COMP 7990
Principles and Practices of Data Analytics

Lecture 4: Unsupervised Learning

Dr. Eric Lu Zhang

Outline for Data Preprocessing and Data Mining

- Data Preprocessing
- Supervised learning
- Regression
  1. Linear regression with one variable
  2. Linear Regression with multiple variables
- Classification
  1. Perceptron
  2. Artificial Neural Network
  3. Support Vector Machine
  4. K Nearest Neighbor
- **Unsupervised learning**
  1. K-means Clustering
  2. Hierarchical Clustering

Classification

Classification

$$\mathbf{X} \longrightarrow y$$

- Input X
  - an m*n matrix
  - Each row represents one data sample

- Output y
  - an m*1 vector
  - Each element in y represents the output (i.e., label) of one data sample
  - y_i is a **discrete value** for classification problem
    - $y_i \in \{0, 1\}$ for binary classification
    - $y_i \in \{1, \dots, k\}$ for multi-class classification

Regression

$$\mathbf{X} \xrightarrow{\text{Regression}} y$$

- Input X
  - an $m*n$ matrix
  - Each row represents one data sample
- Output y
  - an $m*1$ vector
  - Each element in y represents the output (i.e., label) of one data sample
  - y_i **is a continuous value** for regression problem

Clustering

Clustering

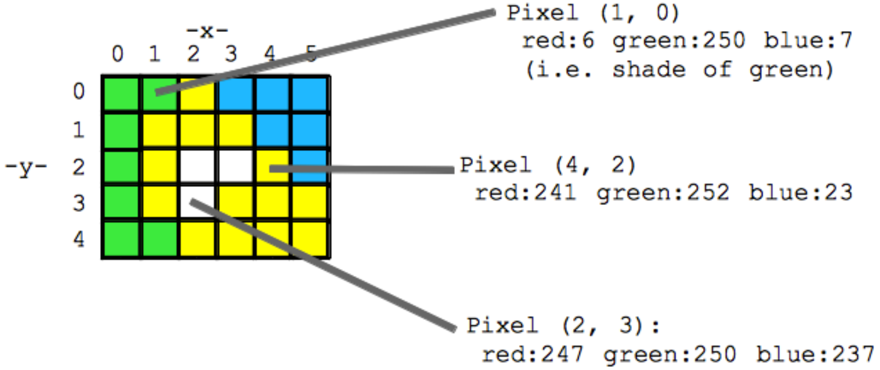
X

- Input X
  - an *m*n* matrix
  - Each row represents one data sample

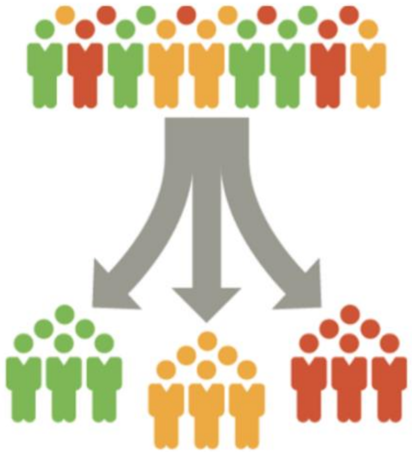
- The given data does not contain any output y
- Clustering tries to group input samples into different groups based on data similarities.

Clustering: Some Real-World Examples

- Clustering pixel values in an image to do image segmentation

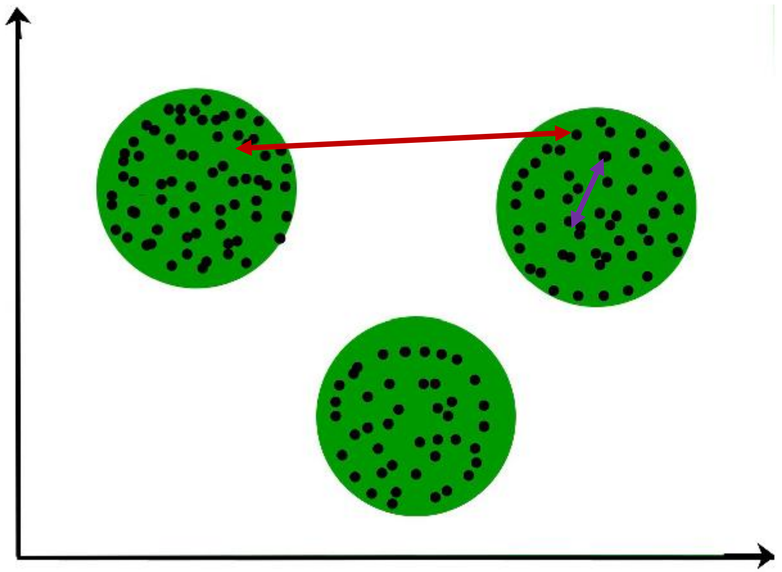


- Clustering customers based on their profile or purchase history



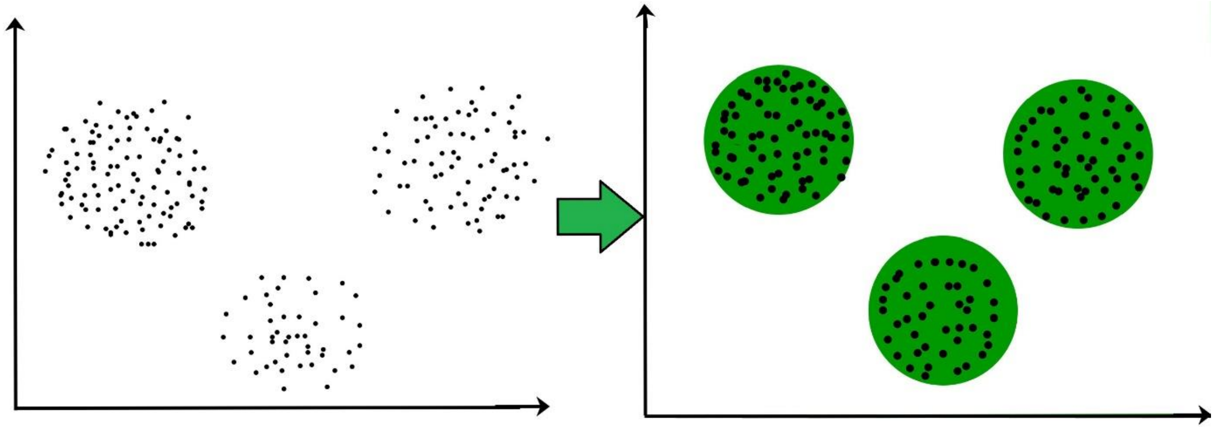
What is Clustering?

- Cluster: A collection of data points
  - With a cluster, the data points are close to each other.
  - For the data points in different clusters, they are far from each other.
- Clustering
  - Compute similarities (distance) between data points
  - Group similar (close) data points into clusters
  - Clusters/Groups/Partitions are used interchangeably in the literature but are essentially the same concept.
- Unsupervised learning: no predefined class labels

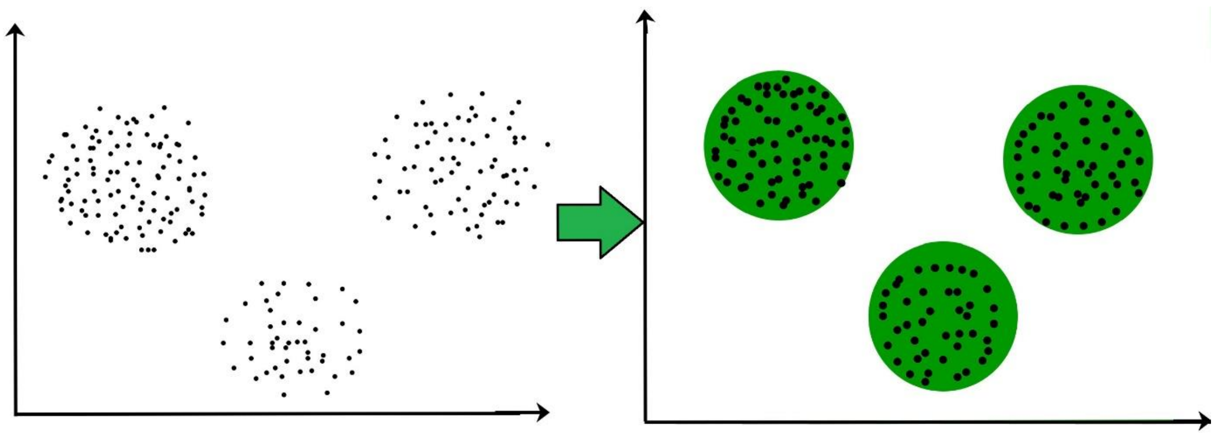


What is Clustering?

- Data Clustering is an unsupervised learning problem.
- Given *m* unlabeled samples $\{\mathbf{x}_i\}_{i=1}^{m}$, where $\mathbf{x}_i$ is a *n* dimensional input feature vector; the number of clusters *K*
- Goal: Group *m* samples into *K* clusters



What is Clustering?



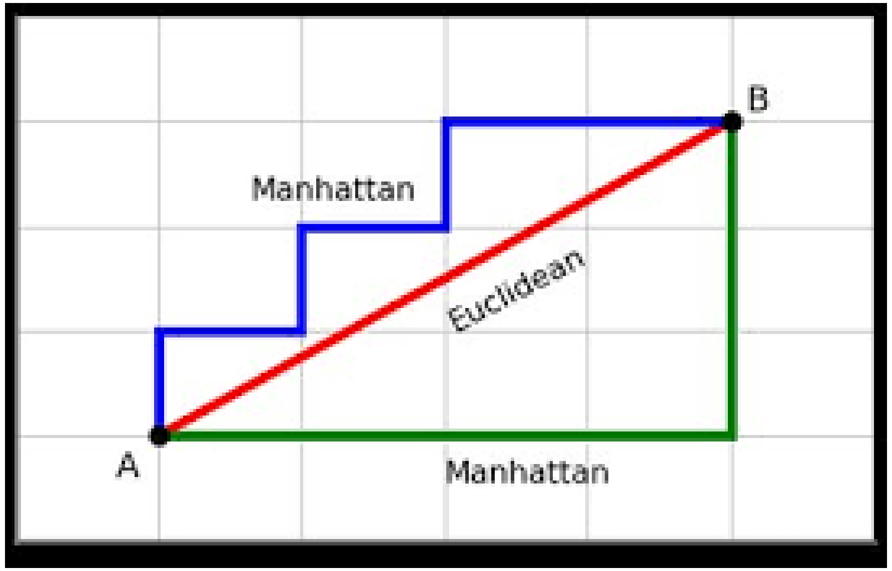
- The only information clustering uses is the similarity between samples
- A good clustering is the one that can achieve:
  - High intra-cluster similarity: cohesive within cluster
  - Low inter-cluster similarity: distinctive between clusters

Notions of Similarity/Distance

- The choice of the similarity measure is very important for clustering.
- Similarity is inversely related to distance.
- There are different ways to measure the distances between two data points.
  - L_2 (Euclidean) distance: $d(\mathbf{x},\mathbf{z}) = \|\mathbf{x}-\mathbf{z}\| = \sqrt{\sum_{j=1}^{n}(x_j - z_j)^2}$
  - L_1 (Manhattan) distance: $d(\mathbf{x},\mathbf{z}) = \sum_{j=1}^{n}\left|x_j - z_j\right|$
  - L_p distance: $d(\mathbf{x},\mathbf{z}) = \left(\sum_{j=1}^{n}\left|x_j - z_j\right|^p\right)^{1/p}$
  - L_∞ distance: $\max\{x_j - z_j\}$, j=1…n
  - Kernelized (non-linear) distance: $d(\mathbf{x},\mathbf{z}) = \|\phi(\mathbf{x}) - \phi(\mathbf{z})\|$

Euclidean and Manhattan Distance: Difference?

E.g. 1



L_2 (Euclidean) distance $= \sqrt{3^2 + 4^2} = 5$

L_1 (Manhattan) distance $= 3 + 4 = 7$

L_∞ distance$=$ $\max(\{3, 4\}) = 4$

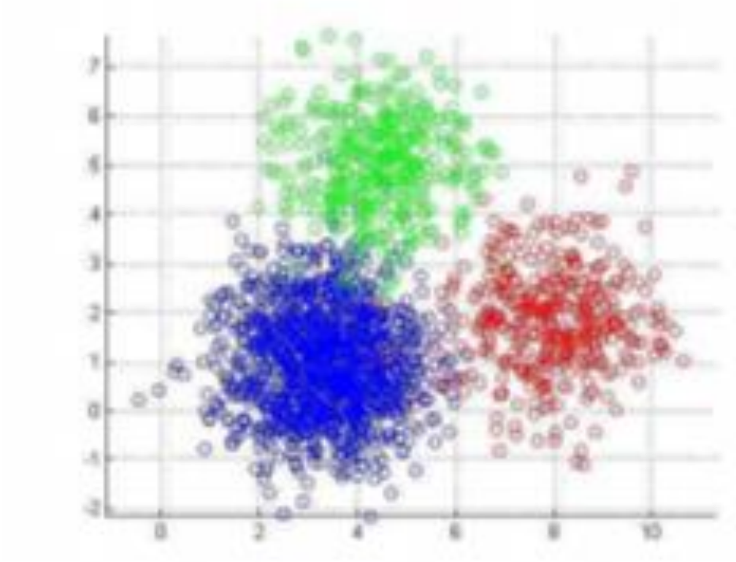
E.g. 2 Distance along different dimensions $= (2, 3, 2, 3, 100, 2)$

L_2 distance $= \sqrt{4 + 9 + 4 + 9 + 10000 + 4}$

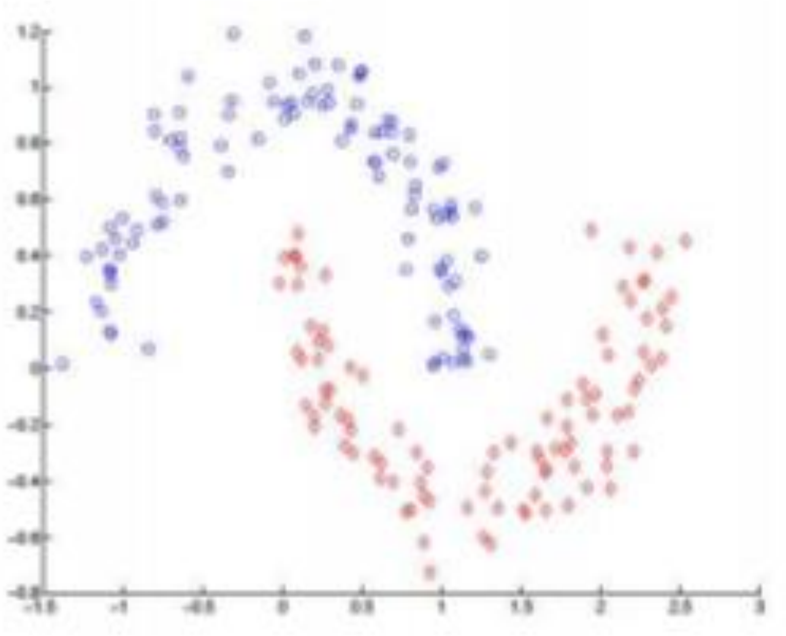
L_1 distance $= 2 + 3 + 2 + 3 + 100 + 2$

L_∞ distance $= 100$

Kernelized (non-linear) Distance



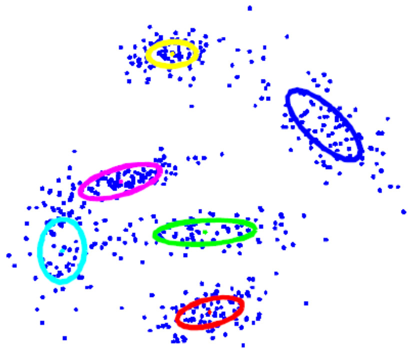
Use of Euclidean distance is reasonable.



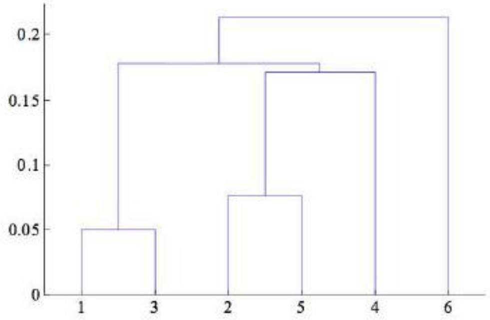
Kernelized distance is needed.

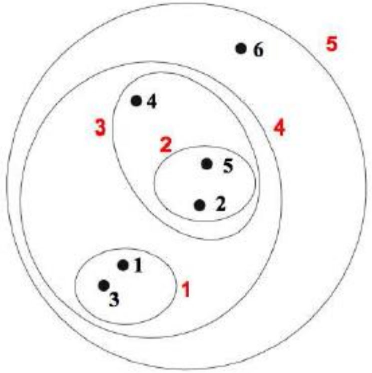
$$d(\mathbf{x}, \mathbf{z}) \\ = \|\phi(\mathbf{x}) - \phi(\mathbf{z})\|$$

Types of Clustering



- Partitional Clustering (e.g., *K*-means)
  - Partitions are independent of each other.
  - Hierarchical relationship not considered.
- Hierarchical Clustering (e.g., agglomerative clustering, divisive clustering)
  - Partitions can be visualized using a tree structure (a dendrogram)
  - Does not need the number of clusters as input
  - Allows partitions at different levels of granularities (i.e., can refine/coarsen clusters)





Outline for Data Preprocessing and Data Mining

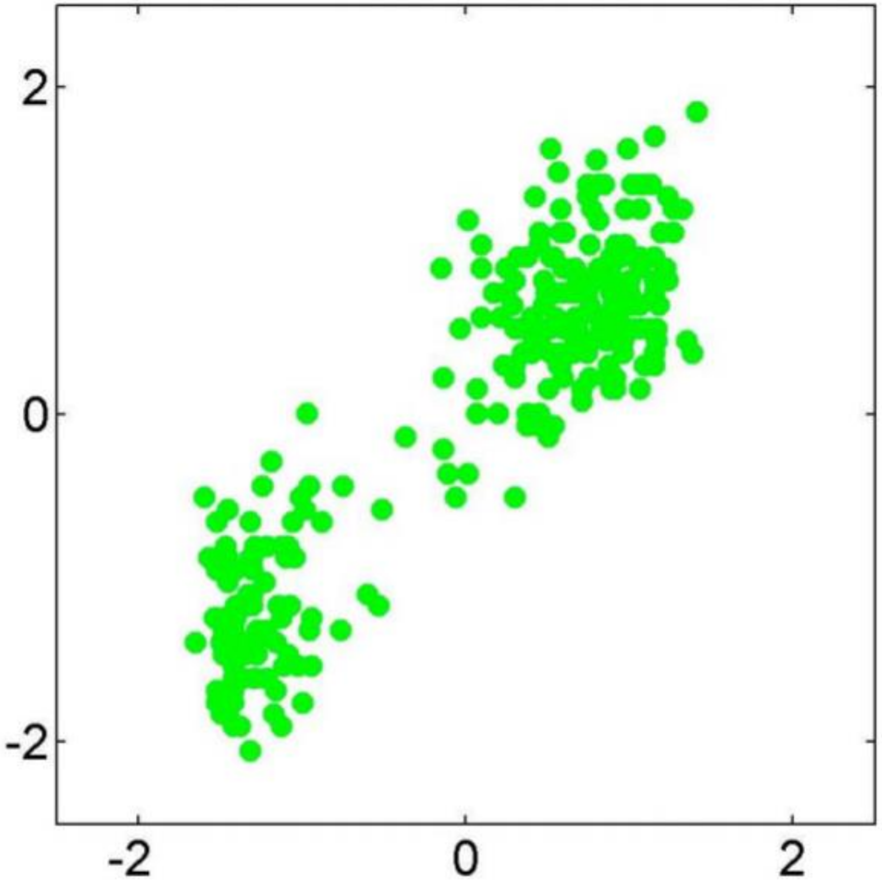
- **Data Preprocessing**
- **Supervised learning**
- Regression
  1. Linear regression with one variable
  2. Linear Regression with multiple variables
- Classification
  1. Perceptron
  2. Artificial Neural Network
  3. K Nearest Neighbor
  4. Support Vector Machine
- **Unsupervised learning**
  1. K-means Clustering
  2. Hierarchical Clustering

K-means Algorithm

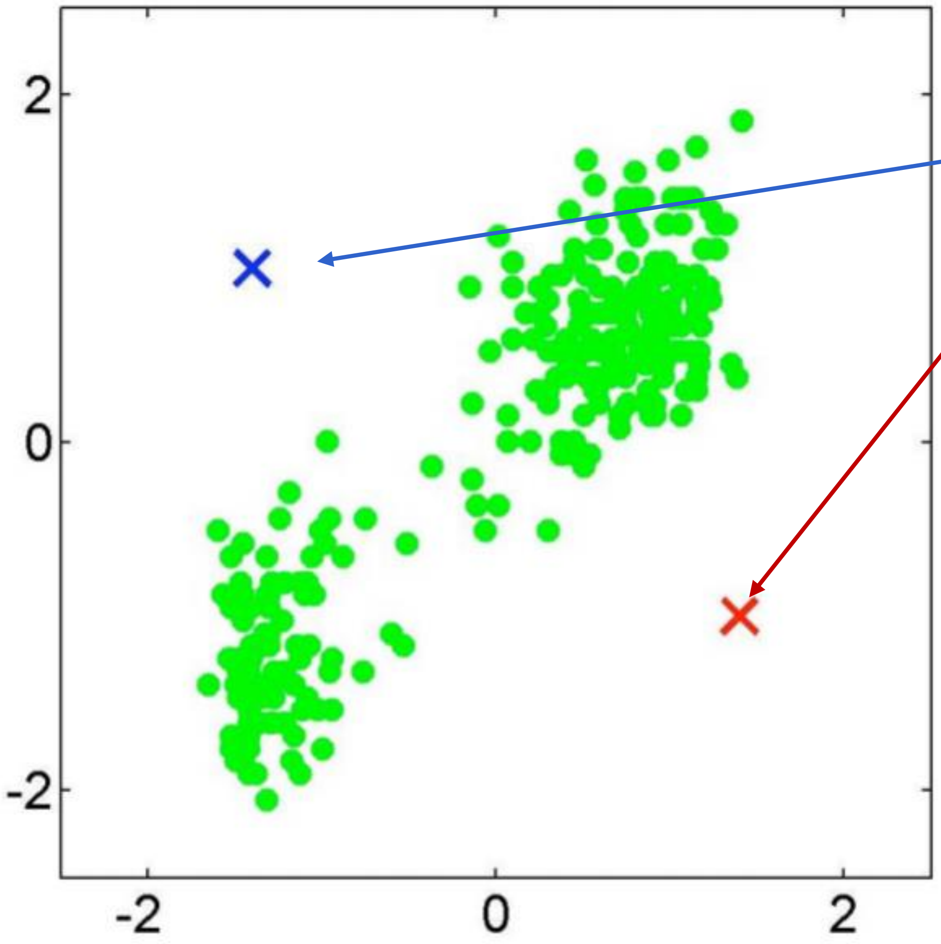
- **Input:** Samples $\{\mathbf{x}_i\}_{i=1}^{m}$, parameter K (i.e., number of clusters)
- **Initialize:** K cluster centers (means) $\mathbf{c}_1, \ldots, \mathbf{c}_k$. Several initialization options:
  - Randomly initialized anywhere in the input space
  - Randomly choose K samples from the data as the cluster centers
- **Iterate:**
  - Assign each sample $\mathbf{x}_i$ to its closest cluster center
    $$k = \arg\min_k \|\mathbf{x}_i - \mathbf{c}_k\|$$
  - Re-compute the cluster center $\mathbf{c}_k$ for every new cluster
    $$\mathbf{c}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$
  - Repeat while not converged
- Converge criteria: Cluster centers do not change anymore

C_k is the set of samples in cluster k
$|C_k|$ denotes the number of samples in C_k

K-means Example (Assume K = 2)

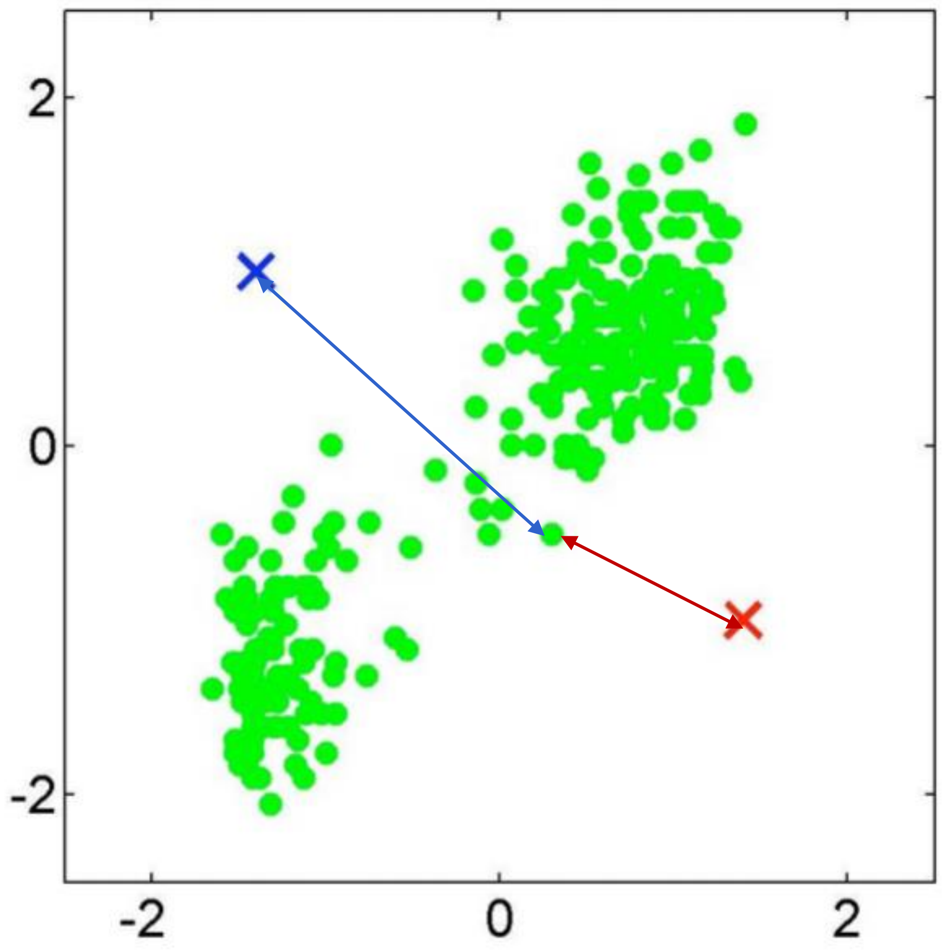


K-means Example: Initialization



- Randomly initialize two data points in the input space as the cluster centers.

K-means Iteration 1: Assign Data Points to Cluster

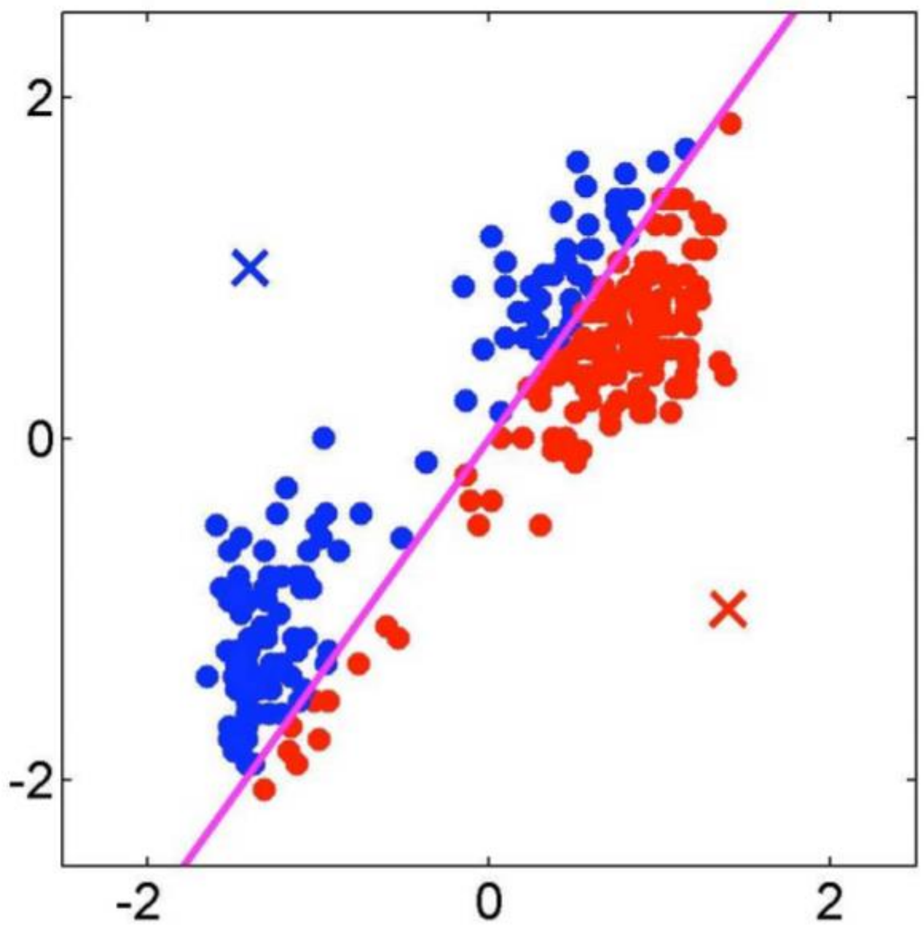


- For each sample, compute its distance from the cluster centers.

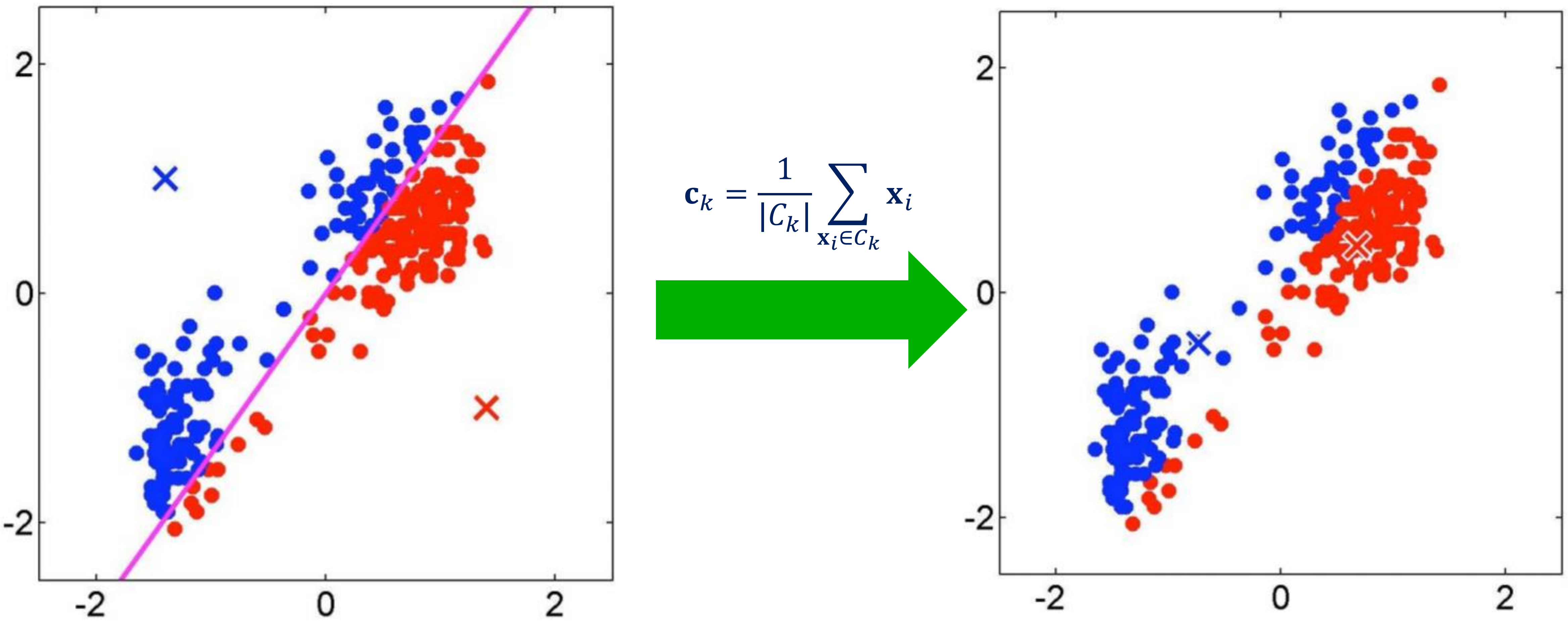
- Assign each sample $\mathbf{x}_i$ to its closest cluster center.

$$k = \arg\min_k \|\mathbf{x}_i - \mathbf{c}_k\|$$

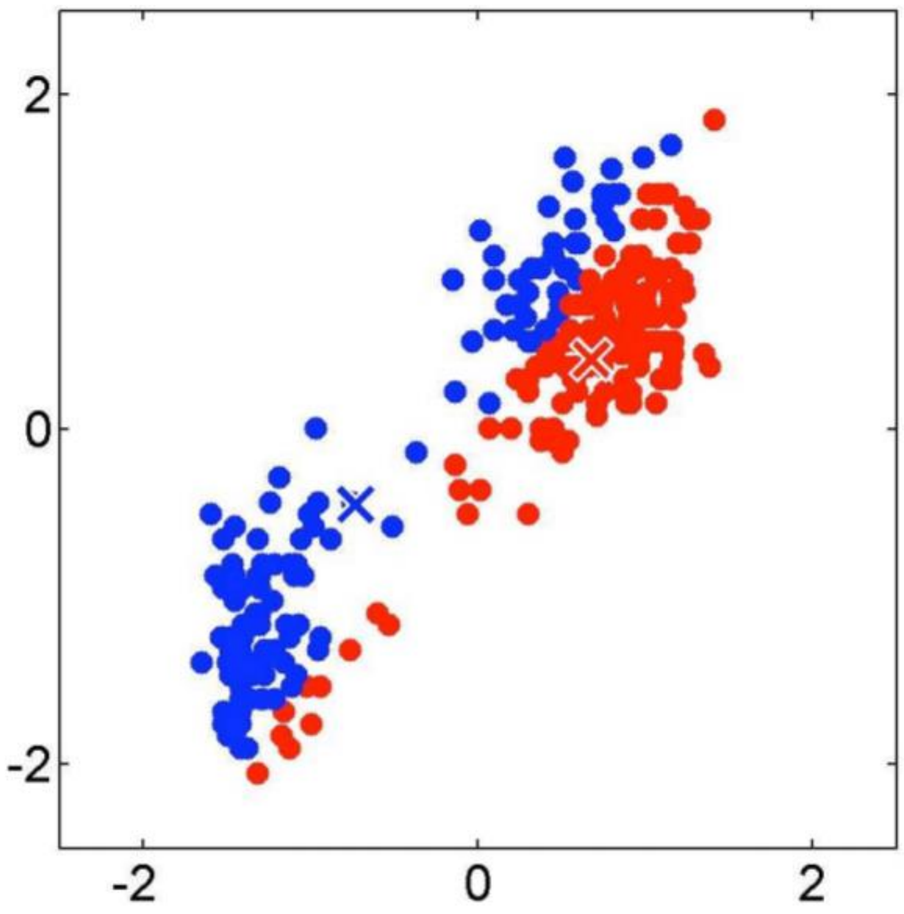
K-means Iteration 1: Assign Data Points to Cluster



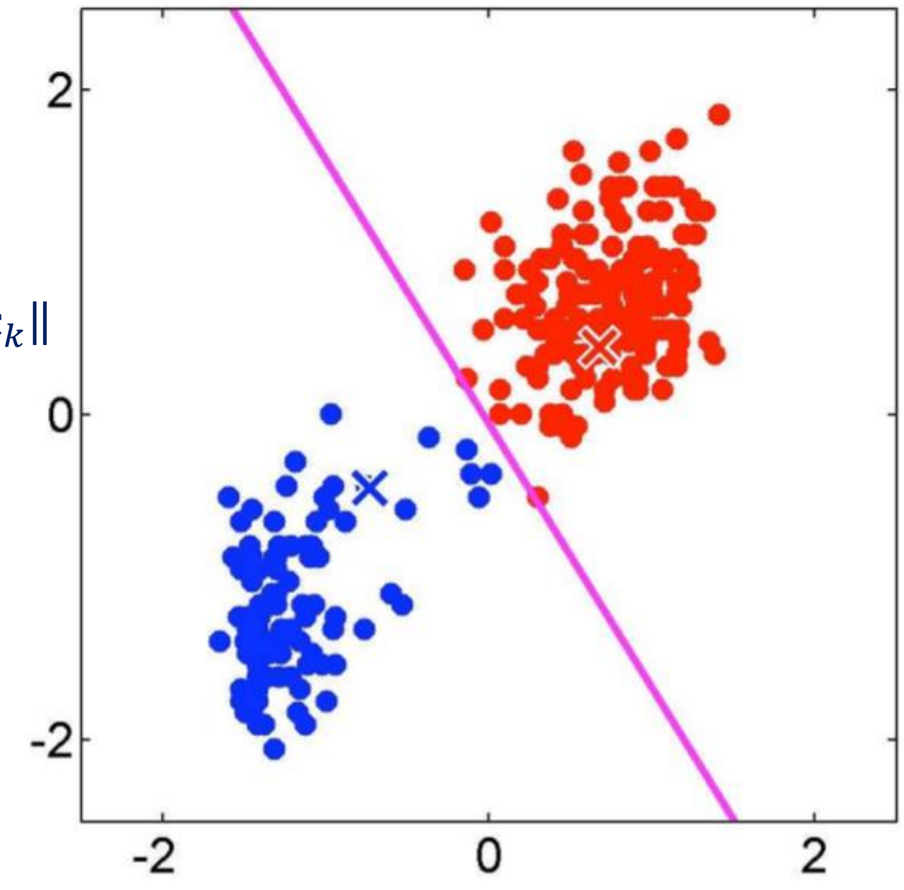
K-means Iteration 1: Recompute the Cluster Centers



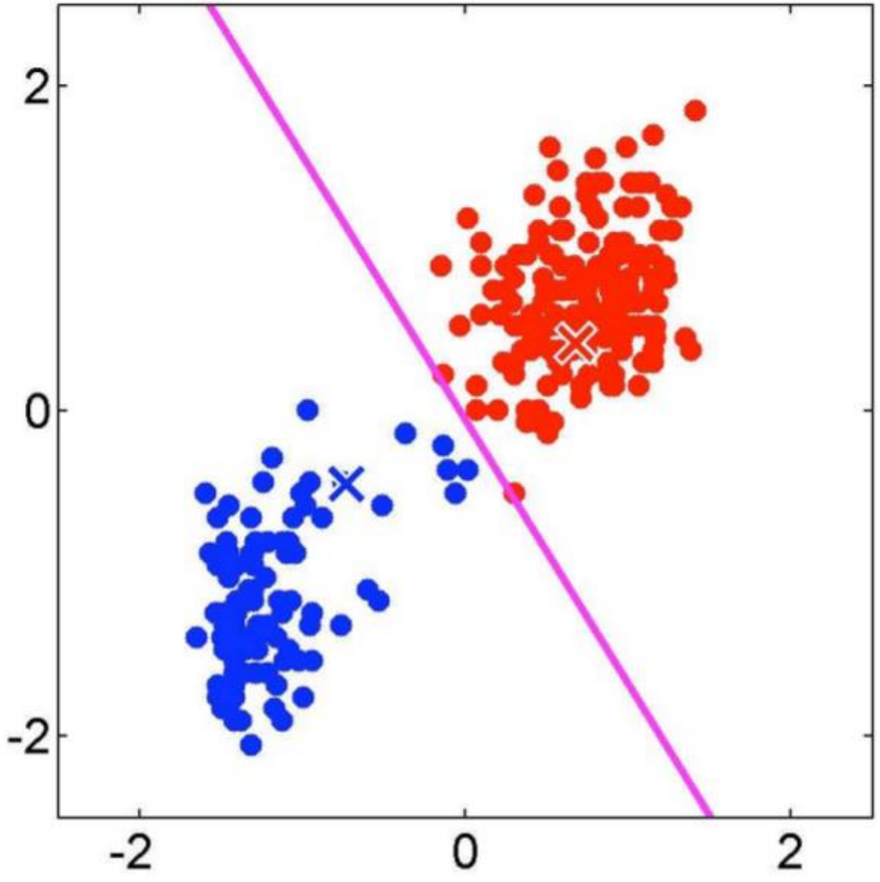
K-means iteration 2: Assign Data Points to Cluster



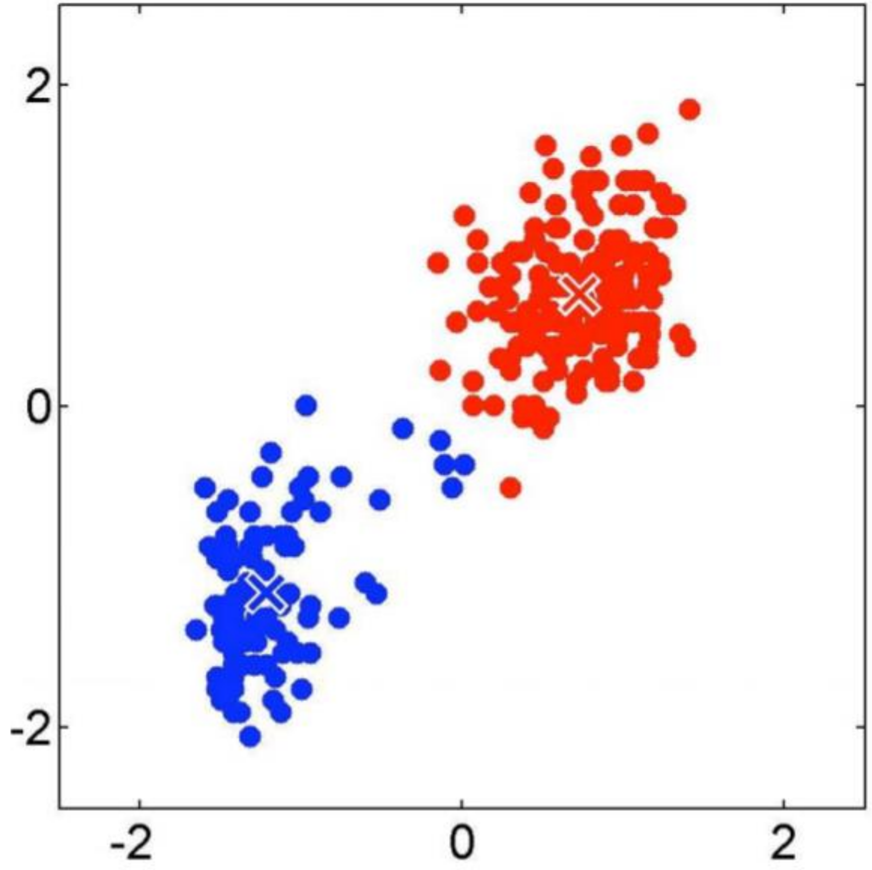
$$k = \arg\min_k \|\mathbf{x}_i - \mathbf{c}_k\|$$



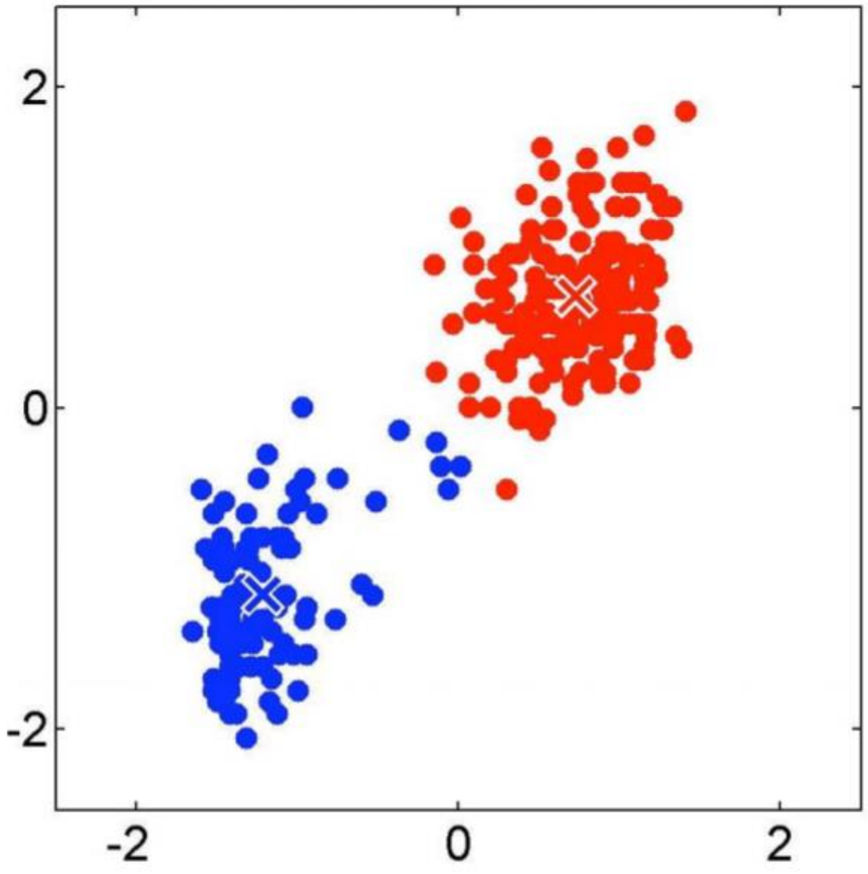
K-means Iteration 2: Recompute the Cluster Centers



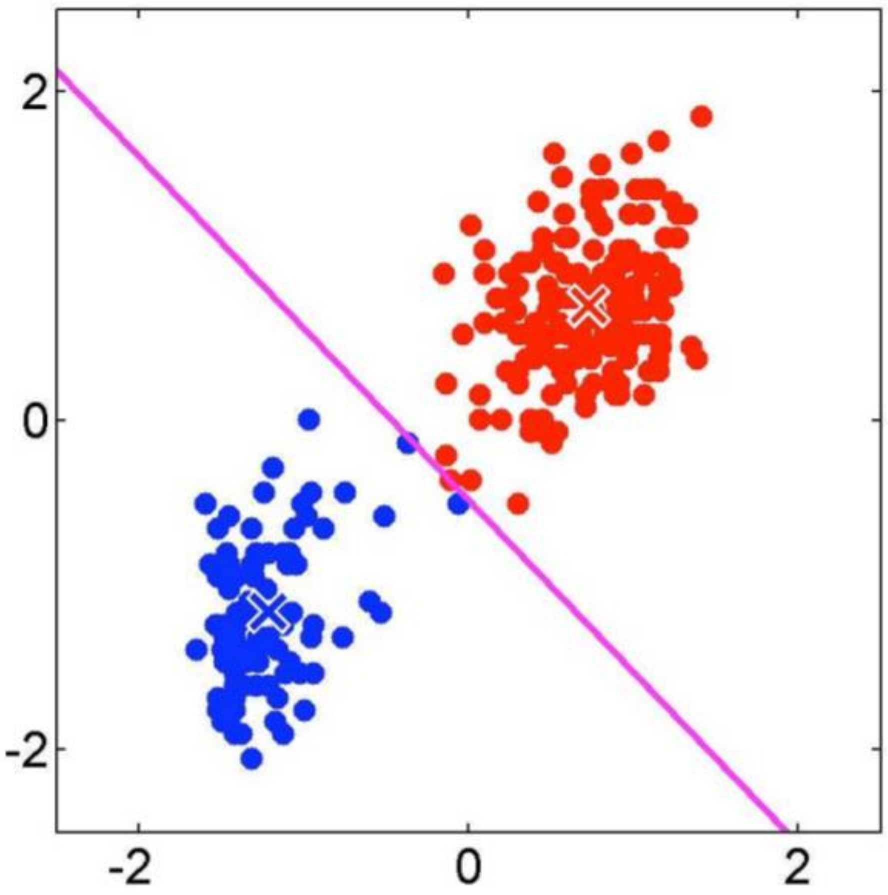
$$\mathbf{c}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$



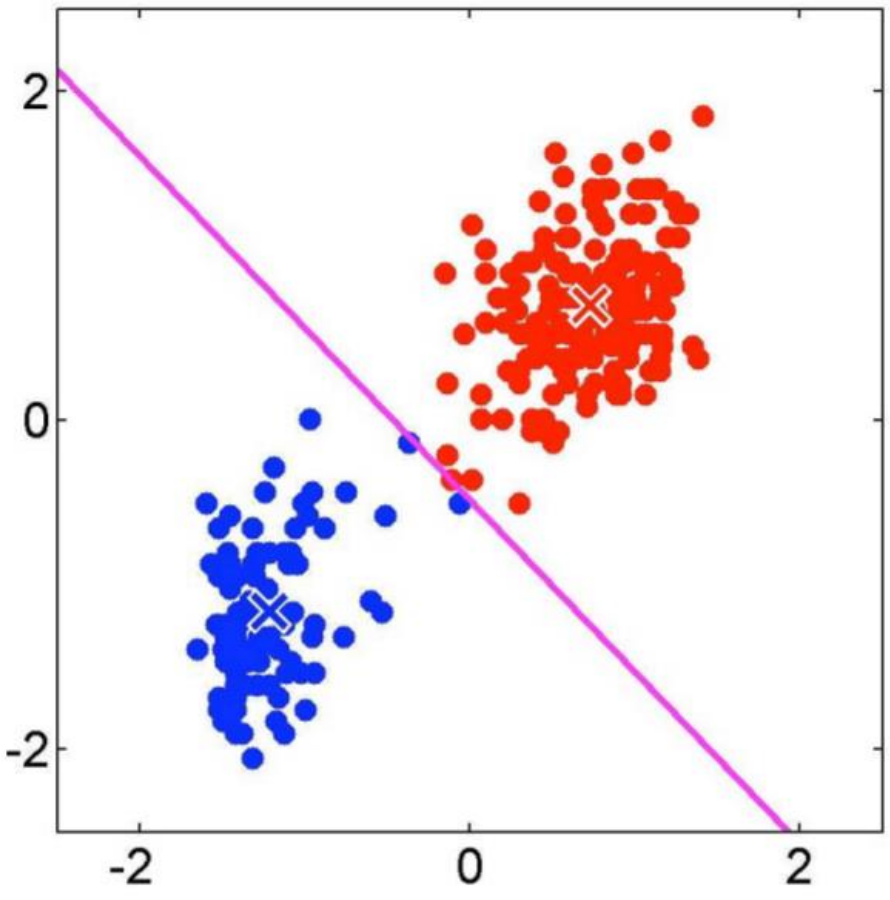
K-means Iteration 3: Assign Data Points to Cluster



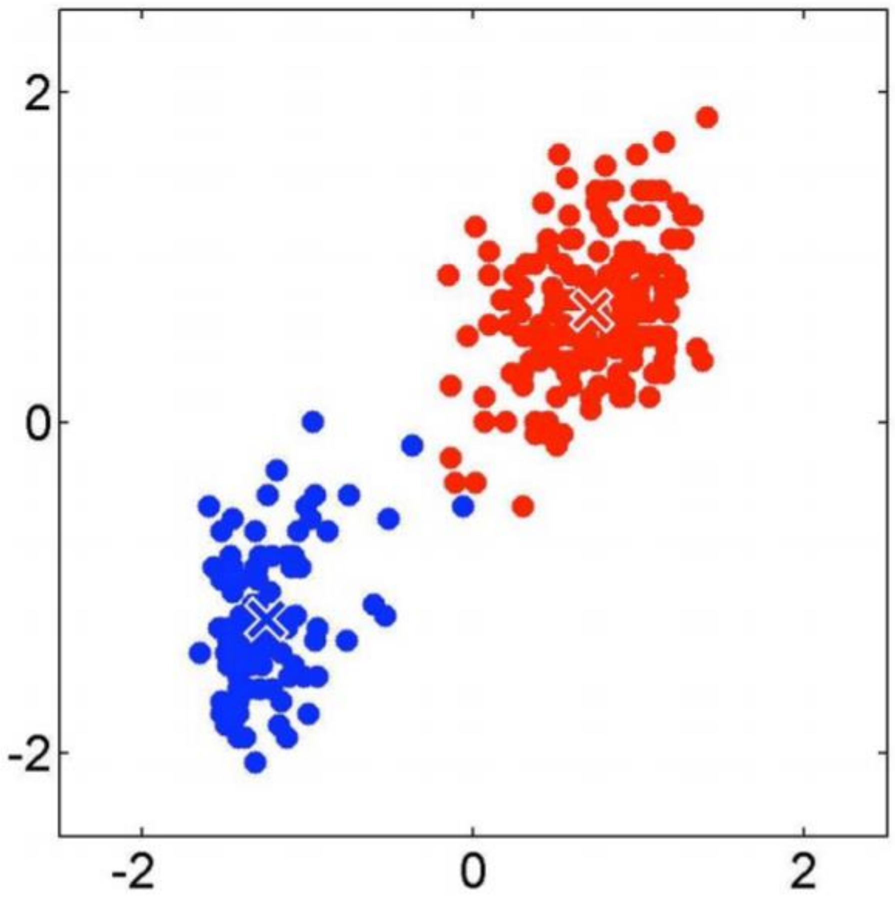
$$k = \arg\min_k \|\mathbf{x}_i - \mathbf{c}_k\|$$



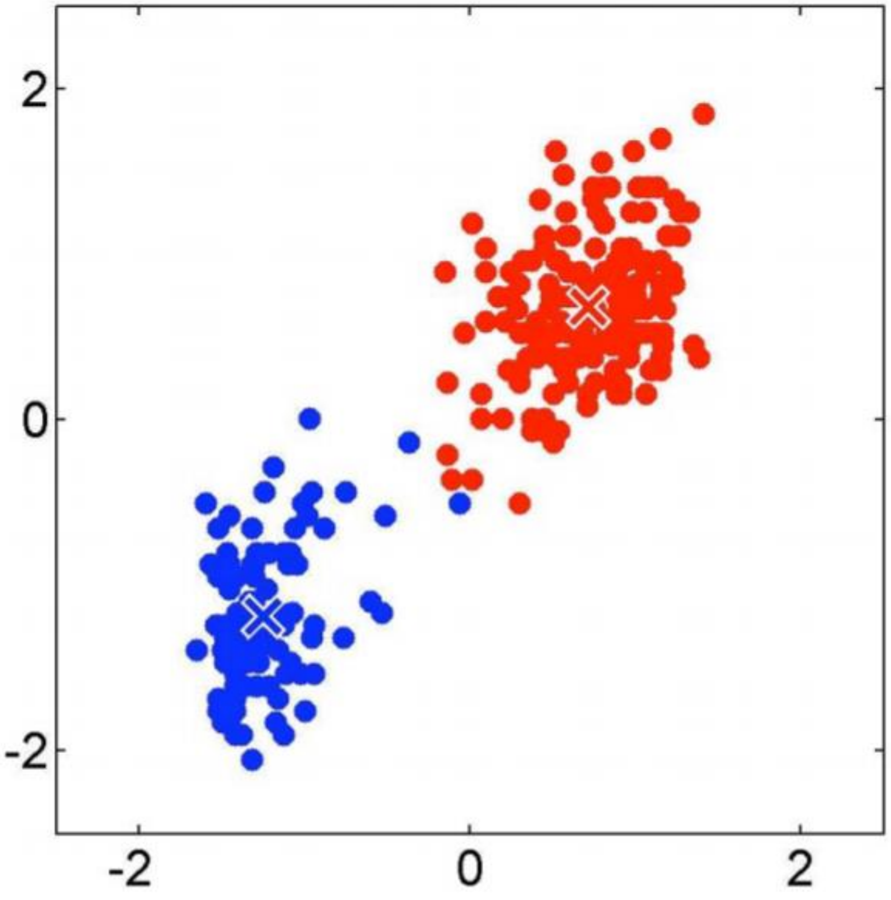
K-means Iteration 3: Recompute the Cluster Centers



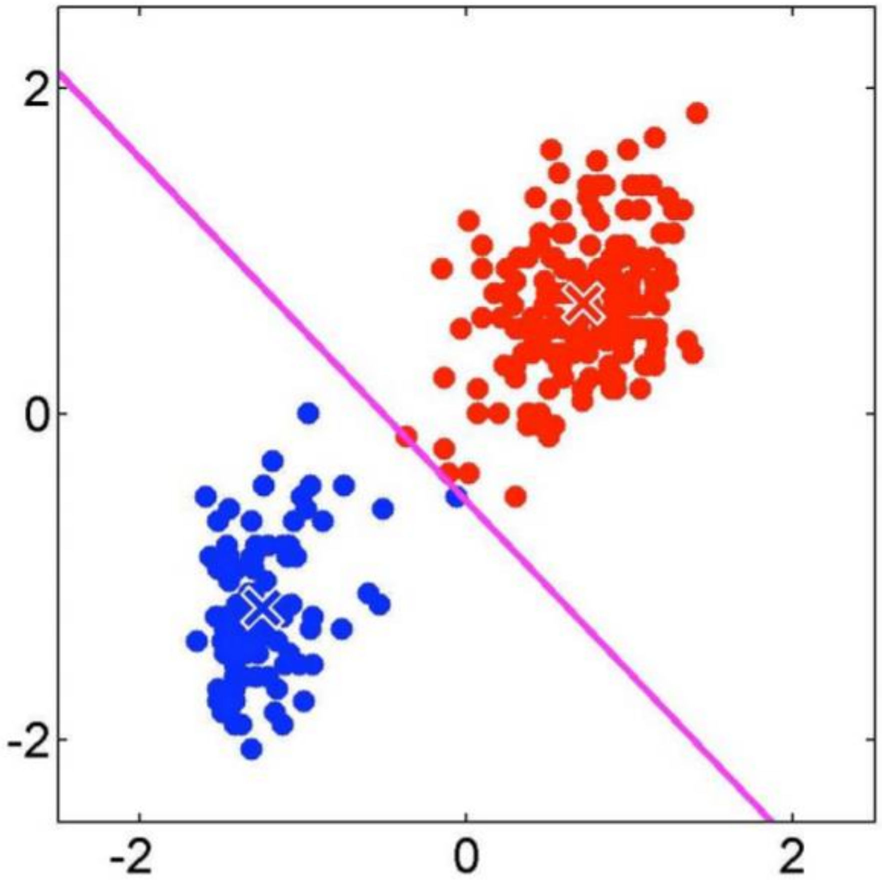
$$\mathbf{c}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$



K-means Iteration 4: Assign Data Points to Cluster

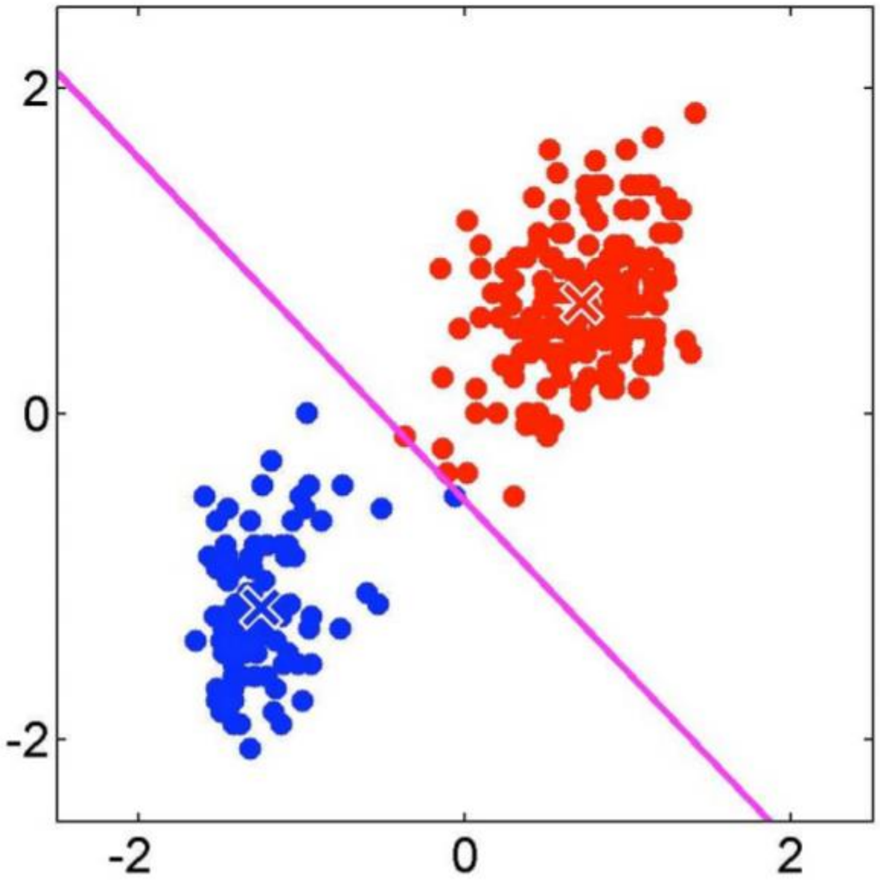


$$k = \arg\min_{k} \|\mathbf{x}_i - \mathbf{c}_k\|$$

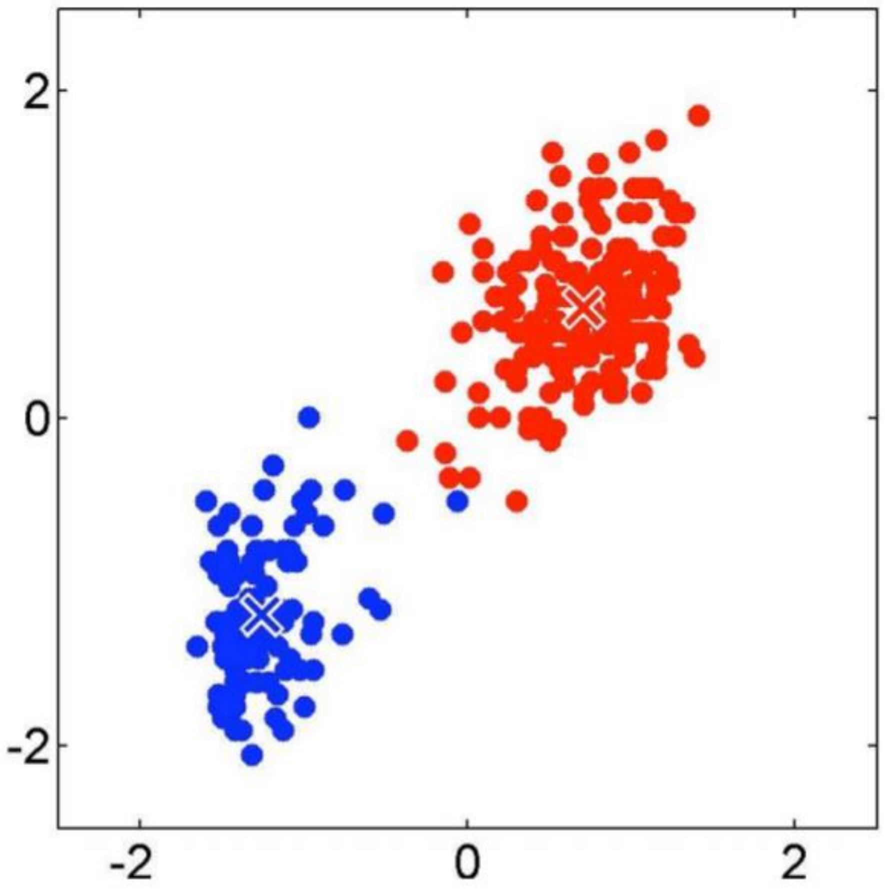


The cluster information does not change. The algorithm converged.

K-means iteration 4: Recompute the Cluster Centers



$$\mathbf{c}_k = \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$



The cluster centers do not change. The algorithm converged.

K-means: The Objective Function for Optimization

- The K-means objective function
  - Let $\mathbf{c}_1, \ldots, \mathbf{c}_K$ be the K cluster centers (means)
  - Let $\gamma_{ik} \in \{0,1\}$ be indicator variable denoting whether data point $\mathbf{x}_i$ belongs to cluster k

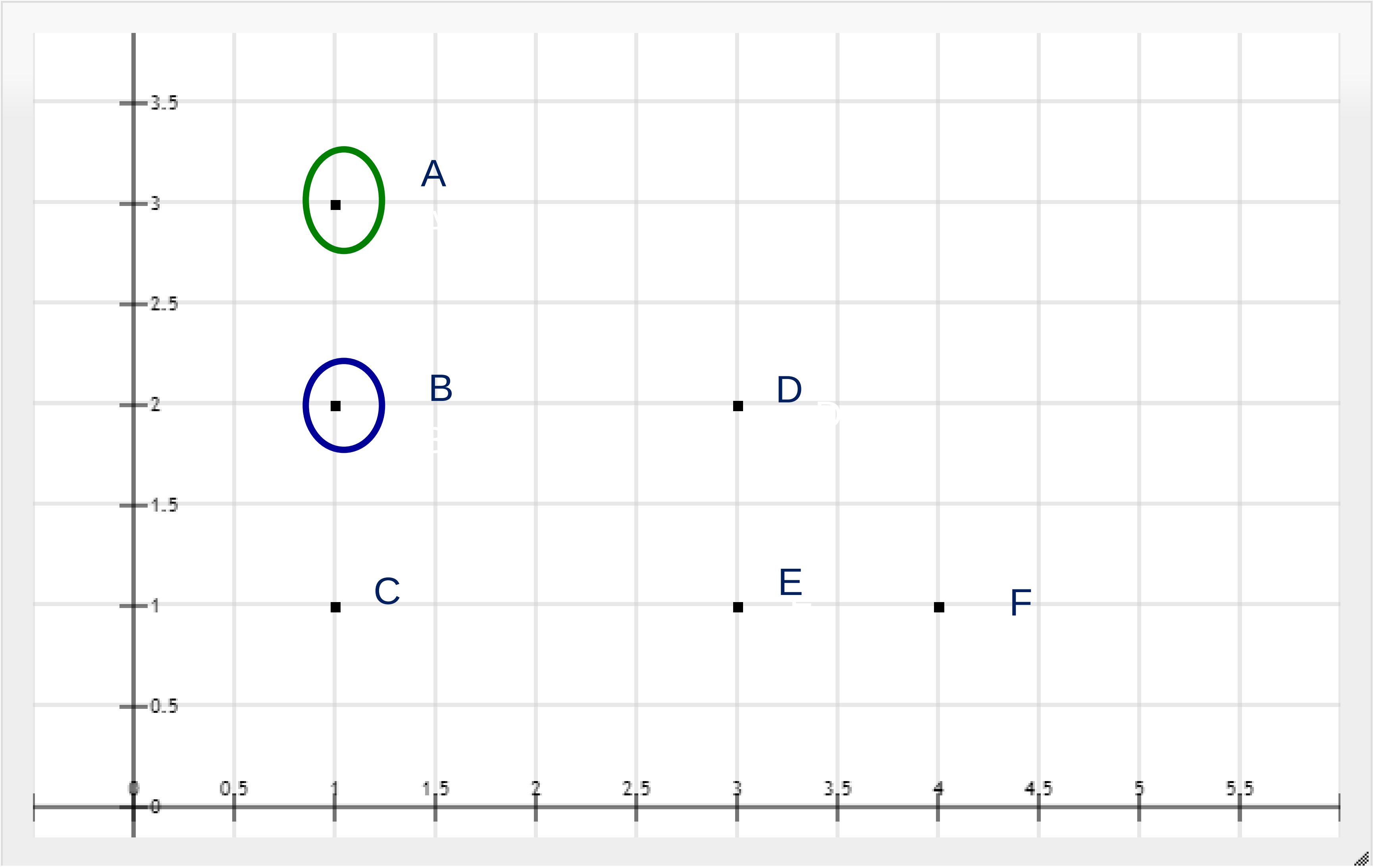
$$\gamma_{ik} = \begin{cases} 1 \text{ if } \mathbf{x}_i \text{ belongs to cluster } k \\ 0 \;\text{ if } \mathbf{x}_i \text{ not belongs to cluster } k \end{cases}$$

  - *K-means algorithm aims to minimize the total sum of distances of points from their cluster centers.*

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} \gamma_{ik} \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

  - **Note:** Exact optimization of the K-means objective function needs exhaustively enumerate all partitions. It is a NP-hard problem (to compute global optimal solution).
  - The K-means algorithm is a **heuristic way** to obtain a local optimal solution.

Another K-means Example



Iteration #1

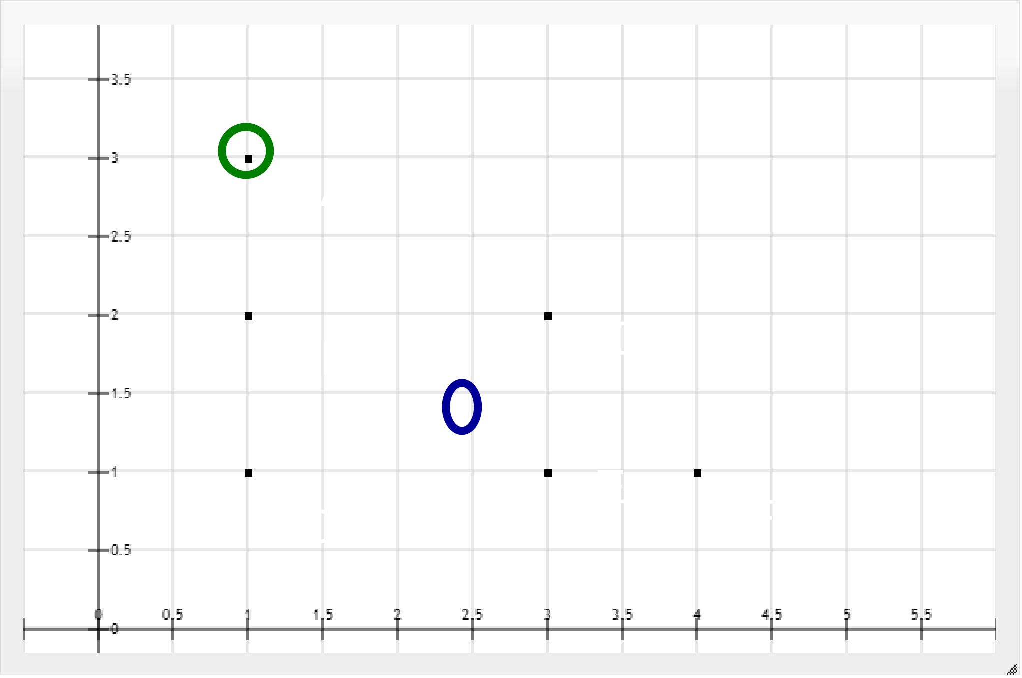
- Assume A and B were randomly picked as initial centroids.
- Computes the distance for each points

Points	Centroids 1 (A)	Centroids 2(B)
A (1,3)	***0***	1
B (1,2)	1	***0***
C (1,1)	2	***1***
D (3,2)	sqrt(5)	***2***
E (3,1)	sqrt(8)	***sqrt(5)***
F (4,1)	sqrt(13)	***sqrt(10)***

Compute New Centroids #2

- New centroids:
- Centroids 1: A
- Centroids 2: Mean of (B,C,D,E,F)

$$(x, y) = \left(\frac{1+1+3+3+4}{5}, \frac{2+1+2+1+1}{5}\right) = (2.4, 1.4)$$



Iteration #2

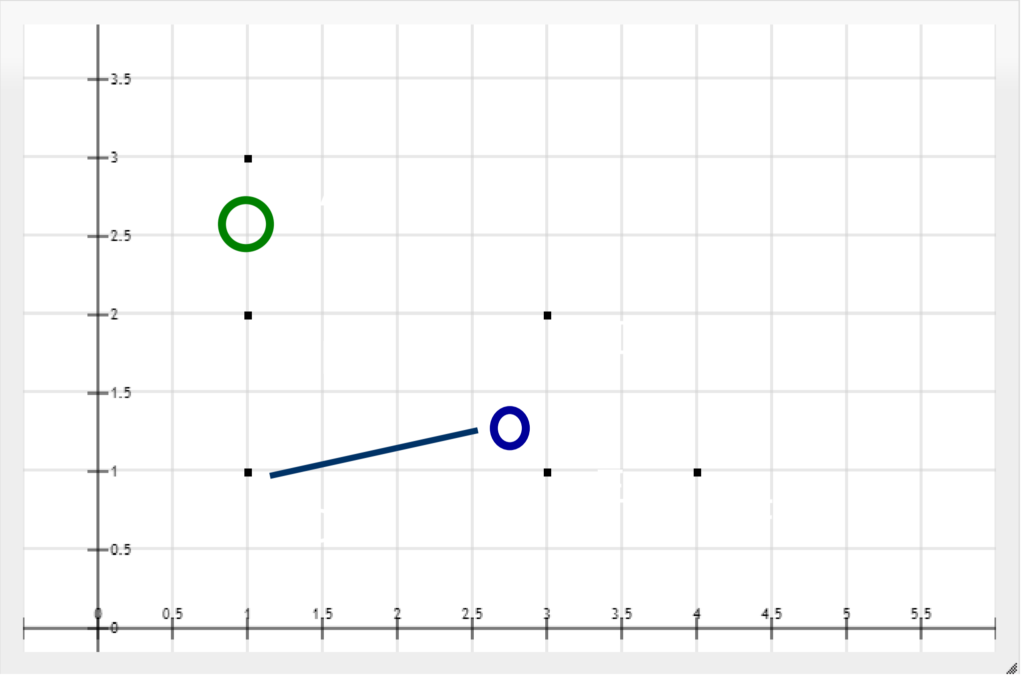
- Computes the distance for each points

Points	Centroids 1 (A)	Centroids 2(2.4,1.4)
A (1,3)	***0***	2.13
B (1,2)	***1***	1.523
C (1,1)	2	***1.46***
D (3,2)	2.24	***0.85***
E (3,1)	2.83	***0.72***
F (4,1)	3.61	***1.65***

Compute New Centroids #3

- New centroids:
- Centroids 1: Mean of (A, B) = (1, 2.5)
- Centroids 2: Mean of (C,D,E,F)

$$(x, y) = \left(\frac{1+3+3+4}{4}, \frac{1+2+1+1}{4}\right) = (2.75, 1.25)$$



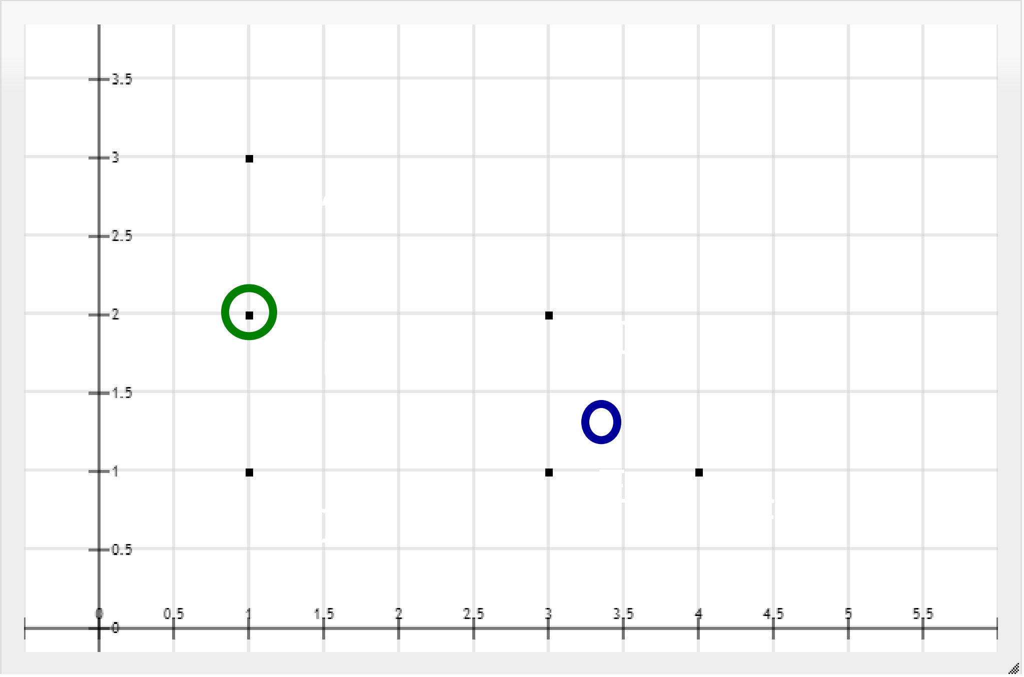
Distance between c and new centroids 2 is1.77

Thus, new group is (A,B,C), (D,E,F)

Compute New Centroids #4

- New centroids:
- Centroids 1: Mean of (A, B, C) = B
- Centroids 2: Mean of (D,E,F)

$$(x, y) = (\frac{3+3+4}{3}, \frac{2+1+1}{3}) = (3.33, 1.33)$$



Obviously the new group is (A,B,C), (D,E,F).

Stop here.

K-means: The Objective Function for Optimization

- The K-means objective function

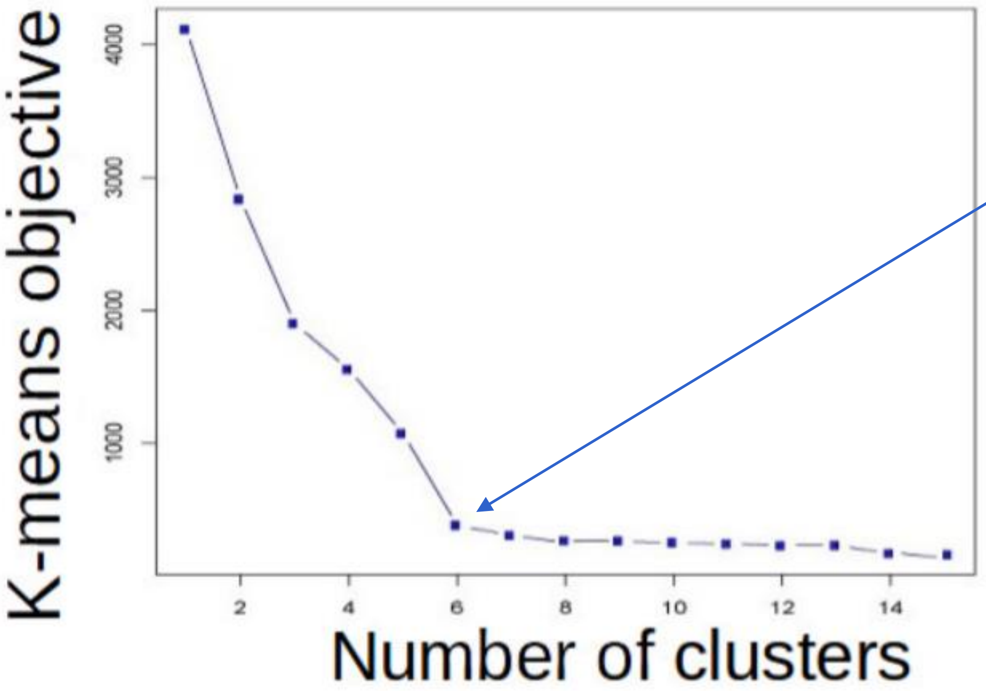
$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} \gamma_{ik} \|\mathbf{x}_i - \mathbf{c}_k\|^2$$

- K-means algorithm is a heuristic to optimize this function. It works iteratively between two steps
  - Fix cluster centers $\mathbf{c}_k$, find best γ_{ik} (assign data points to cluster)
  - Fix γ_{ik}, find the best $\mathbf{c}_k$ (re-compute the cluster center)

- Convergence of K-means algorithm
  - Each step can never increase the objective

How to choose *K* (number of clusters)

- One way to select *K* for the *K*-means algorithm is to try different values of *K*, plot the *K*-means objective versus *K*, and look at the “elbow-point” in the plot.



K = 6 is the elbow point.

K-means: Initialization Issues

- *K*-means is extremely sensitive to cluster center initialization

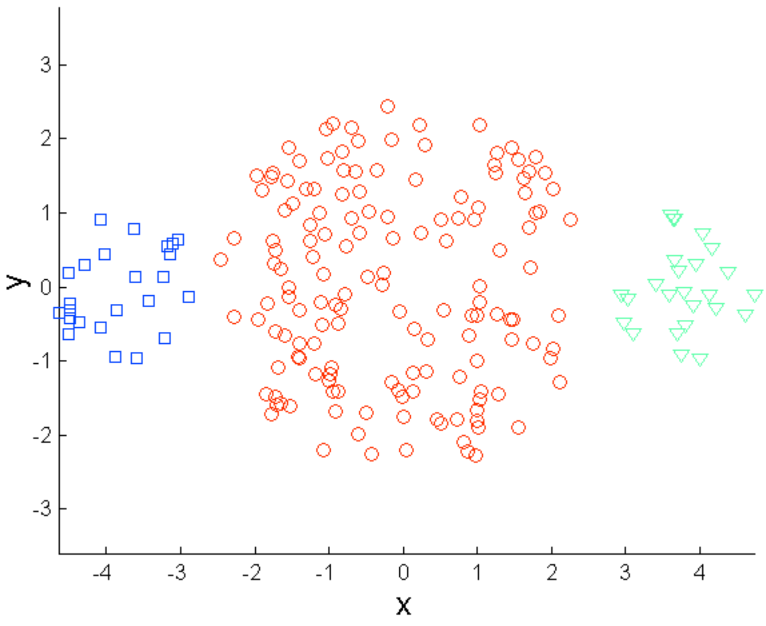
- Bad initialization can lead to
  - Poor convergence speed
  - Bad overall clustering

- Possibly solutions
  - Choose the first center as one of the samples, the second which is the farthest from the first, the third which is the farthest from both, and so on.
  - Try multiple initializations and choose the best result.

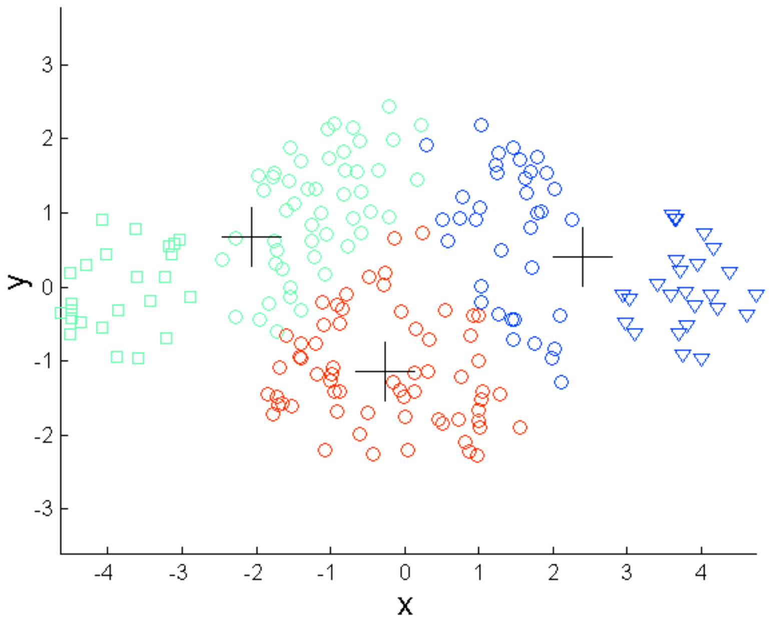
K-means: Limitations

- 1. K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes
- 2. Makes hard assignments of points to clusters
  - A point either completely belongs to a cluster or does not belong
  - Soft assignment ignored (i.e., probability of being assigned to each cluster: say $K = 3$ for some points $\mathbf{x}_i$, $p_1 = 0.7$, $p_2 = 0.2$, $p_3 = 0.1$)

Limitations of K-means: Differing Sizes

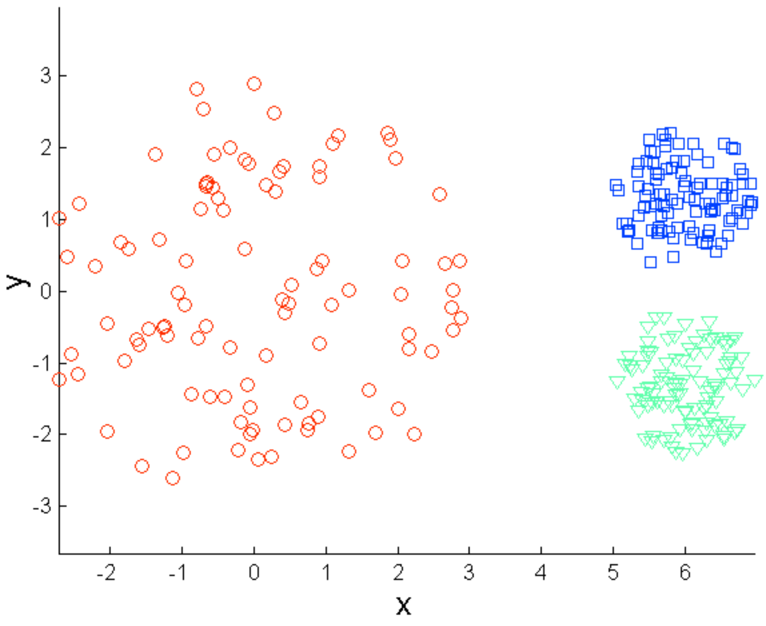


Original Points

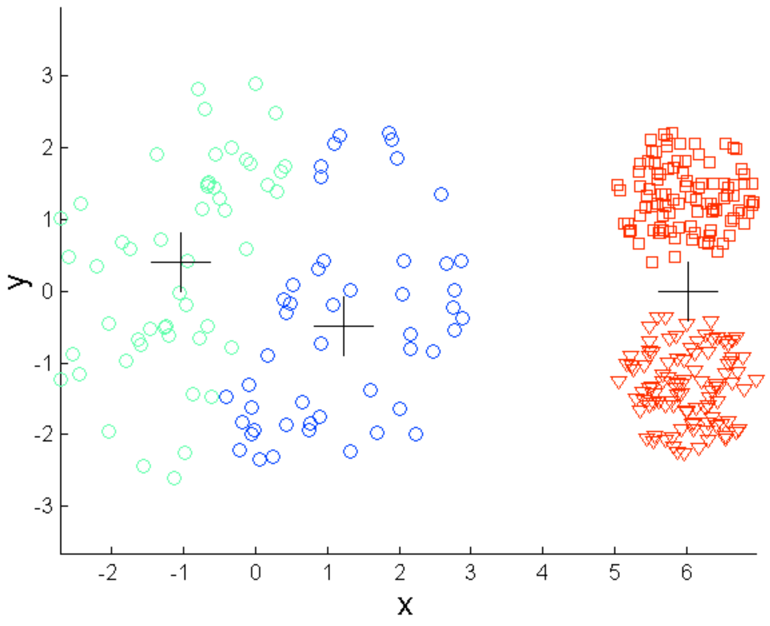


K-means (3 Clusters)

Limitations of K-means: Differing Density

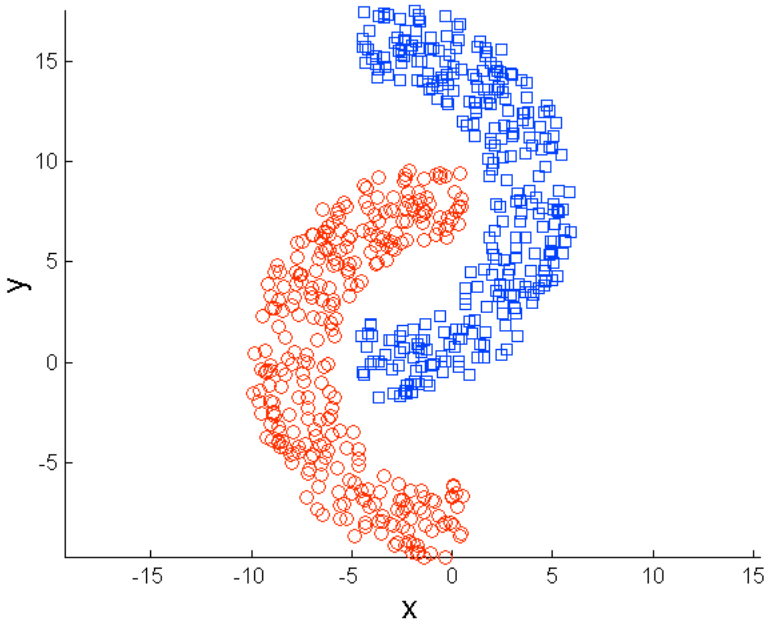


Original Points

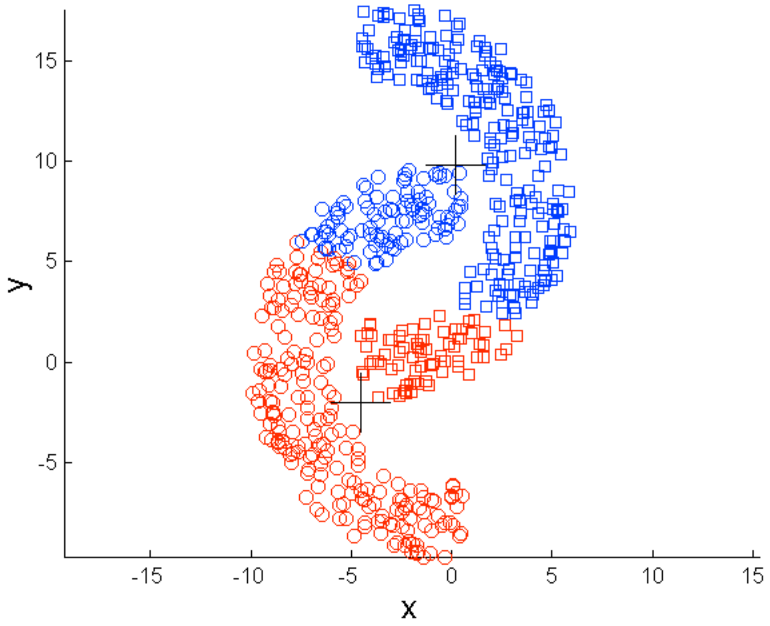


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes



Original Points

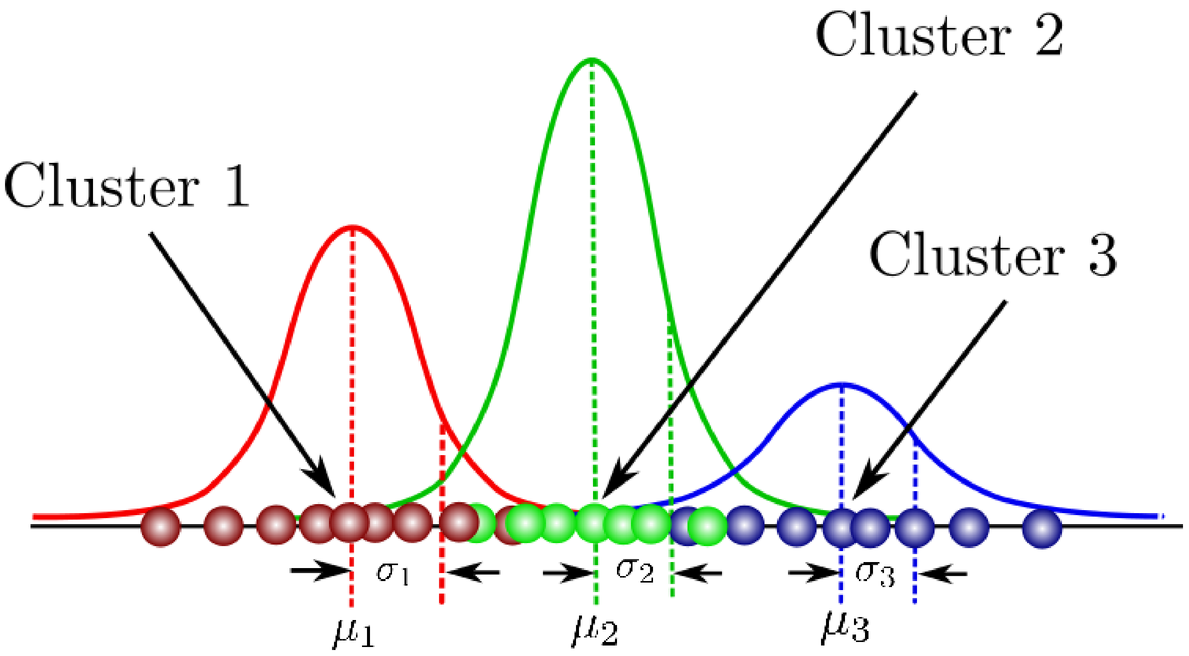


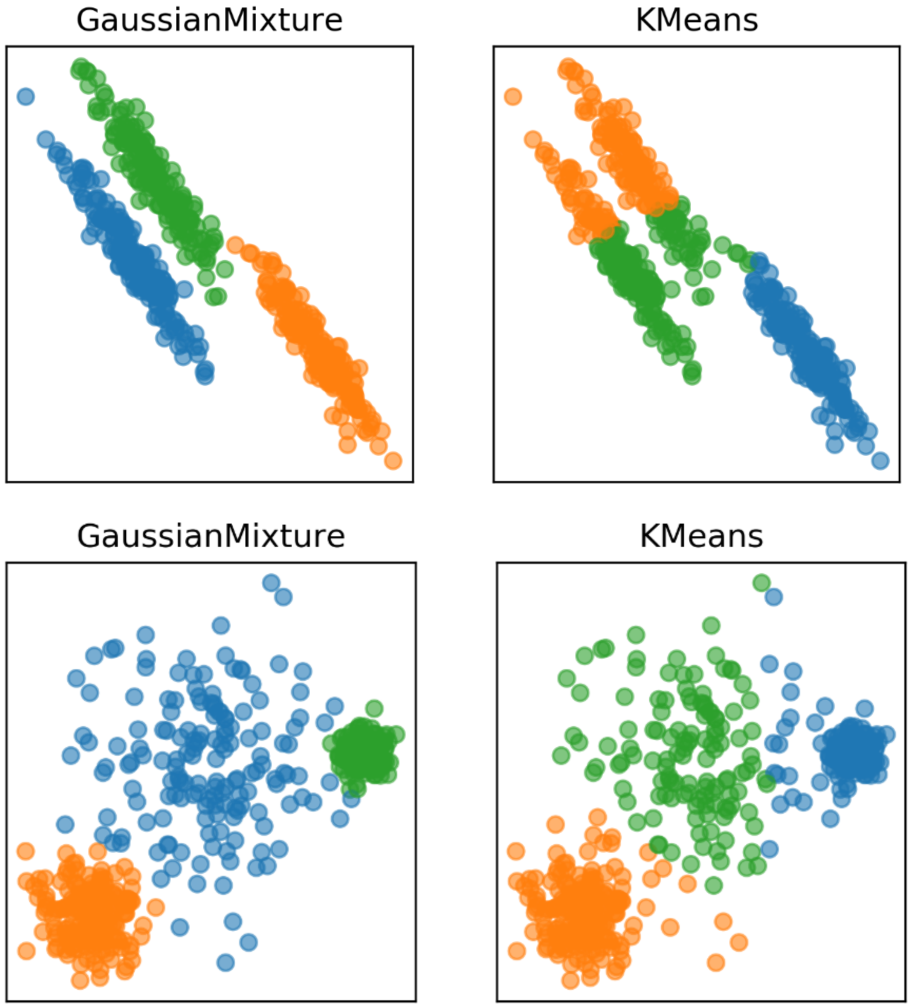
K-means (2 Clusters)

K-means: Limitations

- 1. K-means has problems when clusters are of differing
  - Sizes (*Gaussian Mixture Models*)
  - Densities (*Gaussian Mixture Models*)
  - Non-globular shapes (Kernel *K*-means)
- 2. Makes hard assignments of points to clusters (*Gaussian Mixture Models*)
  - A point either completely belongs to a cluster or does not belong
  - Soft assignment ignored (i.e., probability of being assigned to each cluster: say $K = 3$ for some points $\mathbf{x}_i$, $p_1 = 0.7$, $p_2 = 0.2$, $p_3 = 0.1$)
- *Solution: Gaussian Mixture Models and* Kernel *K*-means

Gaussian Mixture Models





Kernel *K*-means

- The idea: Replace the Euclidean distance/similarity computations in *K*-means by the kernelized version $d(\mathbf{x}_i, \mathbf{c}_k) = \|\phi(\mathbf{x}_i) - \phi(\mathbf{c}_k)\|$

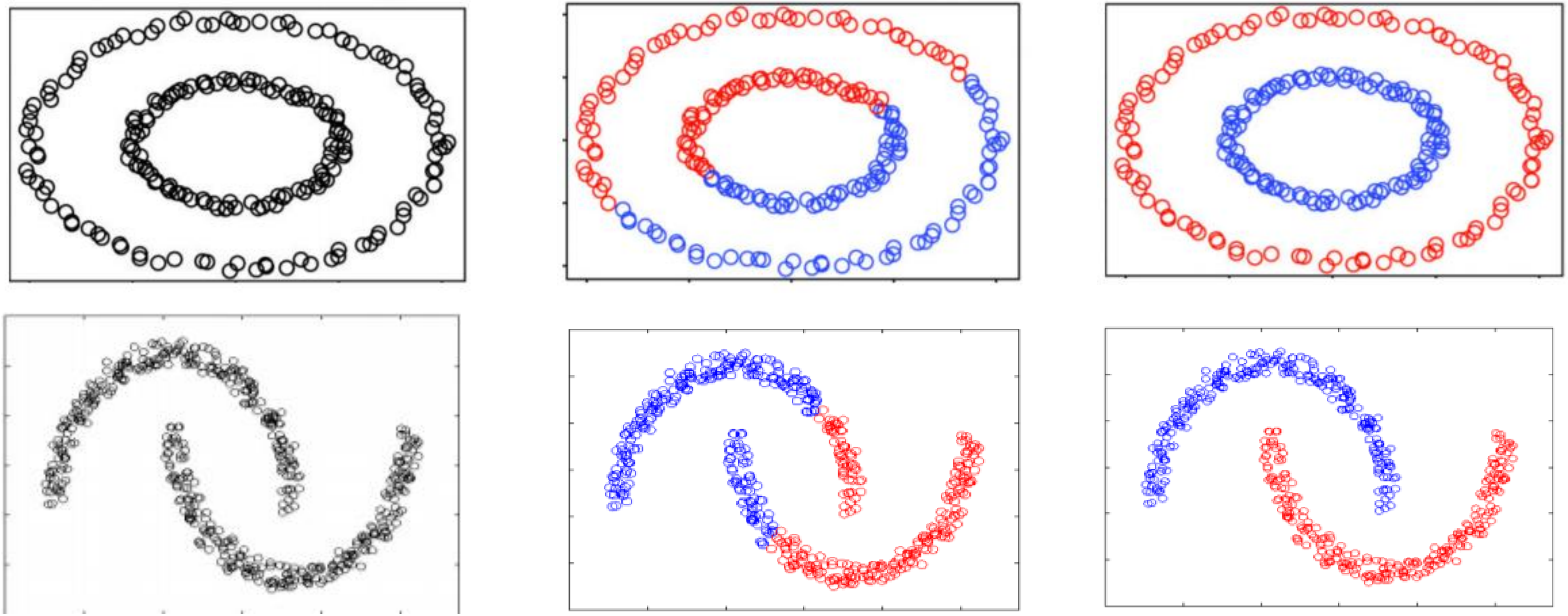
$$
\begin{aligned}
\|\phi(\mathbf{x}_i) - \phi(\mathbf{c}_k)\|^2 &= \|\phi(\mathbf{x}_i)\|^2 + \|\phi(\mathbf{c}_k)\|^2 - 2\phi(\mathbf{x}_i)^T\phi(\mathbf{c}_k) \\
&= k(\mathbf{x}_i, \mathbf{x}_i) + k(\mathbf{c}_k, \mathbf{c}_k) - 2k(\mathbf{x}_i, \mathbf{c}_k)
\end{aligned}
$$

- Here *k*(.,.) denotes the kernel function and ϕ is its (implicit) feature map

- Note: ϕ does not have to be computed/stored because computation only depends on kernel evaluations

K-means vs Kernel *K*-means

Input Data | *K*-means | Kernel *K*-means



Outline for Data Preprocessing and Data Mining

- **Data Preprocessing**
- **Supervised learning**

❖ Regression

1. Linear regression with one variable
2. Linear Regression with multiple variables

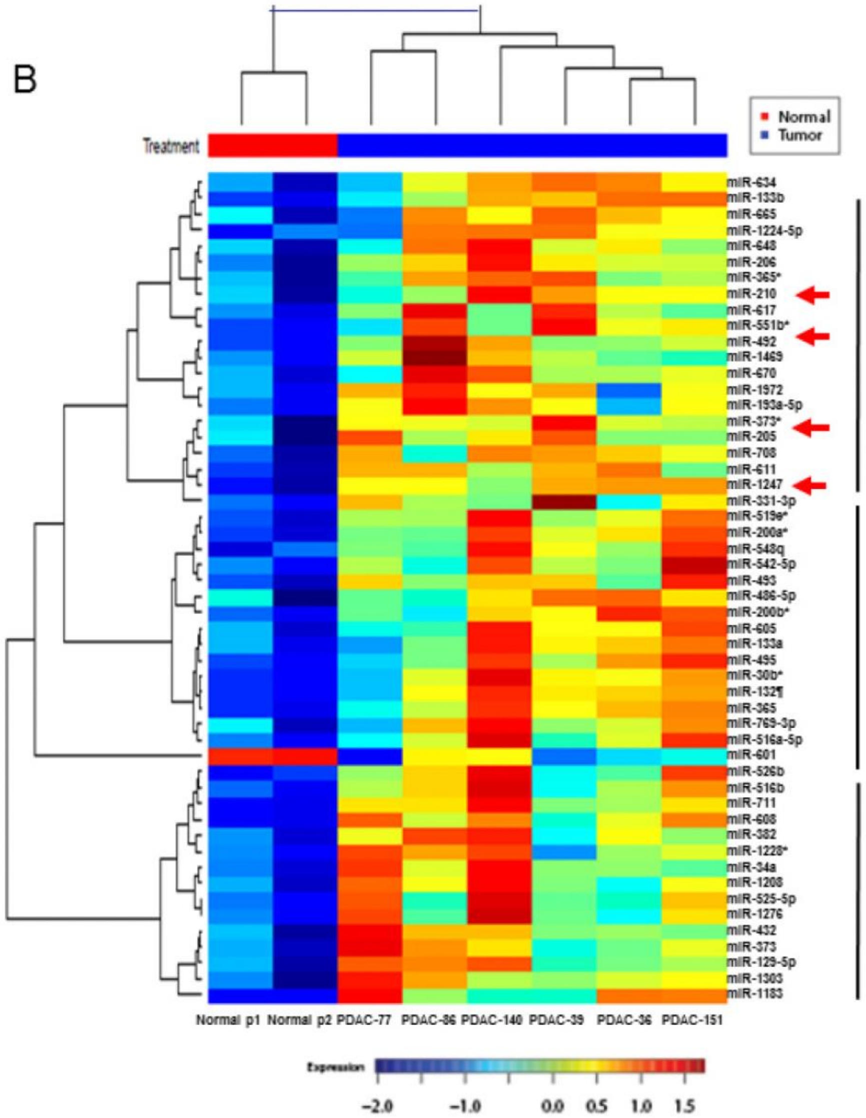
❖ Classification

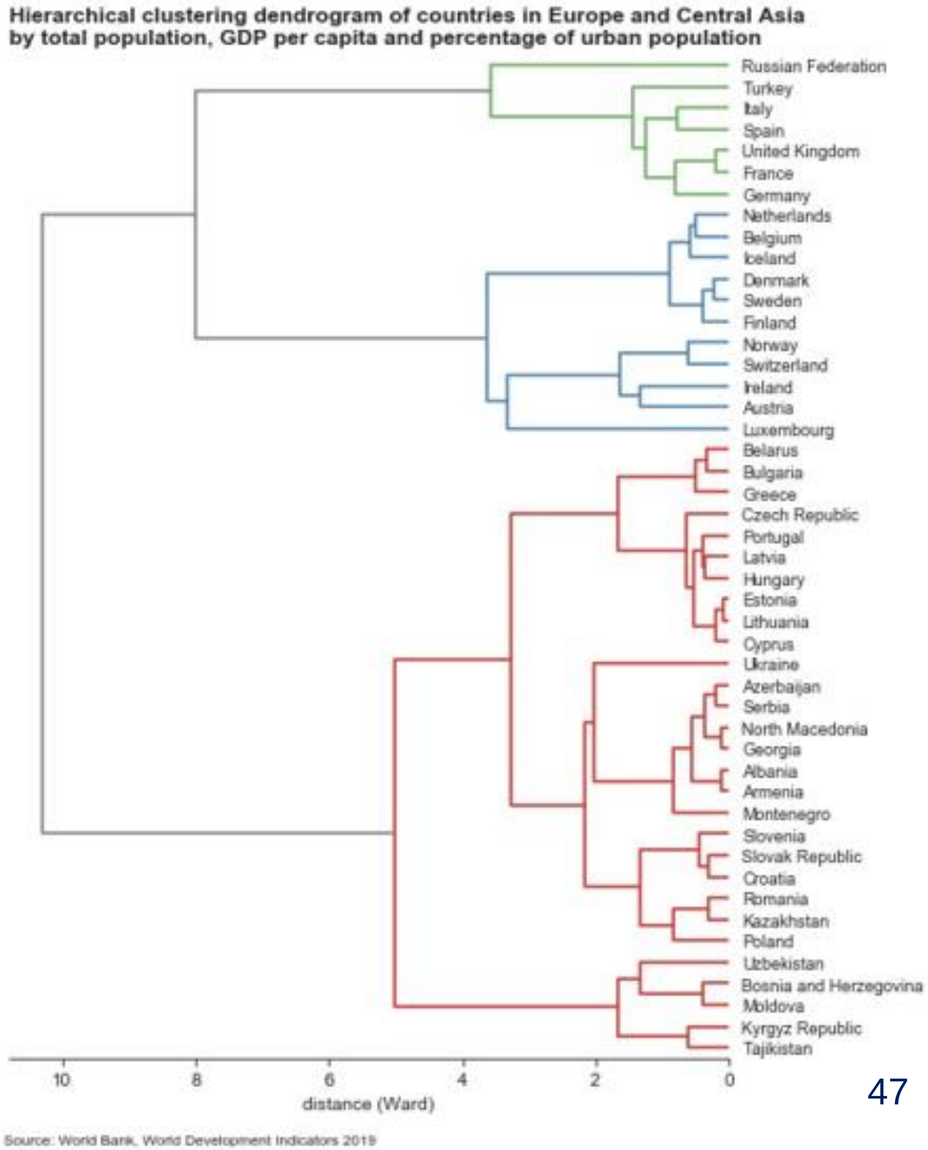
1. Perceptron
2. Artificial Neural Network
3. Support Vector Machine
4. K Nearest Neighbor

- **Unsupervised learning**

1. K-means Clustering
2. Hierarchical Clustering

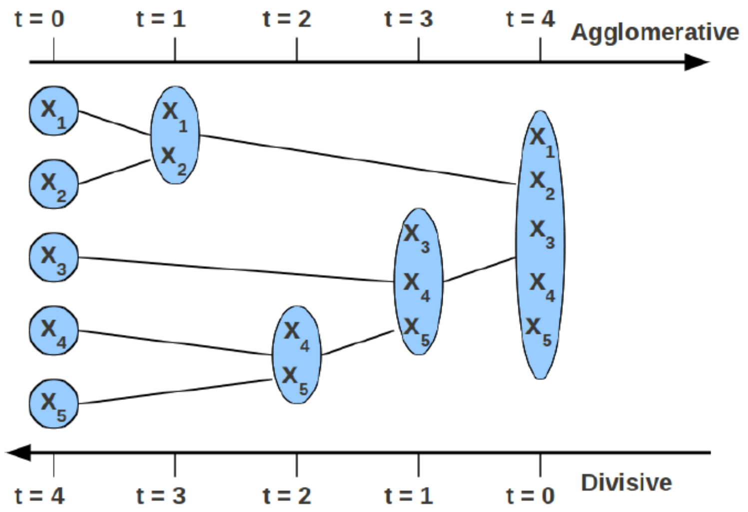
Why Hierarchical Clustering? Some Real-World Examples





Hierarchical Clustering

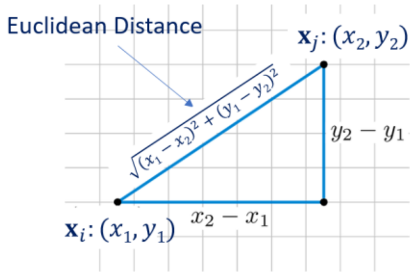
- Agglomerative (bottom-up) clustering
  - Start with each sample in its own singleton cluster.
  - At each iteration, greedily merge two most similar clusters.
  - Stop when there is a single cluster of all samples.
- Divisive (top-down) clustering
  - Start with all samples in a single cluster (i.e, the same cluster)
  - At each iteration, partition cluster(s) into smaller subclusters.
  - Stop when each sample is in its own singleton cluster.
- Agglomerative clustering is more popular and simpler than divisive clustering.



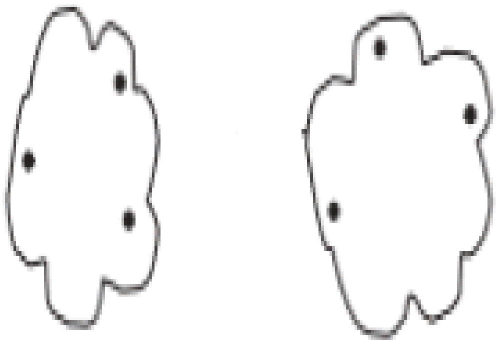
Hierarchical Clustering: (Dis)similarity Between Clusters

- We know how to compute the dissimilarity $d(\mathbf{x}_i,\mathbf{x}_j)$ between two samples (e.g., Euclidean distance).

$$\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2 = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$



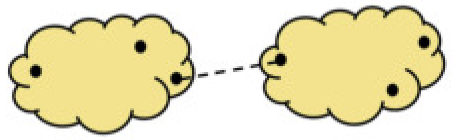
- How to compute the dissimilarity between two clusters R and S?



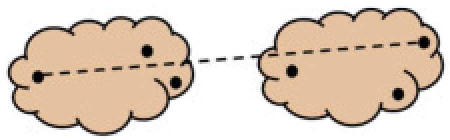
R S

Hierarchical Clustering: (Dis)similarity Between Clusters

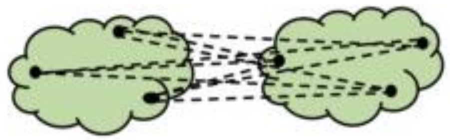
- **Single Linkage**
  - **Smallest** distances between samples, where each one is taken from one of the two groups
- **Complete Linkage**
  - **Largest** distances between samples, where each one is taken from one of the two groups
- **Average linkage**
  - **Average** distance between all samples in one cluster to all points in another cluster
- **Centroid linkage**
  - Distance between their **centroids**.



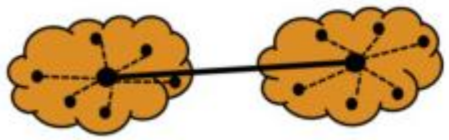
Single Linkage



Complete Linkage

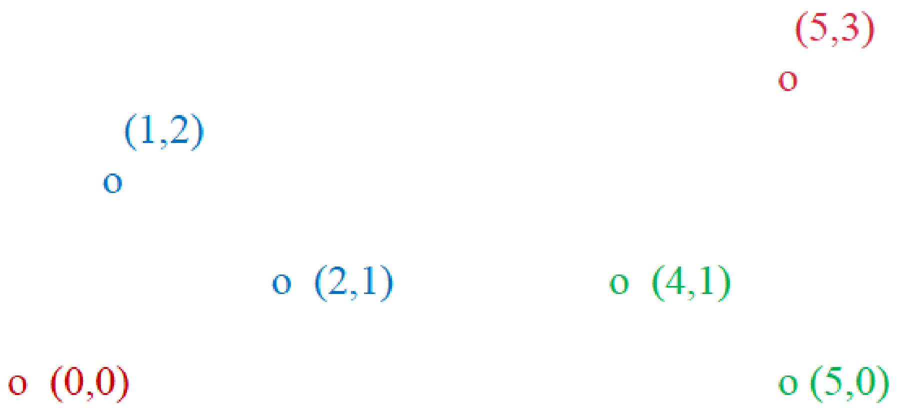


Average Linkage



Centroid Linkage

Example: Hierarchical Clustering (with Centroid Linkage)

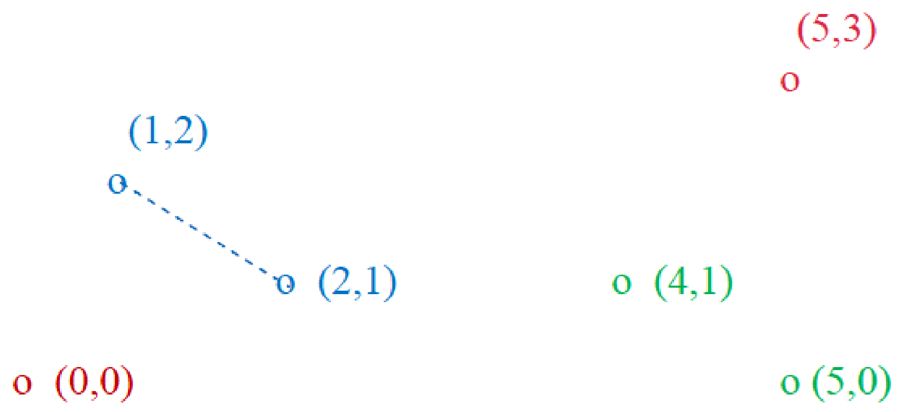


x_1	x_2
0	0
1	2
2	1
4	1
5	0
5	3

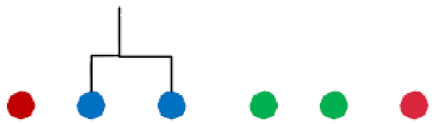
Data:
o … data point

Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 1

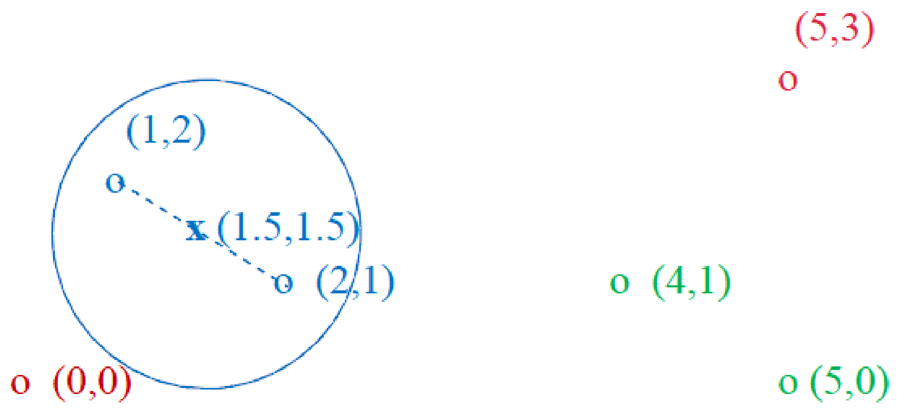


Data:
o … data point

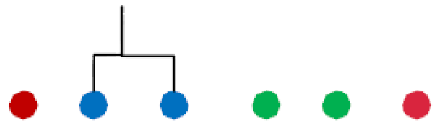


Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 2

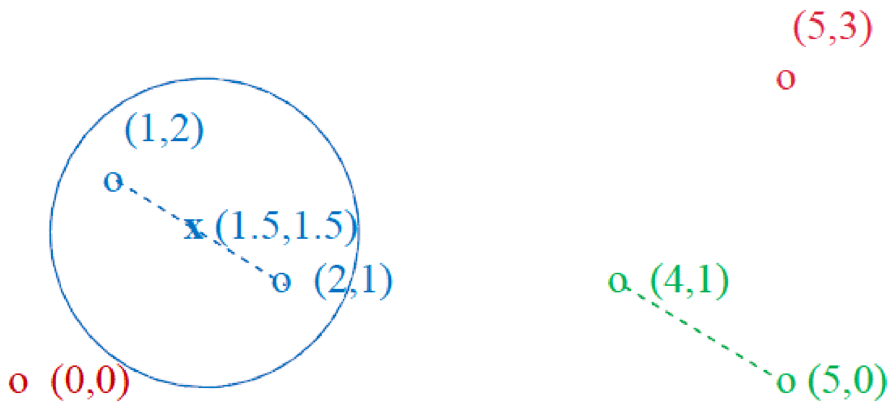


Data:
o … data point
x … centroid

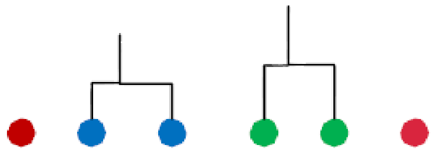


Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 3

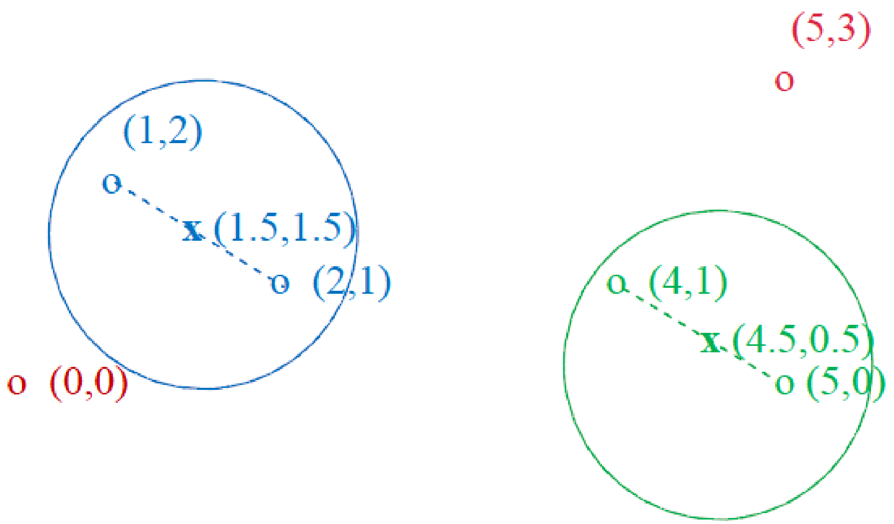


Data:
o … data point
x … centroid

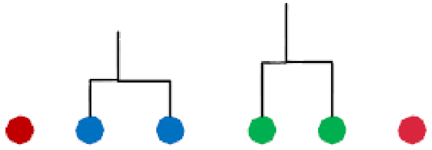


Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 4

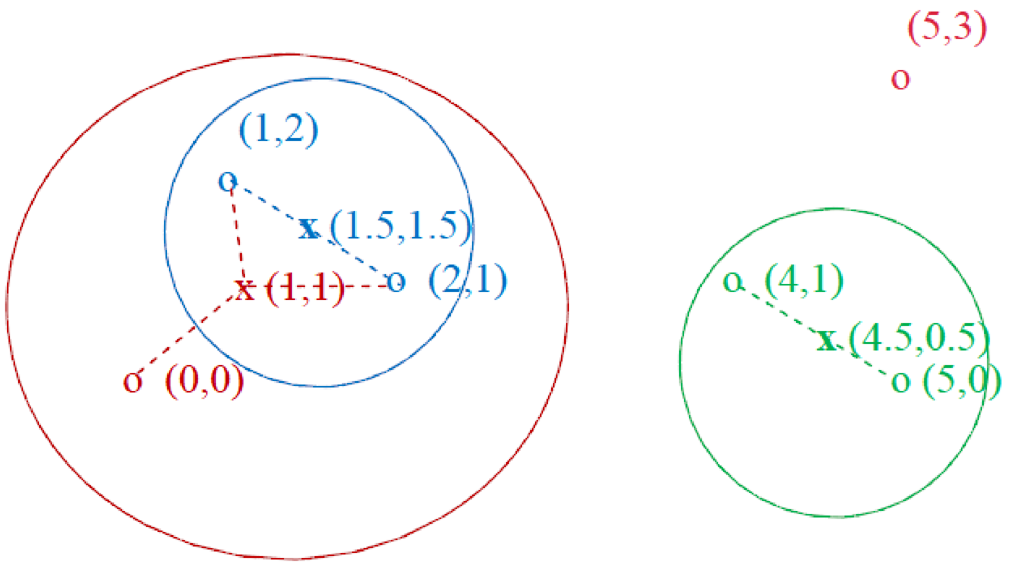


Data:
o … data point
x … centroid

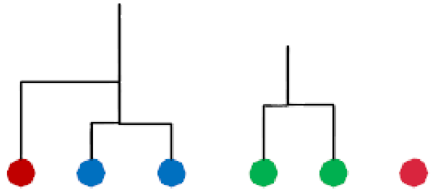


Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 5

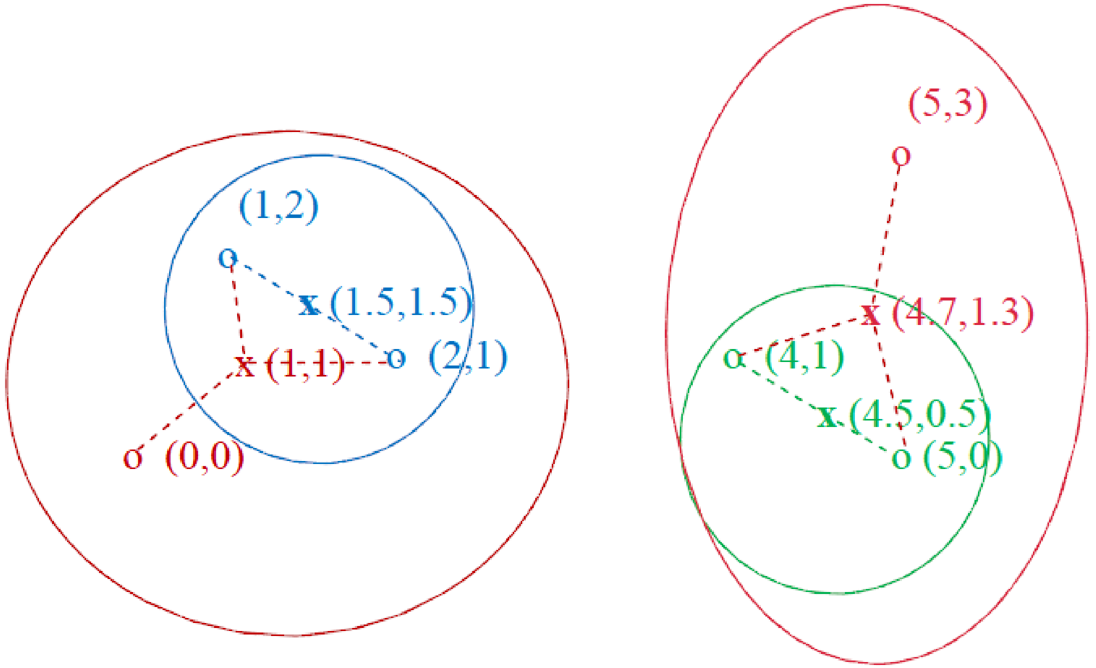


Data:
o … data point
x … centroid

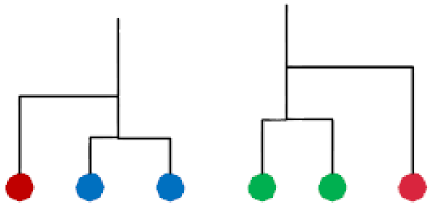


Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 6

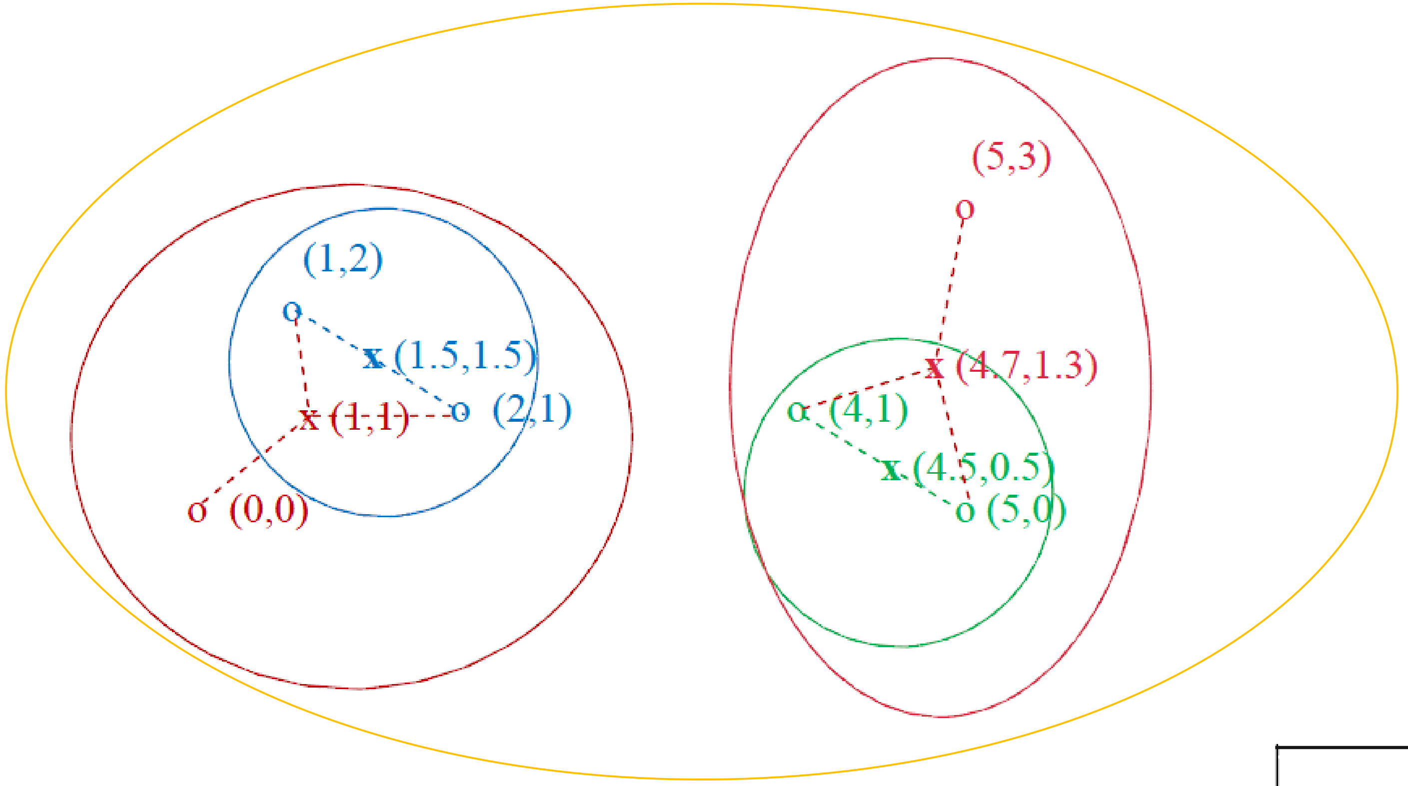


Data:
o … data point
x … centroid

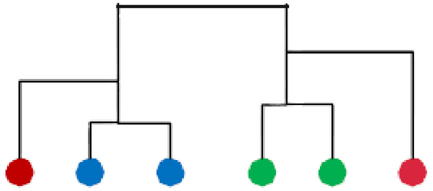


Dendrogram

Example: Hierarchical Clustering (with centroid linkage) Step 7



Data:
o … data point
x … centroid

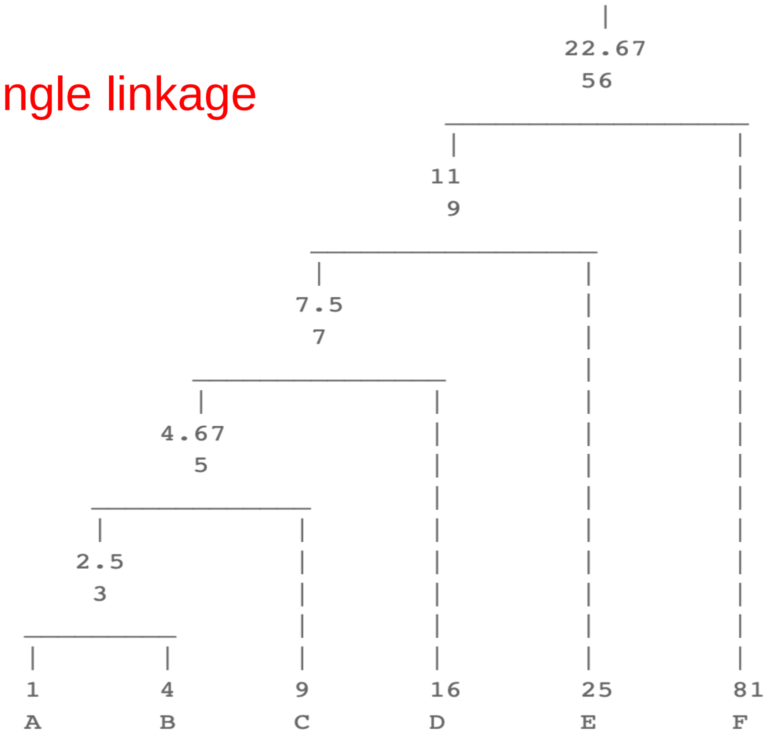


Dendrogram

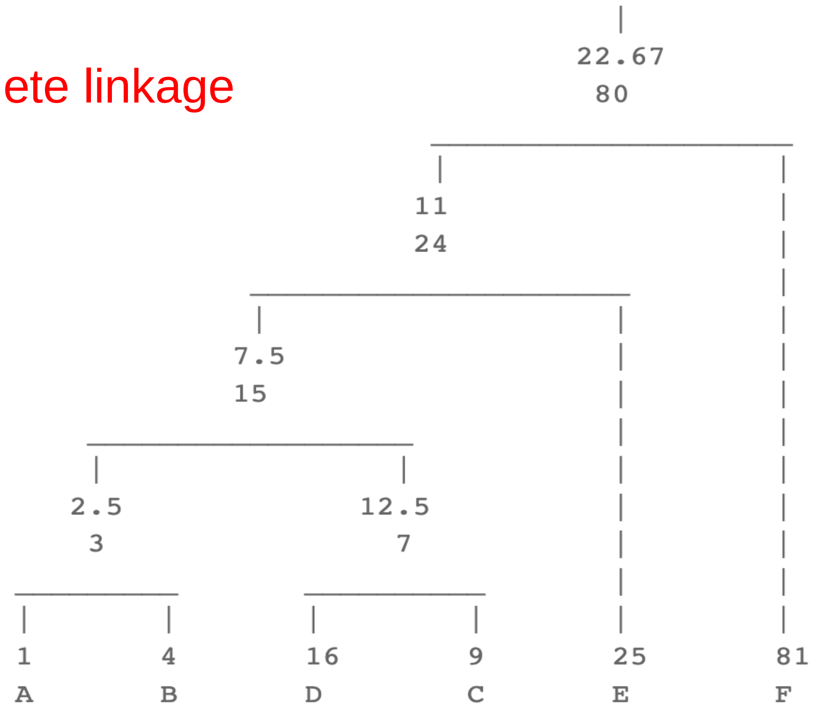
An Exercise on Bottom-up Hierarchical Clustering

- Perform a bottom-up hierarchical clustering on a one-dimensional data set {1, 4, 9, 16, 25, 81} and draw the dendrogram. Assume that the distance between clusters is computed using single linkage or complete linkage

single linkage



complete linkage



Partitional clustering vs Hierarchical Clustering

- Partitional clustering (e.g., *k*-means) produces a single partitioning.
- Hierarchical Clustering can give different partitionings depending on the level-of-granularity we are looking at.
- Partitional clustering needs the number of clusters to be specified.
- Hierarchical clustering doesn’t need the number of clusters to be specified.
- Partitional clustering is usually more efficient.
- Hierarchical clustering can be slow (due to the merge/split decisions)
- No clear consensus on which of the two produces better clustering.