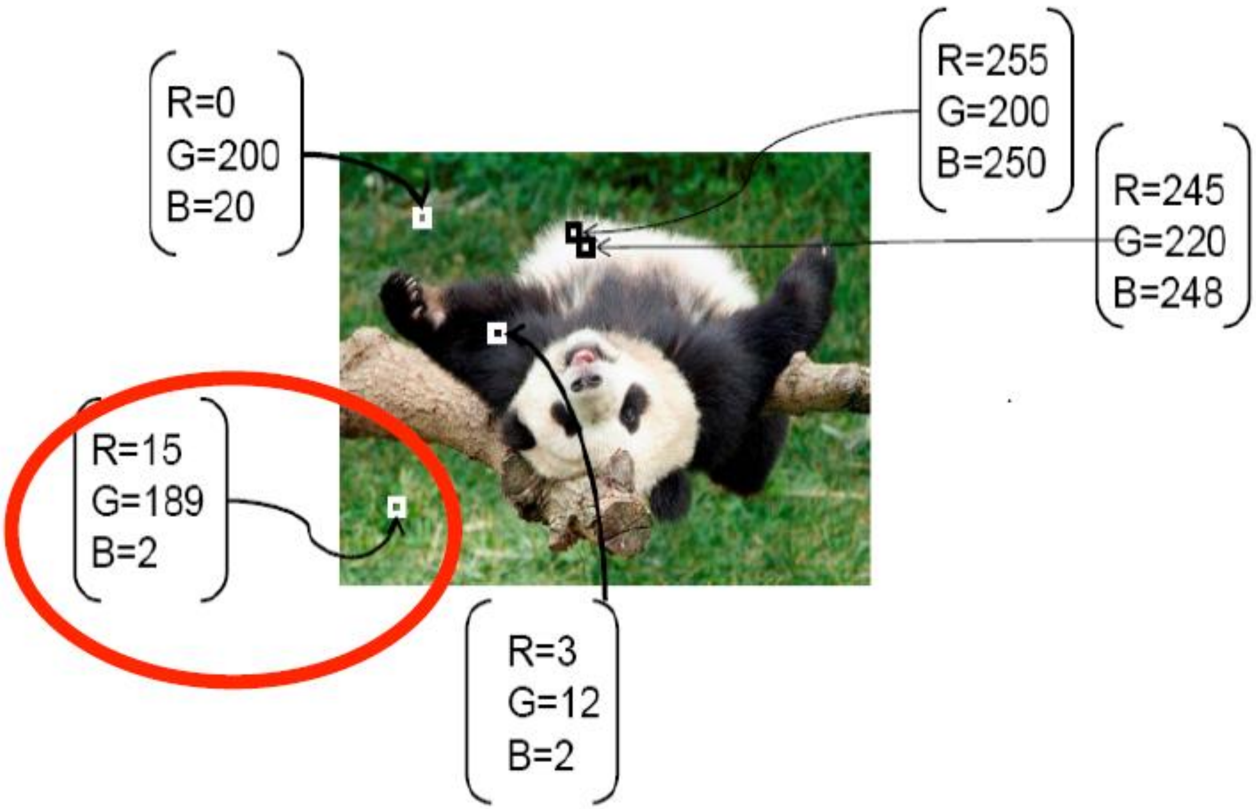
Source: A. K. Jain and R. C. Dubes. Alg. for Clustering Data, Prentice Hall, 1988.

Feature Space

- Every token is identified by a set of salient visual characteristics. For example:
  - Position
  - Color
  - Texture
  - Motion vector
  - Size, orientation (if token is larger than a pixel)

Slide credit: Christopher Rasmussen

Feature Space - Example

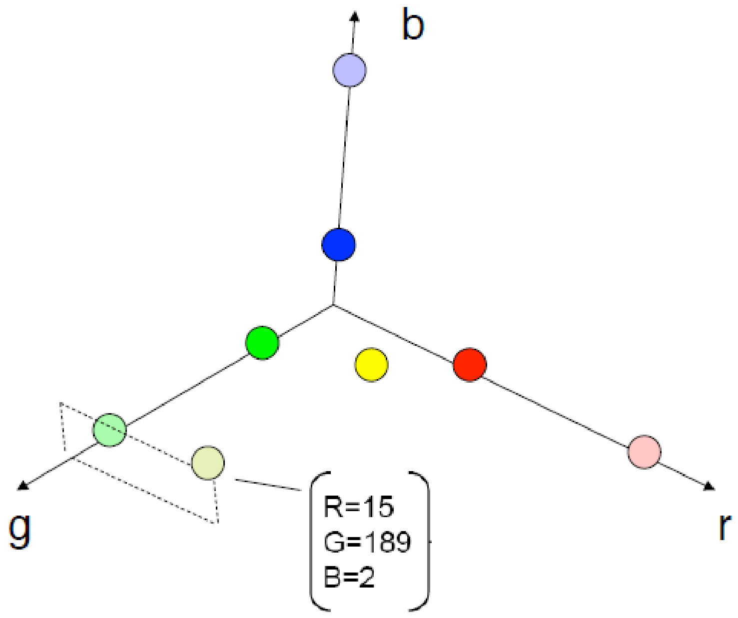


Source: K. Grauman

Source: Savarese slides.

Feature Space - Example

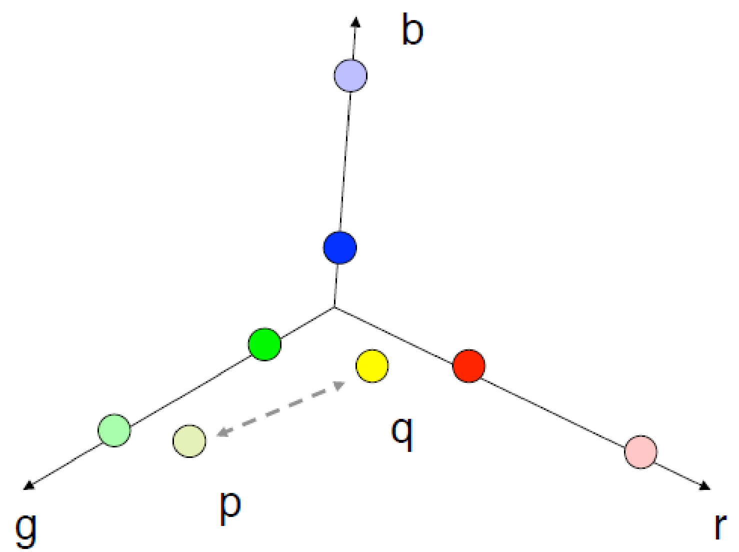
Feature space:
each token is represented by a point



Source: Savarese slides.

Feature Space - Similarity

Token similarity is thus measured by distance between points ("feature vectors") in feature space

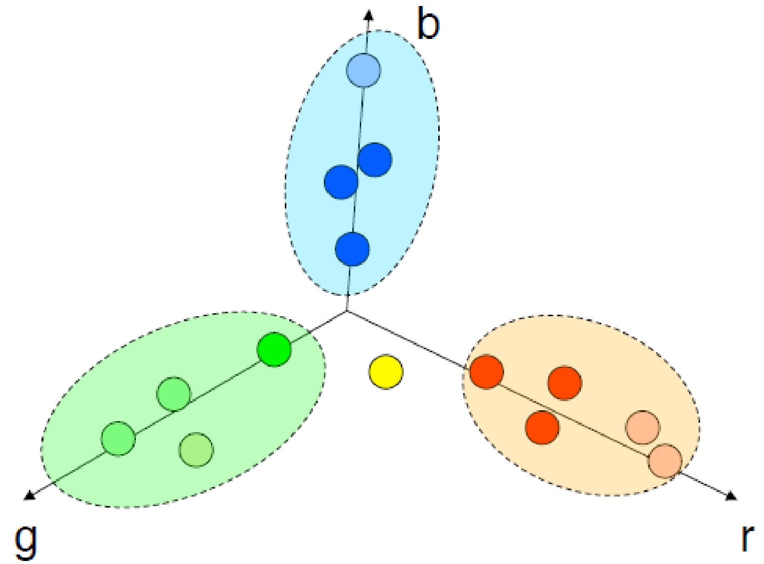


$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.$$

Source: Savarese slides.

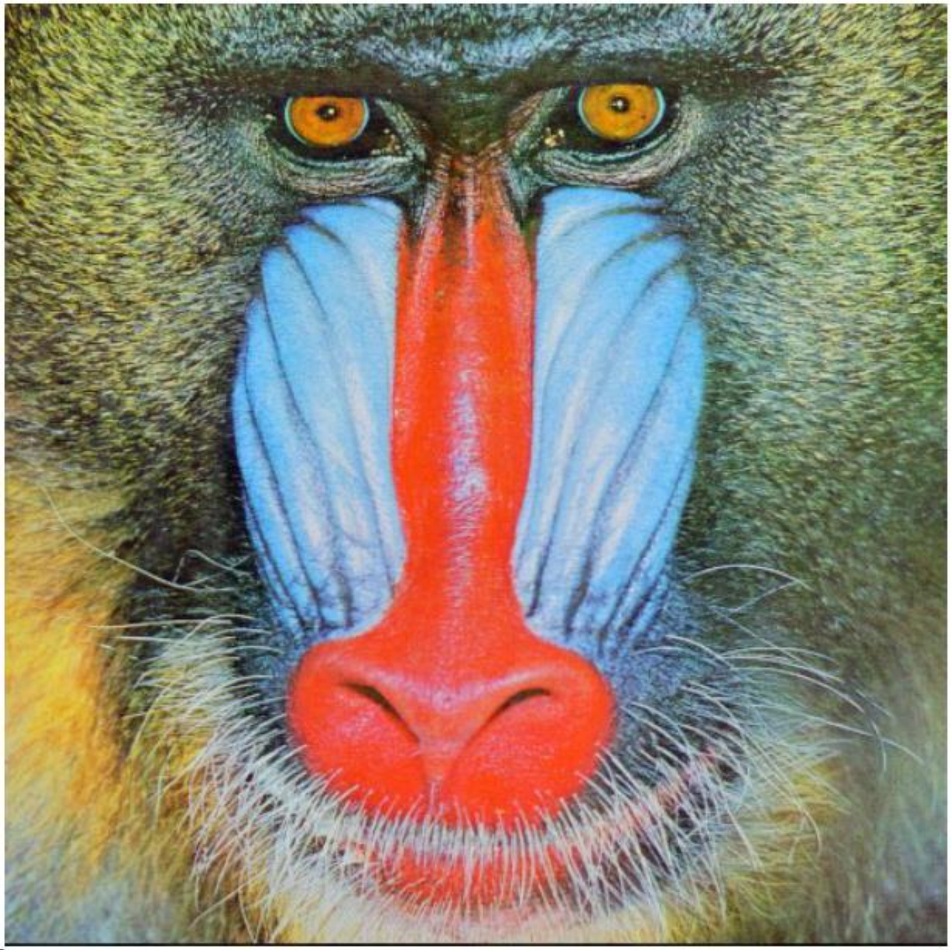
Feature Space - Similarity

Cluster together tokens with high similarity



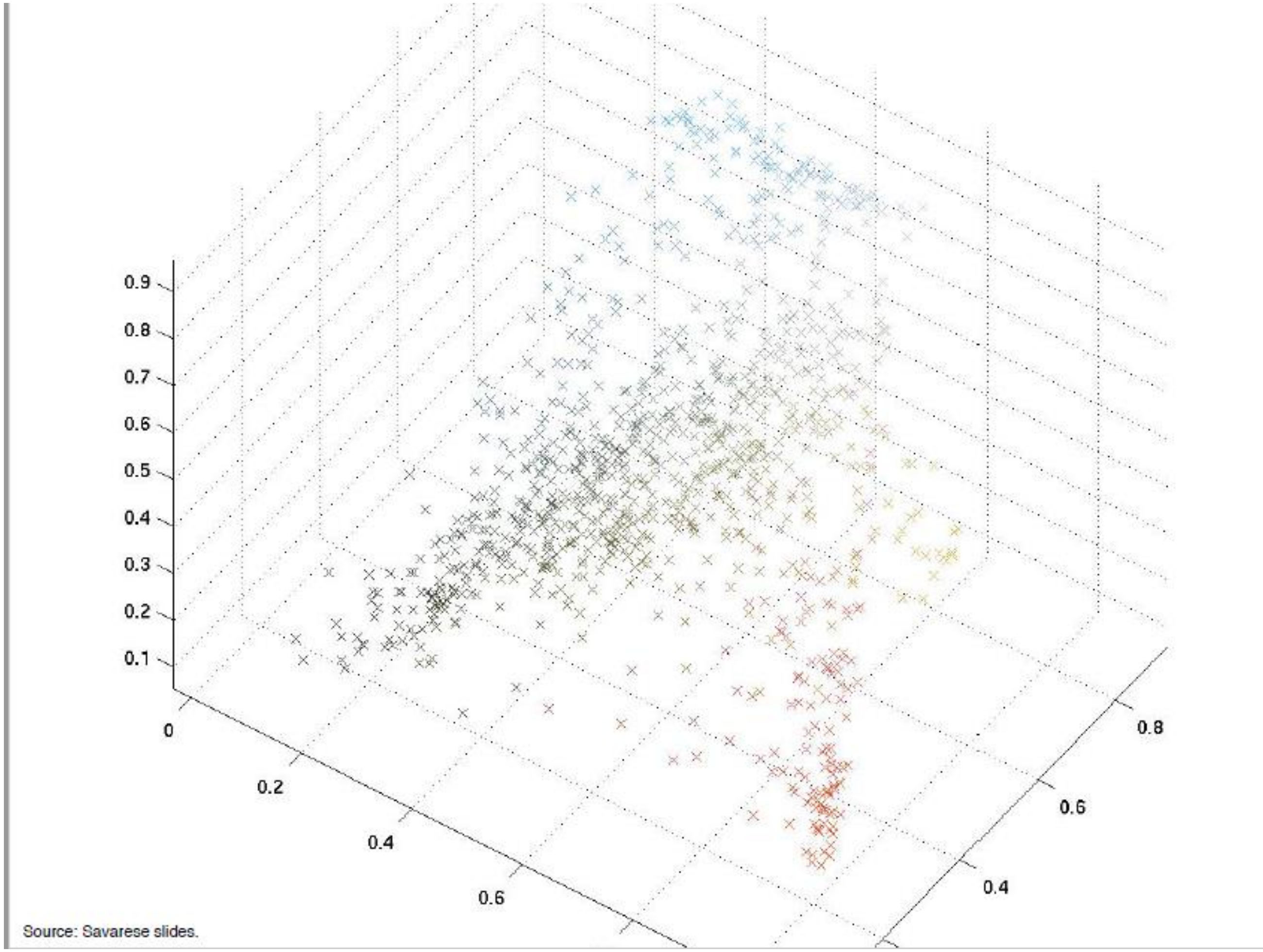
Source: Savarese slides.

Color based Clustering - Example



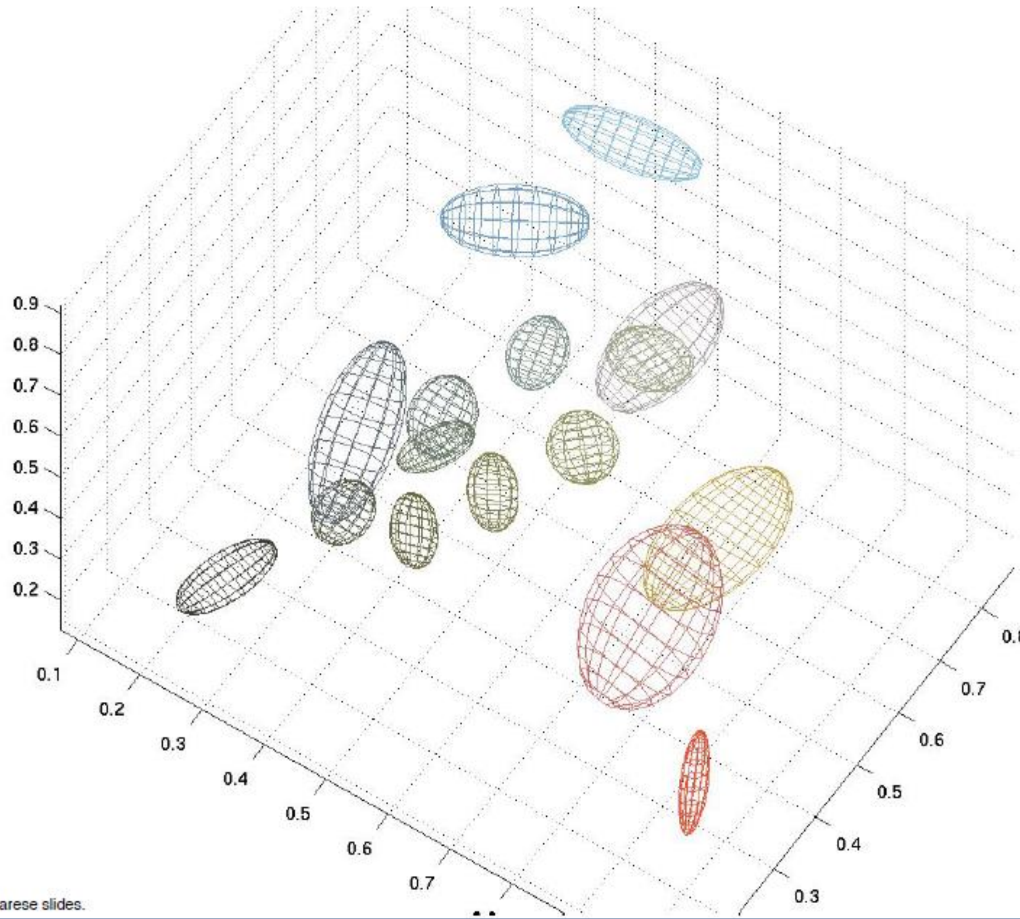
e: Savarese slides.

Color based Clustering - Example



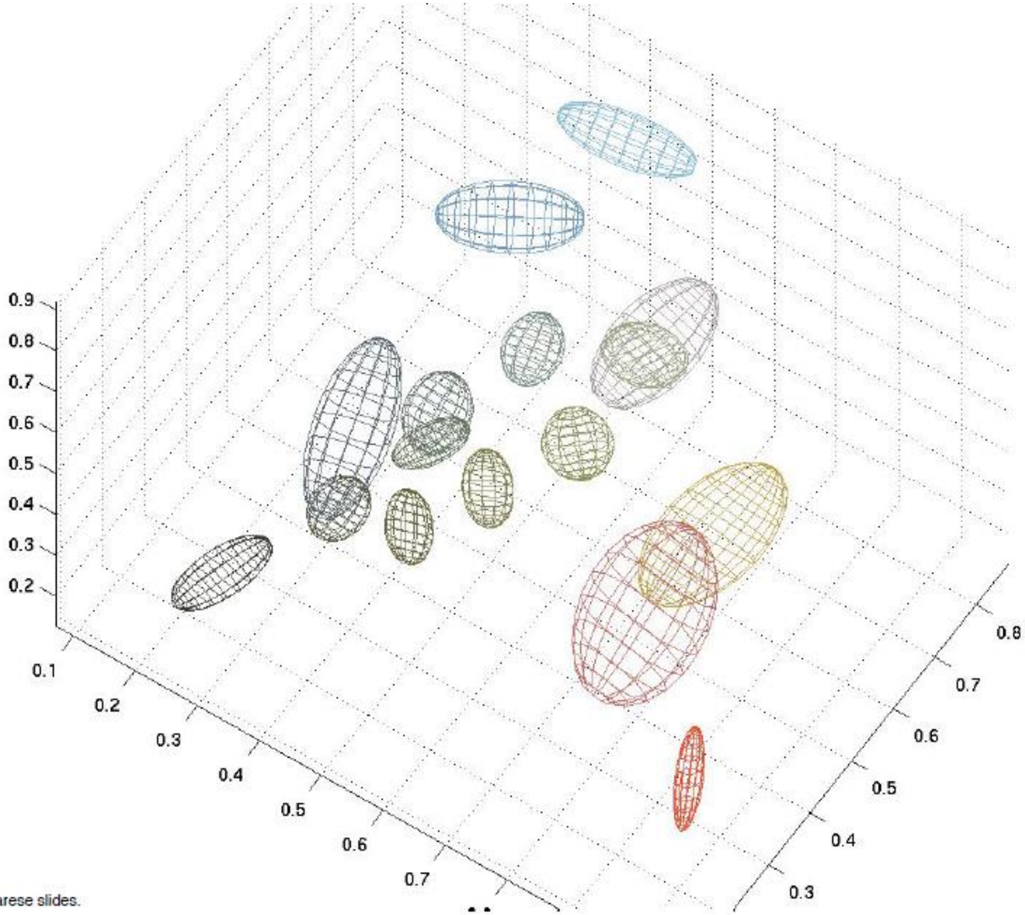
Source: Savarese slides.

Color based Clustering - Example



Source: Savarese slides.

Color based Clustering - Example



Source: Savarese slides.

Clustering – formal definition

- Given a set of N data samples $D = x_1, x_2, \ldots, x_N$ in a d-dimensional feature space, D is partitioned into a number of disjoint subsets D_j:

$$D = \bigcup_{j=1}^{k} D_j \quad \text{where} \quad D_i \cup D_j = \varnothing \quad \forall i \neq j$$

where the points in each subset are similar to each other according to the given similarity function.

- A partition is denoted by

$$\pi = (D_1, D_2, \ldots, D_k)$$

and clustering is then formulated as

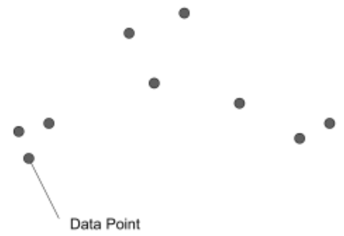
$$\pi^* = \arg\min_{\pi} f(\pi)$$

for $f(\cdot)$ that captures the desired cluster properties.

K-Means Clustering

Iterative process. Takes multiple iterations to converge

1. Randomly initialize $\mu_1, \mu_2, \ldots, \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i

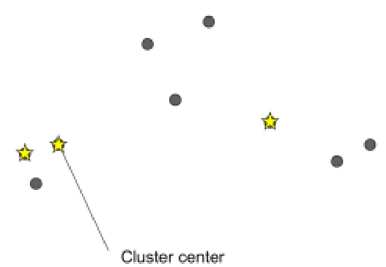


Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Initialize K centers

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, \ldots, \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



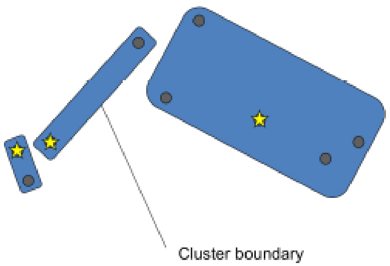
First choose *k* arbitrary centers

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Assign nearest points to clusters

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, ..., \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



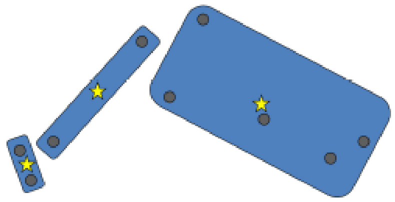
Assign points to closest centers

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Re-compute centers of clusters

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, ..., \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i

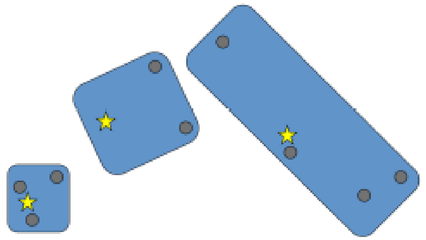


Recompute centers

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Re-Assign points to clusters

1. Randomly initialize $\mu_1, \mu_2, \ldots, \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



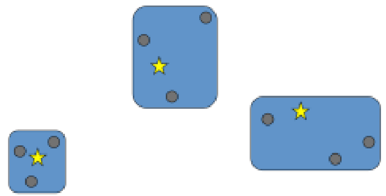
Assign points to closest centers

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Iterate….

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, \ldots, \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



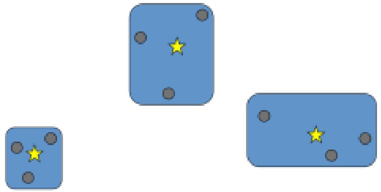
Assign points to closest centers

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Iterate….

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, \ldots, \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



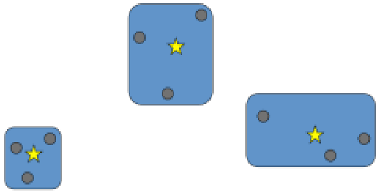
Recompute centers

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Stop! When the centers do not move

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, \ldots, \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



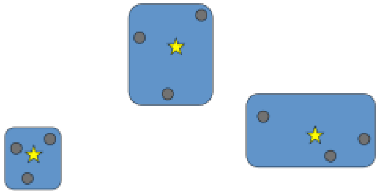
Points already assigned to nearest centers: Algorithm ends

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

Stop! When the centers do not move

K-Means Clustering

1. Randomly initialize $\mu_1, \mu_2, ..., \mu_c$
2. Repeat until no change in μ_i:
   (a) Classify N samples according to nearest μ_i
   (b) Recompute μ_i



Points already assigned to nearest centers: Algorithm ends

Source: D. Aurthor, S. Vassilivitskii. "k-Means++: The Advantages of Careful Seeding".

K-Means Clustering Animation

http://shabal.in/visuals/kmeans/1.html

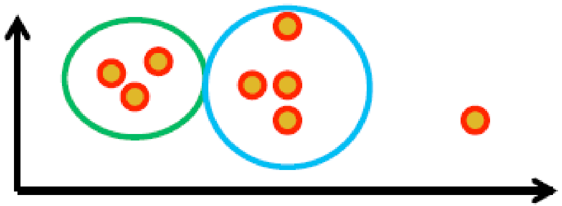
http://simplystatistics.org/2014/02/18/k-means-clustering-in-a-gif/

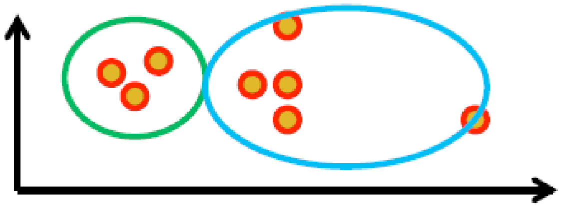
K-Means Clustering – Pros and Cons

- Pros
  - Simple and fast
  - (Always) converges to a local minimum of the error function
  - Available implementations (e.g., in Matlab)

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

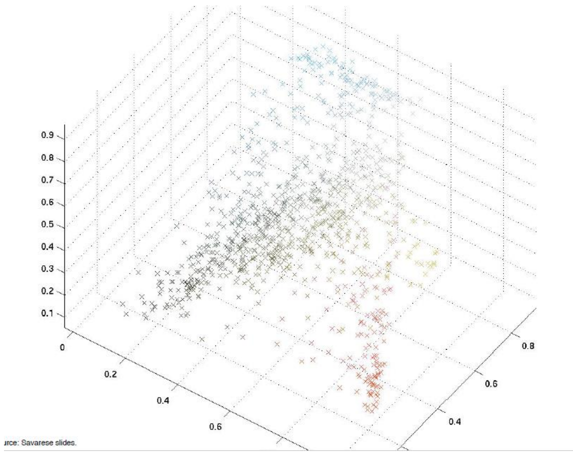
- Cons
  - Need to pick K
  - Sensitive to initialization
  - Only finds "spherical" clusters
  - Sensitive to outliers

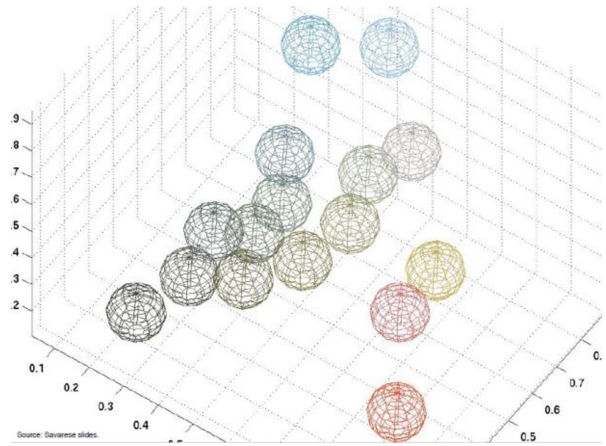




Source: Savarese slides.

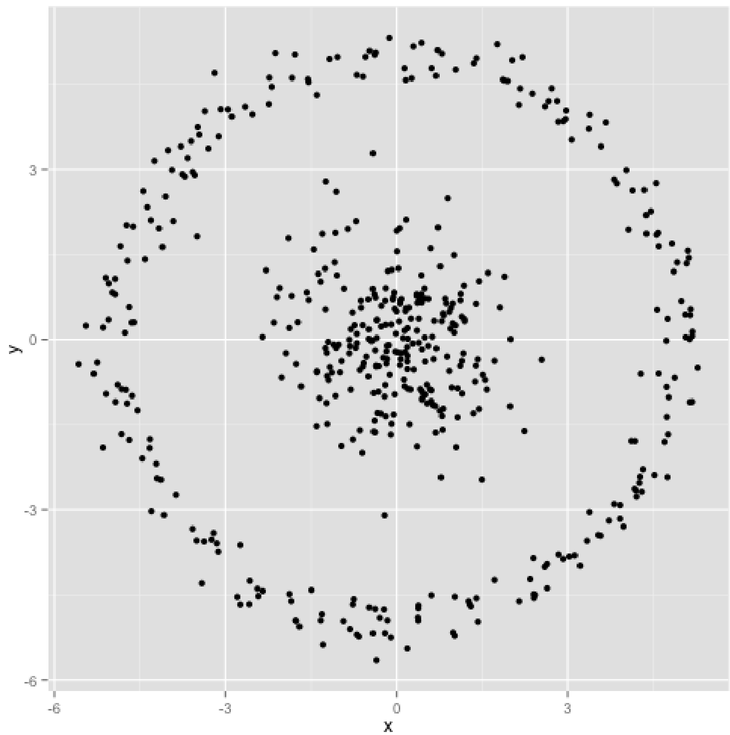
K-Means Clustering – Example



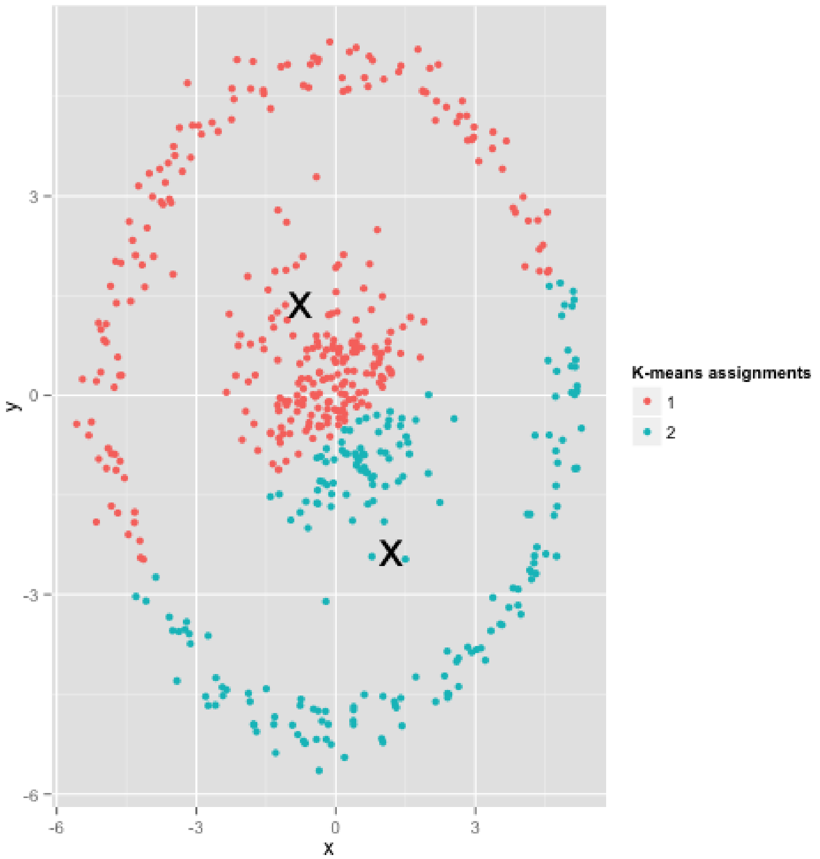


Source: Savarese slides.

What are the clusters here?



K-Means Clustering fails



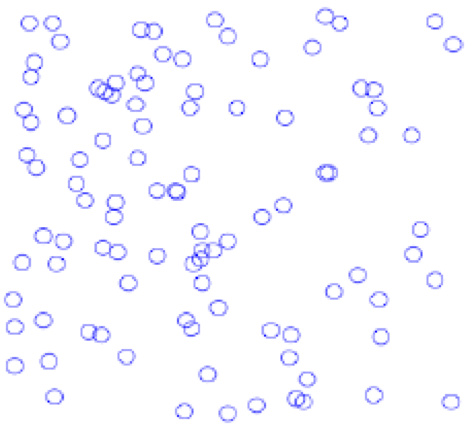
Challenges

1. What is a cluster?
2. How to define pair-wise similarity?
3. Which features? Which normalizations scheme?
4. How many clusters?
5. Which clustering method?
6. Are the discovered clusters and partitioning valid?
7. Does the data have any clustering tendency?

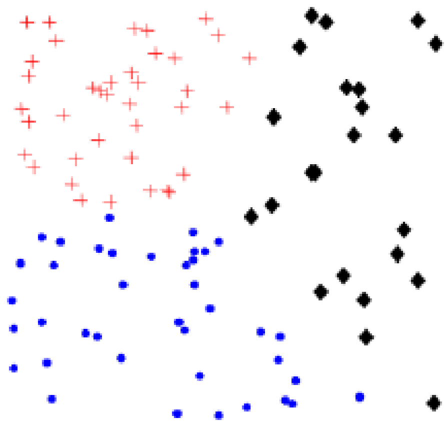
Source: R. Dubes and A. K. Jain. “Clustering Techniques: User’s Dilemma” PR 1976.

Cluster Validity

- Clustering algorithms find clusters, even if there are no **natural** clusters in the data.



100 2D uniform data points



k-Means with k=3

Soft vs Hard Clustering

- Kmeans performs Hard clustering:
  - Data point is deterministically assigned to one and only one cluster
  - But in reality clusters may overlap
- Soft-clustering:
  - Data points are assigned to clusters with certain probabilities

Probabilistic Clustering

- Basic questions
  - what's the probability that a point **x** is in cluster m?
  - what's the shape of each cluster?
- K-means doesn't answer these questions

- Basic idea
  - instead of treating the data as a bunch of points, assume that they are all generated by sampling a continuous function
  - This function is called a **generative model**

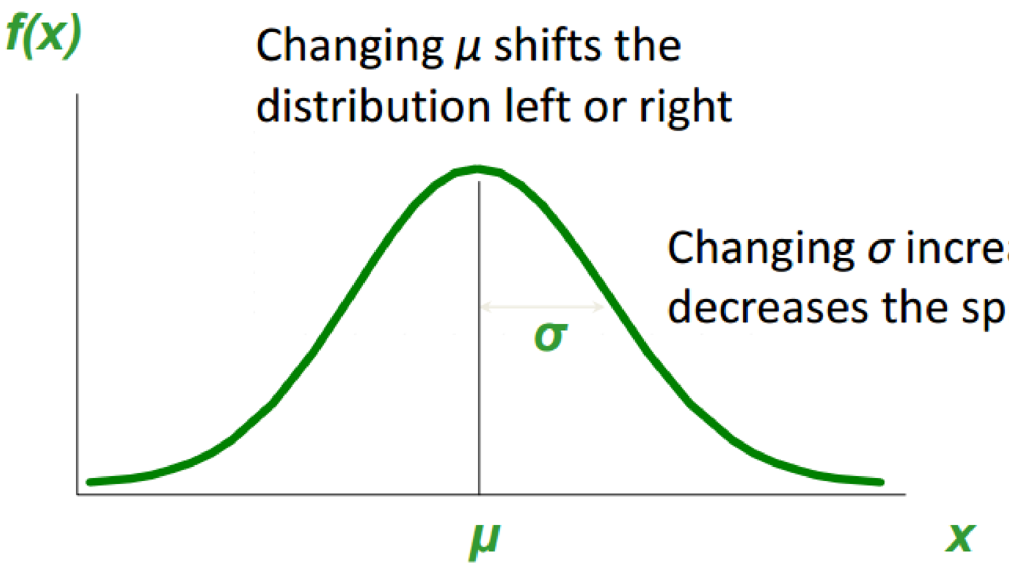
How can we extend K-means to make soft clustering

- Given a set of clusters centers $\mu_1, \mu_2, \ldots, \mu_k$, instead of directly assign all data points to their closest clusters, we can assign them **partially (probabilistically) based on the distances**
- This can be done by
  - assuming a probabilistic distribution (model) for each cluster
  - compute the probability that each point belongs to each cluster
  - Often referred to as Model-based clustering

Gaussian for representing a cluster

- What exactly is a cluster?
  - Intuitively it is a tightly packed ball-shape like thing
- We can use a Gaussian (normal) distribution to describe it
- Let’s first review what is a Gaussian distribution

Gaussian Distribution

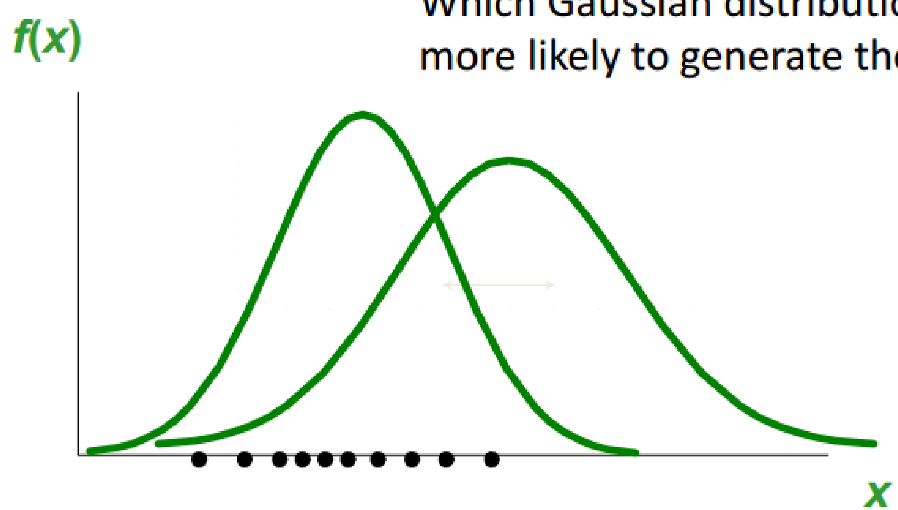


Probability density function $f(x)$ is a function of x given μ and σ

$$N(x\mid\mu,\sigma^2)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Gaussian Distribution

Which Gaussian distribution is more likely to generate the data?

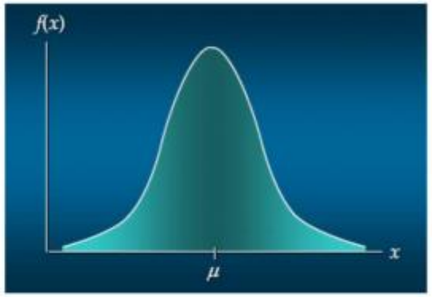


Define likelihood as a function of μ and σ given x_1, x_2, …, x_n

$$\prod_{i=1}^{n} N(x_i \mid \mu, \sigma^2)$$

Multi-variate Gaussian

- Univariate Gaussian distribution:

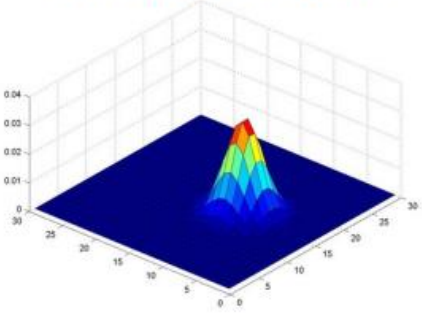


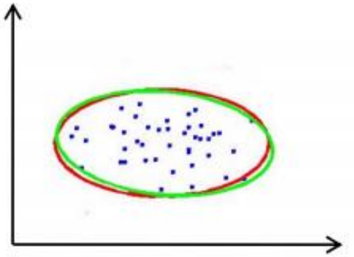
$N(\mu, \sigma^2)$

μ – mean, center of the mass

σ^2 – standard deviation, spread of the mass

- Multivariate Gaussian distribution:





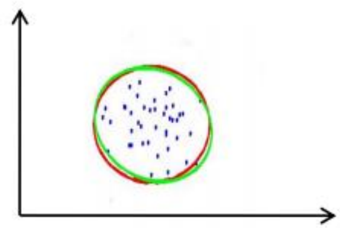
$N(\mu, \Sigma)$

μ – (μ_1, μ_2)

Σ – Covariance matrix

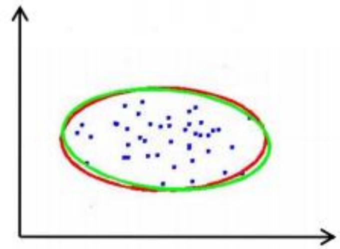
$$\begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$$

Different Co-Variance Matrices



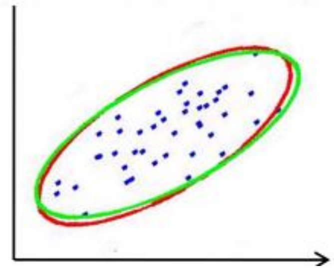
Covariance matrix $\Sigma =$

$$\begin{matrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{matrix}$$



Covariance matrix $\Sigma =$

$$\begin{matrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{matrix}$$



Covariance matrix $\Sigma =$

$$\begin{matrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{matrix}$$

Log-Likelihood

- Multivariate Gaussian

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

mean

covariance

- Log likelihood

$$L(\mu,\Sigma) = \sum_{i=1}^{n} \ln N(x_i|\mu,\Sigma) = \sum_{i=1}^{n} (-\frac{1}{2}(x_i-\mu)^T \Sigma^{-1}(x_i-\mu)) - \pi \ln|\Sigma|)$$

Gaussian Mixture Models (GMMs)

$$\mathcal{N}(x|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

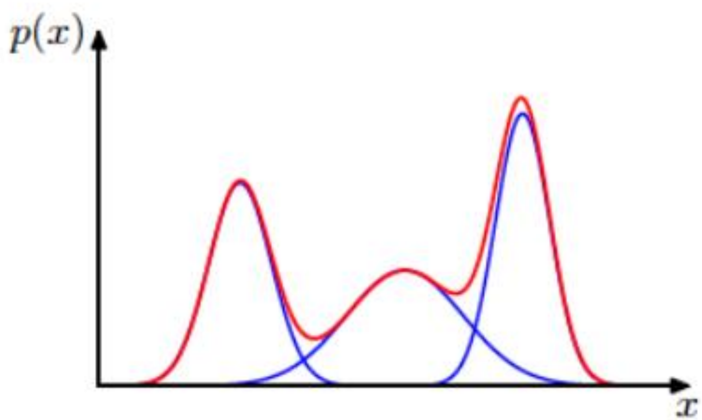
- It forms the basis for the mixture of Gaussians density
- The Gaussian mixture is linear superposition of Gaussians:

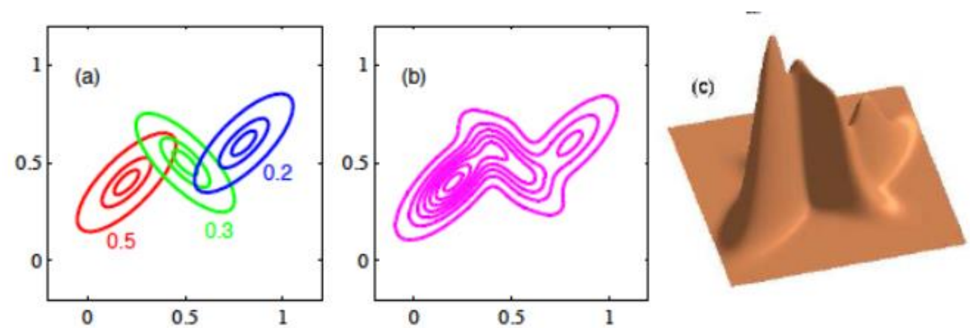
$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- The π_k are non-negative scalars called mixing coefficients and they govern the relative importance between the various Gaussians in the mixture density. $\sum_k \pi_k = 1$

Gaussian Mixture Models (GMMs)

In below examples – the distribution is a mixture of three gaussians with different means and variances





Clustering using GMMs

- Given a set of data points, and assume that we know there are k clusters in the data, we need to:
  - Assign the data points to the k clusters (soft assignment)
  - Learn the gaussian distribution parameters for each cluster: μ and Σ

Expectation Maximizaton (EM)

- Randomly initialize the Gaussian parameters
- Repeat until converge

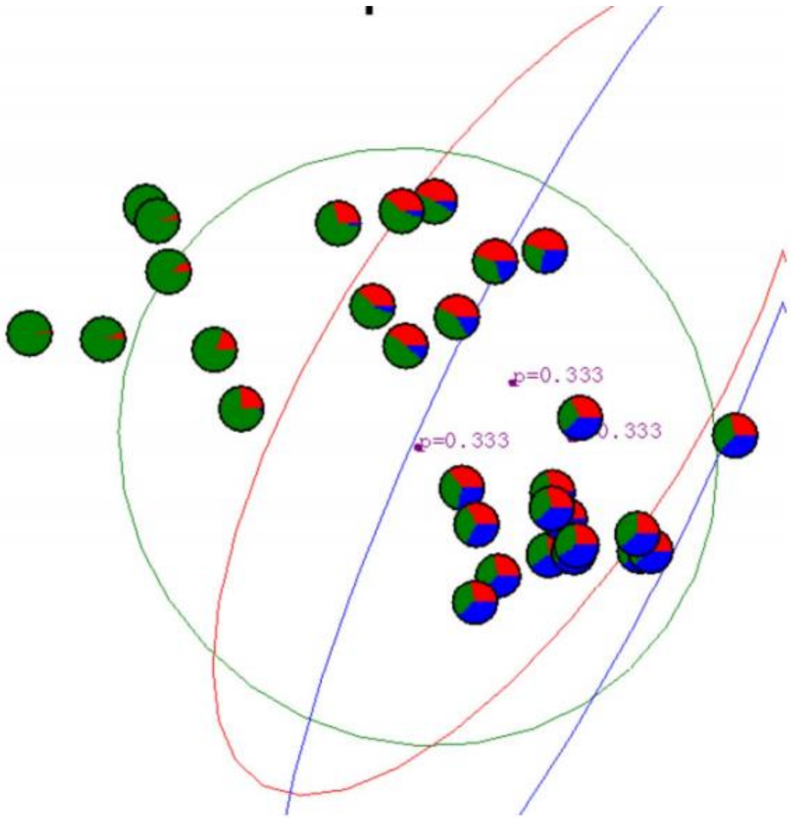
1. Compute $P(\mathbf{x}^j \in C_i \mid \mathbf{x}^j)$ for all data points and all clusters

   This is called the E-step for it computes the expected values of the cluster memberships for each data point

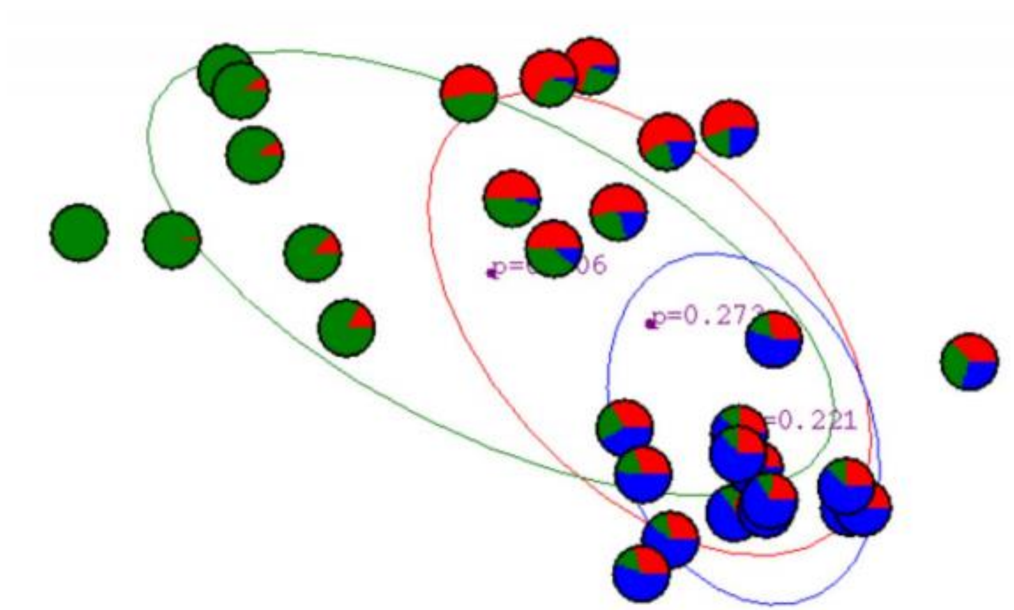
2. Re-compute the parameters of each Gaussian

   This is called the M-step for it performs maximum likelihood estimation of parameters

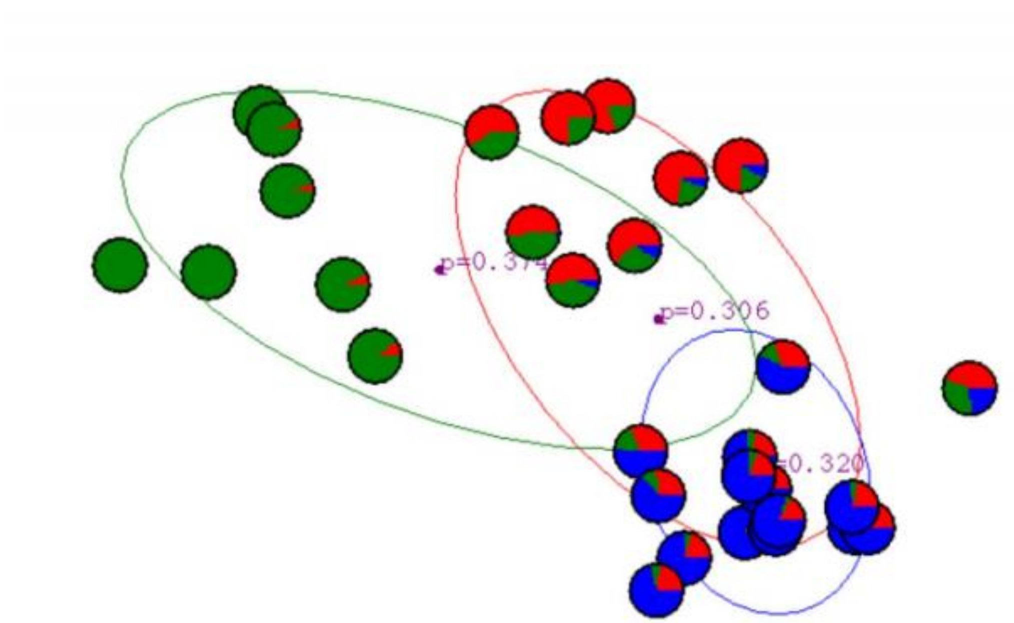
GMM Example - Initialization



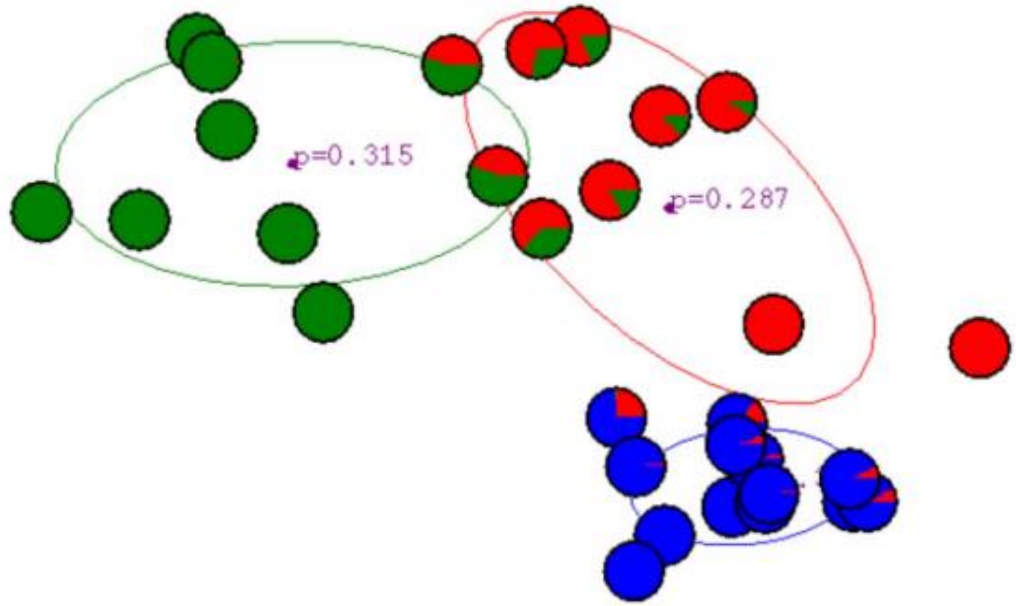
After First Iteration



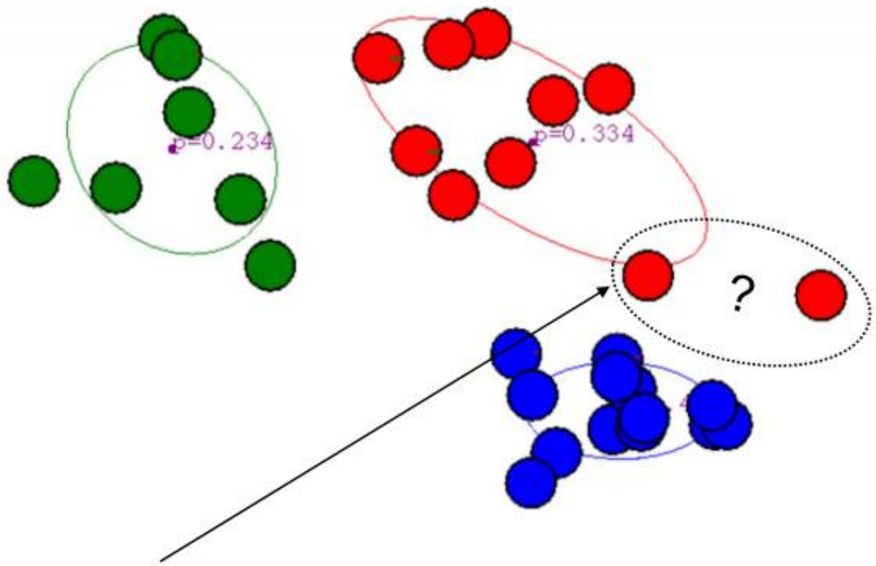
After Second Iteration



After Sixth Iteration



After 20th Iteration



Q: Why are these two points red when
they appear to be closer to blue?

K-Means vs GMMs

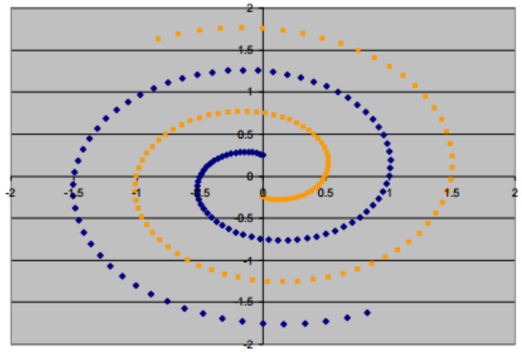
- Objective function
  - Minimize sum of squared Euclidean distance
- Can be optimized by an EM algorithm
  - E-step: assign points to clusters
  - M-step: optimize clusters
  - Performs hard assignment during E-step
- Assumes spherical clusters with equal probability of a cluster

- Objective function
  - Maximize log-likelihood
- EM algorithm
  - E-step: Compute posterior probability of membership
  - M-step: Optimize parameters
  - Perform soft assignment during E-step
- Can be used for non-spherical clusters
- Can generate clusters with different probabilities

Spectral Clustering

Spectral clustering techniques make use of the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions

- **Spectral approach**
  - Use similarity graphs to encode local neighborhood information
  - Data points are vertices of the graph
  - Connect points which are "close"

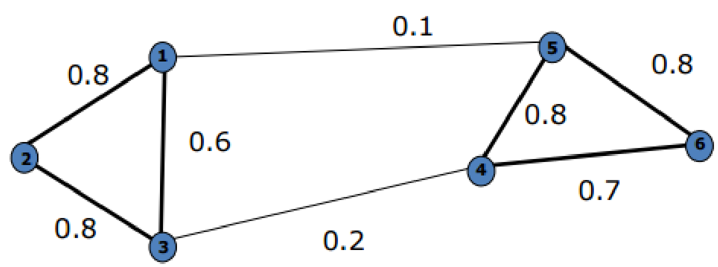


Similarity Graph

- Represent dataset as a weighted graph $G(V,E)$
- All vertices which can be reached from each other by a path form a connected component
- Only one connected component in the graph—The graph is fully connected

$V=\{x_i\}$ Set of n vertices representing data points

$E=\{W_{ij}\}$ Set of weighted edges indicating pair-wise similarity between points

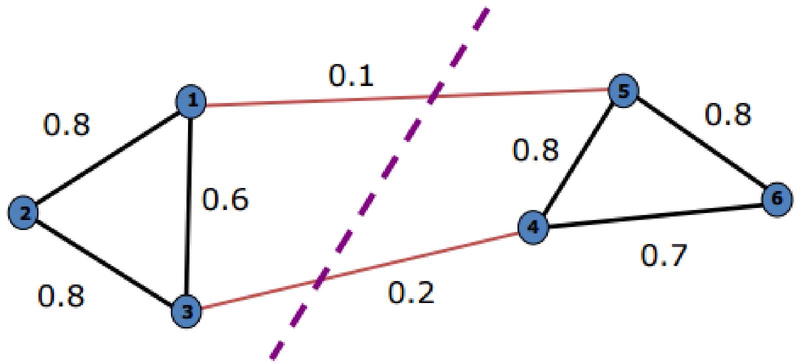


Clustering Objective

Traditional definition of a "good" clustering

- Points assigned to same cluster should be highly similar
- Points assigned to different clusters should be highly dissimilar

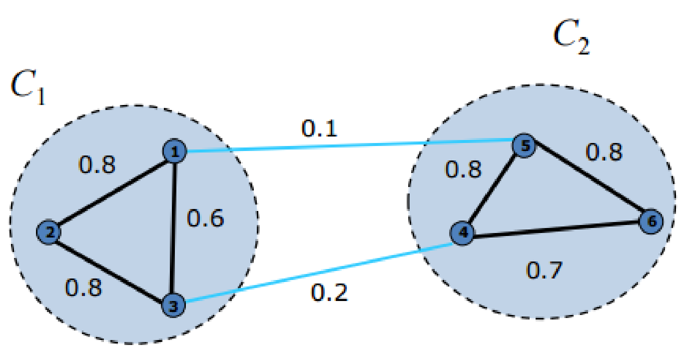
Apply this objective to our graph representation



Minimize weight of between-group connections

Graph Cut

- Express clustering objective as a function of the ***edge cut*** of the partition
- ***Cut****:* Sum of weights of edges with only one vertex in each group
- We wants to find the ***minimal cut*** between groups

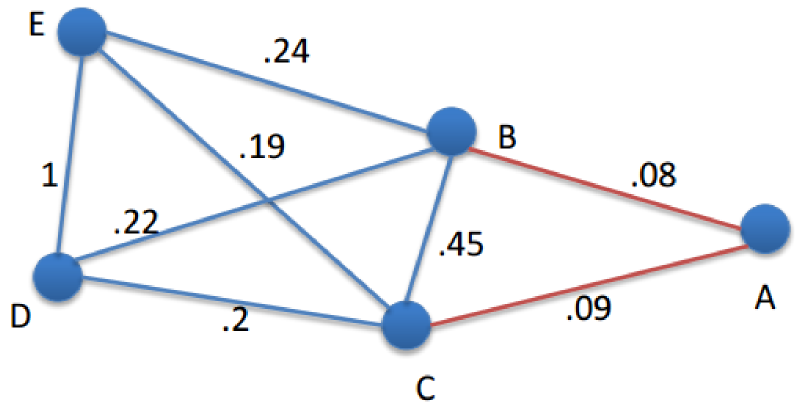


$$cut(C_1, C_2) = \sum_{i \in C_1, j \in C_2} w_{ij}$$

$$cut(C_1, C_2) = 0.3$$

Minimum Cut - Example

- Minimum Cut



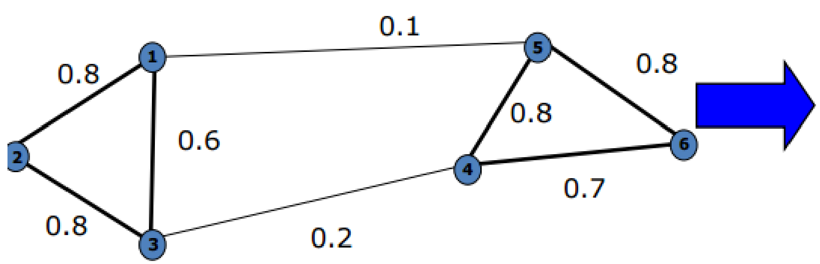
$$Cut(BCDE, A) = 0.17$$

Problem

- Identifying a minimum cut is NP-hard
- There are efficient approximations using linear algebra
- Based on the Laplacian Matrix, or **graph Laplacian**

Similarity Matrix Representation

- **Similarity matrix (W)**
  - $n \times n$ matrix
  - $W = [w_{ij}]$: edge weight between vertex x_i and x_j

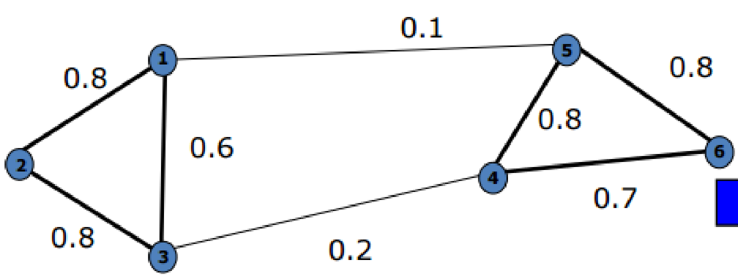


	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

- **Important properties**
  - Symmetric matrix

Degree Matrix

- **Degree matrix (D)**
  - $n \times n$ diagonal matrix
  - $D(i,i) = \sum_j w_{ij}$: total weight of edges incident to vertex x_i



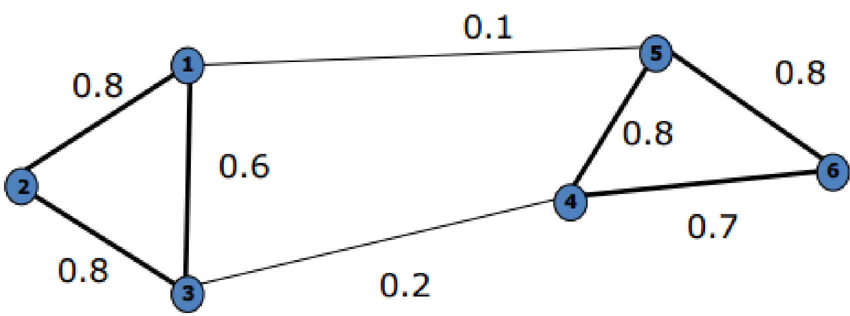
	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	0	0	0	0	0
x_2	0	1.6	0	0	0	0
x_3	0	0	1.6	0	0	0
x_4	0	0	0	1.7	0	0
x_5	0	0	0	0	1.7	0
x_6	0	0	0	0	0	1.5

- **Used to**
  - Normalize adjacency matrix

Laplacian Matrix

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

$$L = D - W$$



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

Find an Optimal mincut

- Express a bi-partition (C_1, C_2) as a vector

$$f_i = \begin{cases} 1 & \text{if } x_i \in C_1 \\ -1 & \text{if } x_i \in C_2 \end{cases}$$

- We can minimize the cut of the partition by finding a non-trivial vector *f* that minimizes the function

$$g(f) = \sum_{i,j \in V} w_{ij} (f_i - f_j)^2 \;\; = f^T L \, f$$

Laplacian matrix

How it works

- How eigen decomposition of L relates to clustering?

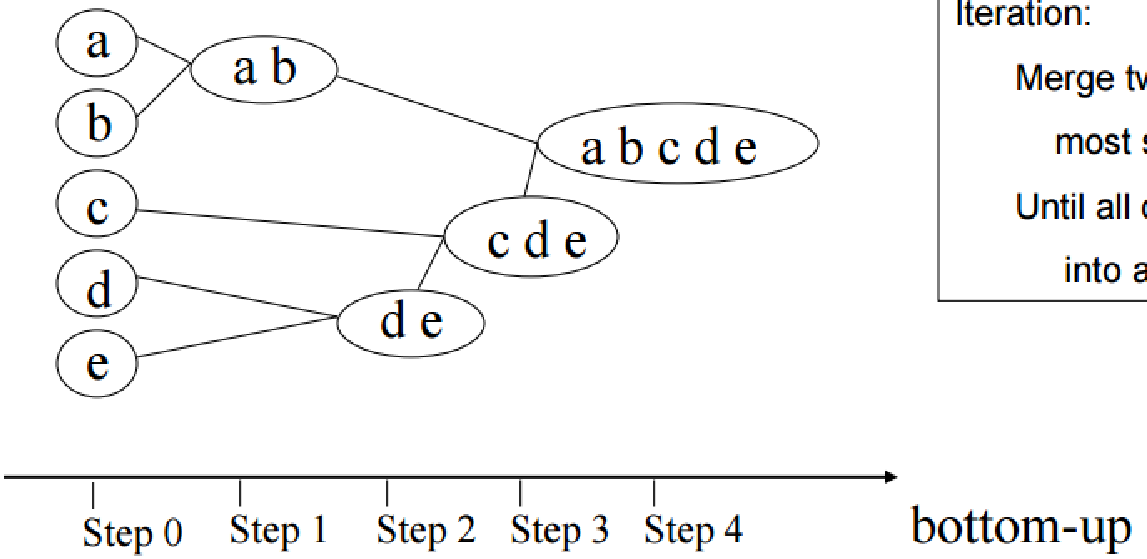
$$L = D - W \qquad f(x_j) = f_j \text{ cluster assignment}$$

$$\begin{aligned} f^T L f &= f^T D f - f^T W f \\ &= \frac{1}{2}\left(\sum_i \left(\sum_j w_{ij}\right) f_i^2 - 2\sum_{ij} f_i f_j w_{ij} + \sum_j \left(\sum_i w_{ij}\right) f_j^2\right) \\ &= \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2 \end{aligned}$$

--Cluster objective function

Hierarchical Clustering

- **Agglomerative approach**



Initialization:

Each object is a cluster

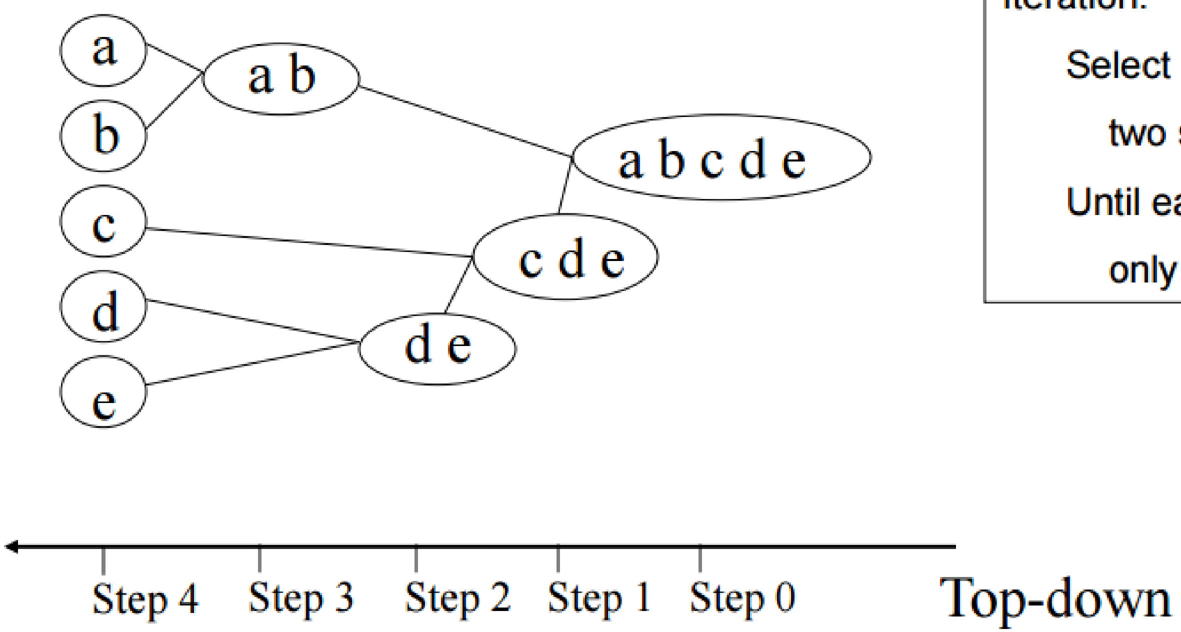
Iteration:

Merge two clusters which are most similar to each other;

Until all objects are merged into a single cluster

Hierarchical Clustering

- **Divisive Approaches**



Initialization:

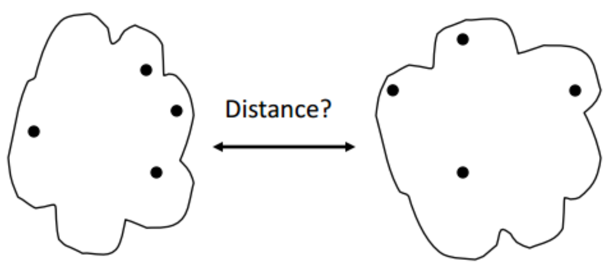
All objects stay in one cluster

Iteration:

Select a cluster and split it into two sub clusters

Until each leaf cluster contains only one object

Inter-Cluster Distance



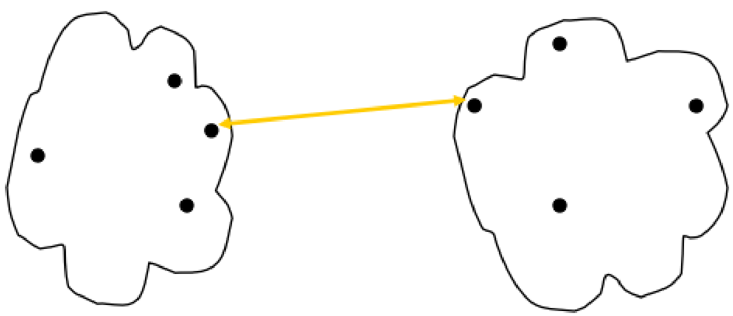
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Distance Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- ……

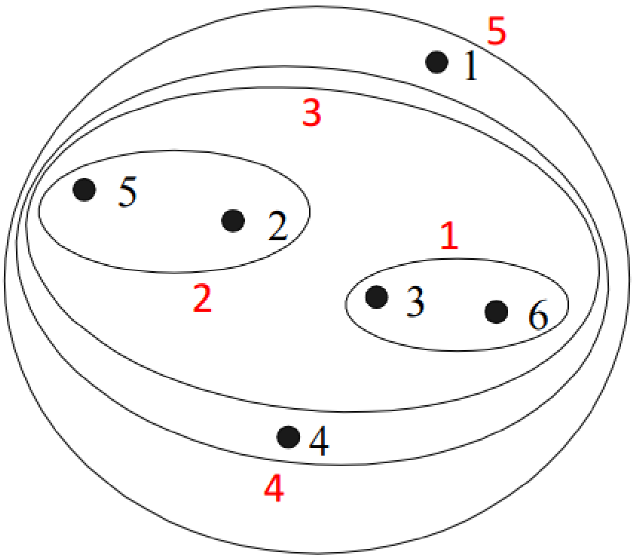
MIN or Single Link

- **Inter-cluster distance**
  - The distance between two clusters is represented by the distance of the *closest pair of data objects* belonging to different clusters.
  - Determined by one pair of points, i.e., by one link in the proximity graph

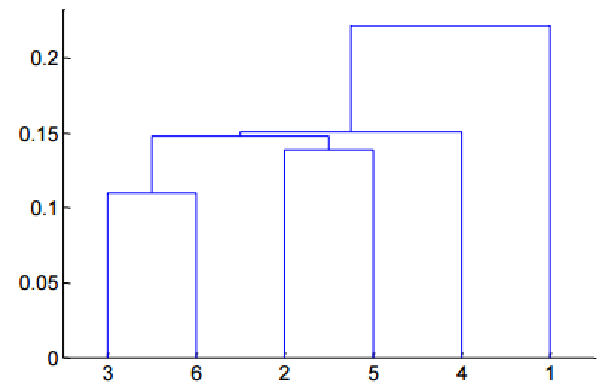


$$d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)$$

MIN



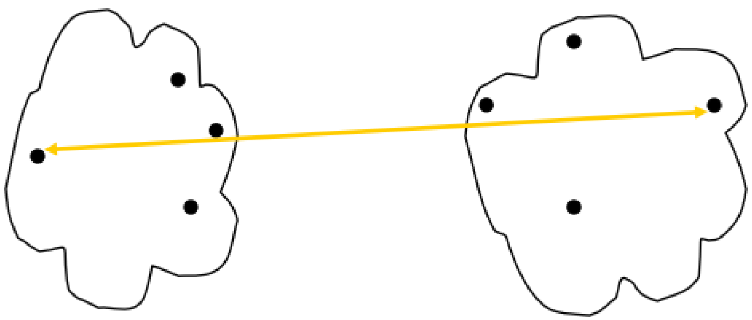
Nested Clusters



Dendrogram

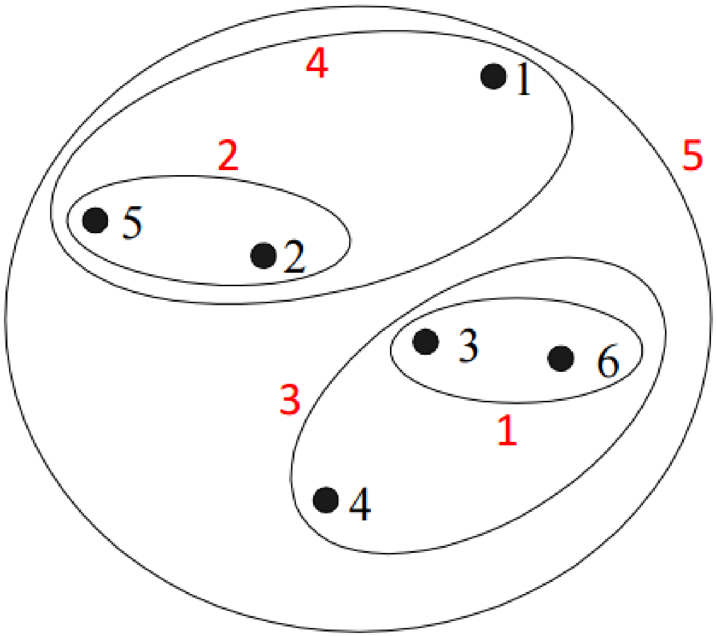
MAX or Complete Link

- **Inter-cluster distance**
  - The distance between two clusters is represented by the distance of the *farthest pair of data objects* belonging to different clusters

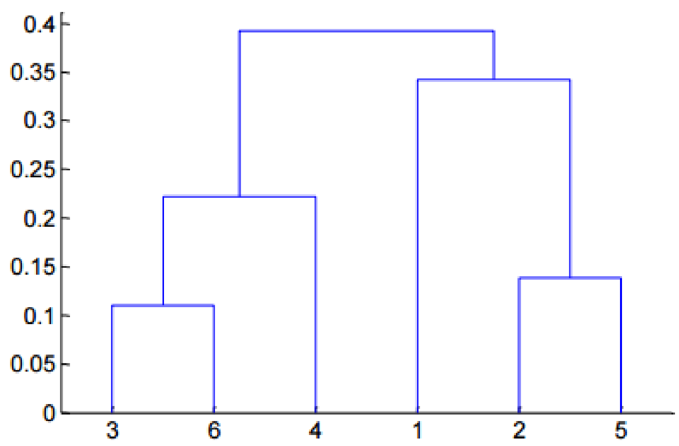


$$d_{\min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

MAX or Complete Link

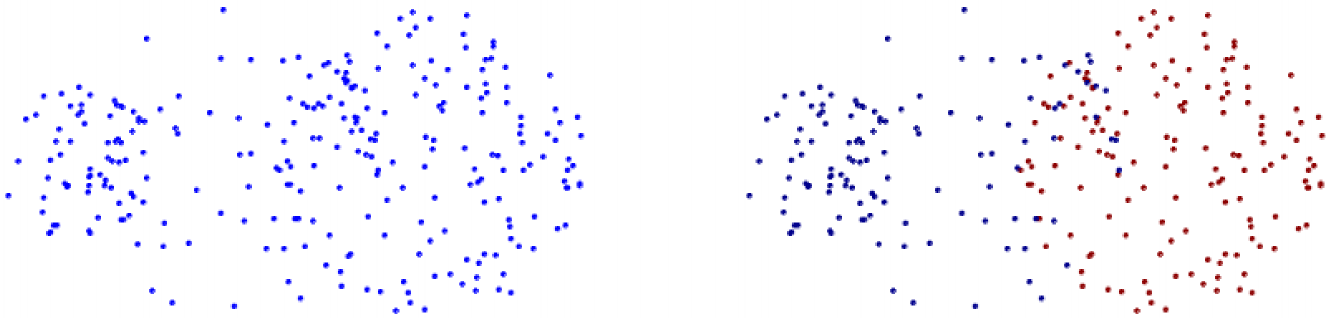


Nested Clusters



Dendrogram

Limitations of MIN

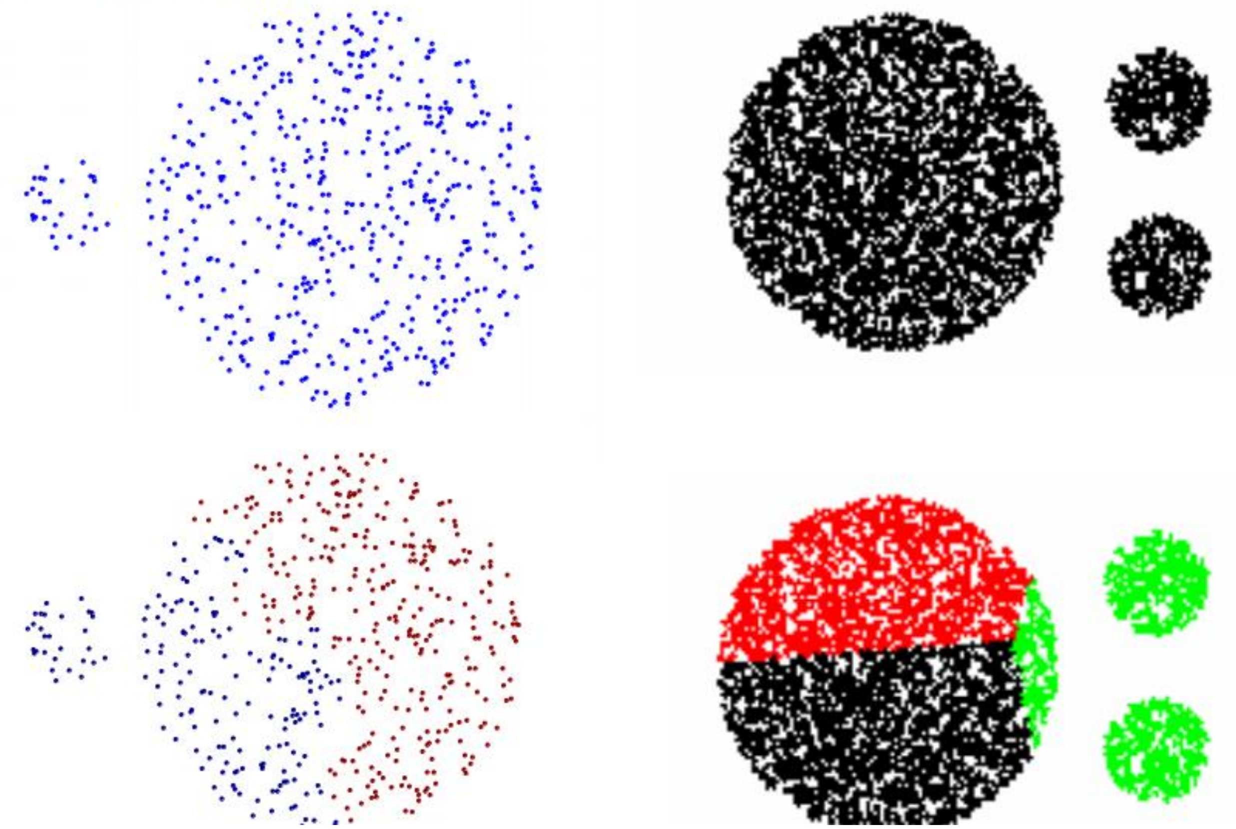


Original Points   Two Clusters

- Sensitive to noise and outliers

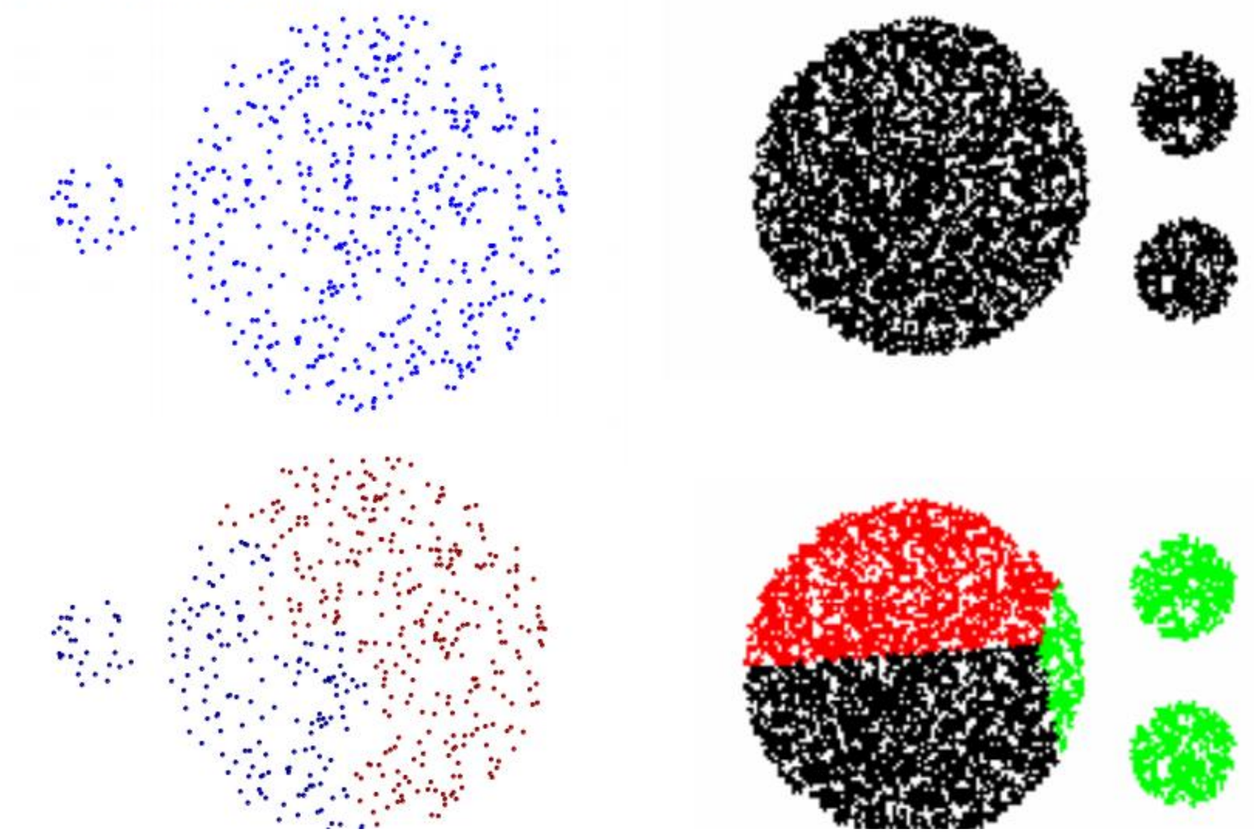
Limitations of MAX

- Tends to break large clusters



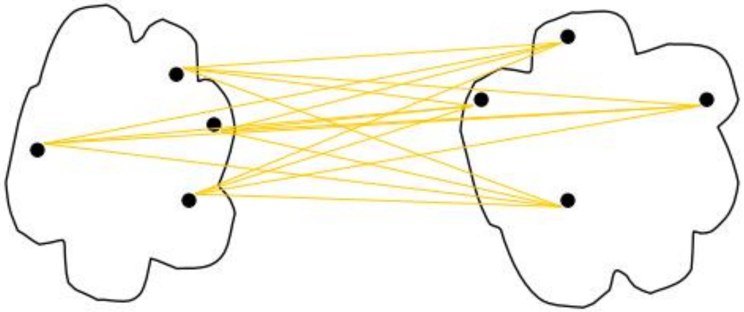
Limitations of MAX

- Tends to break large clusters



More Techniques with increased complexity

- **Inter-cluster distance**
  - The distance between two clusters is represented by the *average* distance of *all pairs of data objects* belonging to different clusters
  - Determined by all pairs of points in the two clusters



$$d_{\min}(C_i, C_j) = \underset{p \in C_i, q \in C_j}{avg} d(p,q)$$