Unsupervised Learning

Some slides were adapted/taken from various sources, including Prof. Andrew Ng's Coursera Lectures, Stanford University, Prof. Kilian Q. Weinberger's lectures on Machine Learning, Cornell University, Prof. Sudeshna Sarkar's Lecture on Machine Learning, IIT Kharagpur, Prof. Bing Liu's lecture, University of Illinois at Chicago (UIC), CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University, Dr. Luis Serrano, Prof. Alexander Ihler and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

Road map

- **Basic concepts**
- **K-means algorithm**
- **Representation of clusters**
- **Hierarchical clustering**
- **Distance functions**
- **Data standardization**
- **Handling mixed attributes**
- **Which clustering algorithm to use?**
- **Cluster evaluation**
- **Discovering holes and data regions**
- **Summary**

Supervised learning vs. unsupervised learning

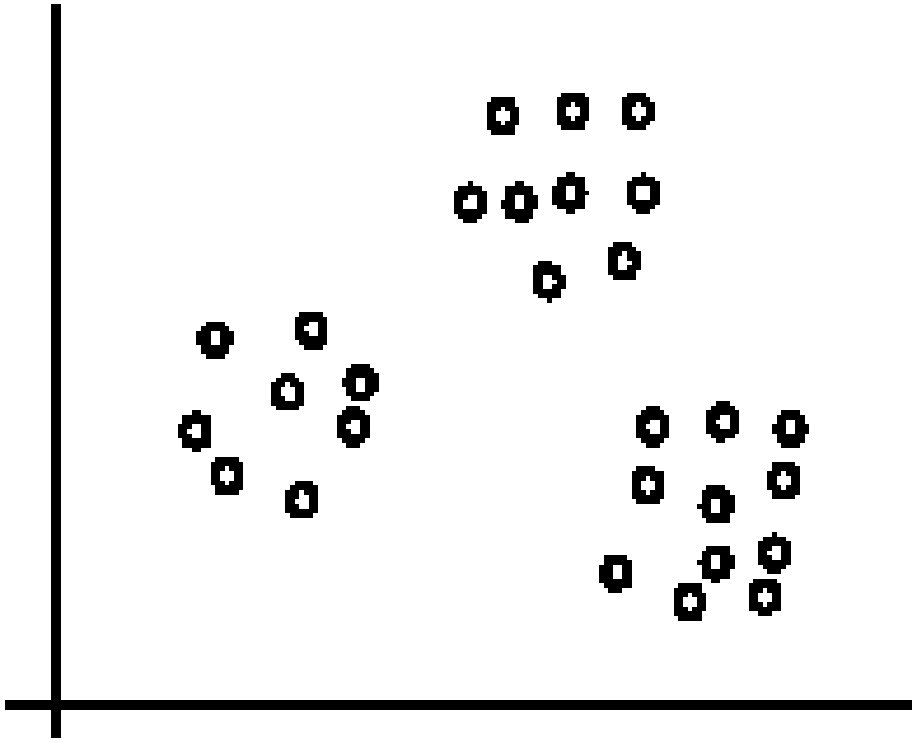
- Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.
  - These patterns are then utilized to predict the values of the target attribute in future data instances.
- Unsupervised learning: The data have no target attribute.
  - We want to explore the data to find some intrinsic structures in them.

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
  - In fact, association rule mining is also unsupervised
- This chapter focuses on clustering.

An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

What is clustering for? (cont…)

- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

Aspects of clustering

- A clustering algorithm
  - Partitional clustering
  - Hierarchical clustering
  - …
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance $\Rightarrow$ maximized
  - Intra-clusters distance $\Rightarrow$ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) D be

  $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$,

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters S = $\{S_1, S_2, \ldots, S_k\}$.
  - Each cluster has a cluster **center**, called **centroid** $(\boldsymbol{\mu_i})$. *k* is specified by the user

Optimization criterion

- WCSS (within cluster sum of square)

$$\sum_{i=1}^{k} \sum_{x \in S_i} ||x_i - \mu_i||^2$$

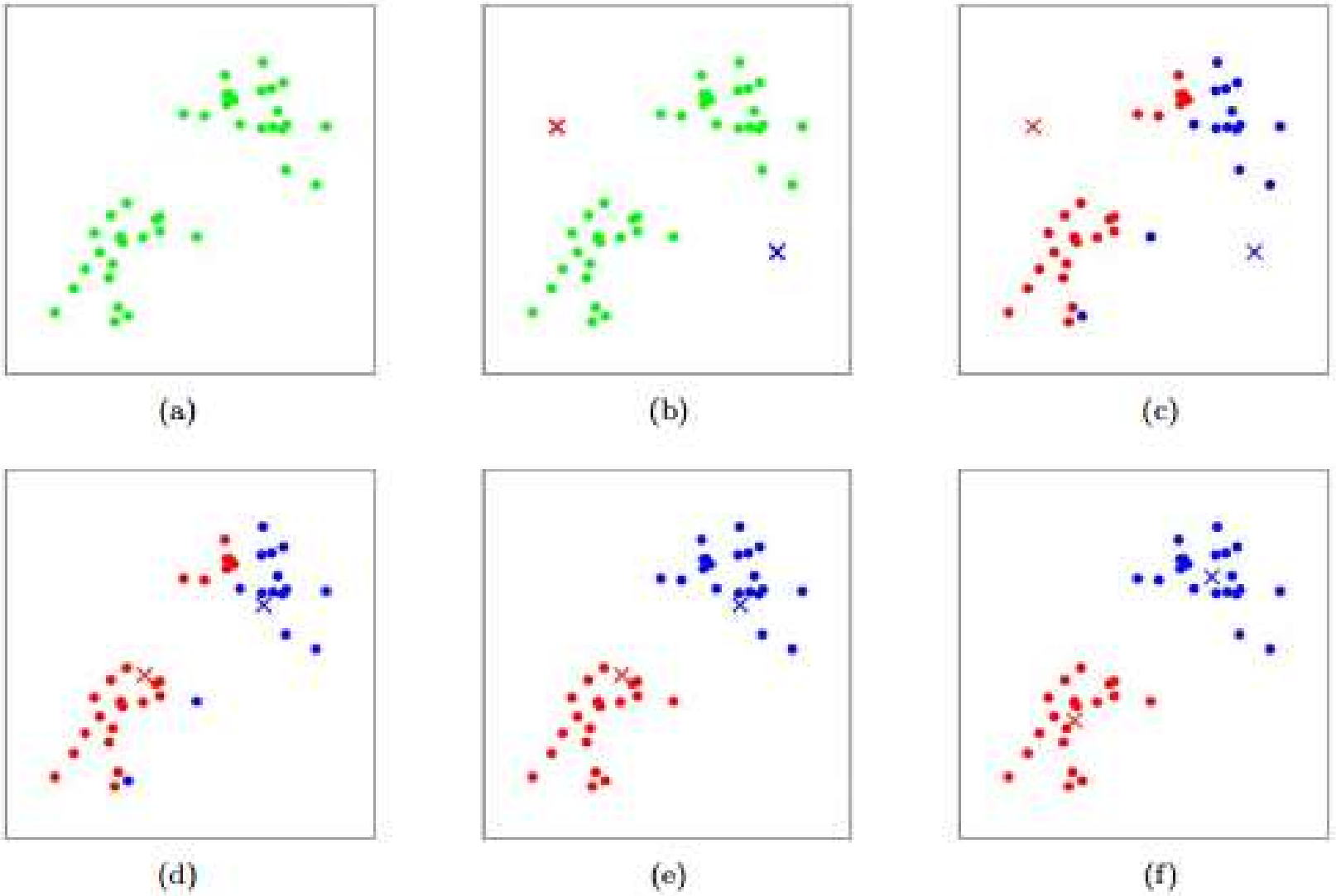
- Find out that set of clusters where WCSS is minimum i.e. optimization criteria is

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{x \in S_i} ||x_i - \mu_i||^2$$

K-means algorithm

Macqueen, 1967

- Given *k*, the *k-means (heuristic based)* algorithm works as follows:
  1) Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers
  2) Assign each data point to the closest centroid
  3) Re-compute the centroids using the current cluster memberships.
  4) If a convergence criterion is not met, goto step 2.


(a)
(b)
(c)
(d)
(e)
(f)

K-means algorithm – (cont …)

Algorithm *k*-means(*k*, *D*)

1 Choose *k* data points as the initial centroids (cluster centers)
2 **repeat**
3 **for** each data point $\mathbf{x} \in D$ do
4 compute the distance from **x** to each centroid;
5 assign **x** to the closest centroid // a centroid represents a cluster
6 **endfor**
7 re-compute the centroids using the current cluster memberships
8 **until** the stopping criterion is met

Stopping/convergence criterion

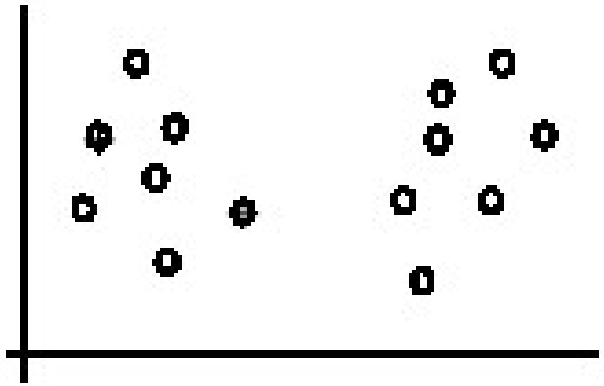
1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{i=1}^{k} \sum_{x \in S_i} ||x_i - \mu_i||^2 \qquad (1)$$

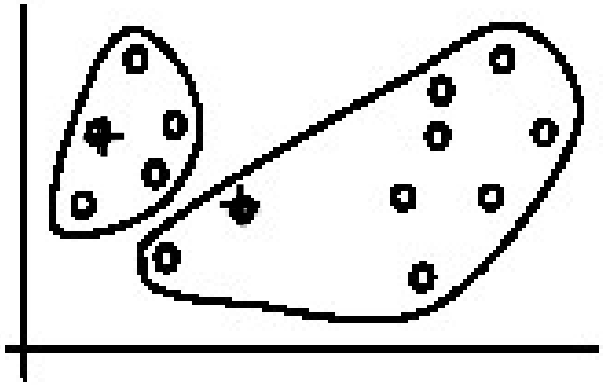
where S_i is the *i*th cluster, μ_i is the centroid of cluster S_i (the mean vector of all the data points in S_i), and *dist*$(x_i, \mu_i) = ||x_i - \mu_i||^2$ is the distance between data point **x** and centroid S_i.

- The **sum of squared error** (SSE) is keep on decreasing in subsequent iterations.
- The process is bound to converge.
- It will converge to local minima of within cluster sum of square distance which may or may not meet the global minima.

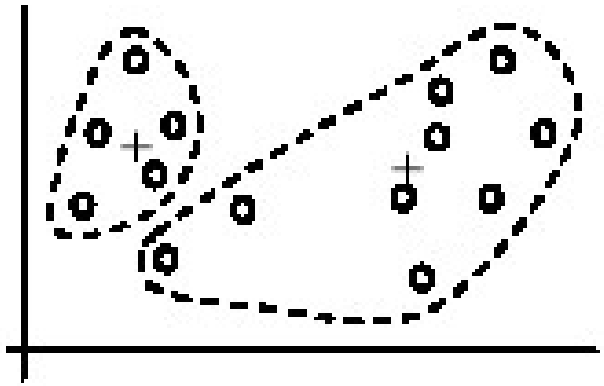
An example



(A). Random selection of k centers

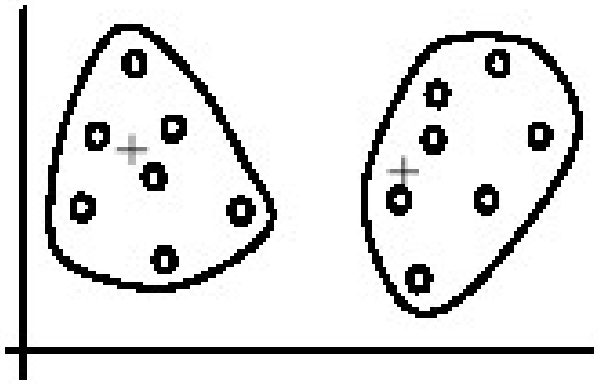


Iteration 1: (B). Cluster assignment

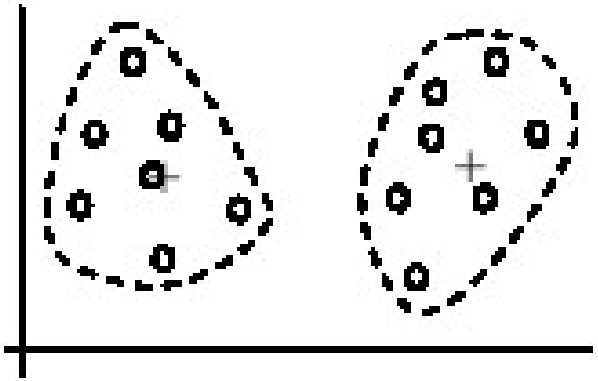


(C). Re-compute centroids

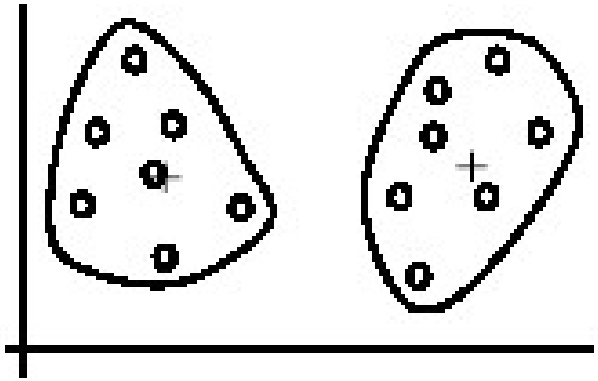
An example (cont …)



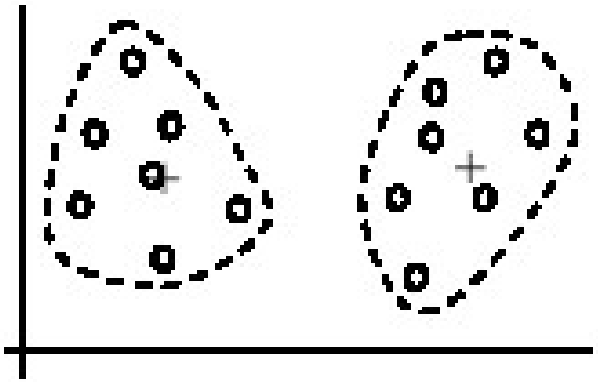
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

An example distance function

- Minkowski Distance:

$$d(x_i, x_j) = \left(\sum_{s=1}^{n} |x_{is} - x_{js}|^p\right)^{\frac{1}{p}}$$

- Euclidean Distance (p=2):

$$\sqrt{\sum_{s=1}^{n} |x_{is} - x_{js}|^2}$$

- Manhattan (City-Block) Distance (p=1):

$$\sum_{s=1}^{n} |x_{is} - x_{js}|$$

An example distance function

- Mahalanobis Distance: Dissimilarity measure between two random vectors $x_i \; and \; x_j$ of the same distribution with the covariance matrix Σ

$$d(x_i, x_i) = \sqrt{(x_i - x_j)\Sigma^{-1}(x_i - x_j)}$$

- Pearson correlation:

$$r(x_i, x_j) = \frac{Cov(x_i, x_j)}{\sigma_{x_i}\sigma_{x_j}}$$

Computational Complexity

- Computing distance between two items is O(n) where n is the dimensionality of the vectors.
- Reassigning clusters: O(km) distance computations, or O(kmn)
- Computing centroids:Each item gets added once to some centroid: O(mn)
- Assume these two steps are each done once for each t iterations: O(tknm)

Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,

    where n is the number of data points,

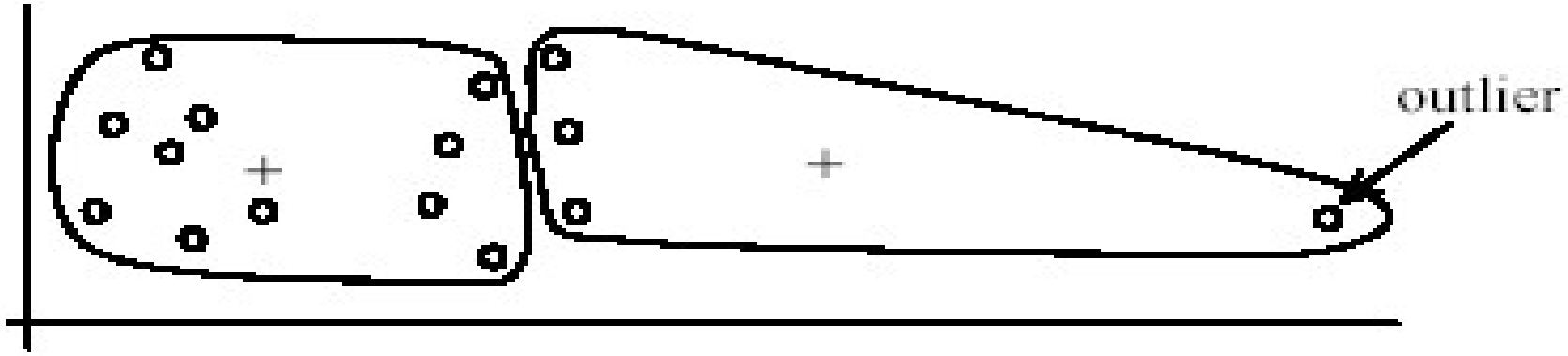
    k is the number of clusters, and

    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

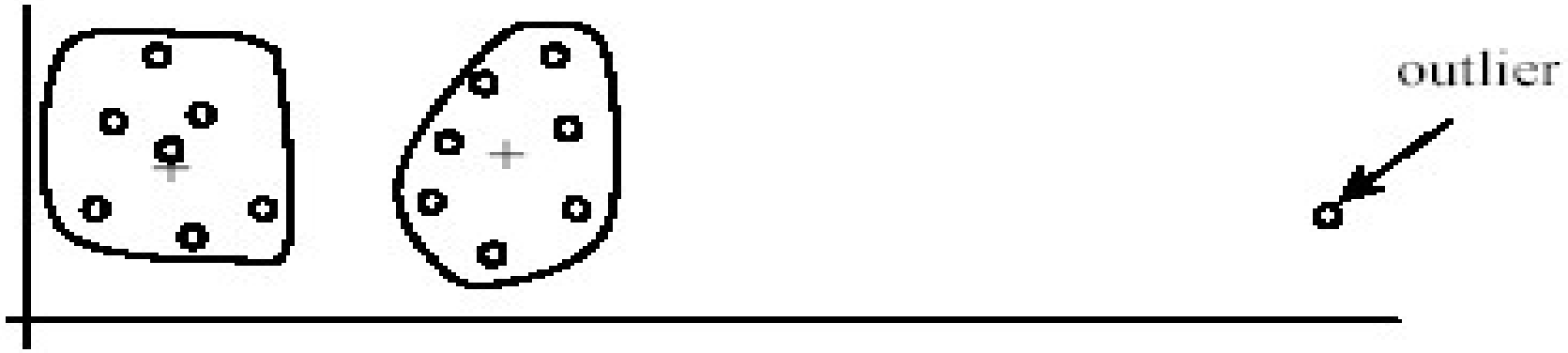
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



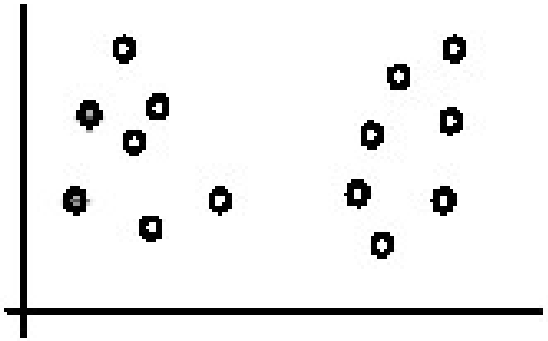
(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

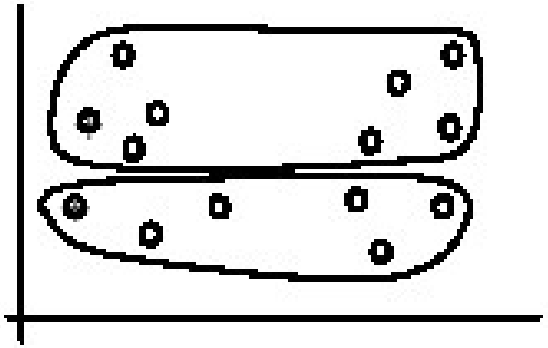
- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

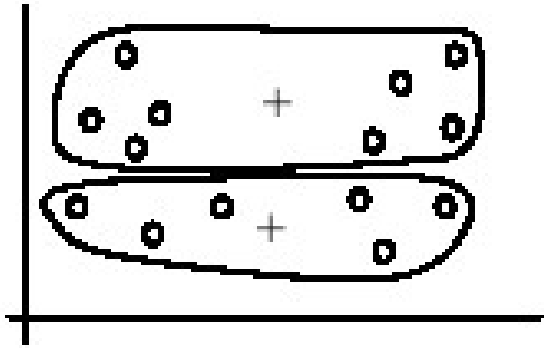
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



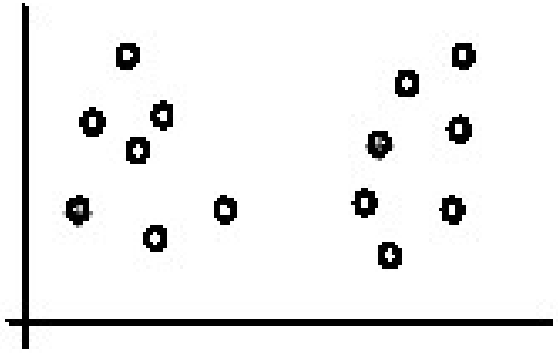
(B). Iteration 1



(C). Iteration 2

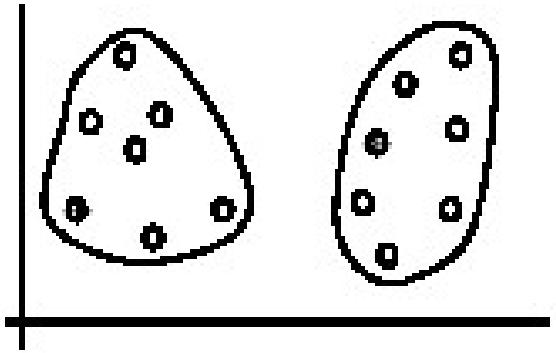
Weaknesses of k-means (cont …)

- If we use different seeds: good results

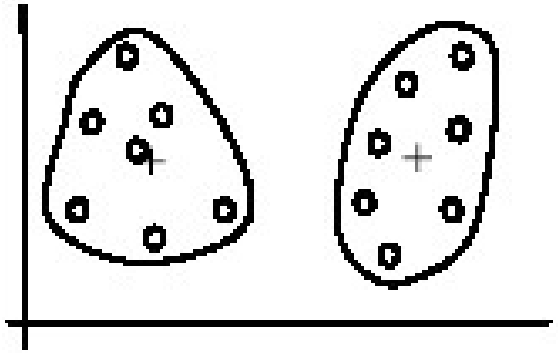


(A). Random selection of k seeds (centroids)

- There are some methods to help choose good seeds



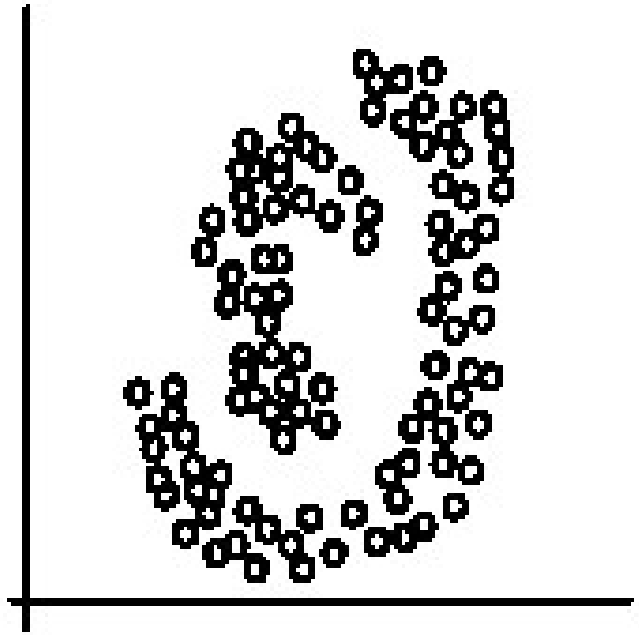
(B). Iteration 1



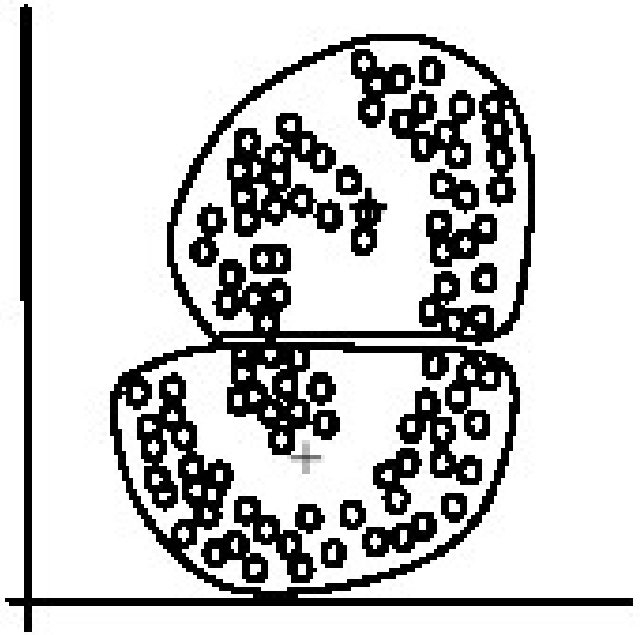
(C). Iteration 2

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): *k*-means clusters

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity, efficiency and
  - other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - although they may be more suitable for some specific types of data or applications.
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Probability Density Estimation
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions
- Summary

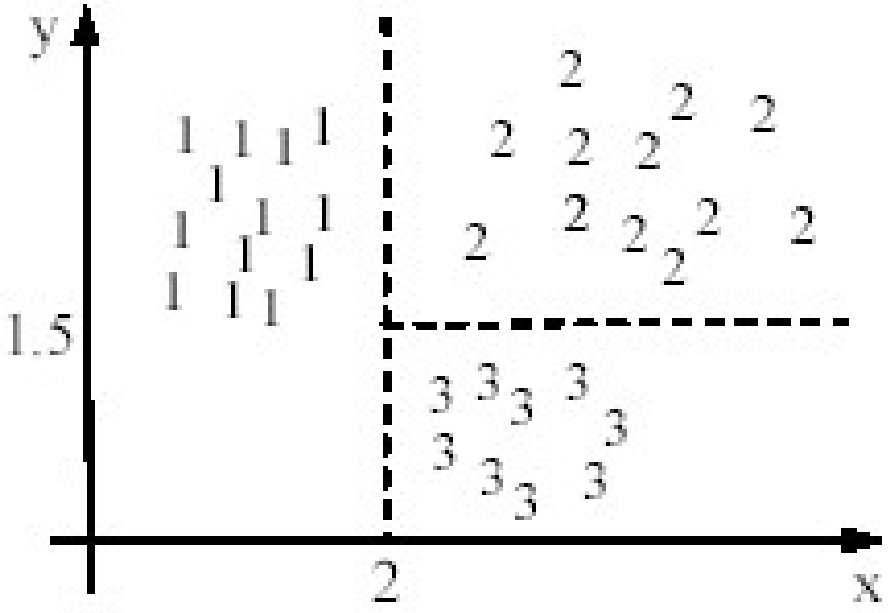
Common ways to represent clusters

- Use the centroid of each cluster to represent the cluster.
  - compute the radius and
  - standard deviation of the cluster to determine its spread in each dimension

  - The centroid representation alone works well if the clusters are of the hyper-spherical shape.
  - If clusters are elongated or are of other shapes, centroids are not sufficient

Using classification model

- All the data points in a cluster are regarded to have the same class label, e.g., the cluster ID.
  - run a supervised learning algorithm on the data to find a classification model.



$x \leq 2 \rightarrow$ cluster 1

$x > 2, y > 1.5 \rightarrow$ cluster 2

$x > 2, y \leq 1.5 \rightarrow$ cluster 3

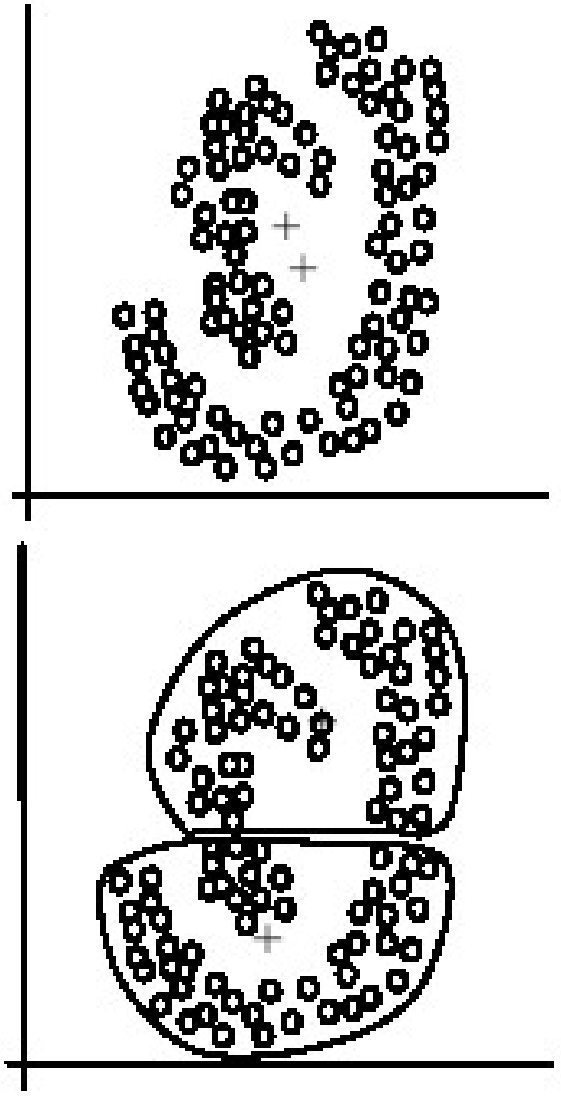
Use frequent values to represent cluster

- This method is mainly for clustering of categorical data (e.g., *k*-modes clustering).

- Main method used in text clustering, where a small set of frequent words in each cluster is selected to represent the cluster.

Clusters of arbitrary shapes

- Hyper-elliptical and hyper-spherical clusters are usually easy to represent, using their centroid together with spreads.
- Irregular shape clusters are hard to represent. They may not be useful in some applications.
  - Using centroids are not suitable (upper figure) in general
  - K-means clusters may be more useful (lower figure), e.g., for making 2 size T-shirts.



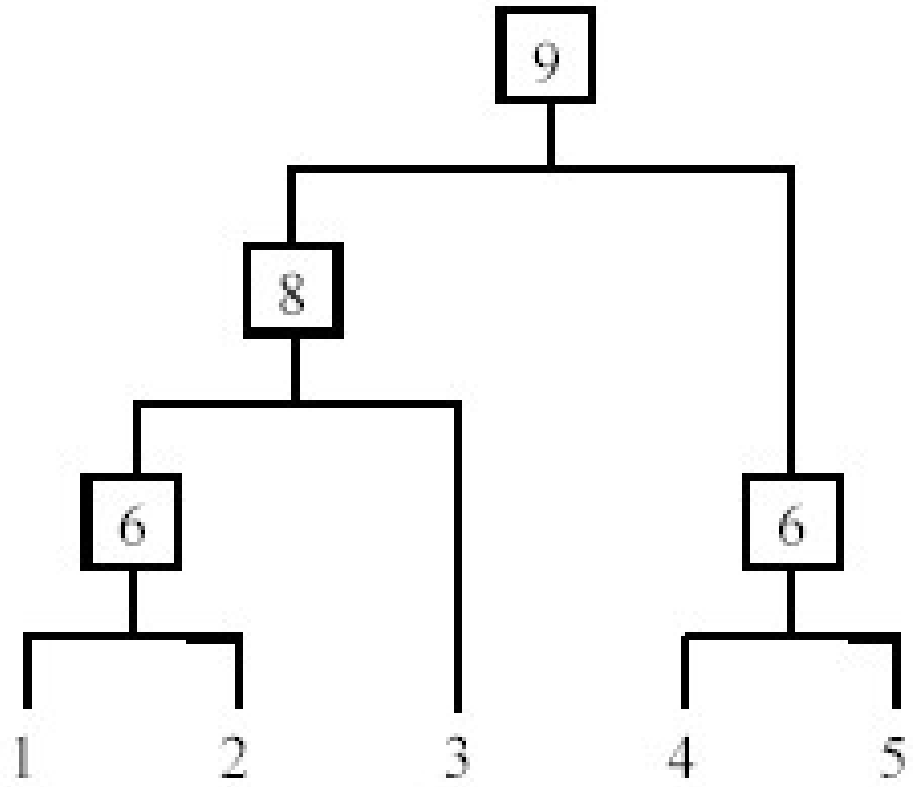
Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Probability Density Estimation
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions

- Summary

Hierarchical Clustering

- Produce a nested sequence of clusters, a tree, also called Dendrogram.



Types of hierarchical clustering

- Agglomerative (bottom up) clustering: It builds the dendrogram (tree) from the bottom level, and
  - merges the most similar (or nearest) pair of clusters
  - stops when all the data points are merged into a single cluster (i.e., the root cluster).
- Divisive (top down) clustering: It starts with all data points in one cluster, the root.
  - Splits the root into a set of child clusters. Each child cluster is recursively divided further
  - stops when only singleton clusters of individual data points remain, i.e., each cluster with only a single point

Agglomerative clustering

It is more popular then divisive methods.

- At the beginning, each data point forms a cluster (also called a node).
- Merge nodes/clusters that have the least distance.
- Go on merging
- Eventually all nodes belong to one cluster

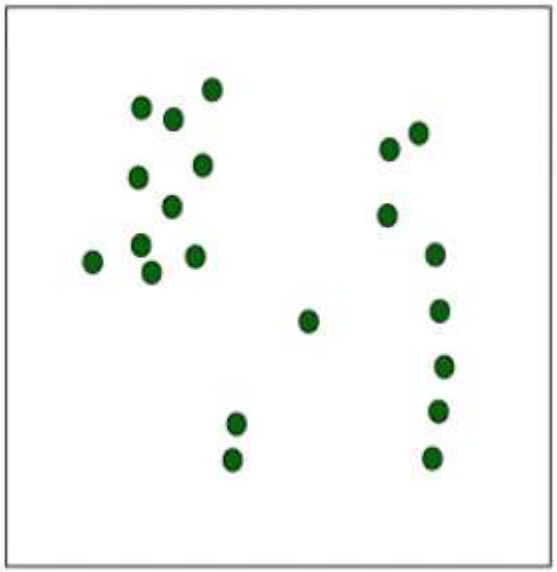
Agglomerative clustering algorithm

Algorithm Agglomerative(D)

1 Make each data point in the data set D a cluster,
2 Compute all pair-wise distances of $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in D$;
2 **repeat**
3 find two clusters that are nearest to each other;
4 merge the two clusters form a new cluster c;
5 compute the distance from c to all other clusters;
12 **until** there is only one cluster left

Visualization

Data:

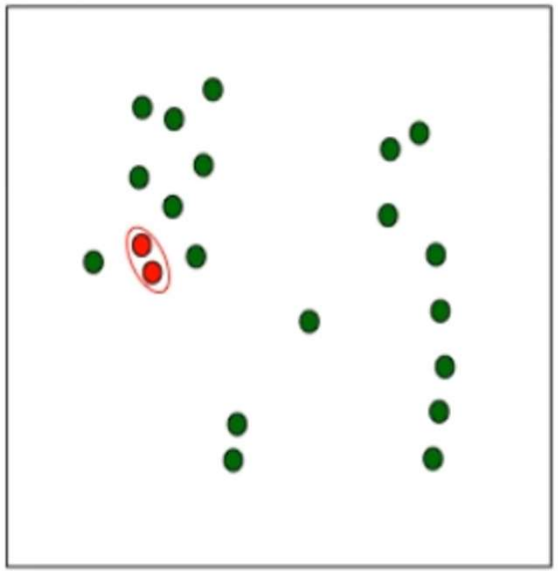


Initially every datum is a cluster.

- A simple algorithm.
- Define a distance (or dissimilarity) between clusters.
- Initialize: every sample is a cluster.
- Iterate:
  - Compute distance between all clusters. Complexity **$O(m^2 \log m)$**
  - Merge two closest clusters.
- Save both clustering and sequence of cluster operations
- A data structure called "**Dendrogram**" is used.

Visualization

Iteration 1



Buildup a sequence of clusters.

Compute distance between all clusters. Find the clusters with smallest distance.

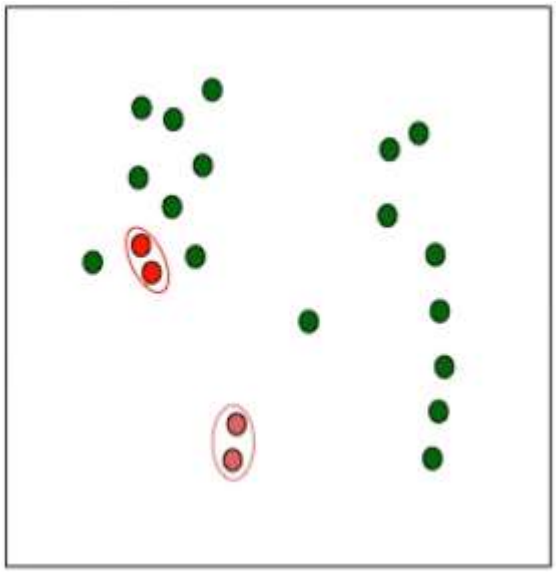
Dendrogram:

Height of the join indicates dissimilarity

Complexity **$O(m^2 \log m) + O(m \log m)$**

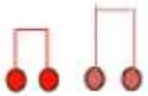
Visualization

Iteration 2



Again, compute distance between all clusters pair including newly formed cluster. Find the cluster pair with smallest distance.

Dendrogram:

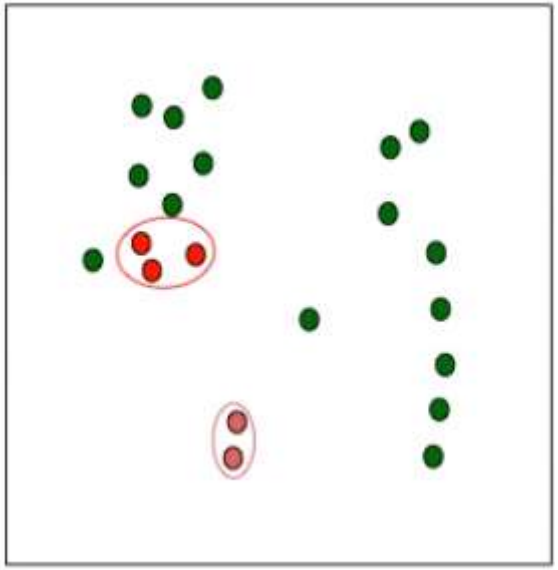


Height of the join indicates dissimilarity

Complexity **$O(m^2 \log m) + 2*O(m \log m)$**

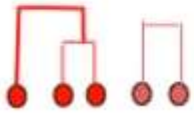
Visualization

Iteration 3



Repeat the same steps to find closest cluster pair.

Dendrogram:

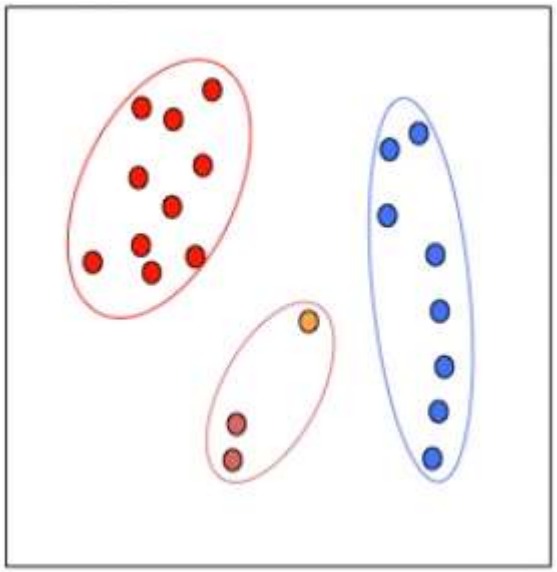


Height of the join indicates dissimilarity

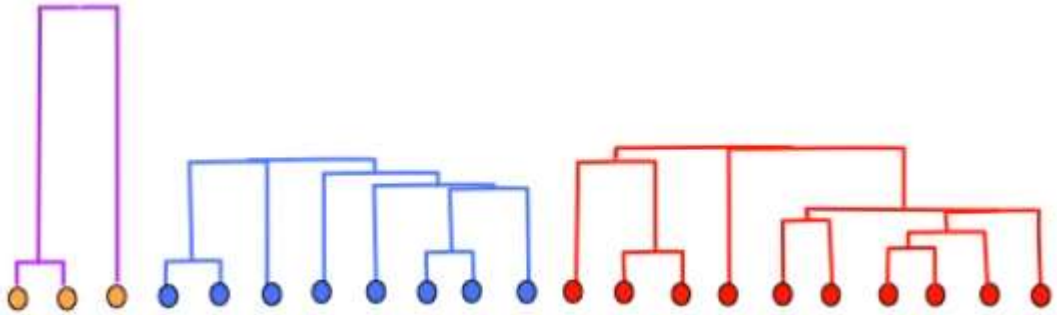
Complexity **$O(m^2 \log m) + 3*O(m \log m)$**

Visualization

Iteration m-3



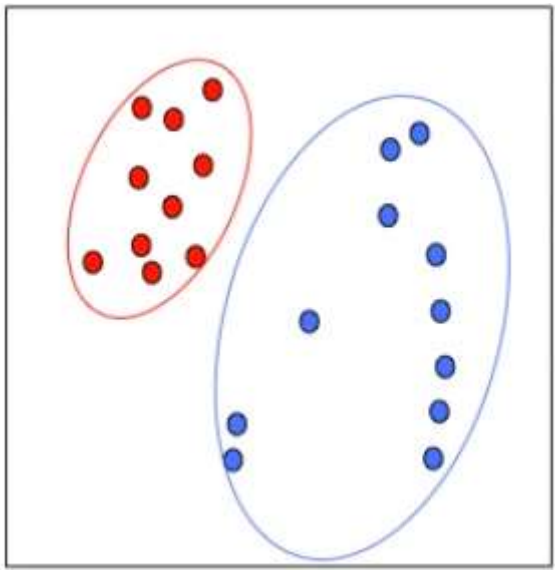
Dendrogram

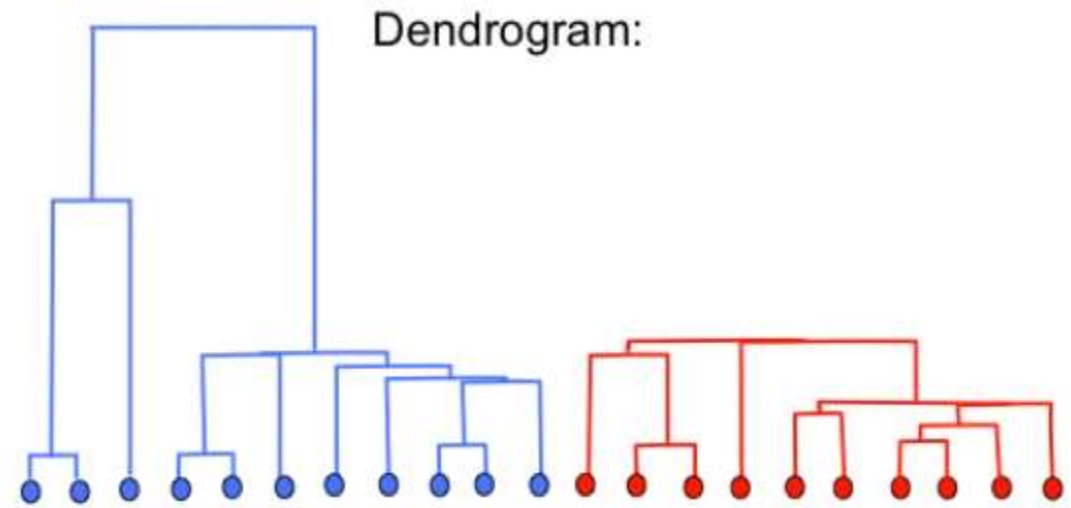


Complexity **$O(m^2 \log m) + (m-3)*O(m \log m)$**

Visualization

Iteration m-2

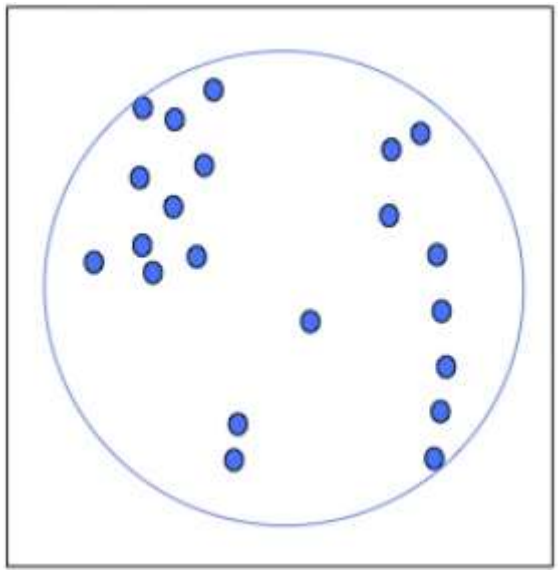


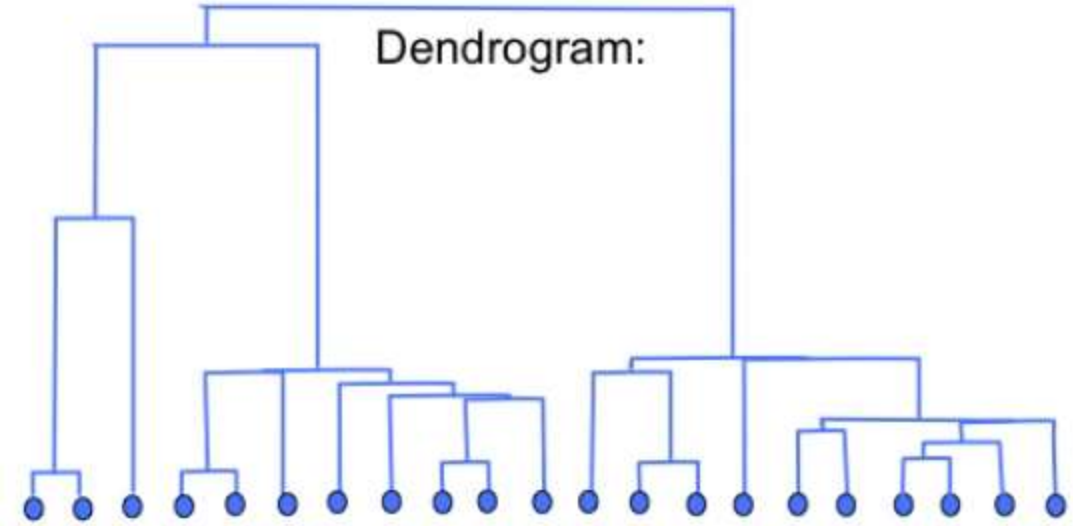


Complexity $\mathbf{O(m^2 \log m) + (m-2)*O(m \log m)}$

Visualization

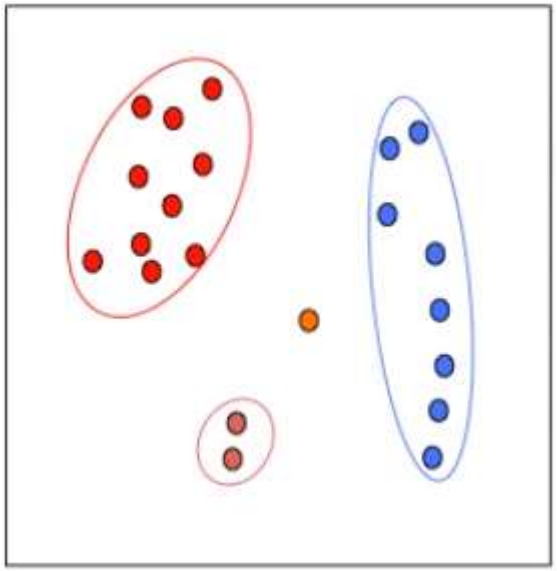
Iteration m-1

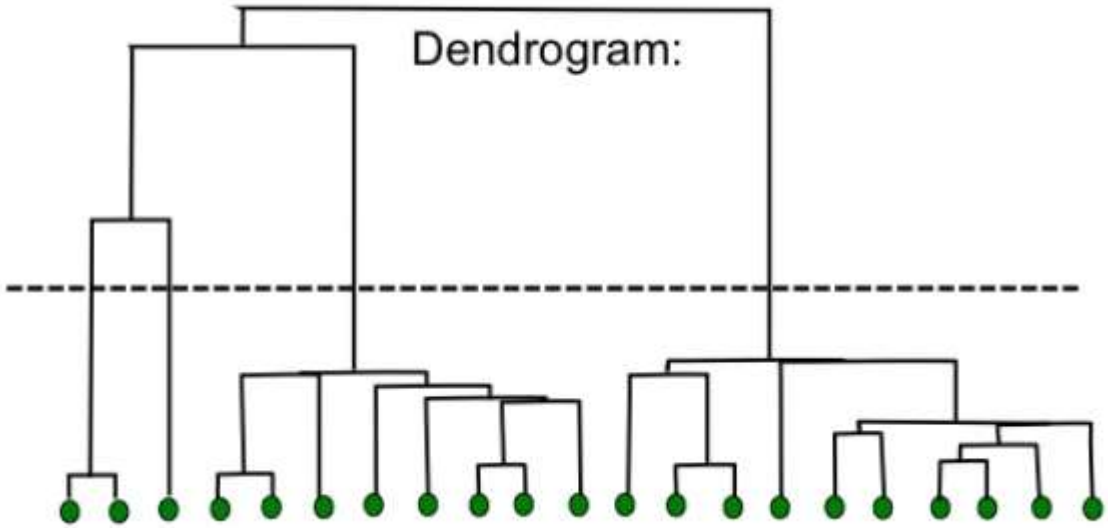




Complexity $\mathbf{O(m^2 \log m) + (m-1)*O(m \log m) = O(m^2 \log m)}$

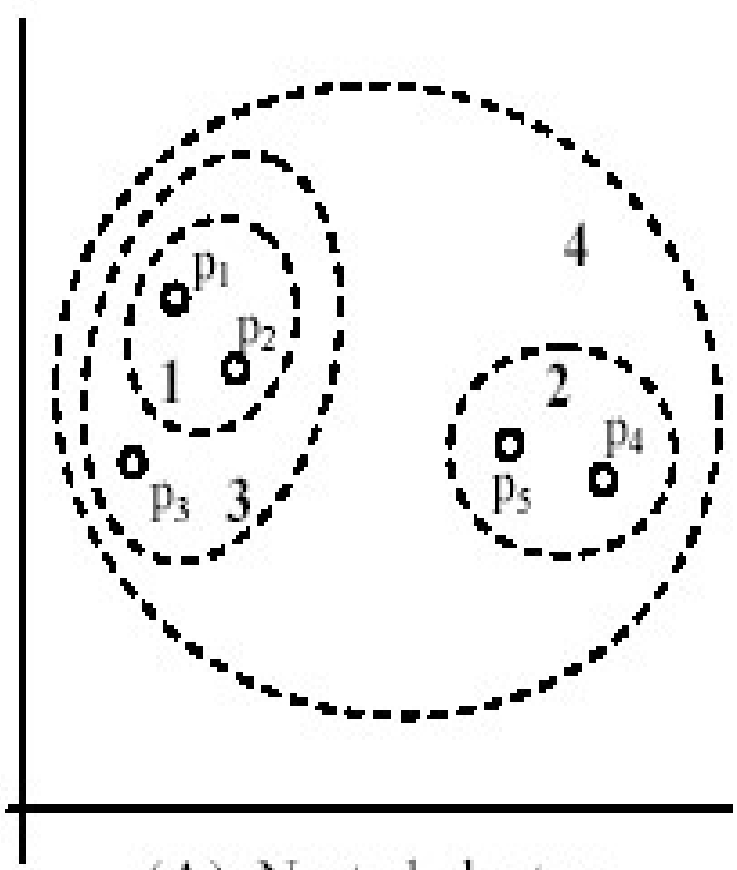
Visualization



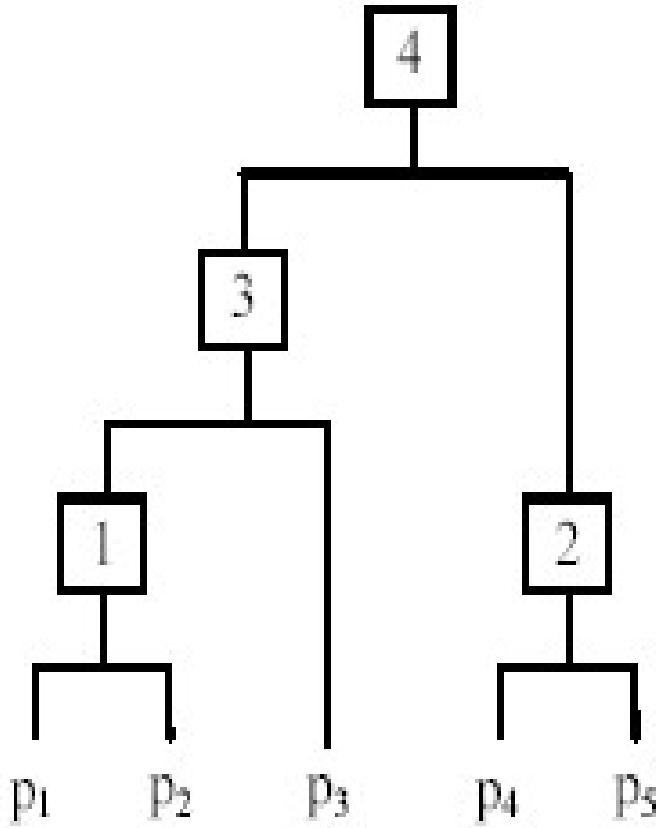


Given the sequence, can select a number of clusters or a dissimilarity threshold

An example: working of the algorithm



(A). Nested clusters



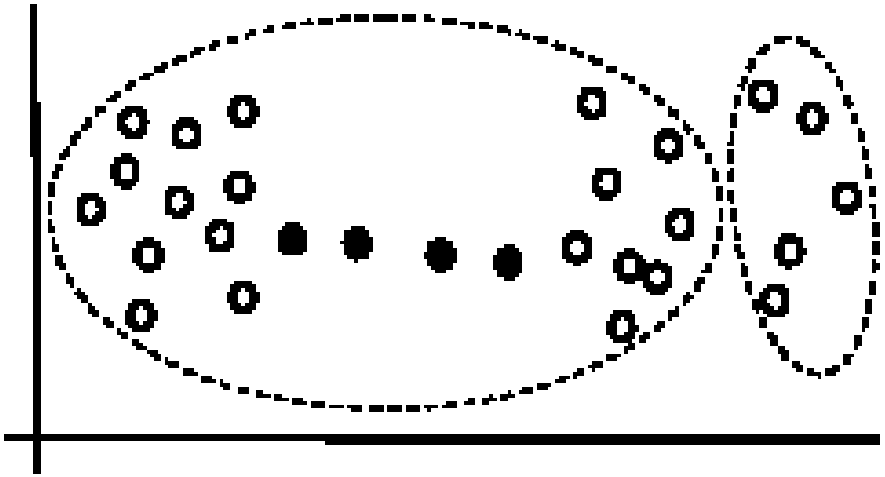
(B) Dendrogram

Measuring the distance of two clusters

- A few ways to measure distances of two clusters.
- Results in different variations of the algorithm.
  - Single link
  - Complete link
  - Average link
  - Centroids
  - …

Single link method

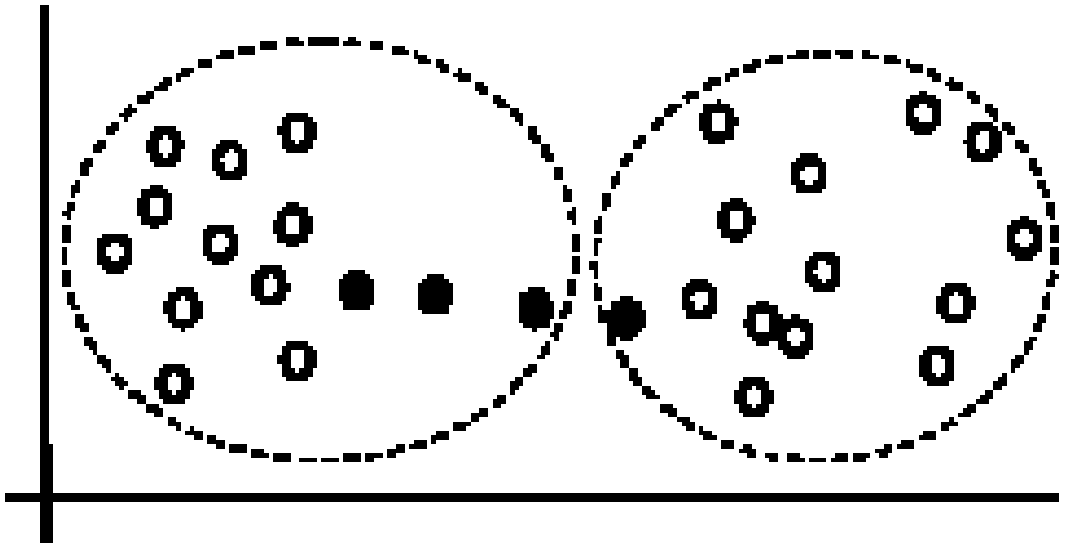
- The distance between two clusters is the distance between two closest data points in the two clusters, one data point from each cluster.
- It can find arbitrarily shaped clusters, but
  - It may cause the undesirable "chain effect" by noisy points



Two natural clusters are split into two

Complete link method

- The distance between two clusters is the distance of two furthest data points in the two clusters.
- It is sensitive to outliers because they are far away



Average link and centroid methods

- Average link: A compromise between
  - the sensitivity of complete-link clustering to outliers and
  - the tendency of single-link clustering to form long chains that do not correspond to the intuitive notion of clusters as compact, spherical objects.
  - In this method, the distance between two clusters is the average distance of all pair-wise distances between the data points in two clusters.
- Centroid method: In this method, the distance between two clusters is the distance between their centroids

Cluster Distances

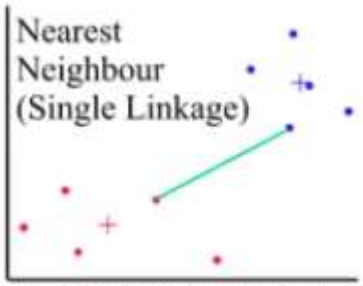
$$D_{\min}(C_i, C_j) = \min_{x \in C_i,\ y \in C_j} \|x - y\|^2$$

$$D_{\max}(C_i, C_j) = \max_{x \in C_i,\ y \in C_j} \|x - y\|^2$$

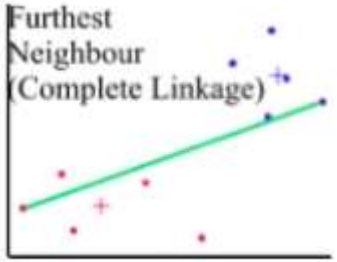
$$D_{\text{avg}}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i,\ y \in C_j} \|x - y\|^2$$

$$D_{\text{means}}(C_i, C_j) = \|\mu_i - \mu_j\|^2$$

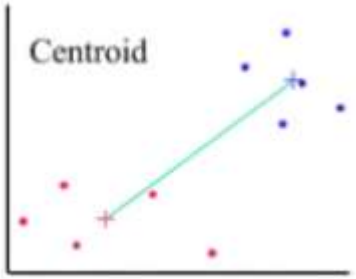
Need:
D(A,C) → D(A+B,C)
D(B,C) →



produces minimal spanning tree.

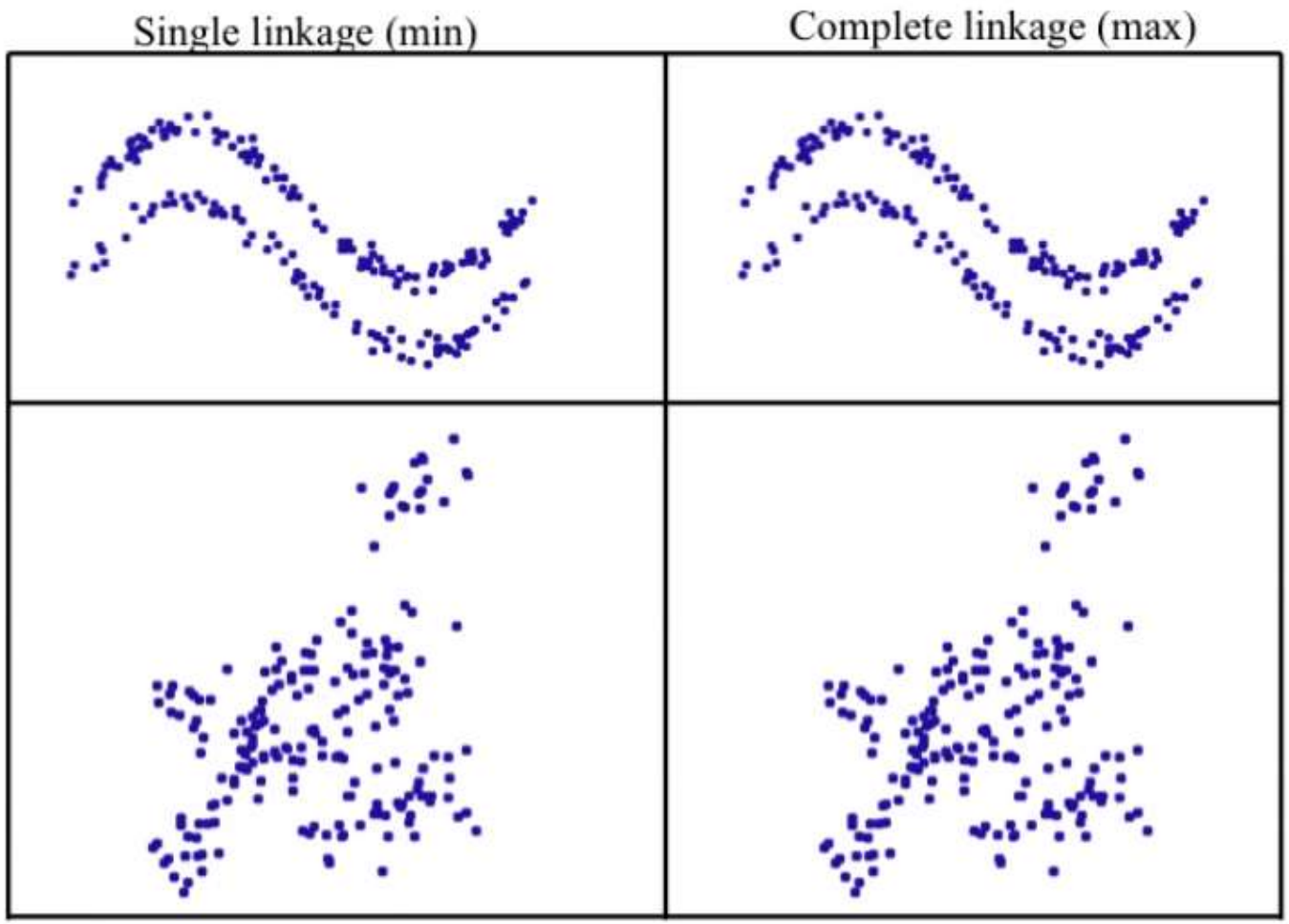


avoids elongated clusters.



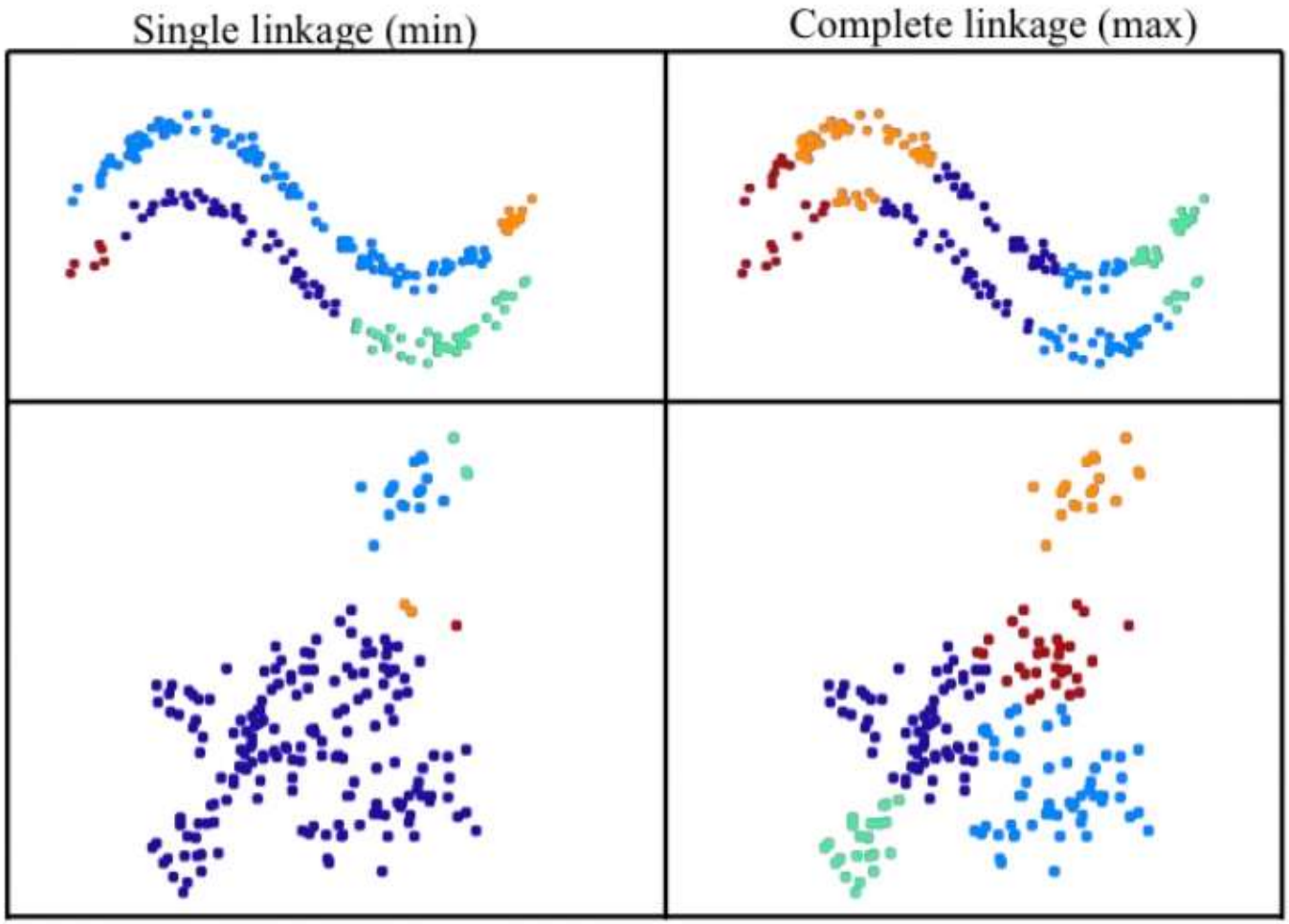
Cluster Distance

Dissimilarity choice will affect the clusters created



Cluster Distance

Dissimilarity choice will affect the clusters created



The complexity

- All the algorithms are at least $O(n^2)$. n is the number of data points.
- Single link can be done in $O(n^2)$.
- Complete and average links can be done in $O(n^2 \log n)$.
- Due the complexity, hard to use for large data sets.
  - Sampling
  - Scale-up methods (e.g., BIRCH).

Road map

- Basic concepts
- K-means algorithm
- Representation of clusters
- Hierarchical clustering
- Probability Density Estimation
- Distance functions
- Data standardization
- Handling mixed attributes
- Which clustering algorithm to use?
- Cluster evaluation
- Discovering holes and data regions

- Summary

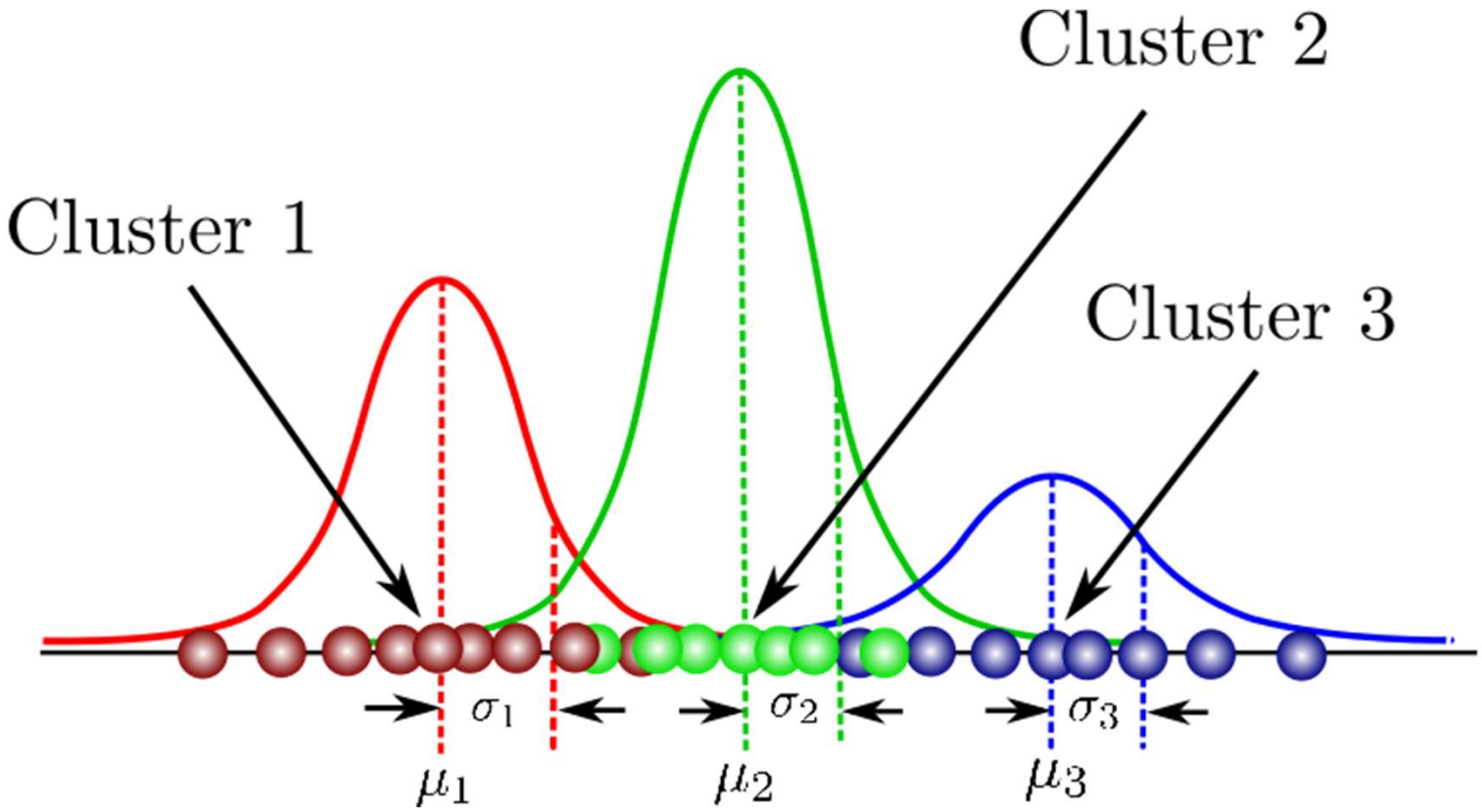
Probability Density Estimation

- Gaussian Mixture Model
- Expectation Maximization

Why Mixture of Gaussian

- Gaussian is in general more closer to the natural distributions and mathematically is more easy to handle (series of differentiations etc.).
- In most cases, specially Mahalanobis Distance, Covariance matrix etc., it has been assumed that the underline data distribution is Gaussian in nature. But, in practical scenario, it may not always necessarily true.
- Sometimes, a dataset with complicated data distribution can be approximated by a weighted summation of multiple Gaussian distributions (popularly called mixture of Gaussian).
- We can cluster the data in several components/parts and assume that each of such components follows Gaussian distribution.

Gaussian Mixture Model



Univariate Gaussian Distribution

- Gaussian or normal distribution, 1D

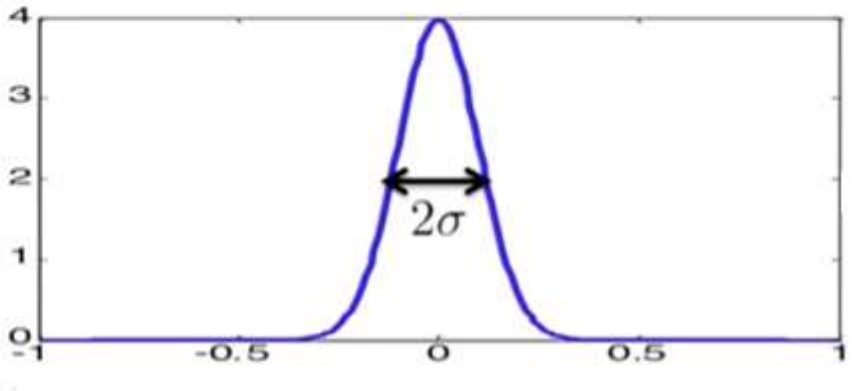
$$\mathcal{N}(x\ ;\ \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}(x-\mu)^2/\sigma^2\right]$$

  - Parameters: mean μ, variance σ^2
    (standard deviation σ)

Maximum Likelihood estimates

$$\hat{\mu} = \frac{1}{N}\sum_i x^{(i)}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_i (x^{(i)} - \hat{\mu})^2$$

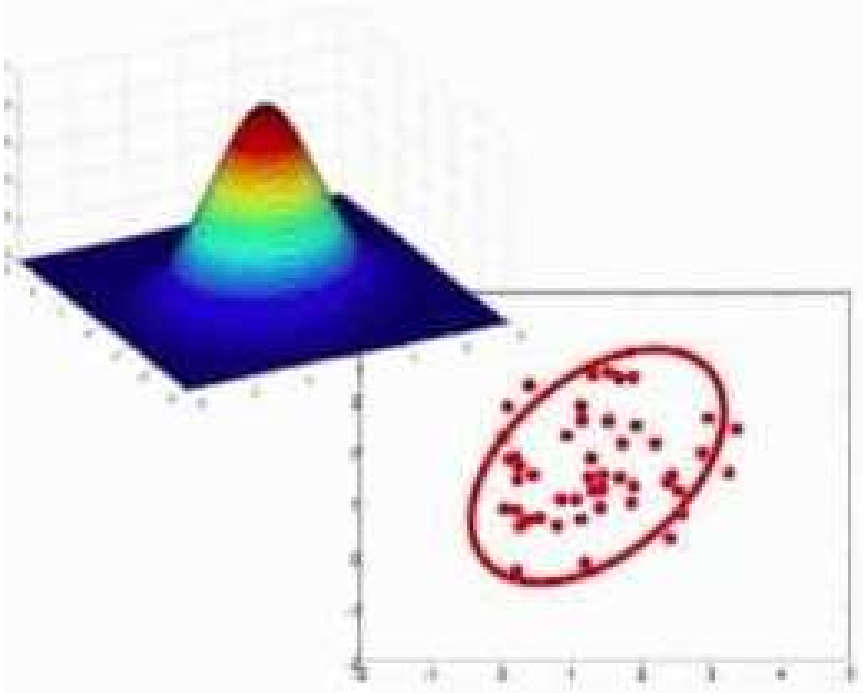


Slide Credit: Alexander Ihler

Multivariate Gaussian Distribution

- Similar to univariate case

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



$\underline{\mu}$ = length-d row vector
Σ = d x d matrix

$|\Sigma|$ = matrix determinant

Maximum likelihood estimate:

$$\hat{\underline{\mu}} = \frac{1}{m} \sum_j \underline{x}^{(j)}$$

$$\hat{\Sigma} = \frac{1}{m} \sum_j (\underline{x}^{(j)} - \hat{\underline{\mu}})^T (\underline{x}^{(j)} - \hat{\underline{\mu}})$$

(average of dxd matrices)

Slide Credit: Alexander Ihler

Goal:

- We want to estimate μ and Σ of a distribution.
- Method: Maximum Likelihood Estimation (MLE)

ML method for estimating parameters

- Consider the log of Gaussian distribution

$$\ln p(x \mid \mu, \Sigma) = -\frac{1}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma| - \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$$

- Take the derivatives and equate them to zero

$$\frac{\partial \ln p(x \mid \mu, \Sigma)}{\partial \mu} = 0 \qquad \frac{\partial \ln p(x \mid \mu, \Sigma)}{\partial \Sigma} = 0$$

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad \Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N} (x_n - \mu_{ML})(x_n - \mu_{ML})^T$$

where N is no. of data samples

Gaussian Mixture

- Linear superposition of Gaussian

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

Number of Gaussians

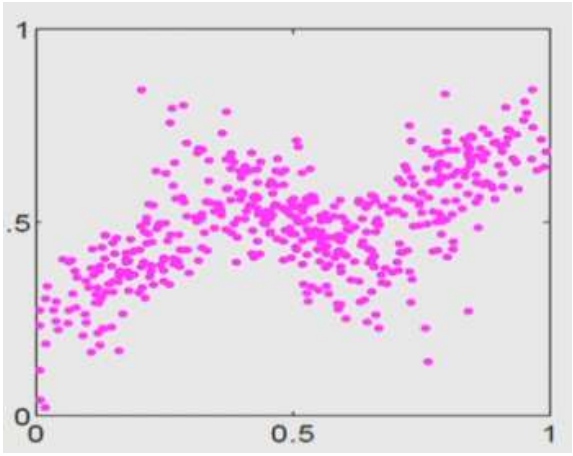
Mixing coefficient: weightage for each Gaussian dist.

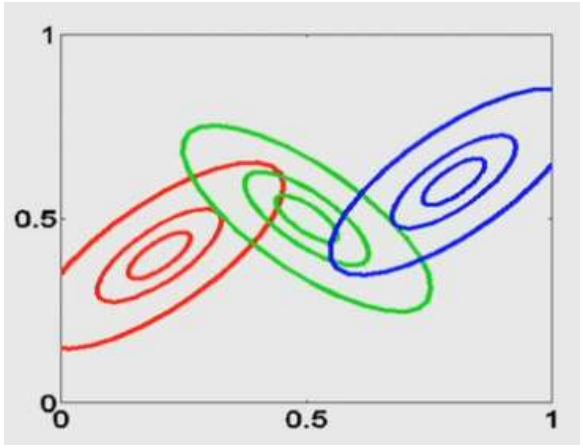
where $0 \le \pi_k \le 1, \quad \sum_{k=1}^{K} \pi_k = 1$

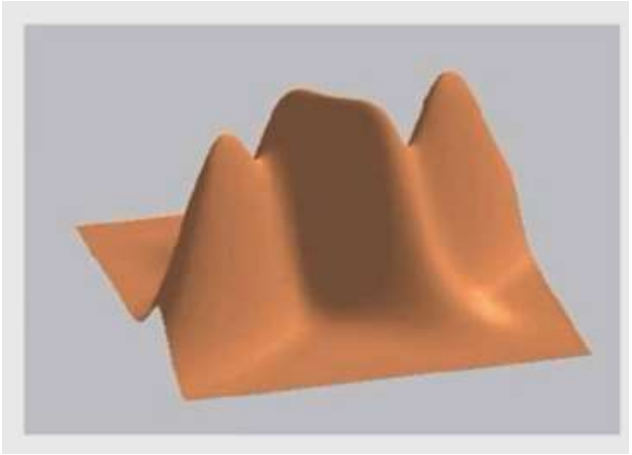
- Log likelihood of the overall function: $\ln p(X|\mu, \Sigma, \pi) =$

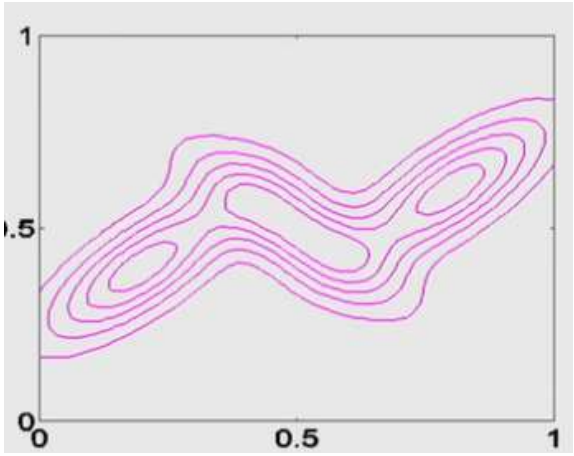
$$\sum_{n=1}^{N} \ln p(x_n) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k) \right\}$$

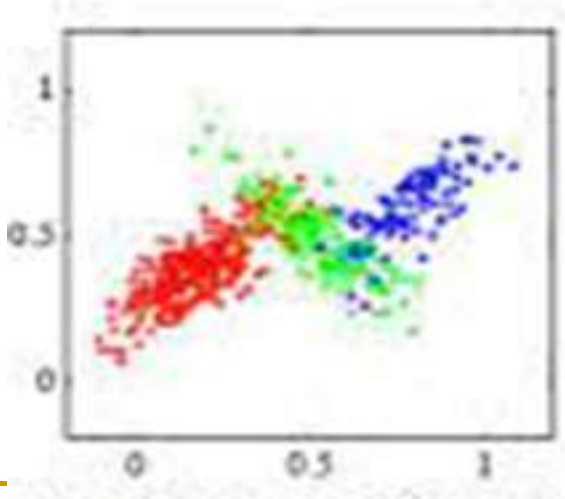
Example: GMM











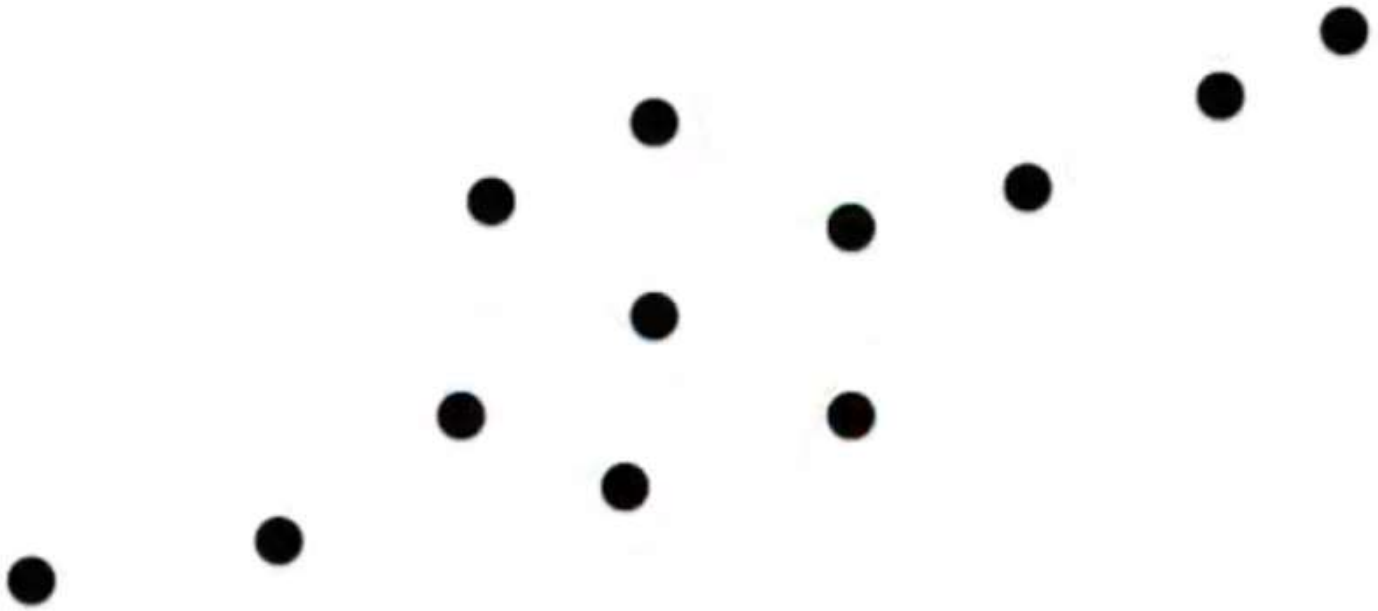
Problem

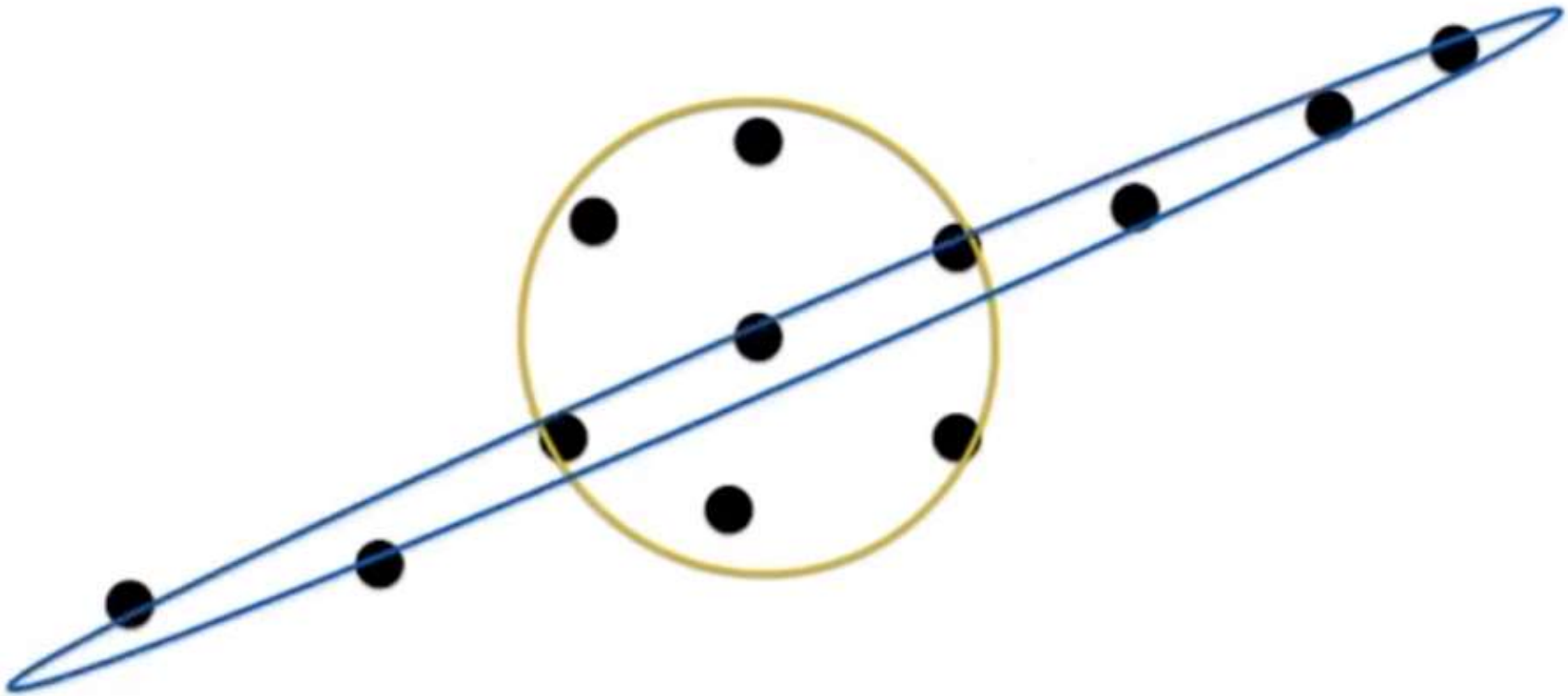
- ML is not suitable here as it does not provide any closed form solution.

- Parameters can be calculated using
  - Expectation Maximization (EM) technique

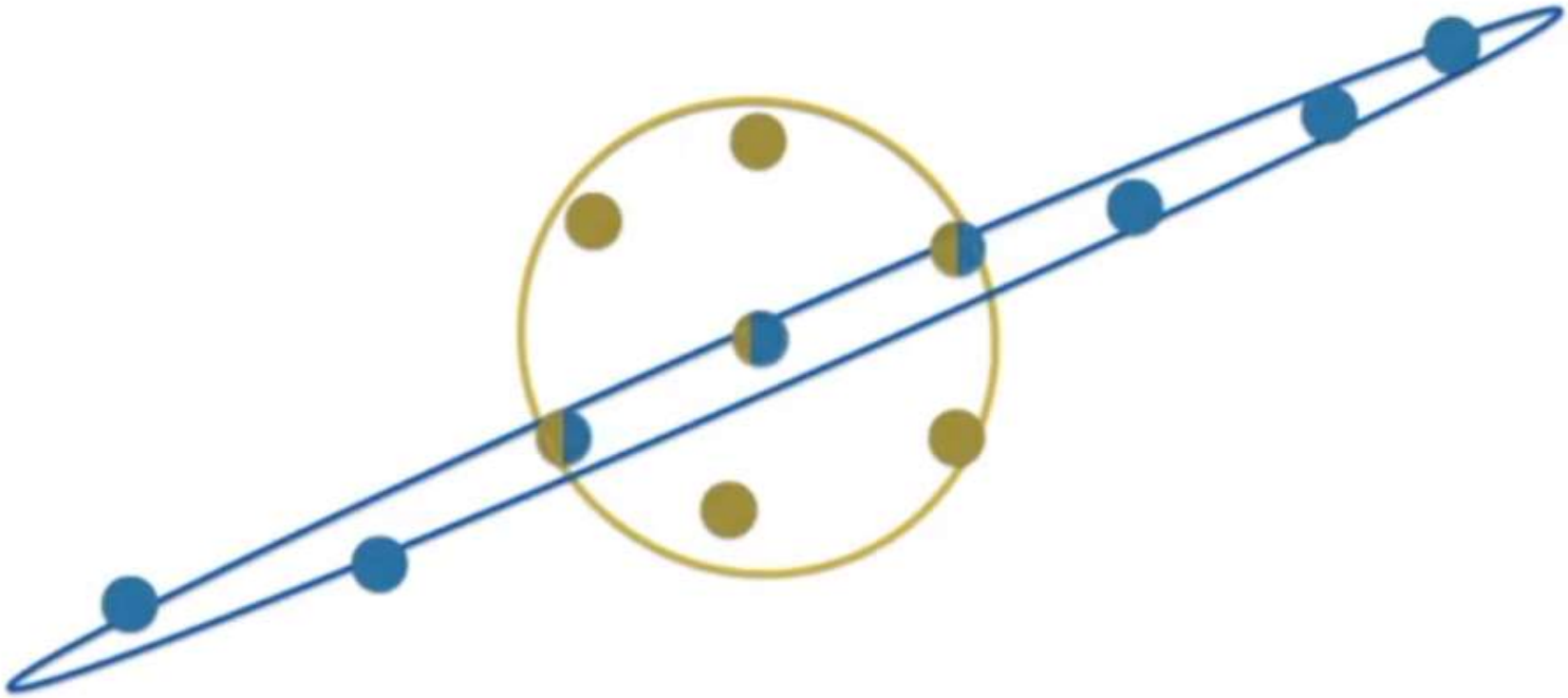
Hard and Soft Clustering

- Hard Clustering
  - Every data point is belong to only one cluster.
- But it is possible that a data point may be belong to multiple clusters
  - Soft assignments

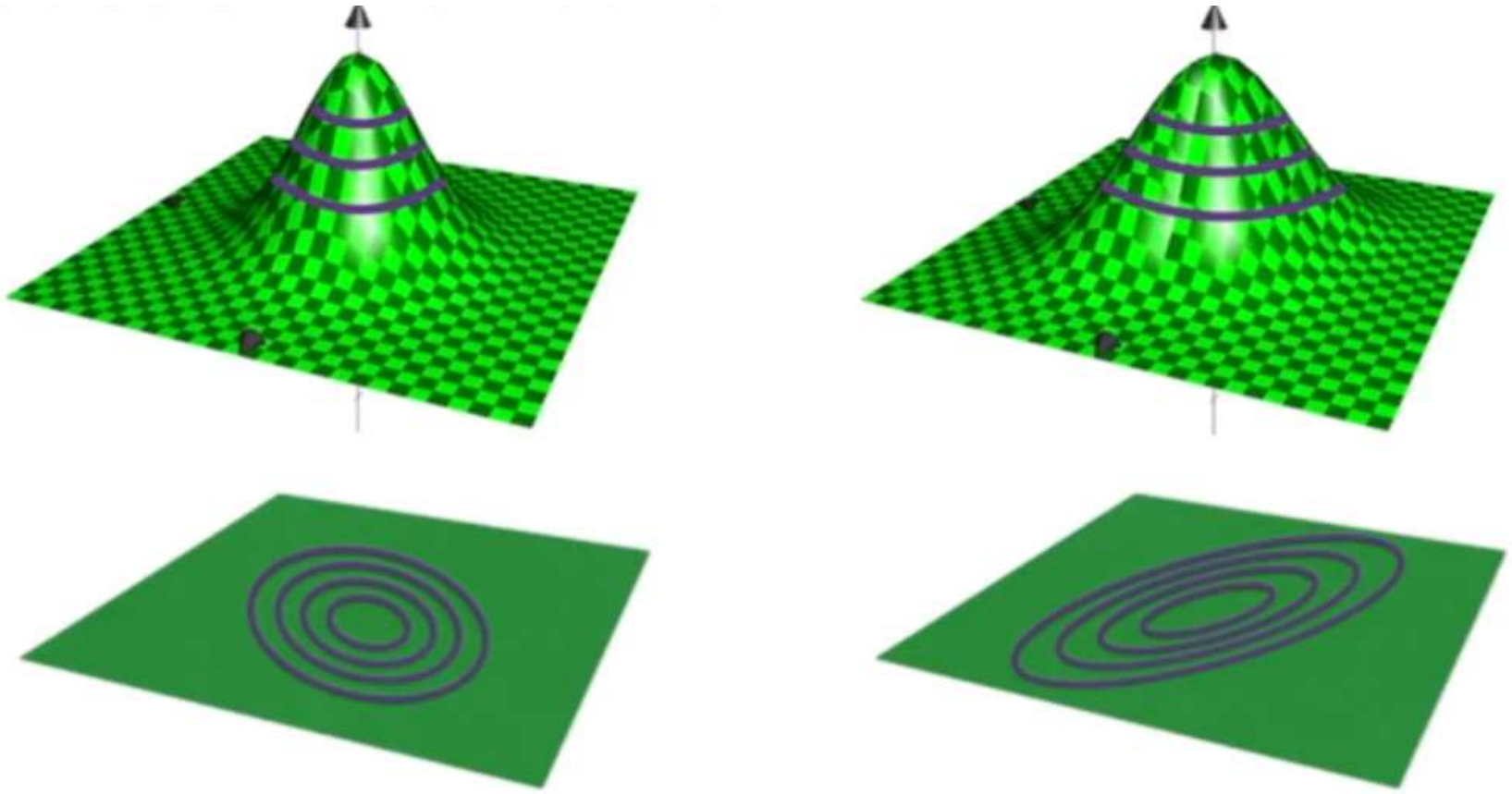




Soft Clustering

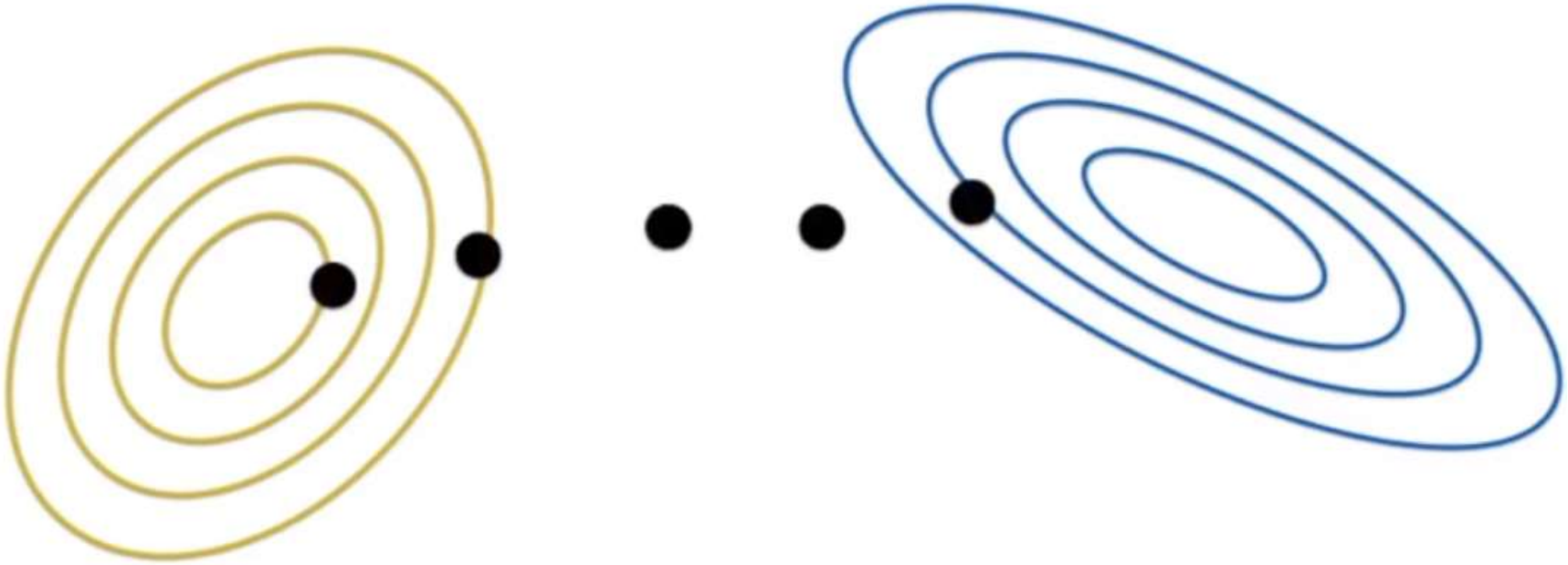


Gaussian Distribution



Soft Clustering

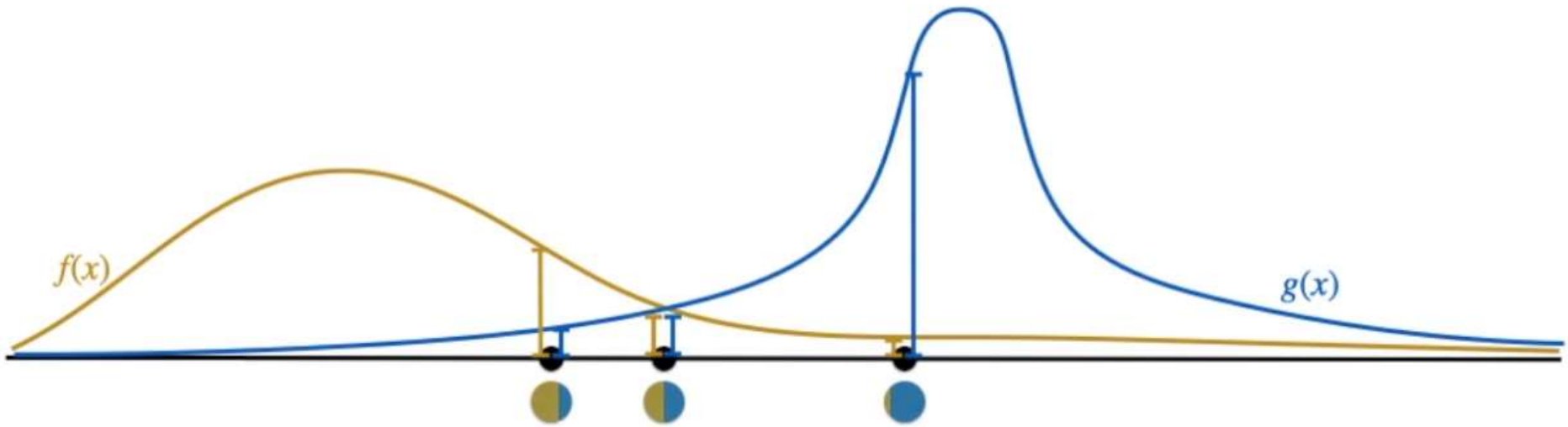
Soft Clustering



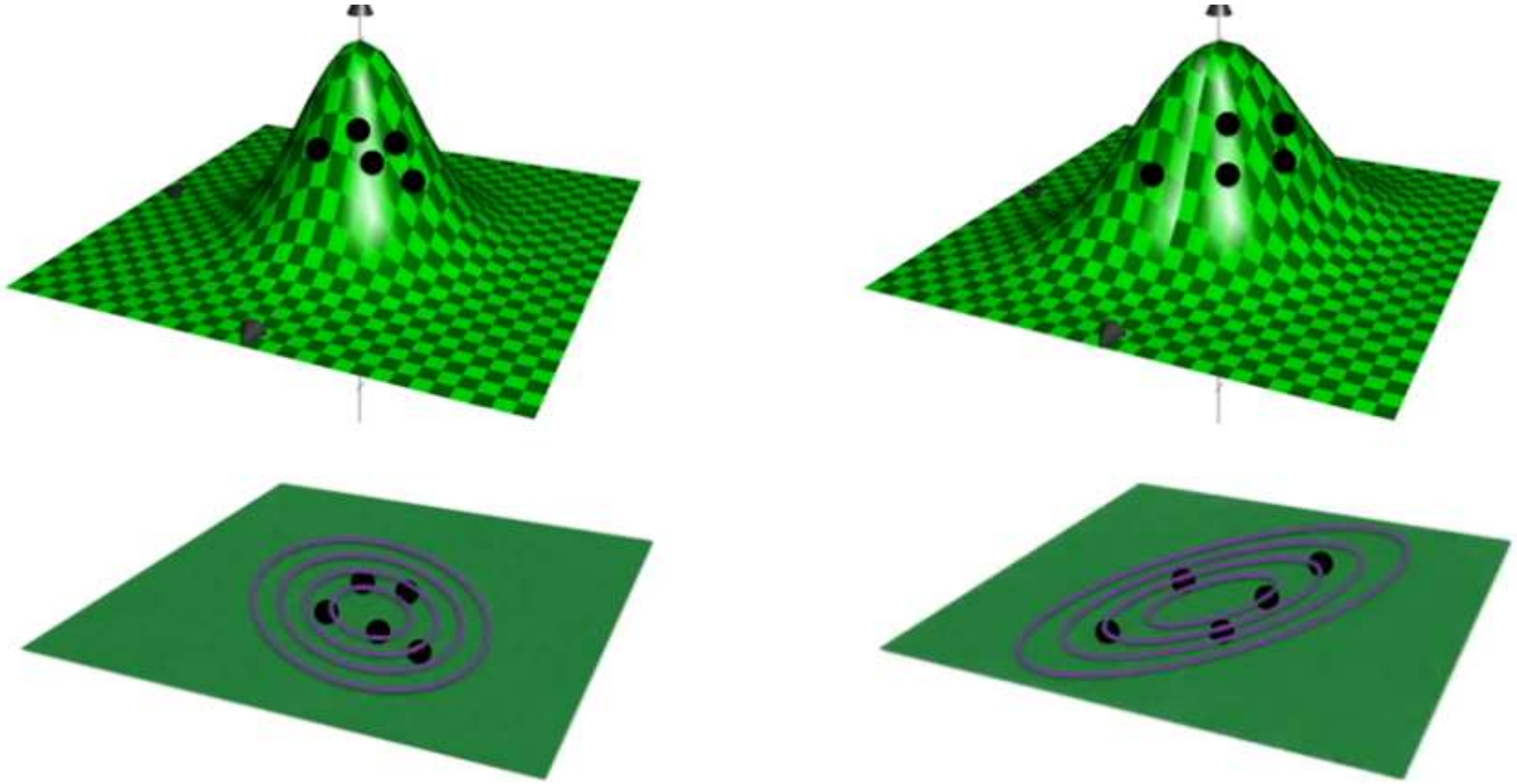
Soft Clustering

Soft Clustering

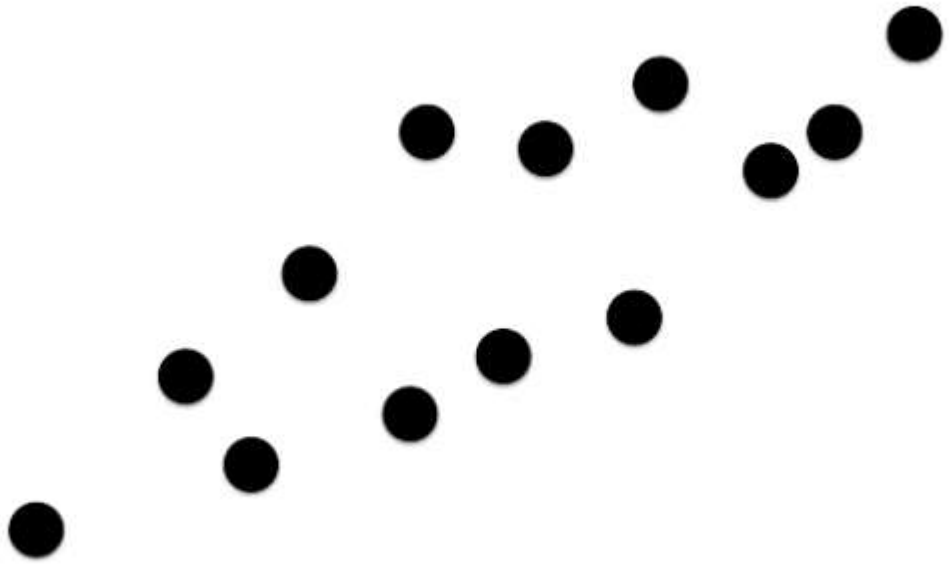
Soft Clustering



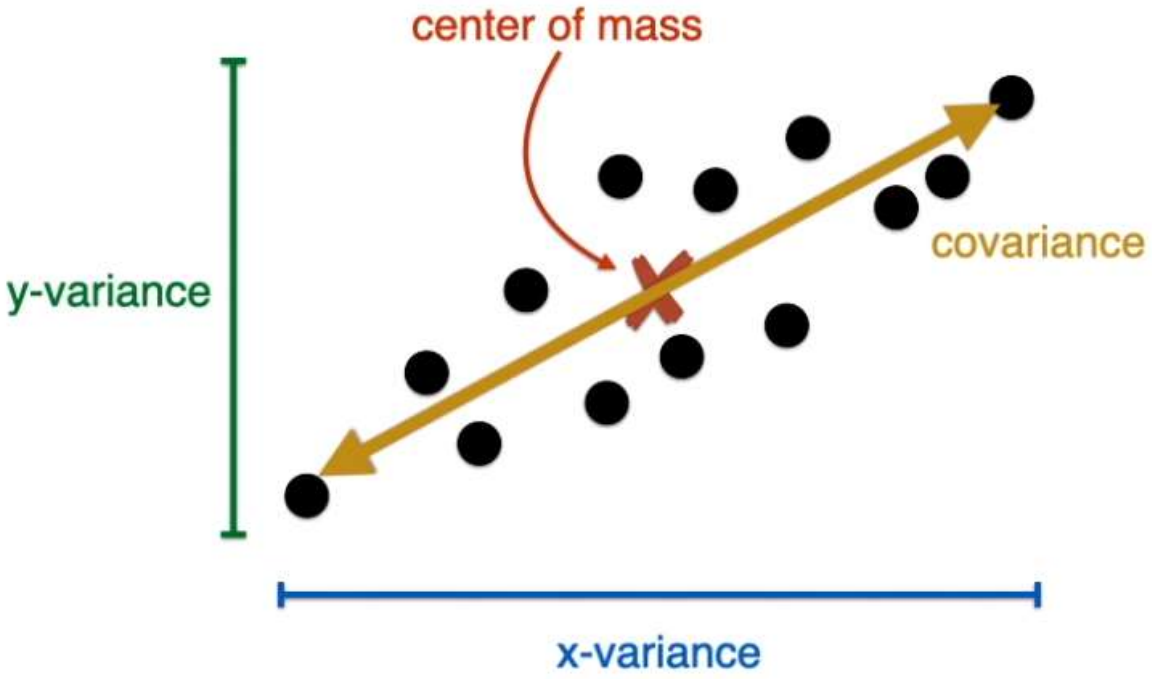
Fitting a Gaussian



Fitting a Gaussian

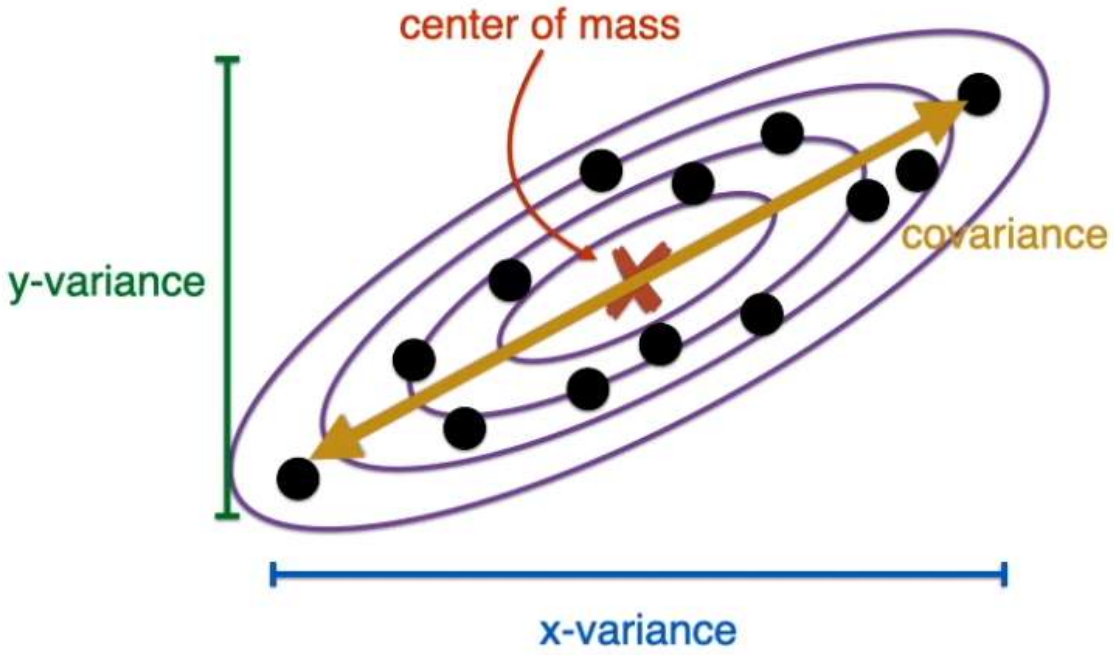


Fitting a Gaussian



$$\mu = Average$$

Fitting a Gaussian



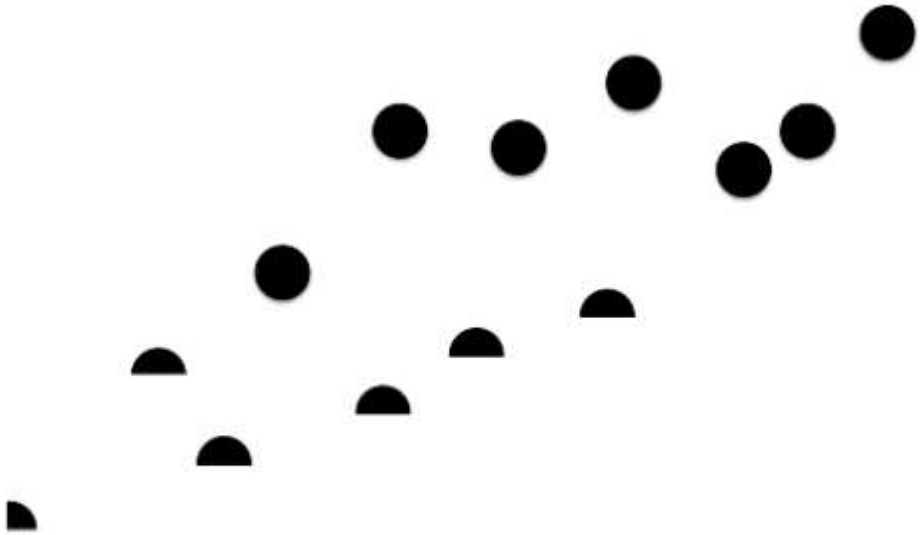
$$\mu = Average$$

$$\Sigma = \begin{pmatrix} Var(x) & Cov(x,y) \\ Cov(x,y) & Var(y) \end{pmatrix}$$

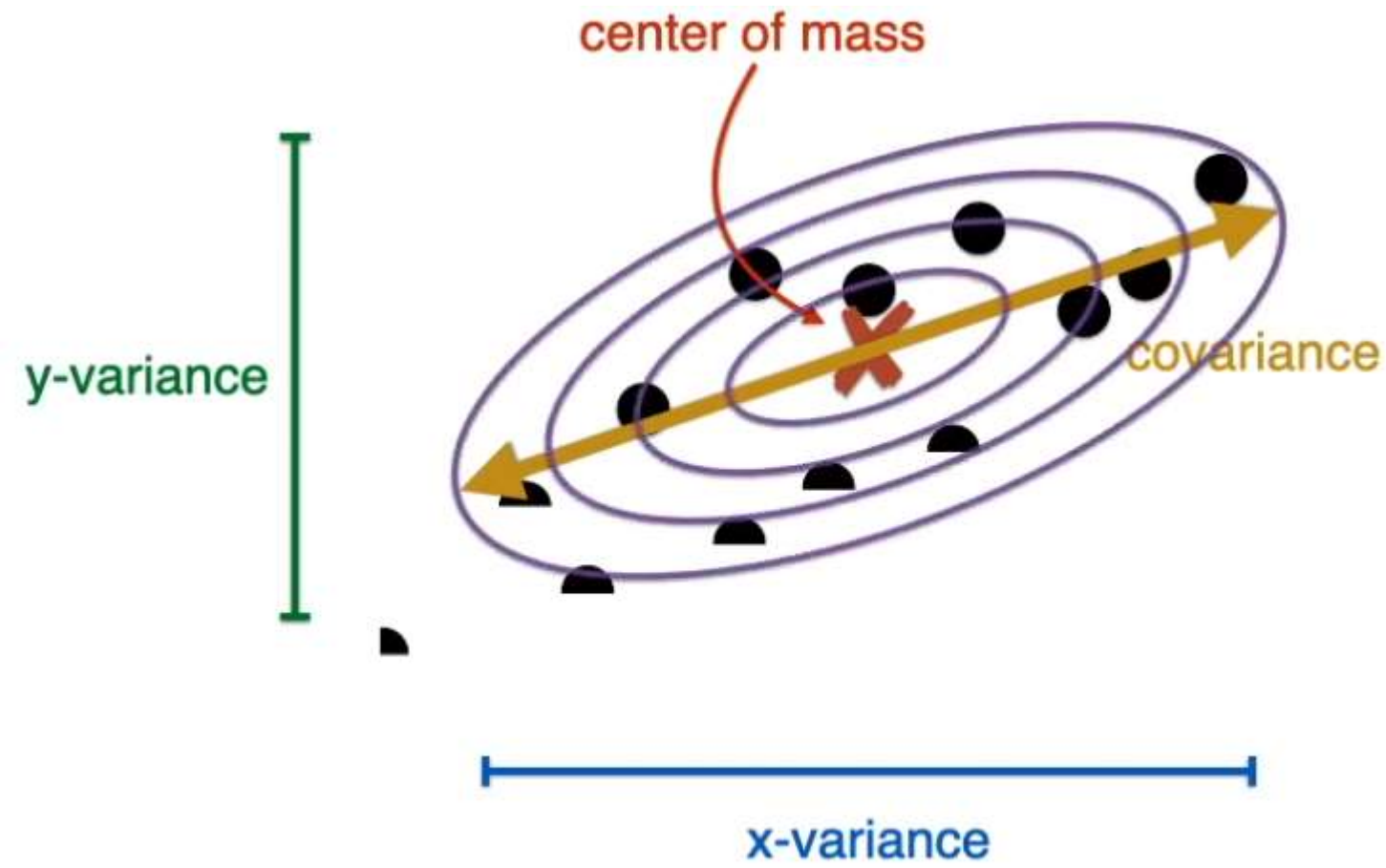
$$f(x) = \frac{\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{2\pi\sqrt{|\Sigma|}}$$

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2)}$$

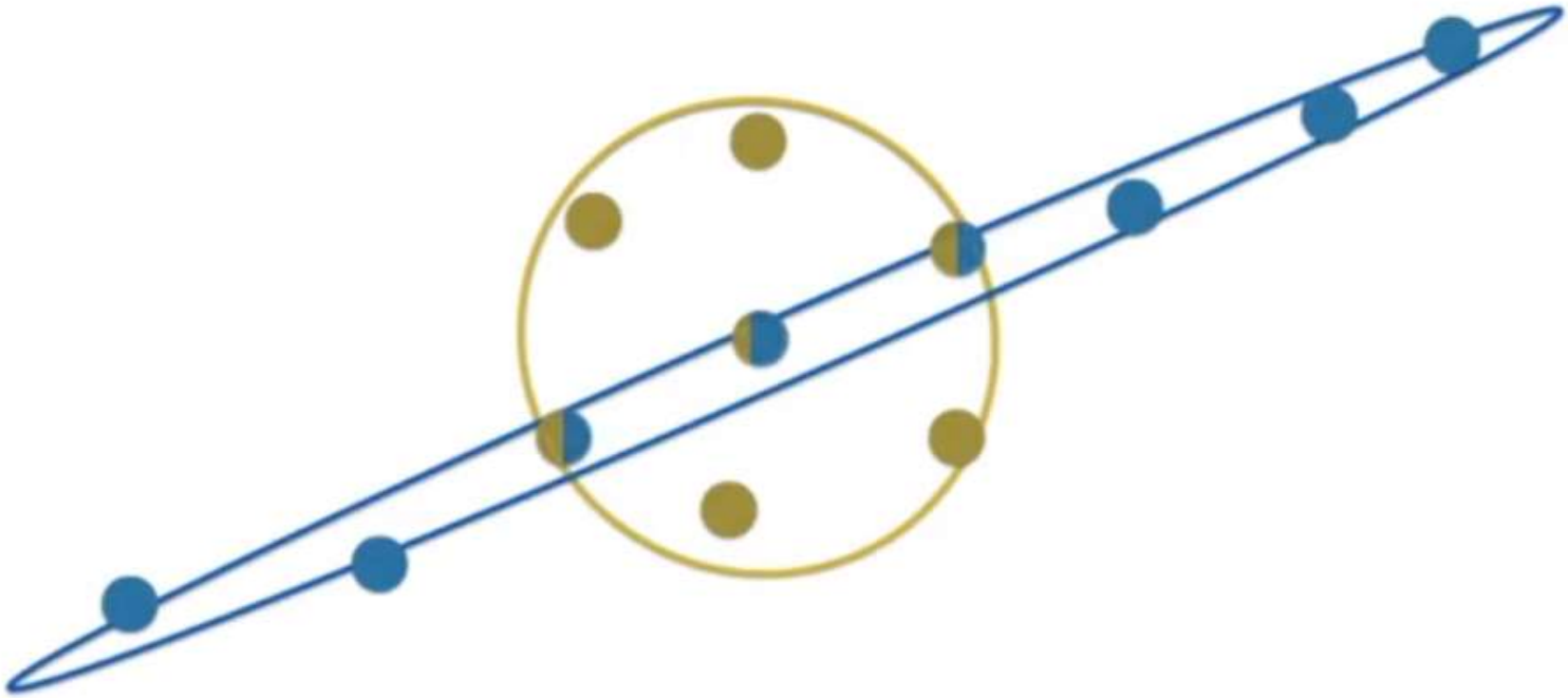
Fitting a Gaussian



Fitting a Gaussian



Gaussian Mixture Model



Gaussian Mixture Model

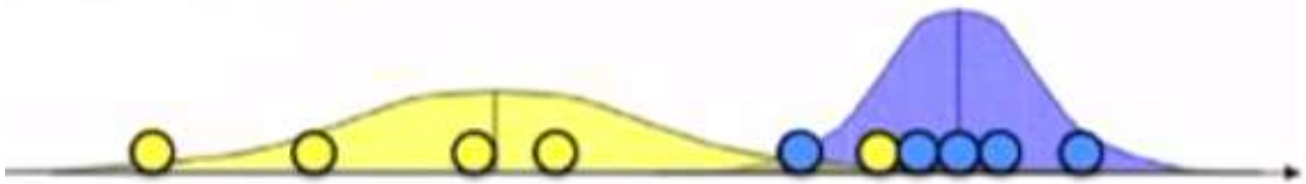
- Probabilistically grounded way of doing soft clustering
- Each cluster: a generative model (Gaussian or multinomial)
- Parameters (e.g. mean and variance are unknown)
- Expectation maximization (EM) Algorithm:
  - Automatically discover all the parameters for the ‘K’ sources

Gaussian Mixture Model

- Observation $x_1 \ldots x_n$
  - K=2 Gaussians with unknown μ and σ^2
  - Estimation is easy if we know the source of each observation

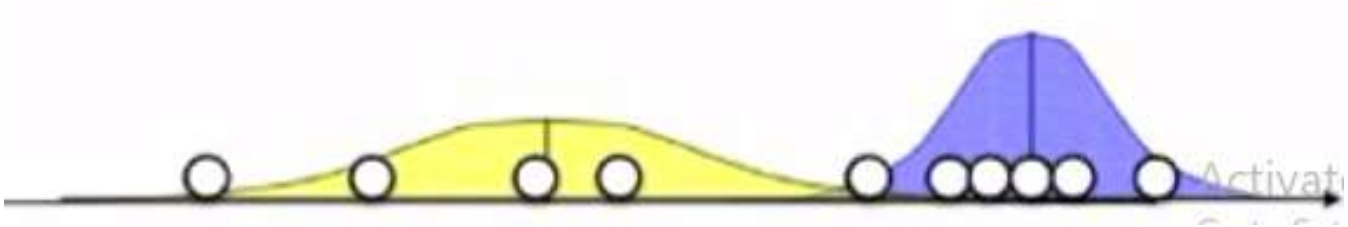
$$\mu_b = \frac{x_1 + x_2 + \ldots + x_{n_b}}{n_b}$$

$$\sigma_b^2 = \frac{(x_1 - \mu_1)^2 + \ldots + (x_n - \mu_n)^2}{n_b}$$



What if we don't know the source

- If we know parameters of Gaussian (μ and σ^2)
  - Can guess whether the point is more likely to be a or b



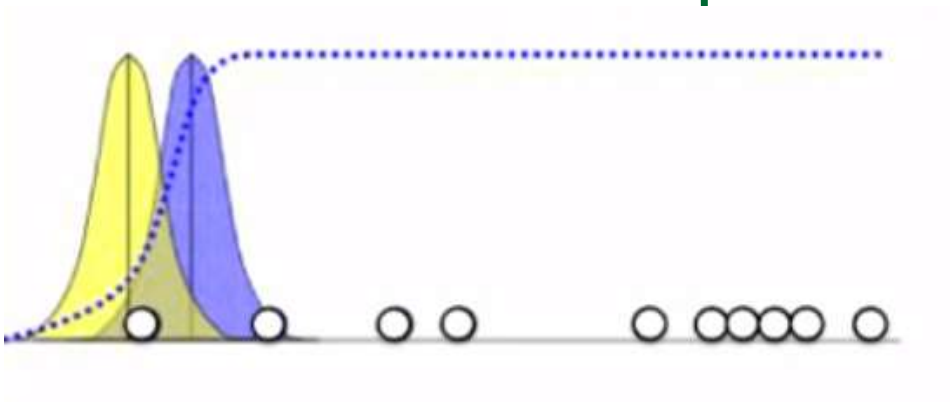
$$P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

Expectation Maximization

- Iterative process
  - Need (μ_a, σ_a^2) and (μ_b, σ_b^2) to guess the source of points
  - Need to know source to estimate (μ_a, σ_a^2) and (μ_b, σ_b^2)
- EM Algorithm
  - Start with two randomly placed Gaussians
  - For each point: $P(b|x_i)$ = does it look like came from b
  - Adjust (μ_a, σ_a^2) and (μ_b, σ_b^2) to fit points assigned to them
  - Iterate until convergence

EM: 1D Example

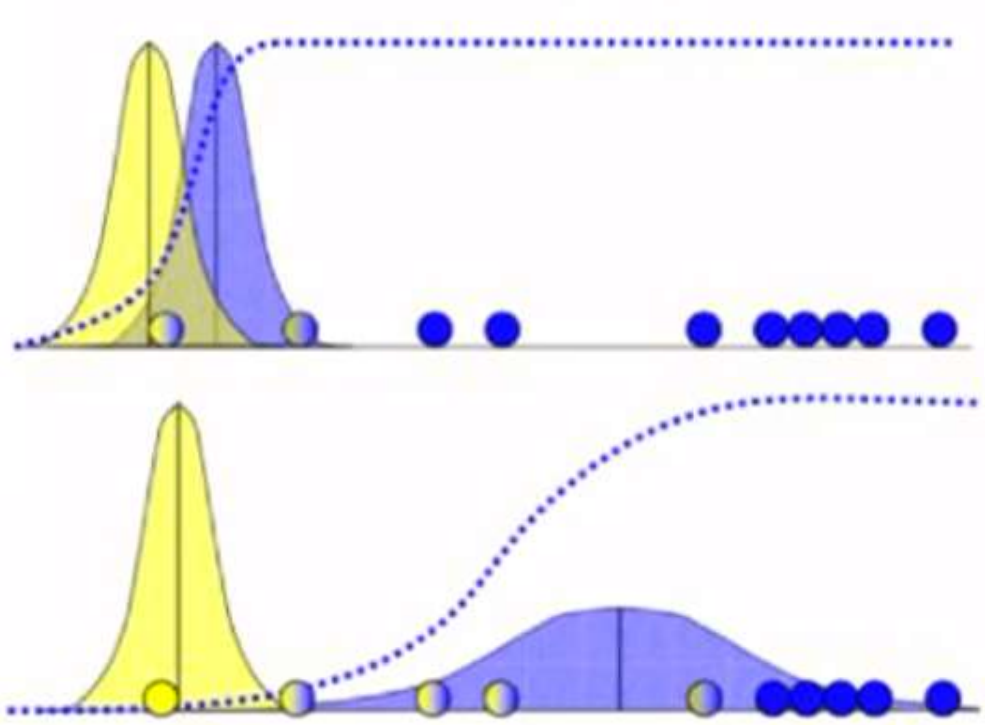


EM: 1D Example

$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

EM: 1D Example



$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

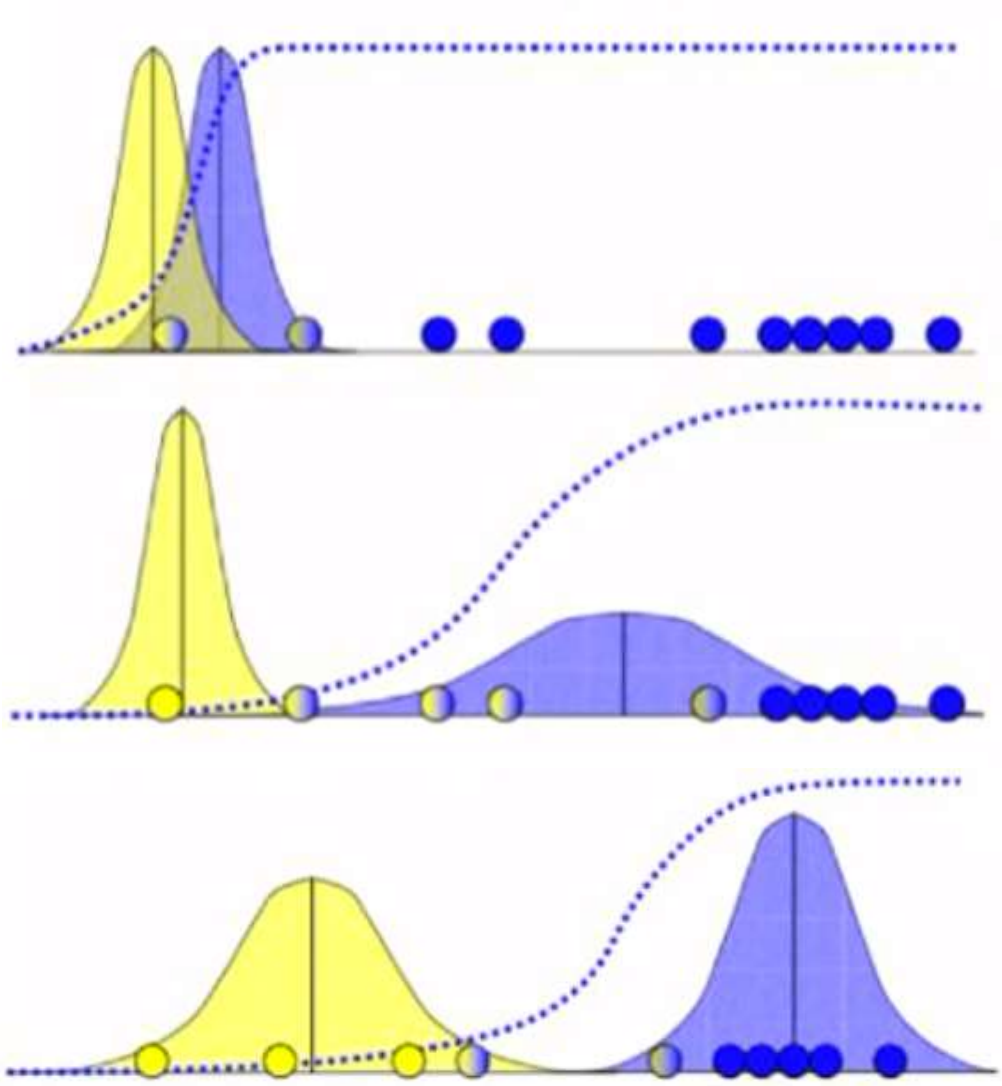
$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \ldots + b_n x_{n_b}}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1 (x_1 - \mu_1)^2 + \ldots + b_n (x_n - \mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \ldots + a_n x_{n_a}}{a_1 + a_2 + \ldots + a_n}$$

$$\sigma_a^2 = \frac{a_1 (x_1 - \mu_1)^2 + \ldots + a_n (x_n - \mu_n)^2}{a_1 + a_2 + \ldots + a_n}$$

EM: 1D Example



$$P(x_i \mid b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

$$b_i = P(b \mid x_i) = \frac{P(x_i \mid b)P(b)}{P(x_i \mid b)P(b) + P(x_i \mid a)P(a)}$$

$$a_i = P(a \mid x_i) = 1 - b_i$$

$$\mu_b = \frac{b_1 x_1 + b_2 x_2 + \ldots + b_n x_{n_b}}{b_1 + b_2 + \ldots + b_n}$$

$$\sigma_b^2 = \frac{b_1 (x_1 - \mu_1)^2 + \ldots + b_n (x_n - \mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

$$\mu_a = \frac{a_1 x_1 + a_2 x_2 + \ldots + a_n x_{n_a}}{a_1 + a_2 + \ldots + a_n}$$

$$\sigma_a^2 = \frac{a_1 (x_1 - \mu_1)^2 + \ldots + a_n (x_n - \mu_n)^2}{a_1 + a_2 + \ldots + a_n}$$

EM: 1D Example

could also estimate priors:

$$P(b) = (b_1 + b_2 + \ldots b_n) / n$$

$$P(a) = 1 - P(b)$$

to continue...