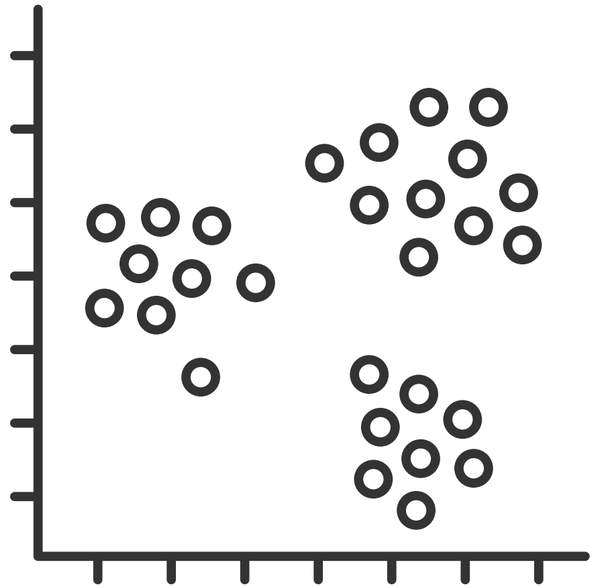
ARTIFICIAL INTELLIGENCE

BACS2003|BACS3074|BMCS2003

CHAPTER 8 MACHINE LEARNING (UNSUPERVISED LEARNING)

OUTCOMES

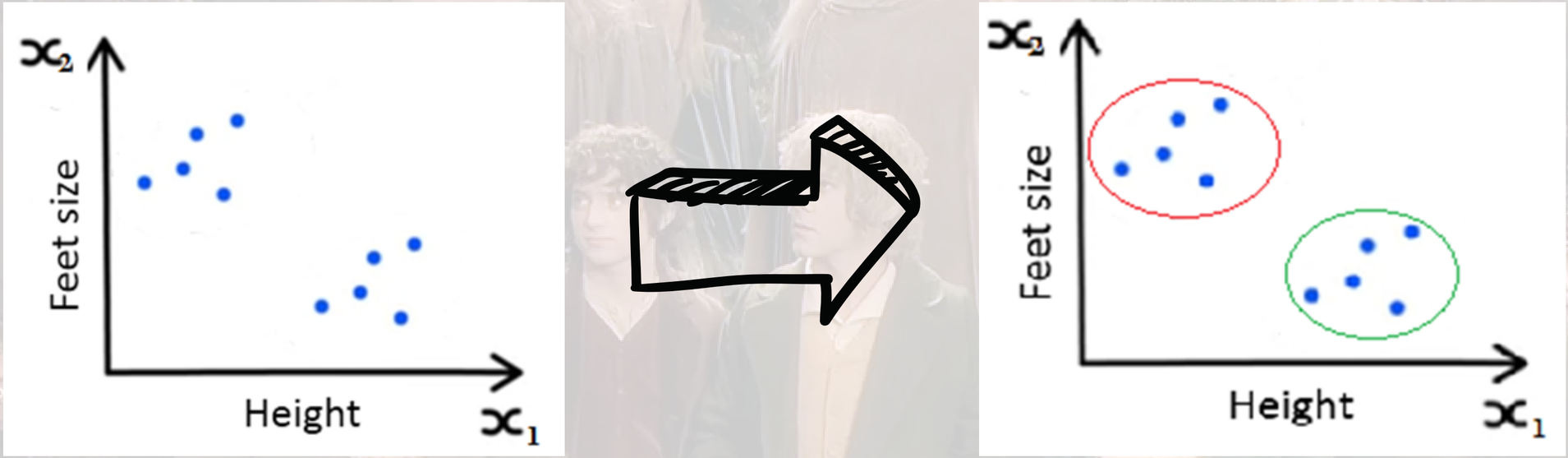
1. **Process flow of Unsupervised ML (UML)**
2. **Types of UML**
3. **Assessing UML performance**



UNSUPERVISED LEARNING

GENERAL IDEA

Categorized data based on similarity

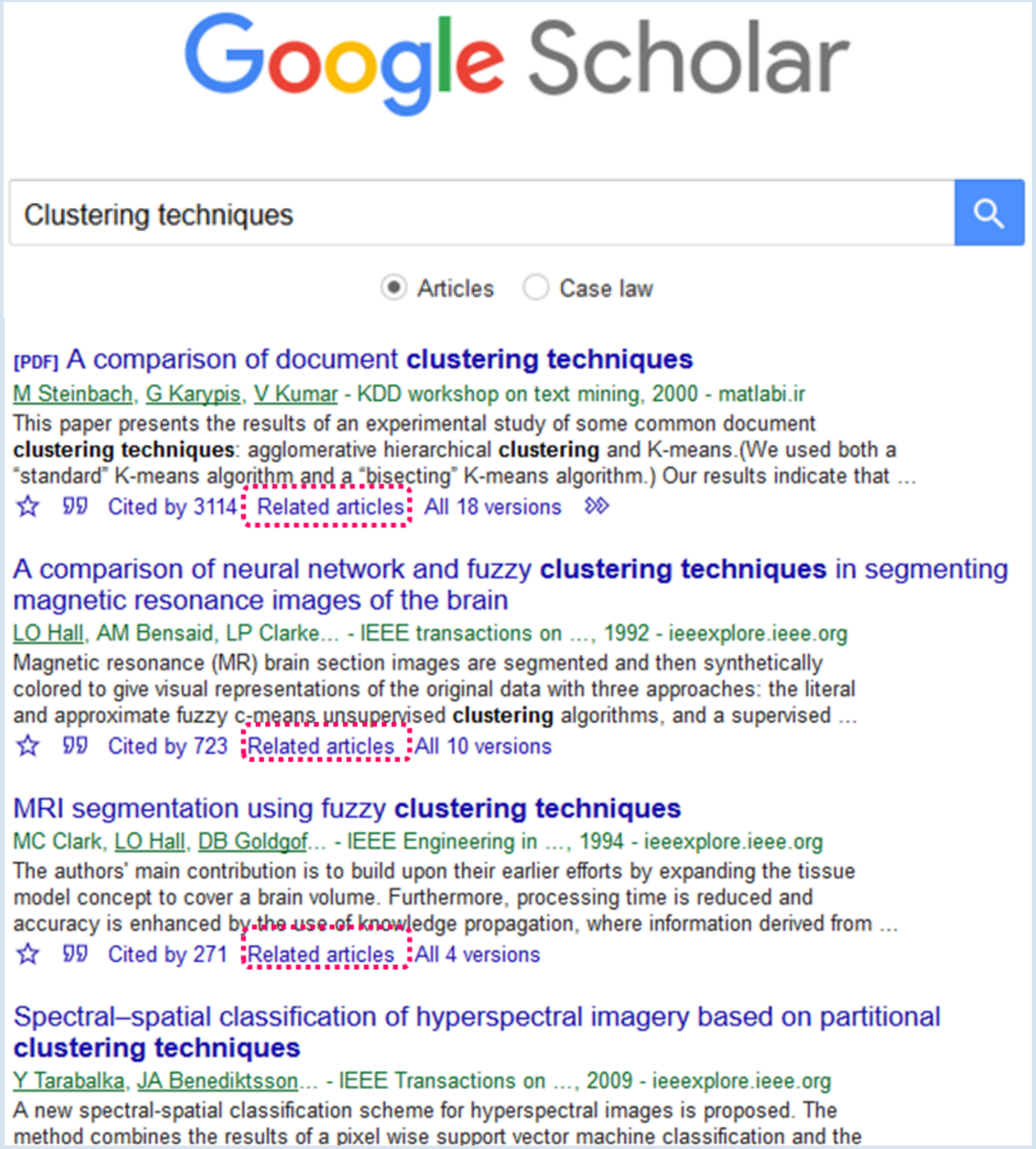


APPLICATION

- Search Engine
- Business and Marketing
- Image segmentation
- Medical Imaging

APPLICATION

- Search Engine
- Business and Marketing
- Image segmentation
- Medical Imaging



APPLICATION

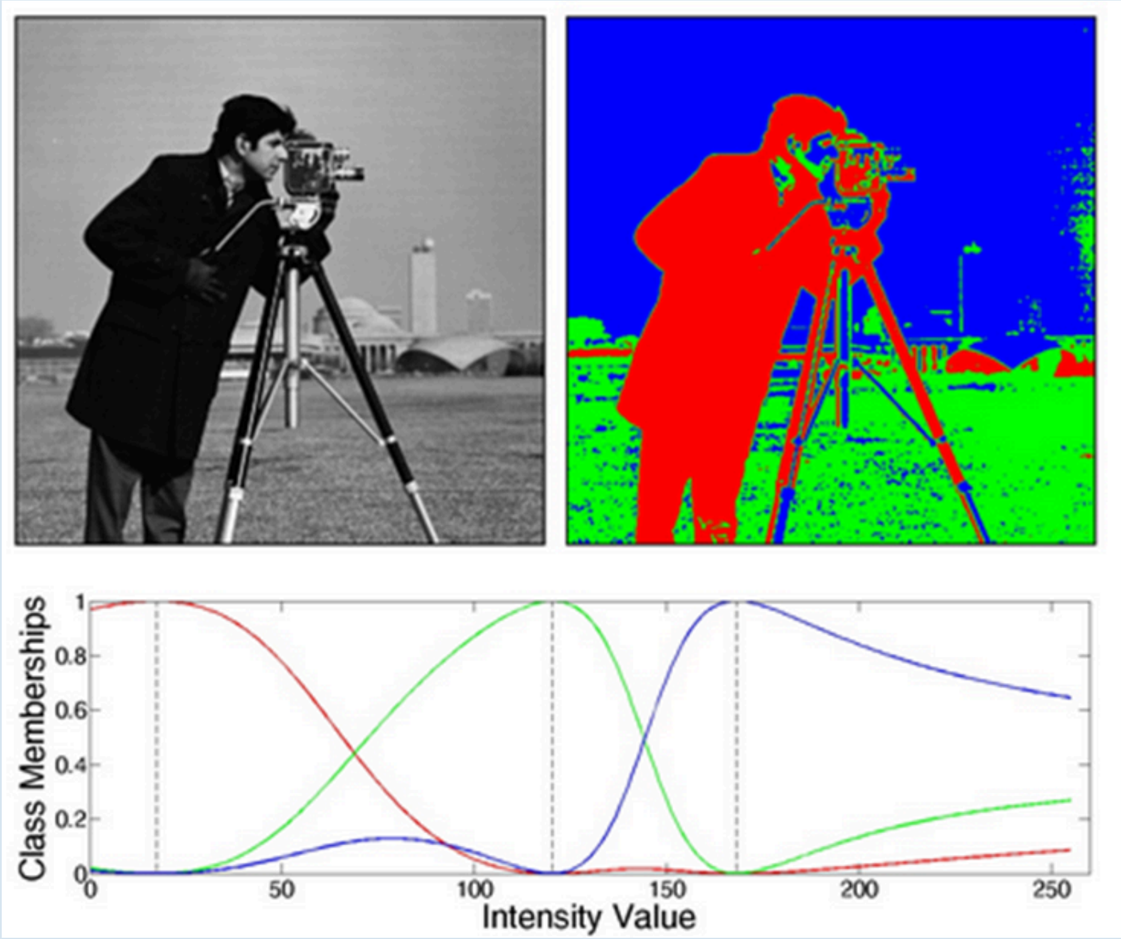
- Search Engine
- Business and Marketing
- Image segmentation
- Medical Imaging

Prepaid Telekom Customers segmentation

Total= 76,753	Calls (%)	Data (%)	SMS (%)
Cluster 1	38	29	7
Cluster 2	79	20	1
Cluster 3	46	24	30
Cluster 4	31	42	27
Cluster 5	7	76	17
Cluster 6	44	41	15
Cluster 7	3	94	3

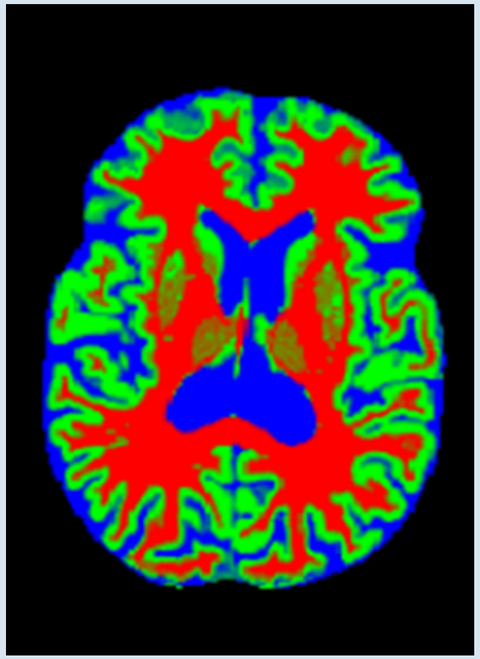
APPLICATION

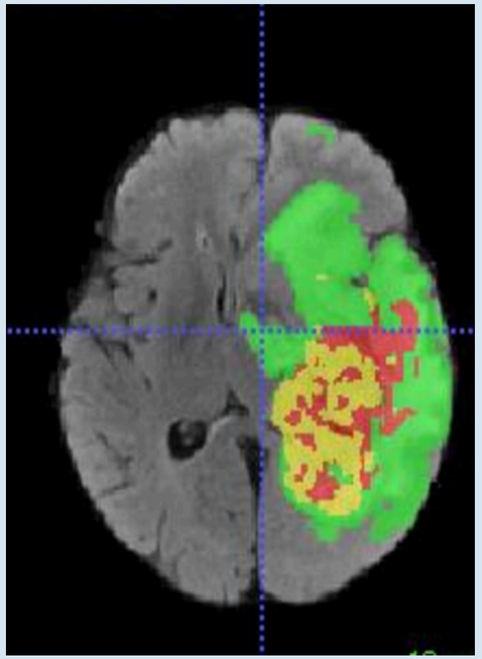
- Search Engine
- Business and Marketing
- Image segmentation
- Medical Imaging



APPLICATION

- Search Engine
- Business and Marketing
- Image segmentation
- Medical Imaging





CLUSTERING METHODS

- K- Means
- Gaussian Mixture Model
- Mean-Shift
- Hierarchical Clustering

K- MEANS

- **K-means partitioning (or clustering) N data points into K disjoint cluster centroid, c_j containing data points so as to minimize the sum-of-squares criterion**

$$J = \sum_{j=1}^{K} \sum_{x \in c_j} \|x - c_j\|^2$$

- Let the set of data points (or instances) $\mathcal{D}$ be $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$,
  - where $\boldsymbol{x}_n = (A_1, A_2, \ldots, A_r)$ is a vector in a real-valued space $A \subseteq R^r$, and r is the number of attributes (dimensions) in the data.

- ***K* is specified by the user**

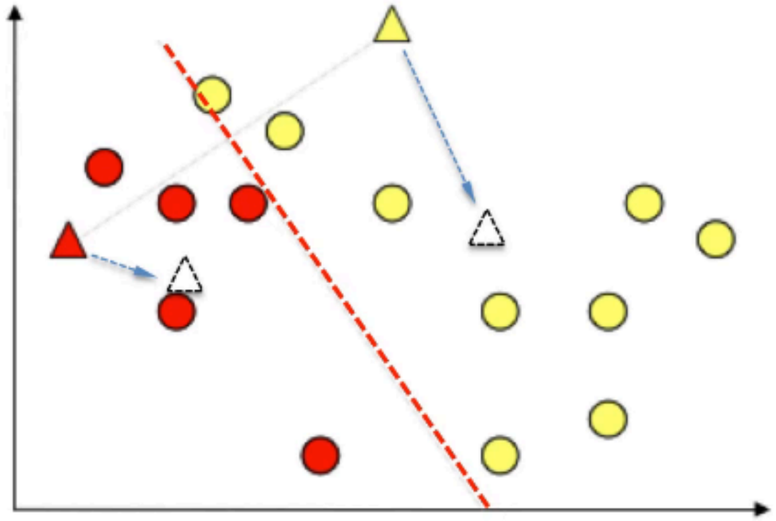
K- MEANS

Algorithm:

- Given ***k***, the *k-means* 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, go to **2**.

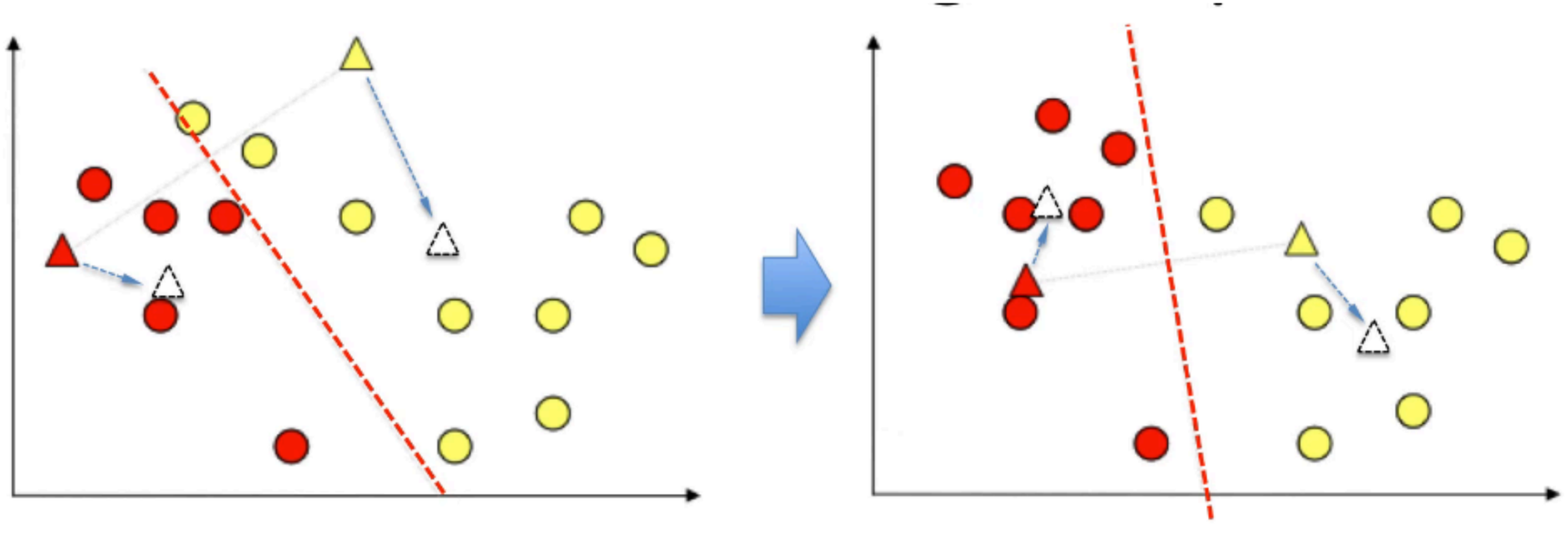
EXAMPLE



Algorithm:

- Given ***k***, the *k-means* 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, go to **2**.

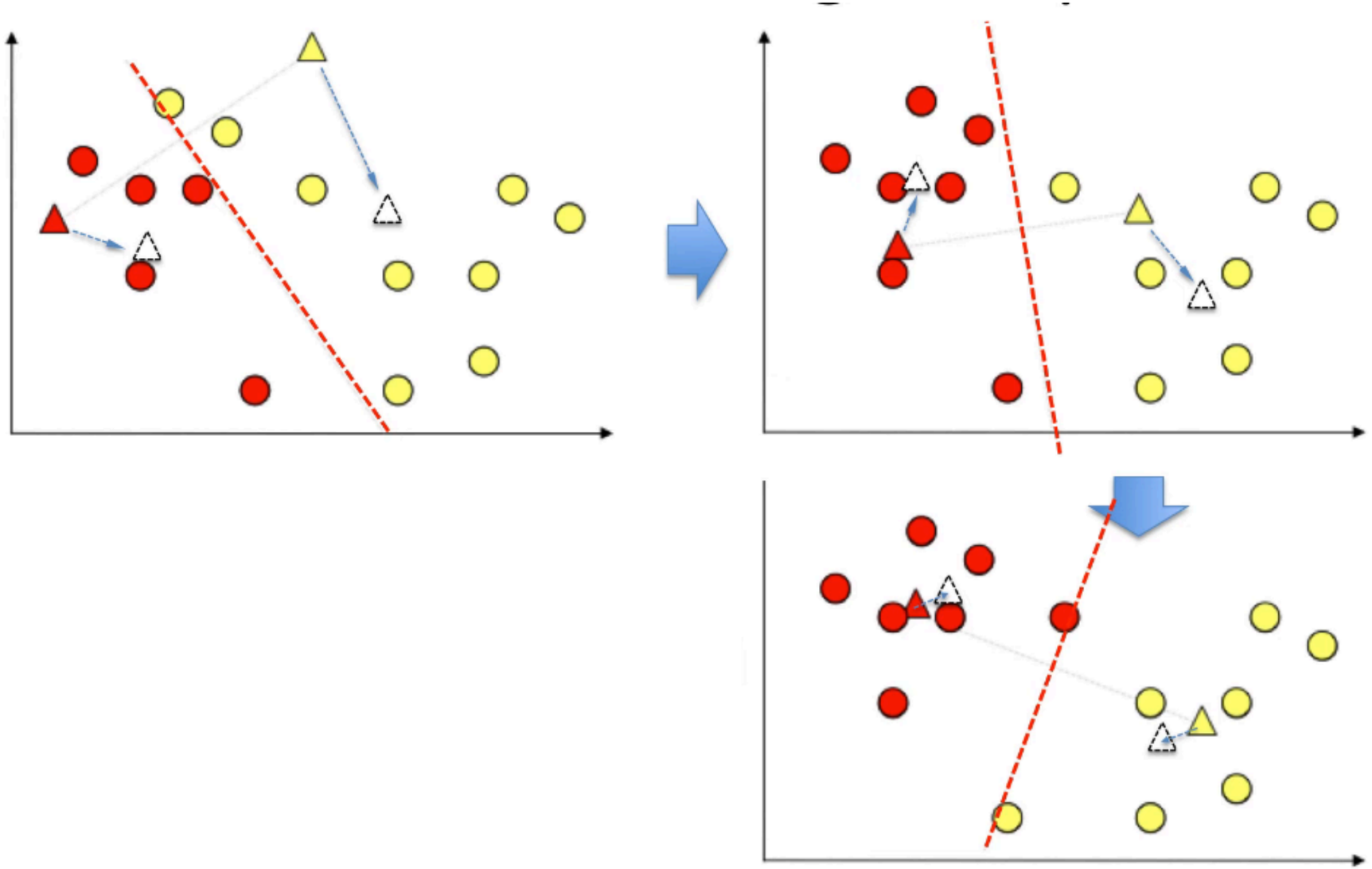
EXAMPLE



Algorithm:

- Given ***k***, the *k-means* 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, go to **2**.

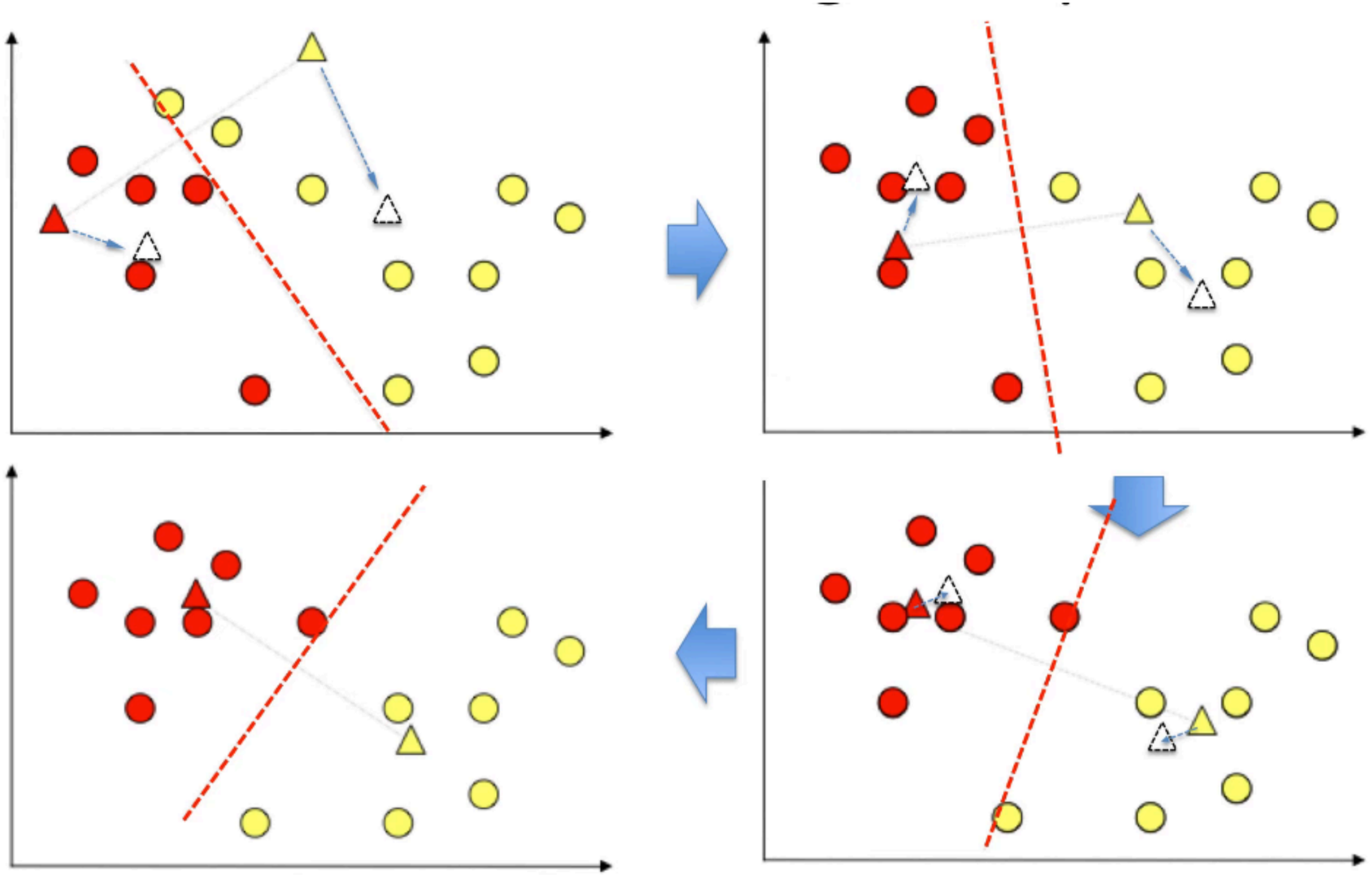
EXAMPLE



Algorithm:

- Given ***k***, the *k-means* 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, go to **2**.

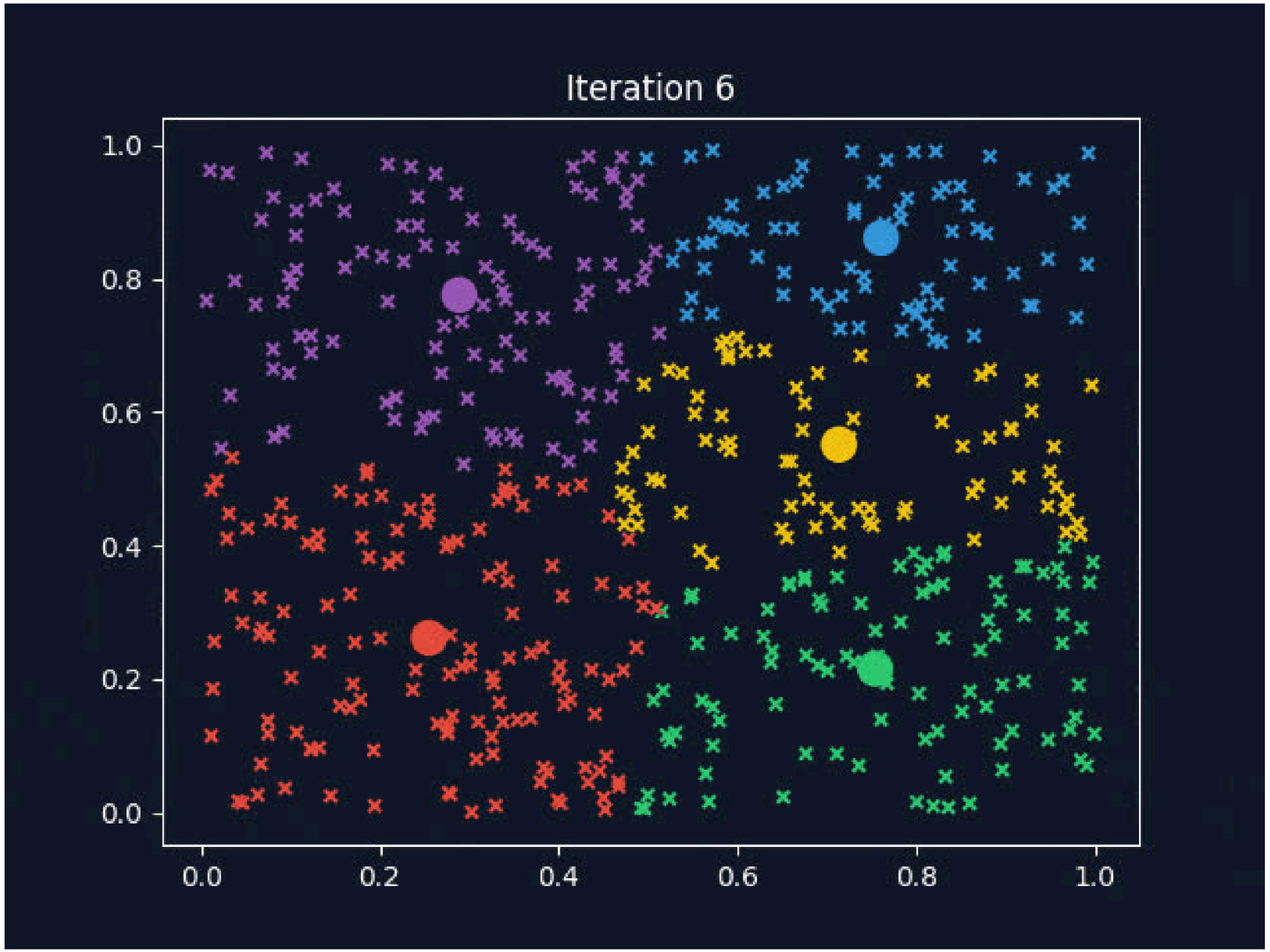
EXAMPLE



Algorithm:

- Given ***k***, the *k-means* 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, go to **2**.

K-MEANS (K=5)



Algorithm:

- Given ***k***, the *k-means* 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, go to **2**.

CALCULATION

Dataset= x1,x2

Data	Height	Weight
1	185	72
2	167	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	169	47
12	177	76

CALCULATION

Dataset= x1,x2

Data	Height	Weight
1	185	72
2	167	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	169	47
12	177	76

1.Assign k=2

2.Randomly guess k cluster Center locations

Centroid,	Height	Weight
c1	**186**	**71**
c2	**165**	**47**

CALCULATION

Dataset= x1,x2

Data	Height	Weight
1	185	72
2	167	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	169	47
12	177	76

1.Assign k=2

2.Randomly guess k cluster Center locations

Centroid,	Height	Weight
c1	**186**	**71**
c2	**165**	**47**

3.Each datapoint finds out which Center it's closest

	Height	Weight	to *c1*	to *c2*	*c1*	*c2*
1	185	72	1.4142	32.0156	☺	

$$d(x_1, c_1) = \sqrt{(185 - \mathbf{186})^2 + (72 - \mathbf{71})^2} = 1.41$$

$$d(x_1, c_2) = \sqrt{(185 - \mathbf{165})^2 + (72 - \mathbf{47})^2} = 32.06$$

CALCULATION

Dataset= x1,x2

Data	Height	Weight
1	185	72
2	167	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	169	47
12	177	76

1.Assign k=2

2.Randomly guess k cluster Center locations

Centroid,	Height	Weight
c1	**186**	**71**
c2	**165**	**47**

3.Each datapoint finds out which Center it's closest

	Height	Weight	to *c1*	to *c2*	*c1*	*c2*
1	185	72	1.4142	32.0156	☺	
2	167	56	24.2074	9.2195		☺
3	168	60	21.0950	13.3417		☺
4	179	68	7.6158	25.2389	☺	
5	182	72	4.1231	30.2324	☺	
6	188	77	6.3246	37.8021	☺	
7	180	71	6.0000	28.3019	☺	
8	180	70	6.0828	27.4591	☺	
9	183	84	13.3417	41.1461	☺	
10	180	88	18.0278	43.6578	☺	
11	169	47	29.4109	4.0000		☺
12	177	76	10.2956	31.3847	☺	

$$d(x_1, c_1) = \sqrt{(185 - \mathbf{186})^2 + (72 - \mathbf{71})^2} = 1.41$$

$$d(x_1, c_2) = \sqrt{(185 - \mathbf{165})^2 + (72 - \mathbf{47})^2} = 32.06$$

CALCULATION

Dataset= x1,x2

Data	Height	Weight
1	185	72
2	167	56
3	168	60
4	179	68
5	182	72
6	188	77
7	180	71
8	180	70
9	183	84
10	180	88
11	169	47
12	177	76

3. Each datapoint finds out which Center it's closest

Height	Weight	to *c1*	to *c2*	*c1*	*c2*
185	72	1.4142	32.0156	☺	
167	56	24.2074	9.2195		☺
168	60	21.0950	13.3417		☺
179	68	7.6158	25.2389	☺	
182	72	4.1231	30.2324	☺	
188	77	6.3246	37.8021	☺	
180	71	6.0000	28.3019	☺	
180	70	6.0828	27.4591	☺	
183	84	13.3417	41.1461	☺	
180	88	18.0278	43.6578	☺	
169	47	29.4109	4.0000		☺
177	76	10.2956	31.3847	☺	

4.Each center finds the new centroid points

New Centroid,	Height	Weight
c1	**181.56**	**75.33**
c2	**168.00**	**54.33**

$$c_1 = \left(\frac{1}{9}(185 + 179 + \cdots), \quad \frac{1}{9}(72 + 68 + \cdots)\right)$$

$$c_2 = \left(\frac{1}{3}(167 + 168 + \cdots), \quad \frac{1}{3}(56 + 60 + \cdots)\right)$$

5.Repeat step 3 & 4 until convergence.

CONVERGENCE

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** J,

$$J = \sum_{j=1}^{K} \sum_{x \in c_j} \|x - c_j\|^2$$

LIMITATIONS

1. ***Very* sensitive to the initial points**.
   - ➢ Do many runs of k-Means, each with different initial centroids.

2. Must **manually choose k**.
   - ➢ Learn the optimal k for the clustering.

   (Note that this requires a performance measure.)

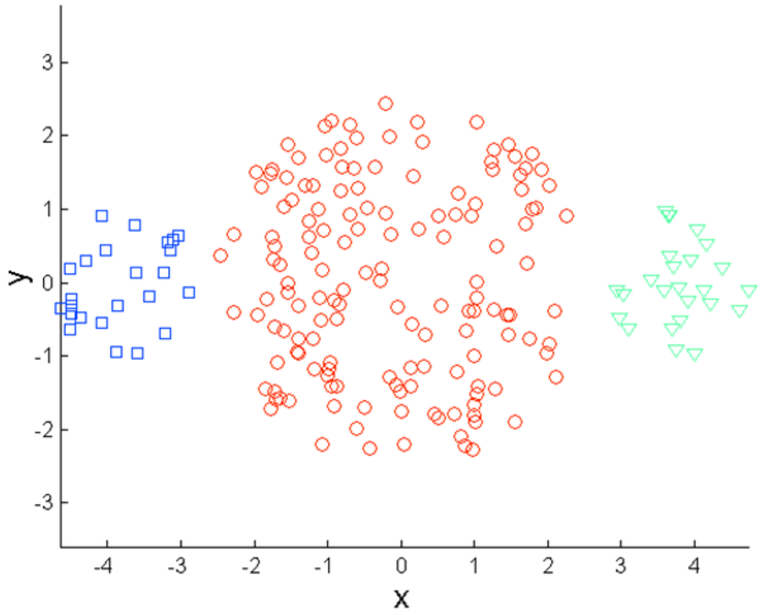
```
from sklearn.cluster import KMeans
from sklearn.datasets import

# Apply K-Means clustering
kmeans = KMeans(n_clusters=n_clusters)
kmeans.fit(X)
```

LIMITATIONS

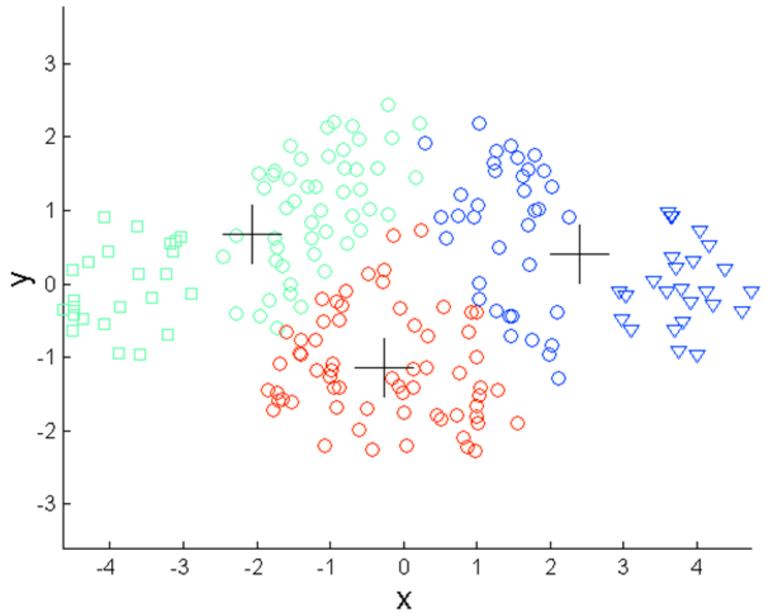
3. K-means has problems when **clusters are of differing**
   - Sizes
   - Densities
   - Non-globular shapes

4. K-means has problems when the data contains **outliers**.

PROBLEM WITH K-MEANS: DIFFERING SIZES

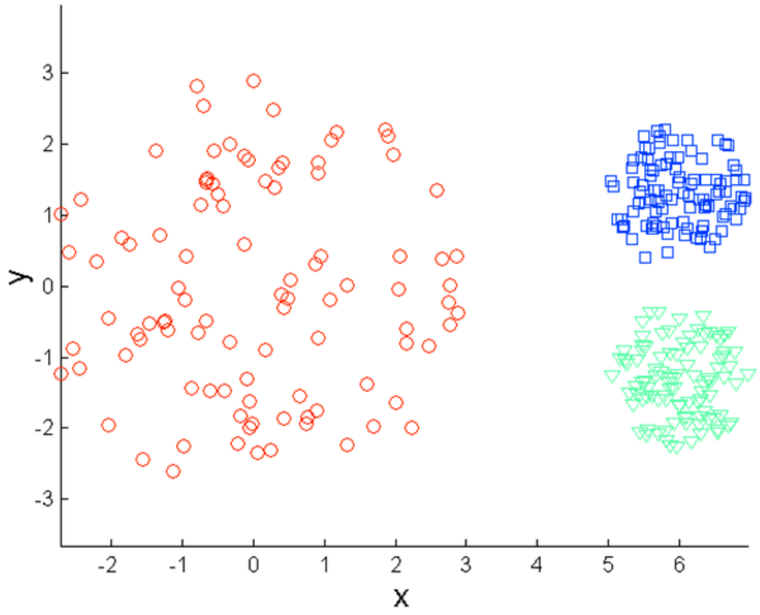


Original Points

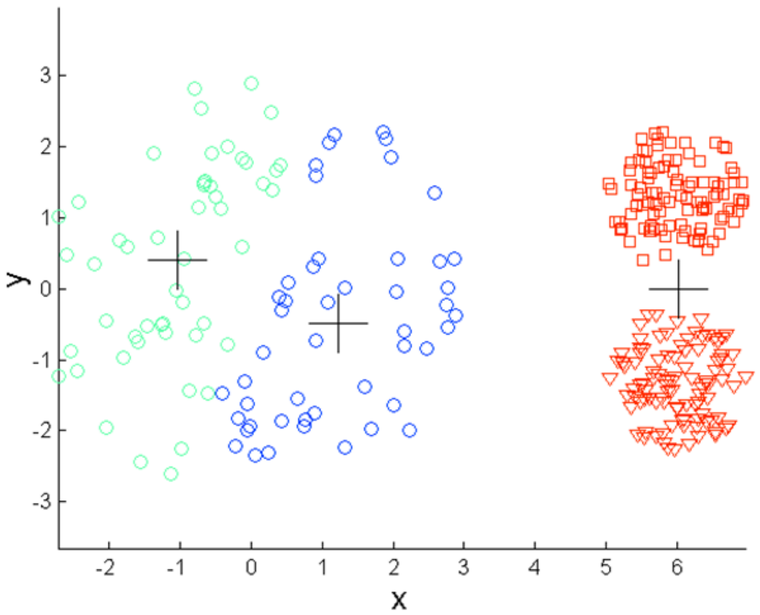


K-means (3 Clusters)

PROBLEM WITH K-MEANS: DIFFERING DENSITY

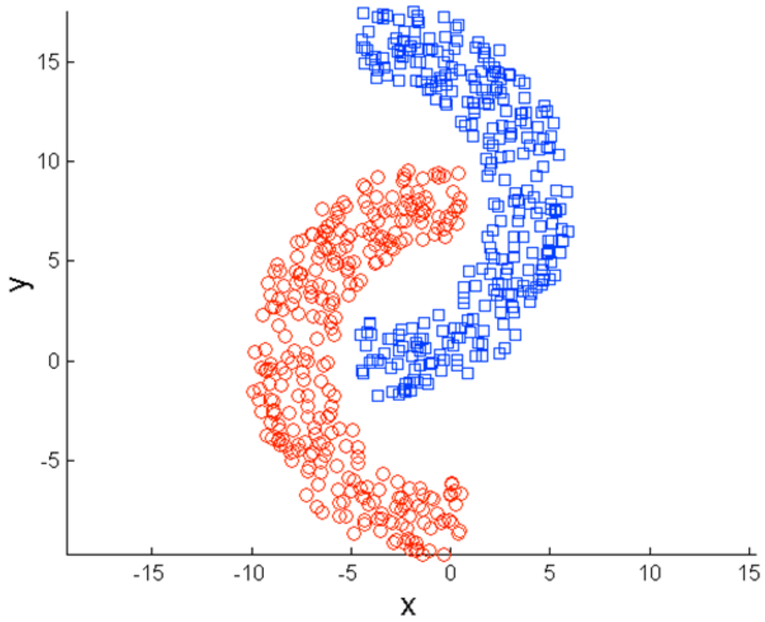


Original Points

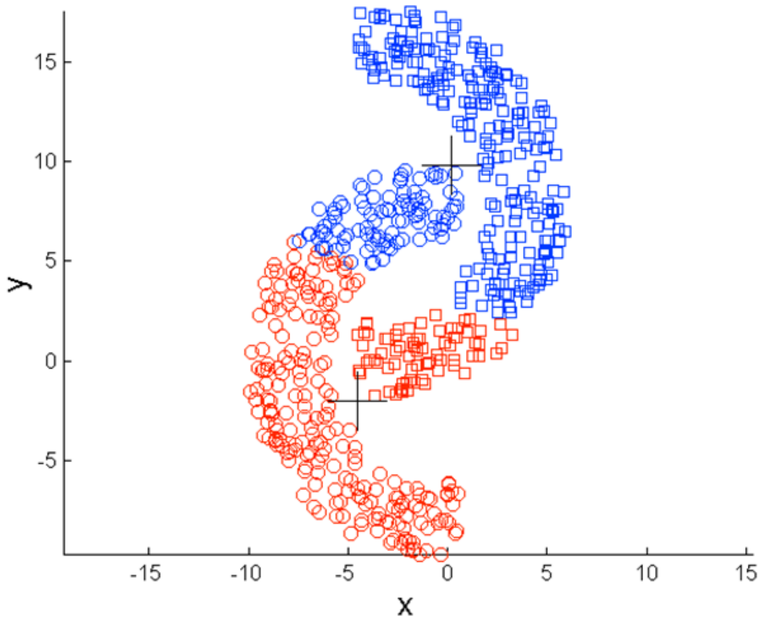


K-means (3 Clusters)

PROBLEM WITH K-MEANS: NON-GLOBULAR SHAPES



Original Points



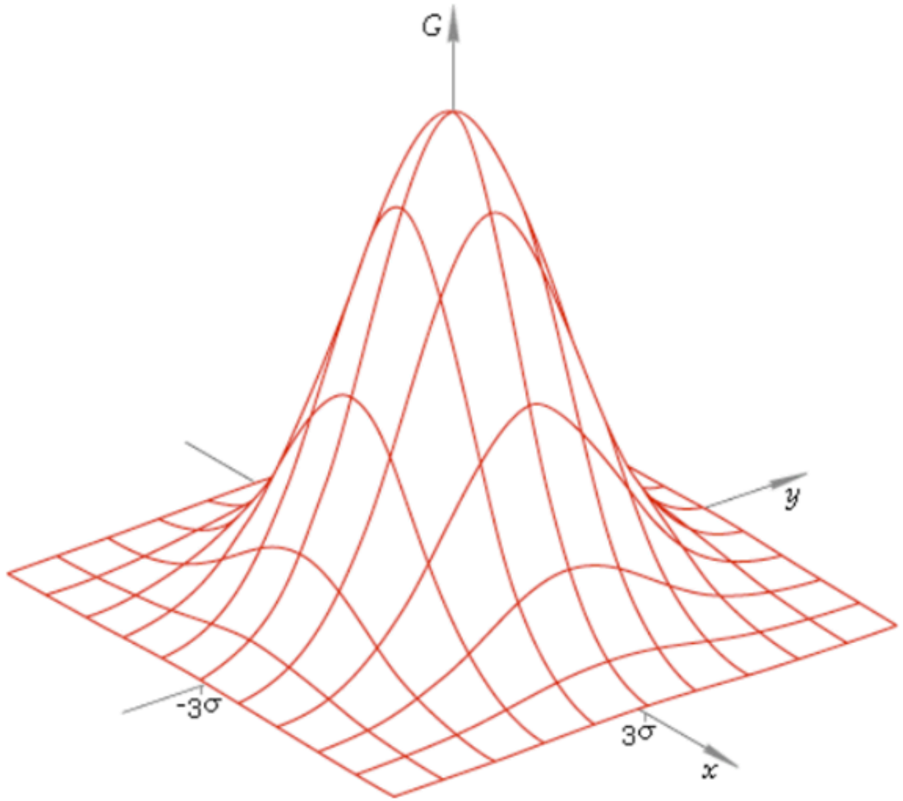
K-means (2 Clusters)

K-MEANS SUMMARY

1. 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.
2. 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.
3. **Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!**

GAUSSIAN MIXTURE MODEL

$$p(X) = \mathcal{N}(X|\mu, \Sigma)$$

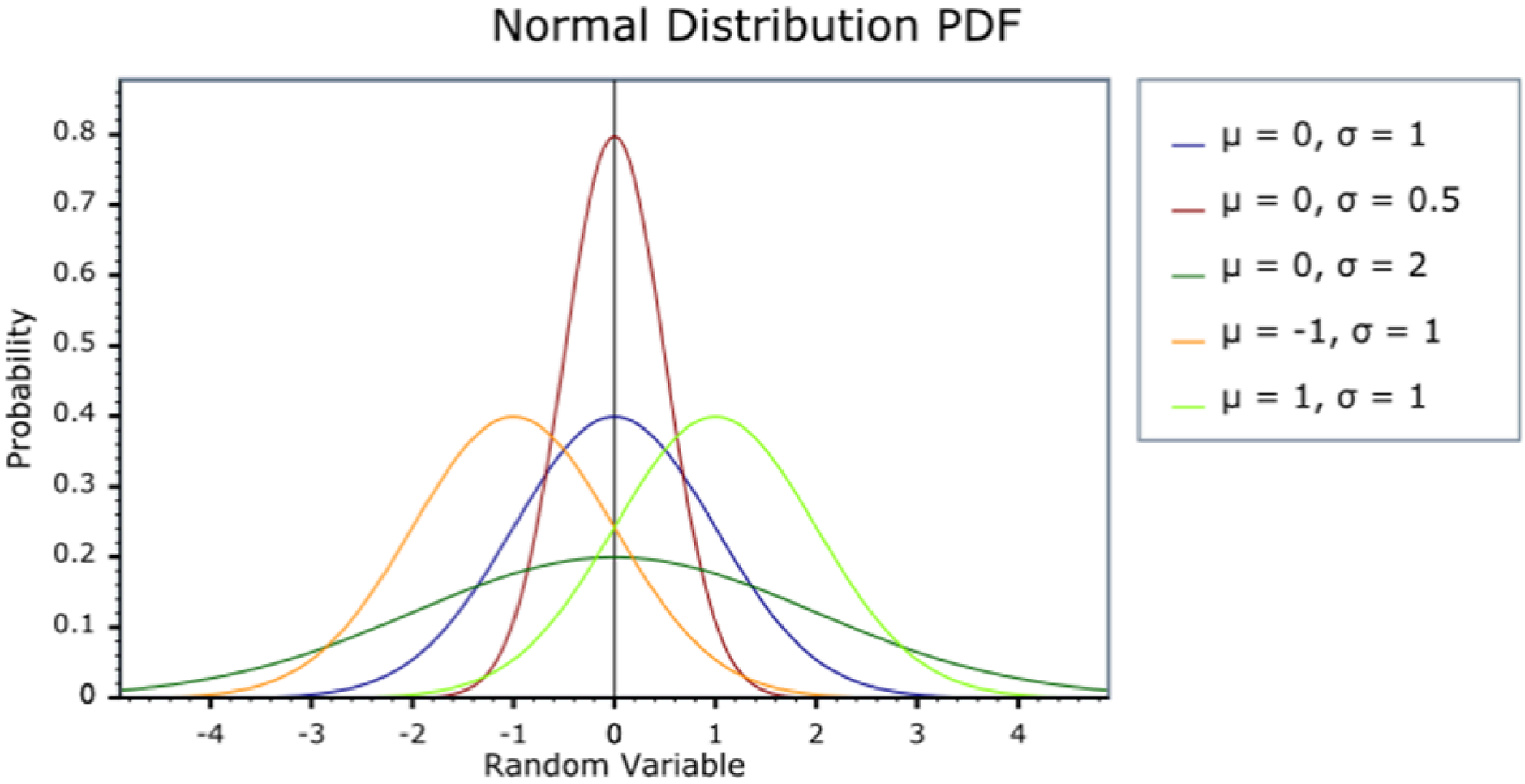


$\mathcal{N}$ = Gaussian == Normal distribution

μ = Mean

Σ = Variance

WHAT IS GAUSSIAN?

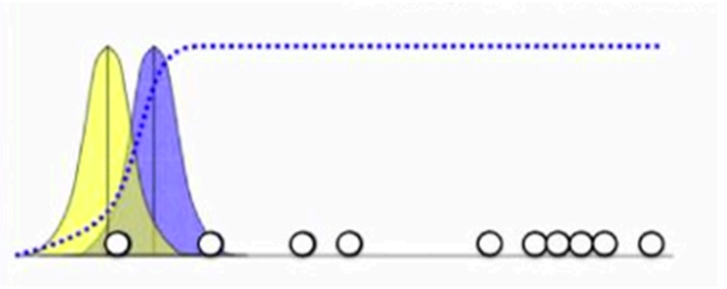


GMM = 2 (2D)

Assumptions:

1. 1D data is a collection of values: $x_1, x_2, \ldots, x_N$

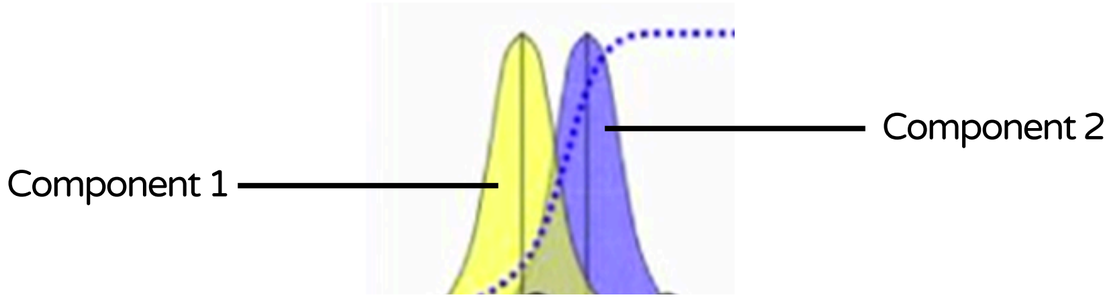
GMM = 2 (2D)



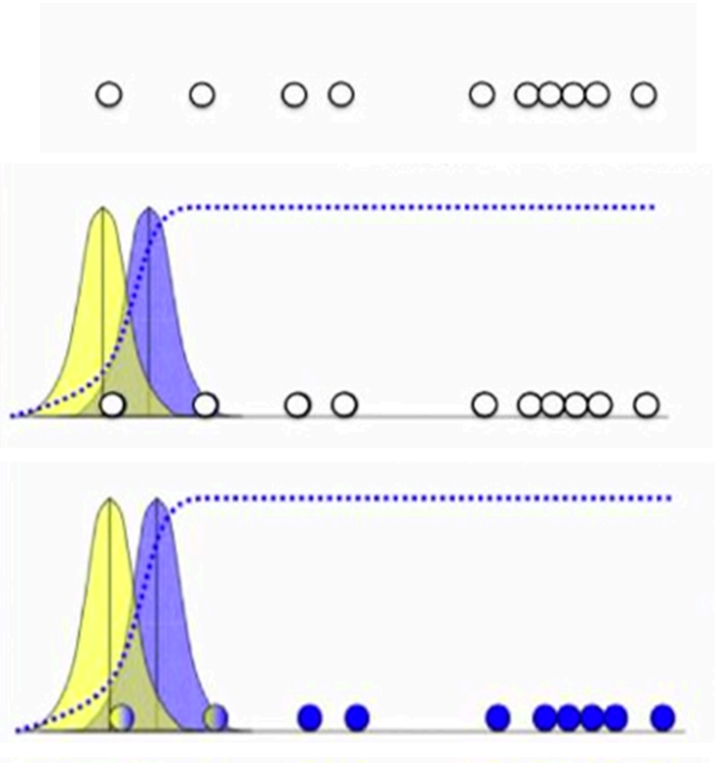
Step 1: Initialisation

The GMM consists of 2 Gaussian components.

Each component k in the mixture model has its own parameters: mean μ_k, variance σ_k^2, and mixing coefficient π_k, where π_k represents the weight of the k-th component in the mixture.



GMM = 2 (2D)



Step 2: Compute Expectation (E) Step

1. **Calculate the Probability Density** of each data point x_i for each Gaussian component k. Use the Gaussian (Normal) distribution formula:
   $P(x_i|\mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}\right)$
2. **Calculate the Weighted Probability** for each data point and each component by multiplying the probability density by the component's mixing coefficient (π_k):
   $W_{ik} = \pi_k \cdot P(x_i|\mu_k, \sigma_k^2)$
3. **Compute Responsibilities** (r_{ik}), which indicate the probability of each data point x_i belonging to each Gaussian component k. This is done by normalizing the weighted probabilities calculated in the previous step across all components for each data point:
   $r_{ik} = \frac{W_{ik}}{\sum_{j=1}^{K} W_{ij}}$
   where the denominator is the sum of weighted probabilities of data point x_i across all K components, ensuring that the sum of responsibilities for each data point across all components equals 1.

GMM = 2 (2D)

Steps 3: Compute Maximization (M) Step

1. Update the Means (μ_k)

The new mean μ_k for each Gaussian component is computed as a weighted average of all data points, where the weights are the responsibilities r_{ik}:

$$\mu_k = \frac{\sum_{i=1}^{N} r_{ik} x_i}{\sum_{i=1}^{N} r_{ik}}$$

2. Update the Variances (σ_k^2)

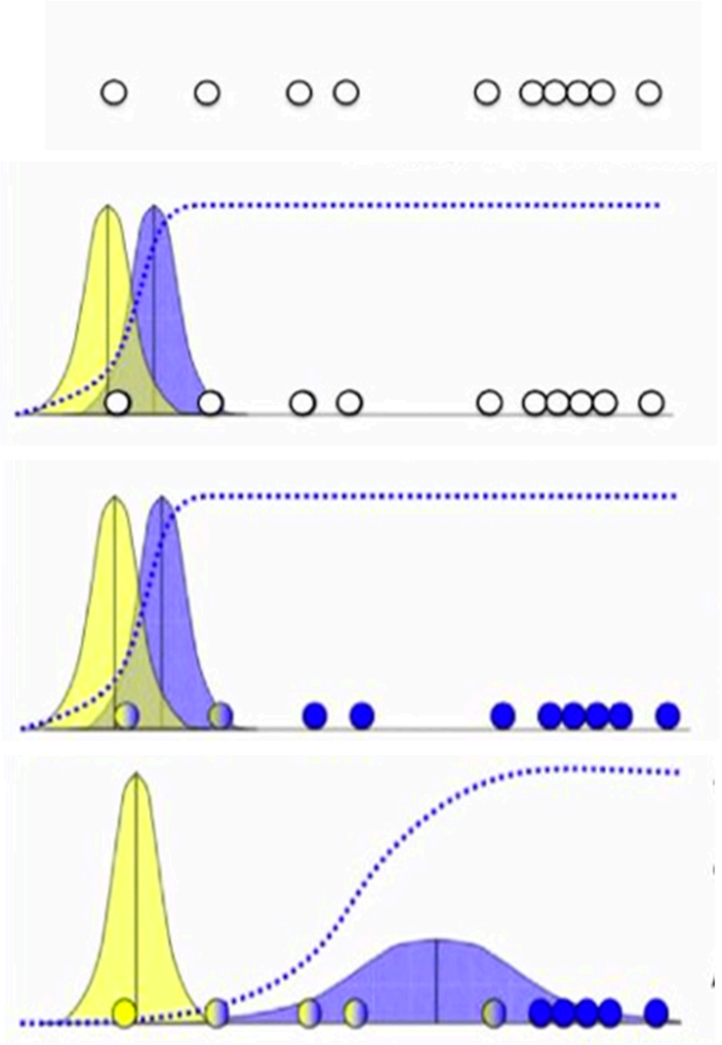
The new variance σ_k^2 for each component is also a weighted average of the squared differences between the data points and the new mean, again using r_{ik} as weights:

$$\sigma_k^2 = \frac{\sum_{i=1}^{N} r_{ik} (x_i - \mu_k)^2}{\sum_{i=1}^{N} r_{ik}}$$

3. Update the Mixing Coefficients (π_k)

The mixing coefficient π_k for each component reflects the proportion of points for which the component is responsible, relative to the whole dataset:

$$\pi_k = \frac{\sum_{i=1}^{N} r_{ik}}{N}$$

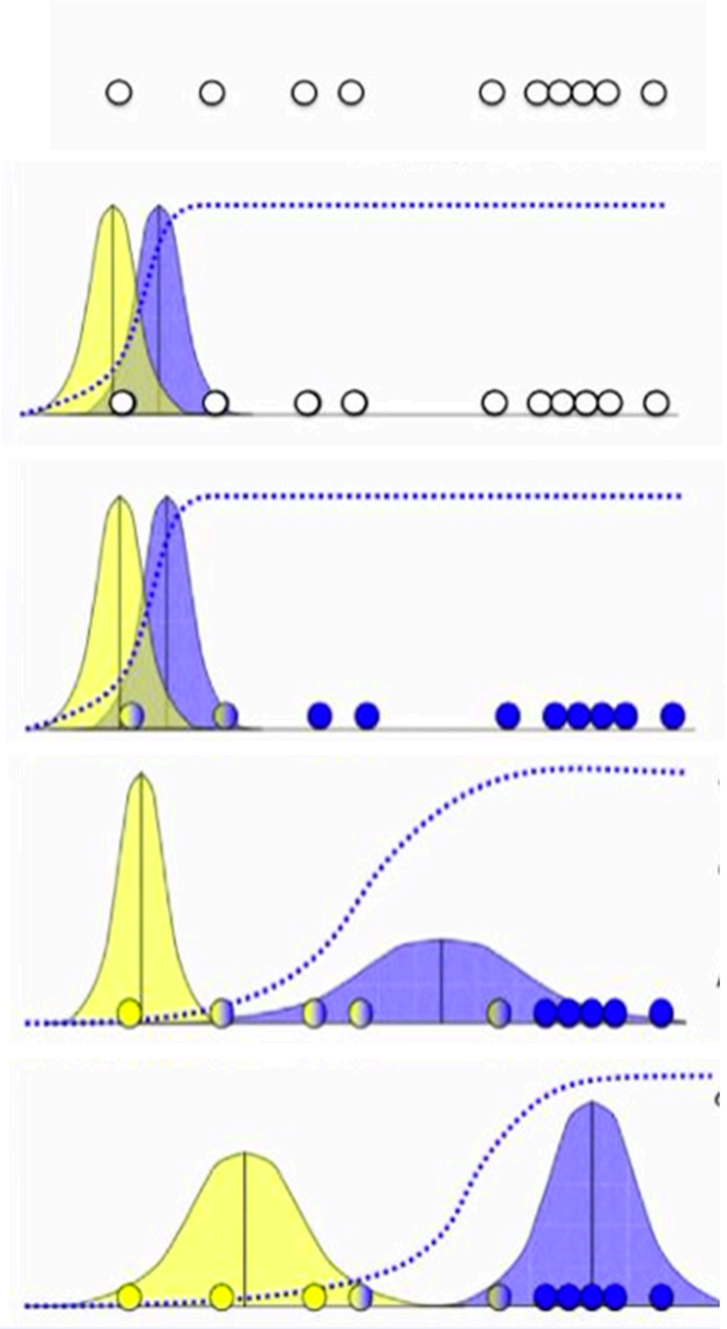


GMM = 2 (2D)

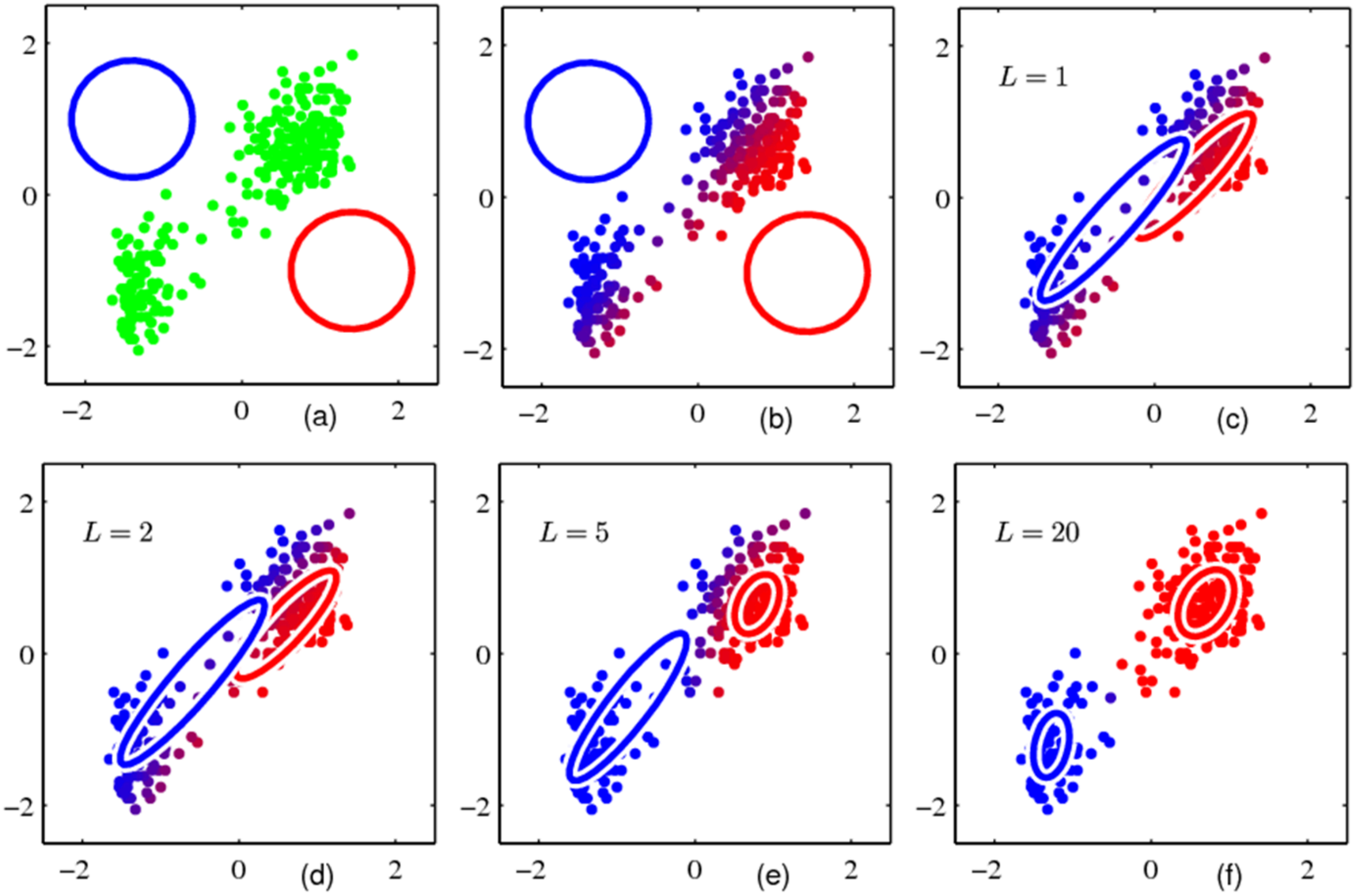
Steps 4: Iterative Refinement

Steps 2 & 3 (E-step and M-step) are repeated iteratively until the algorithm converges, i.e., until the parameters of the Gaussian distributions stop changing significantly between iterations or until a specified number of iterations is reached.

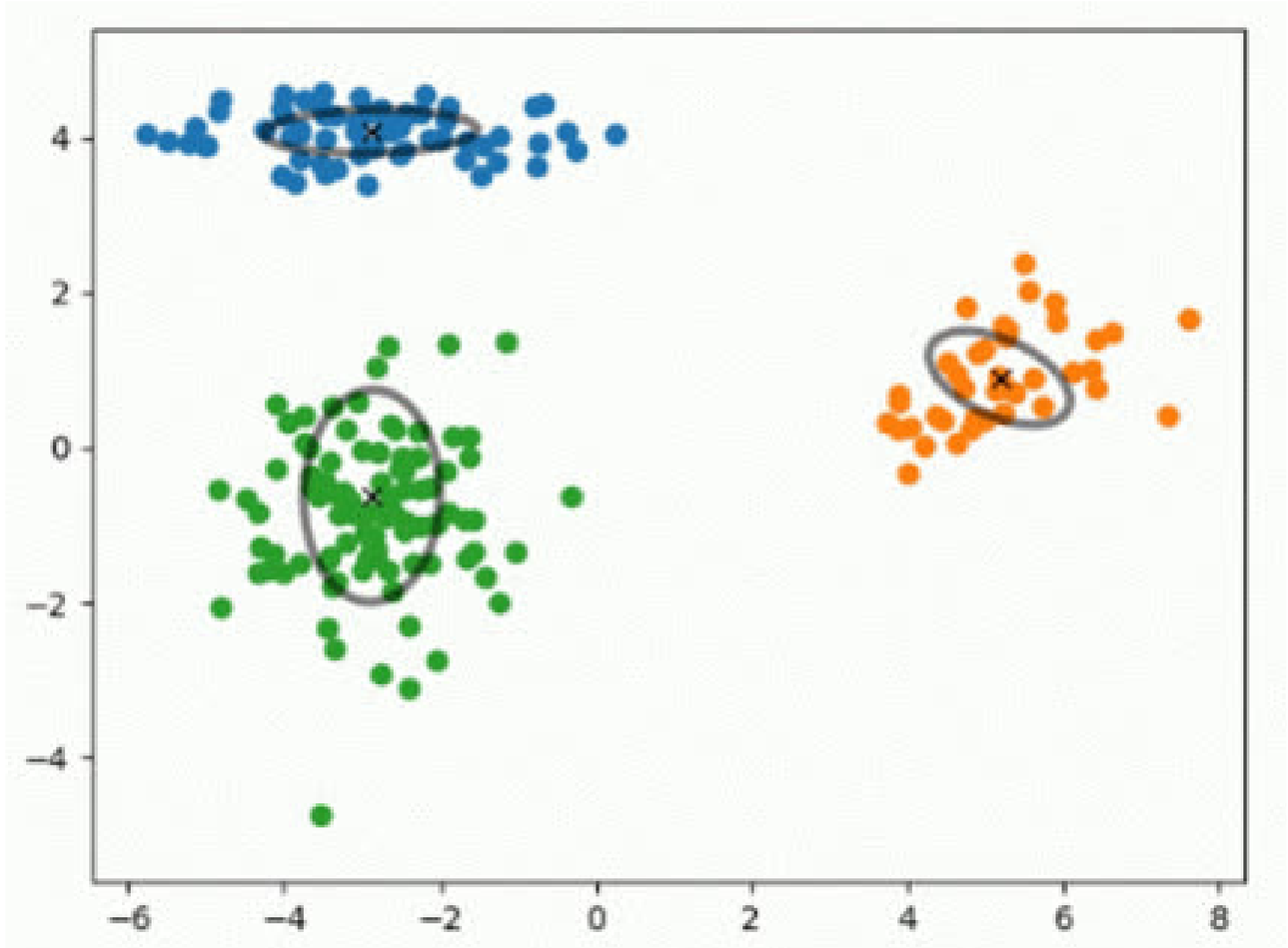
After convergence, each data point is assigned to the Gaussian distribution with the highest probability of generating it. This step essentially clusters the data points into different groups based on their probability distributions.



EXAMPLE 2



EXAMPLE 3




```
from sklearn.mixture import GaussianMixture
# Fit a GMM with 3 components (clusters)
gmm = GaussianMixture(n_components=3, random_state=0)
gmm.fit(data)
```

MEAN SHIFT

1. The mean shift algorithm seeks a ***mode* or local maximum of density** of a given distribution.

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

$K(x)$ is a kernel function that determines the weight of nearby points for re-estimation of the mean. The popular one:
1) Flat Kernel
2) Gaussian Kernel

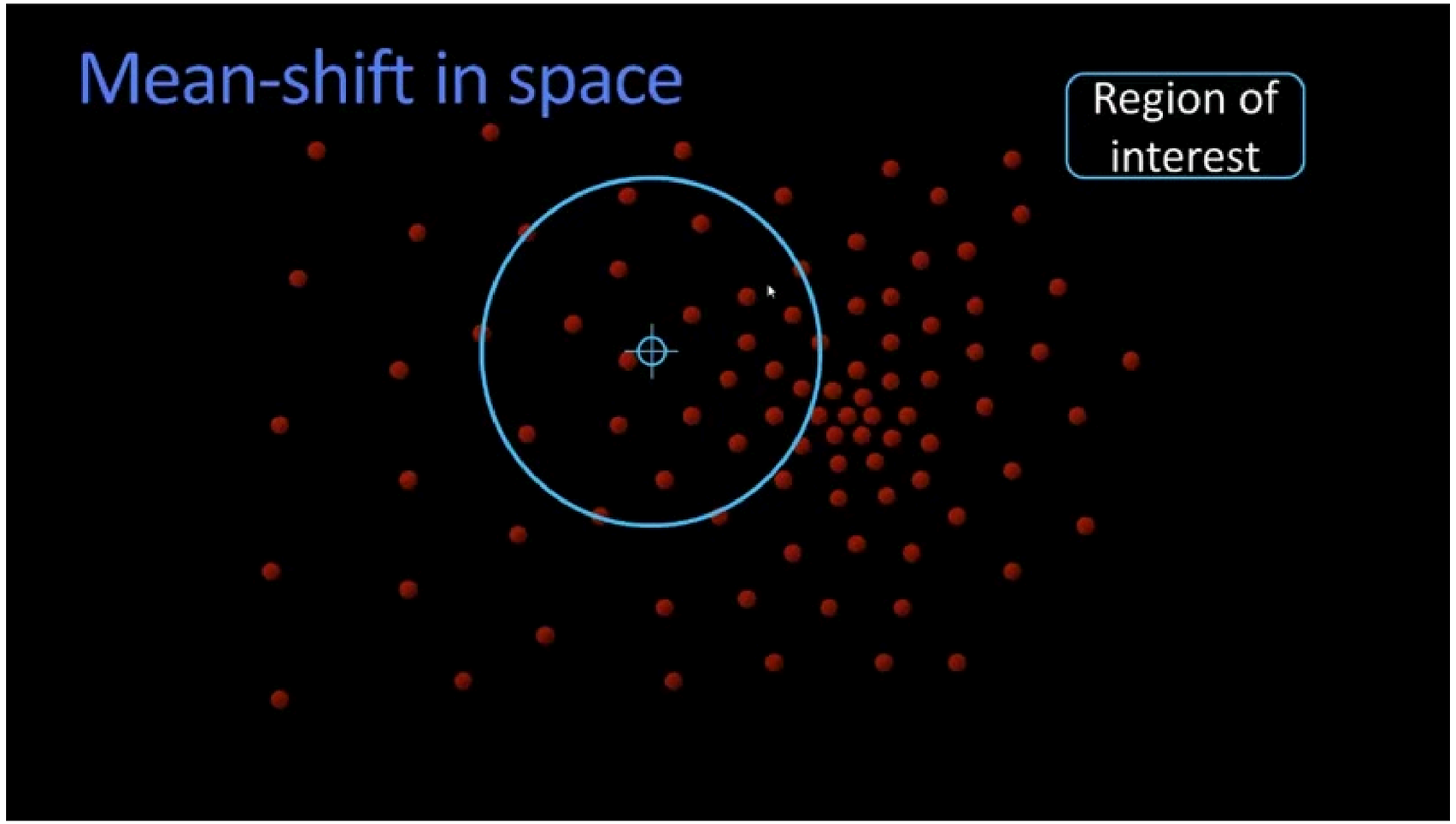
2. where $N(x)$ is the neighborhood of x, a set of points for which $K(x) \neq 0$.
3. The **difference $m(x) - x$** is called mean shift.

MEAN SHIFT

Algorithm:

1. **Choose a search window** (width and location)
2. **Compute the mean** of the data **in the search window**
3. **Center** the search window at the **new mean location**
4. **Repeat until convergence**

MEAN SHIFT

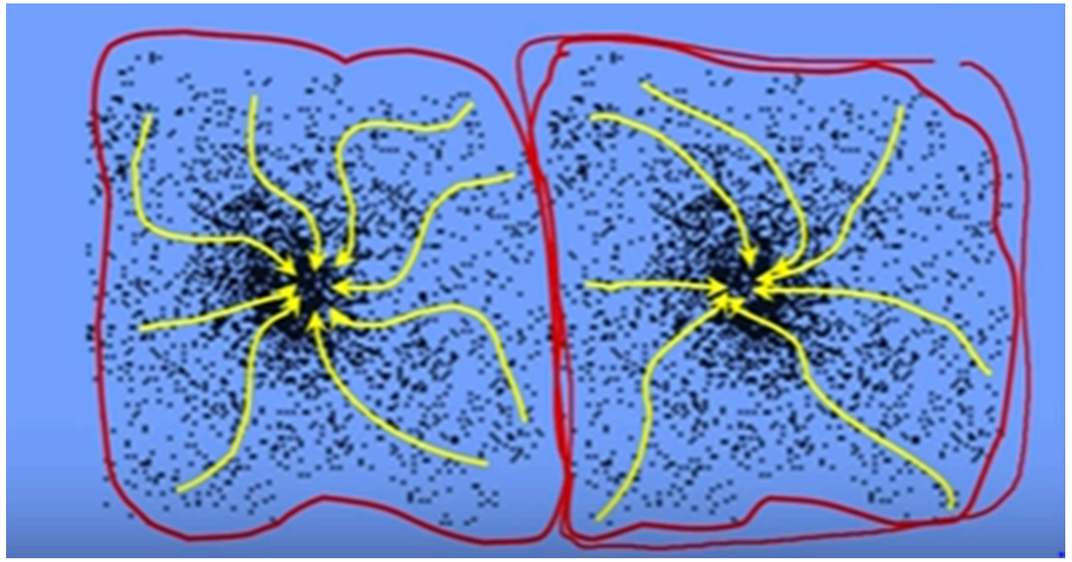


Algorithm:

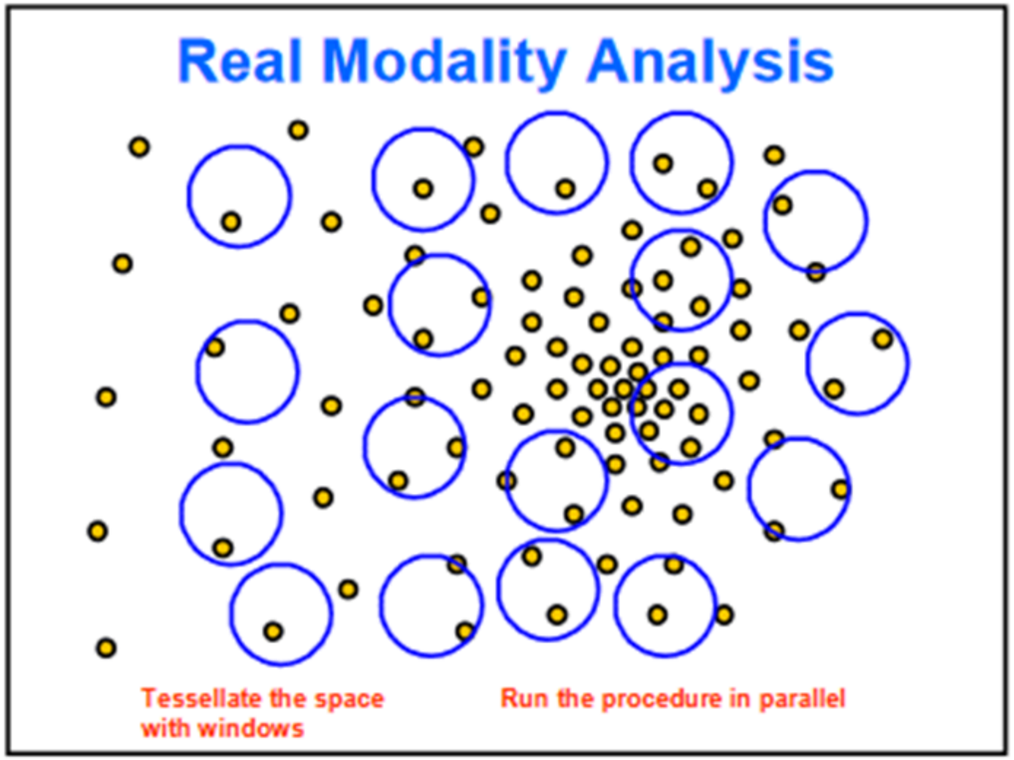
1. **Choose a search window** (width and location)
2. **Compute the mean** of the data **in the search window**
3. **Center** the search window at the **new mean location**
4. **Repeat until convergence**

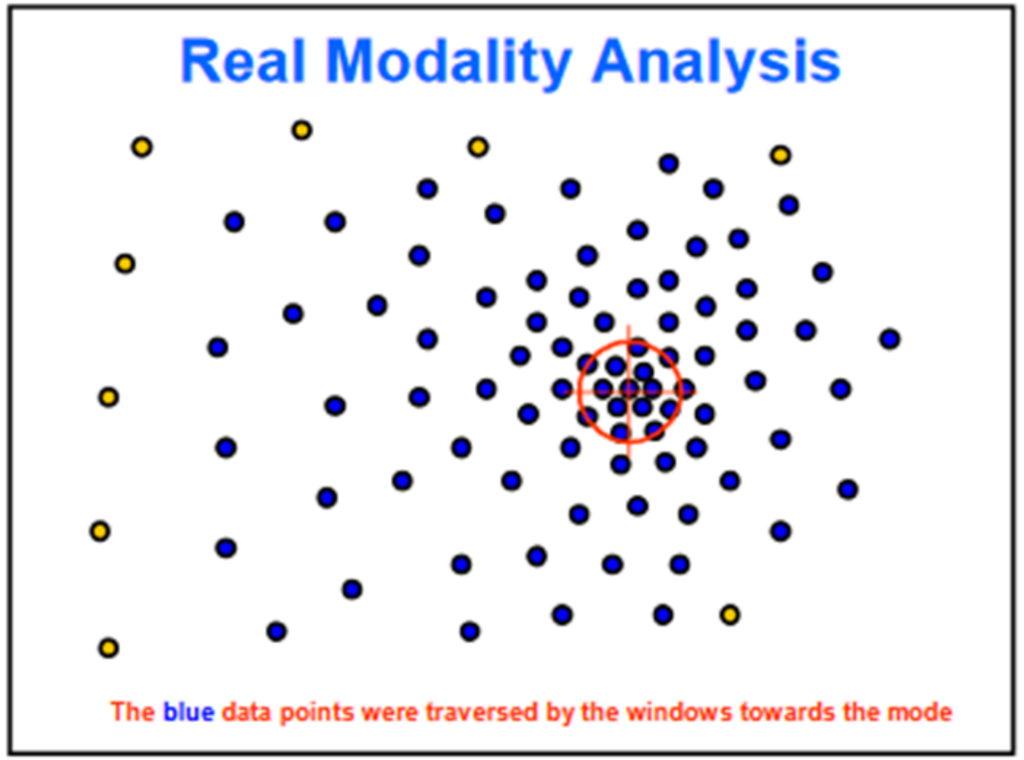
MEAN SHIFT

1. Cluster: all data points in the attraction basin of a **mode**
2. Attraction basin: the region for which all trajectories lead to the same mode



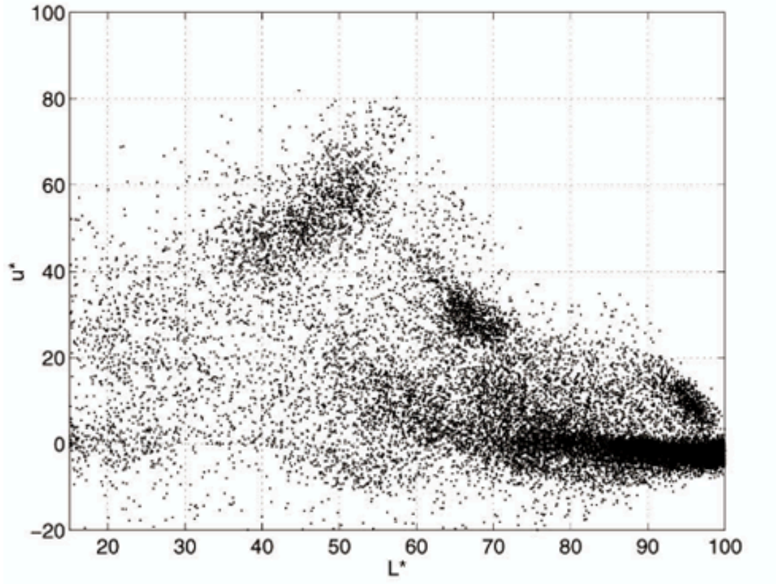
MEAN SHIFT

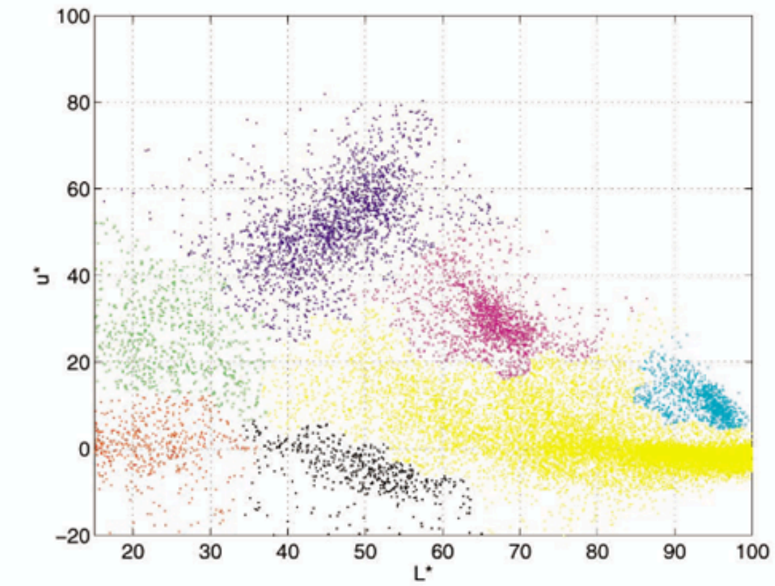


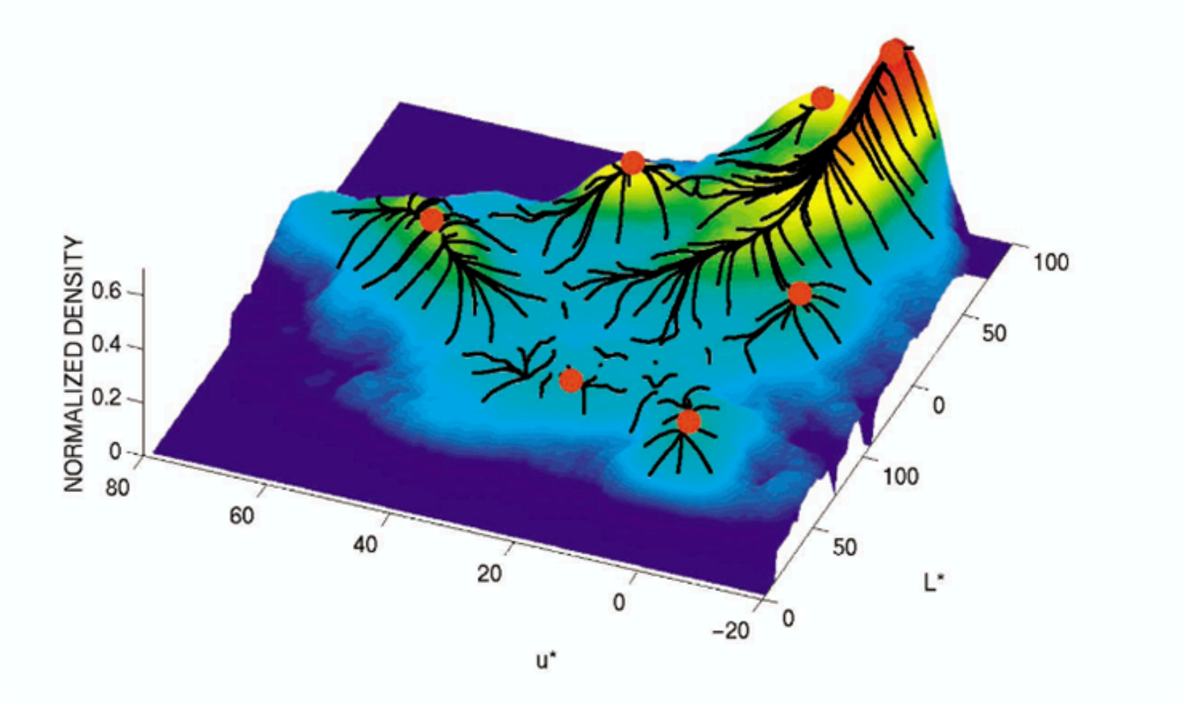



```
from sklearn.cluster import MeanShift, estimate_bandwidth

# Estimate bandwidth for Mean Shift
bandwidth = estimate_bandwidth(X, quantile=0.3)

# Apply Mean Shift clustering
meanshift = MeanShift(bandwidth=bandwidth)
meanshift.fit(X)
```

IMAGE SEGMENTATION





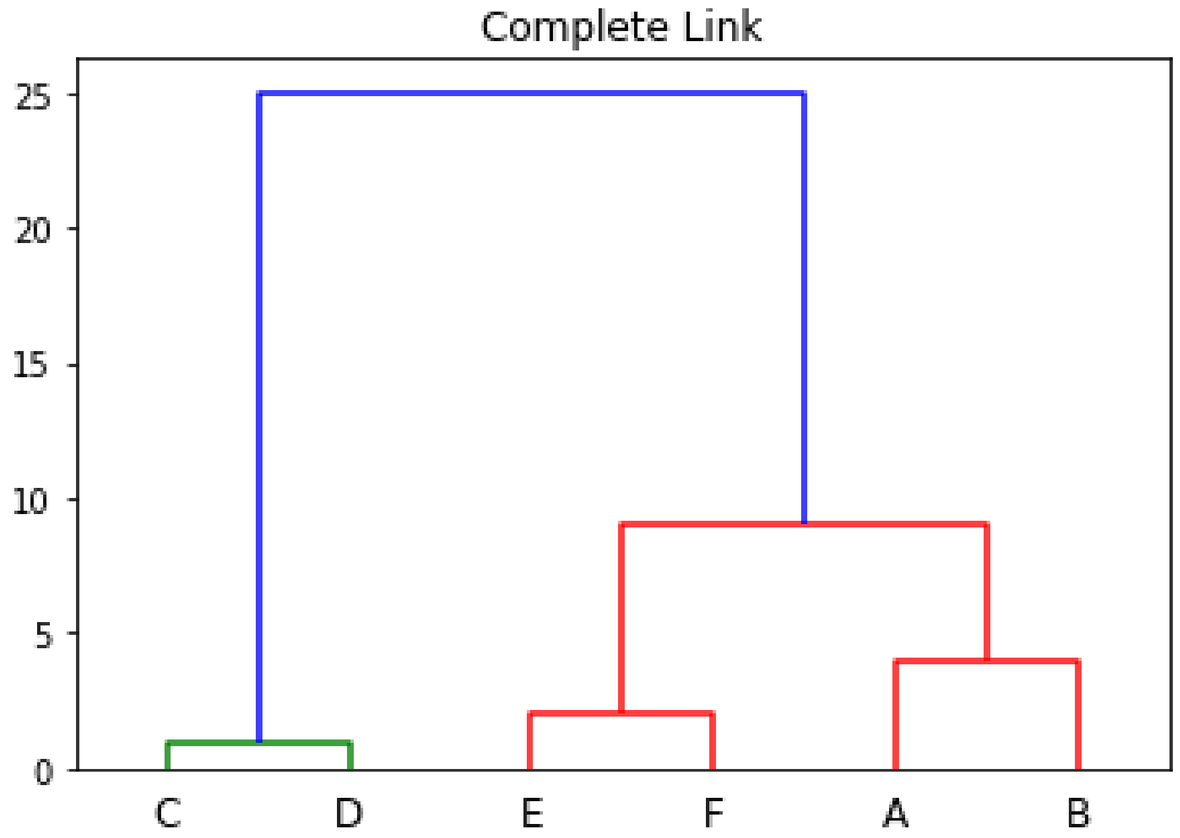


MEAN SHIFT

1. Advantages
   - **Does not assume number of clusters**
   - Just a single parameter **(window size)**
   - Finds variable number of modes
   - **Robust to outliers**
2. Disadvantages
   - Output **depends on window size**
   - **Computationally expensive**

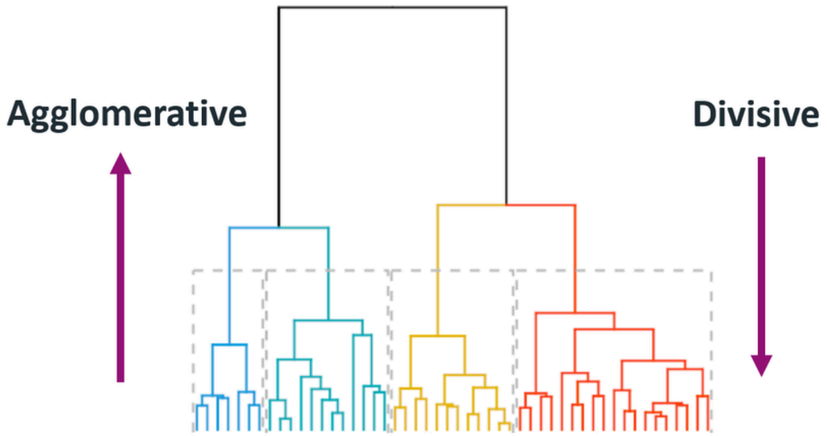
HIERARCHICAL CLUSTERING

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



HIERARCHICAL CLUSTERING

1. **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).
2. **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

Agglomerative clustering (more popular than divisive methods).

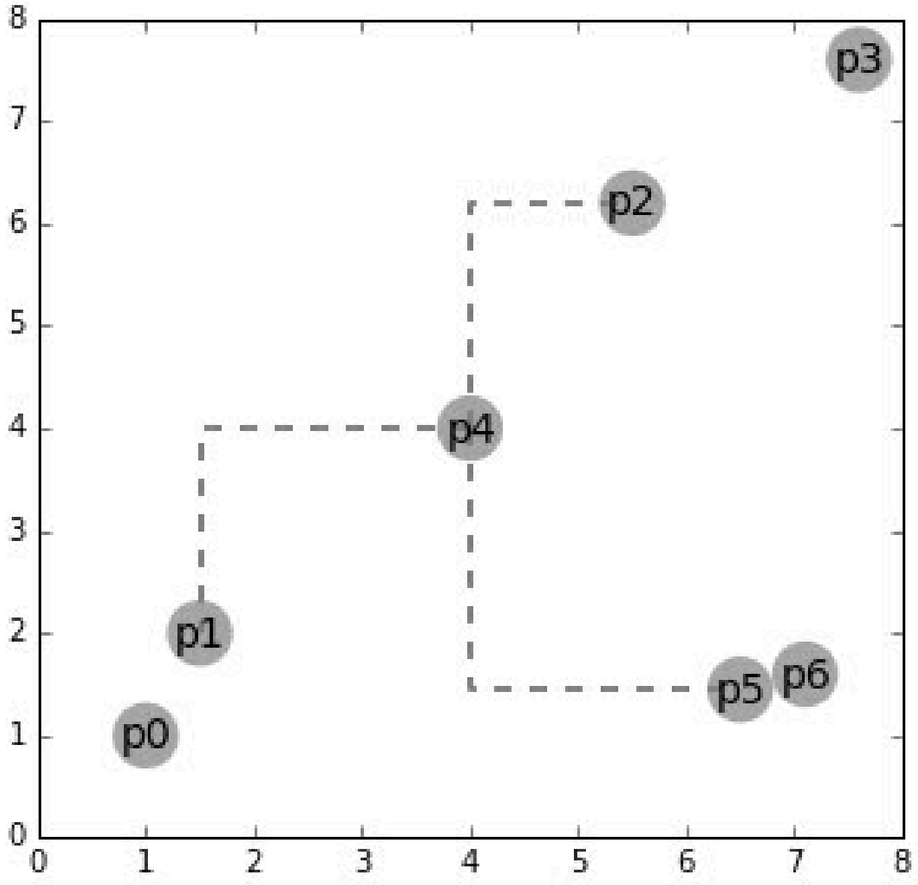
Algorithm:

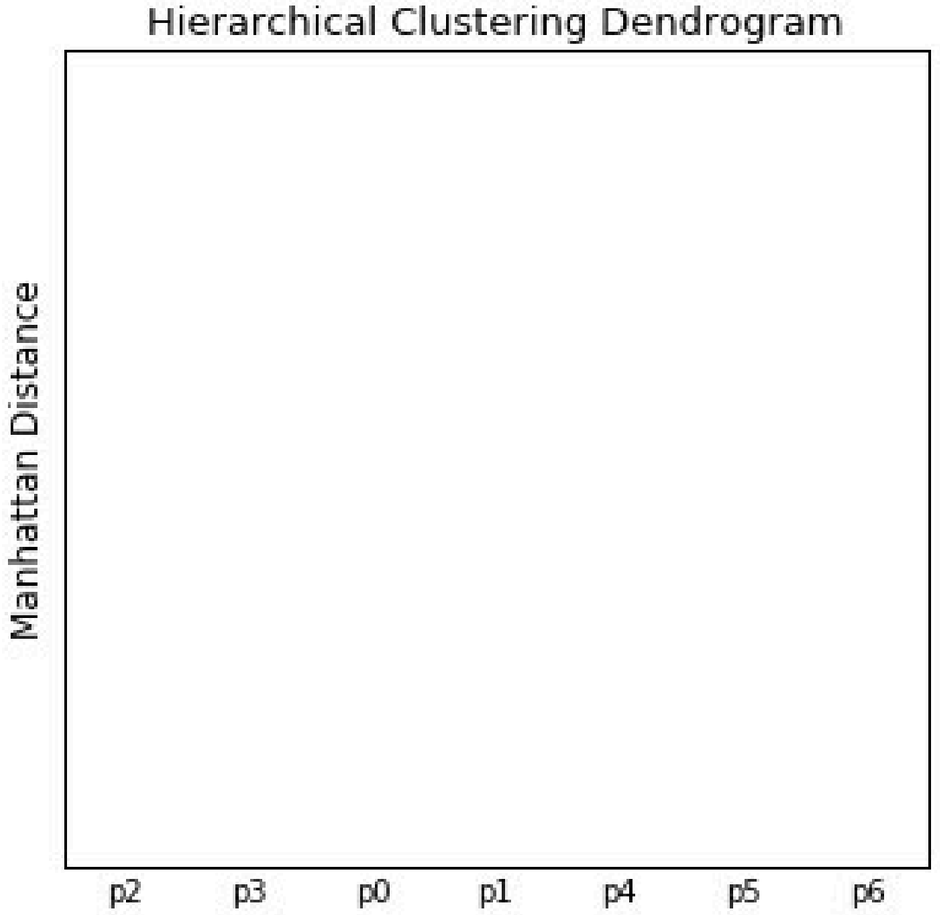
1. At the beginning, **each data point forms a cluster** (also called a node).
2. **Merge** nodes/clusters that have the **least distance**.
3. Go on merging until there is **only one cluster left.**

AGGLOMERATIVE CLUSTERING

Algorithm:

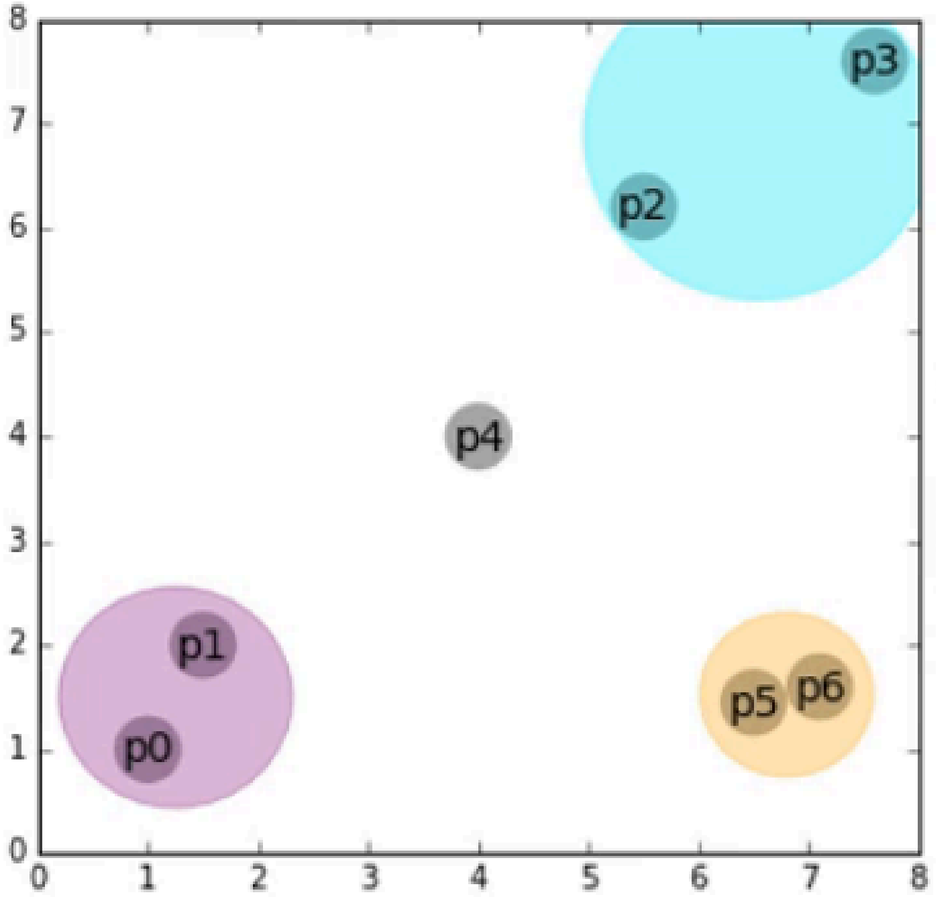
1. At the beginning, **each data point forms a cluster** (also called a node).
2. **Merge** nodes/clusters that have the **least distance**.
3. Go on merging until there is **only one cluster left.**

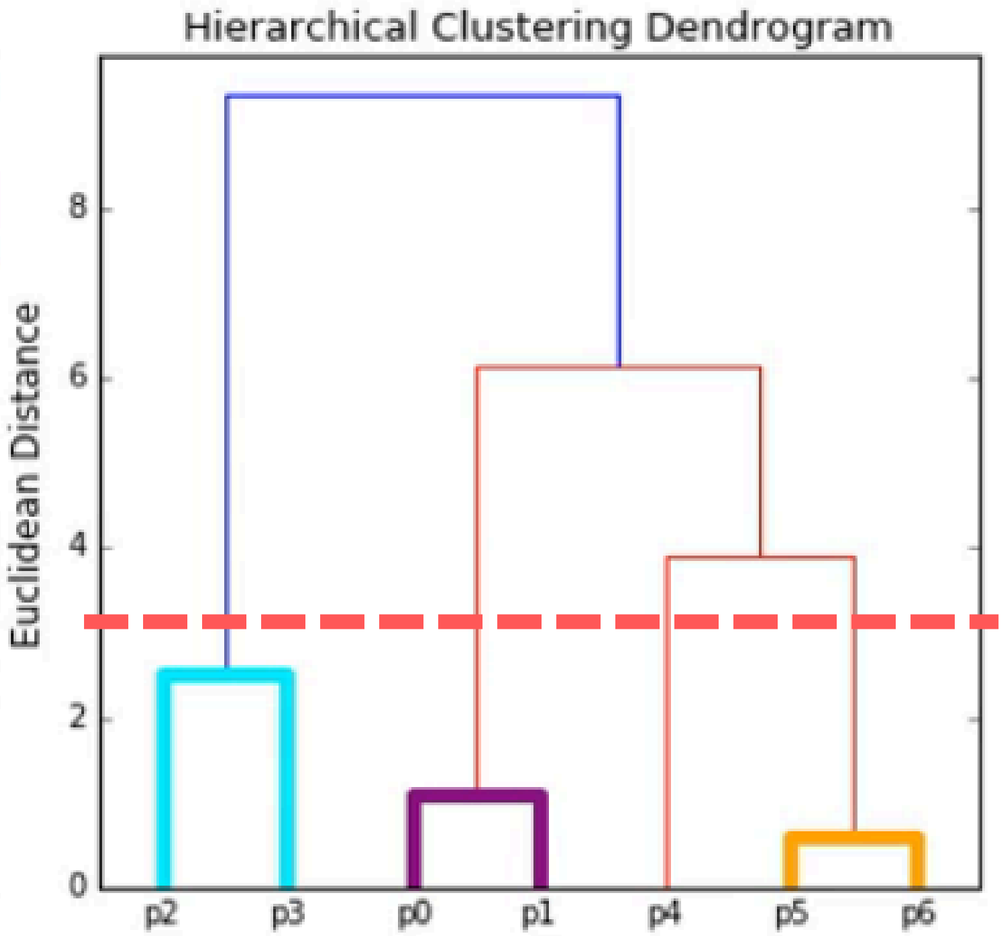




Manhattan (Taxicab or Cityblock) Distance

AGGLOMERATIVE CLUSTERING

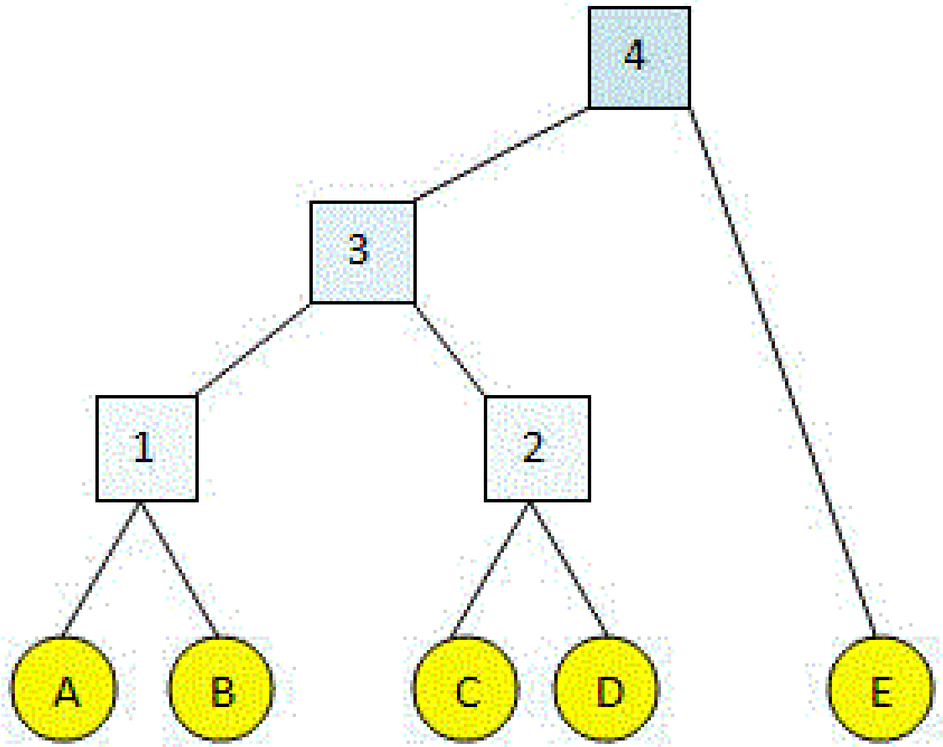




4 CLUSTERS

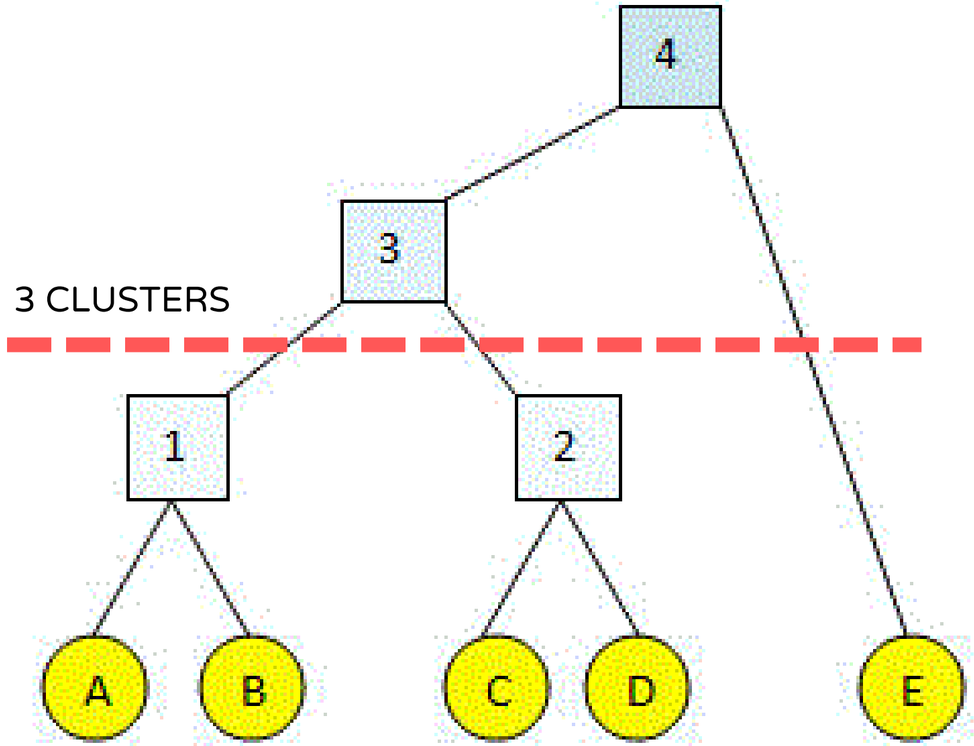
TRY THIS (3 CLUSTERS)

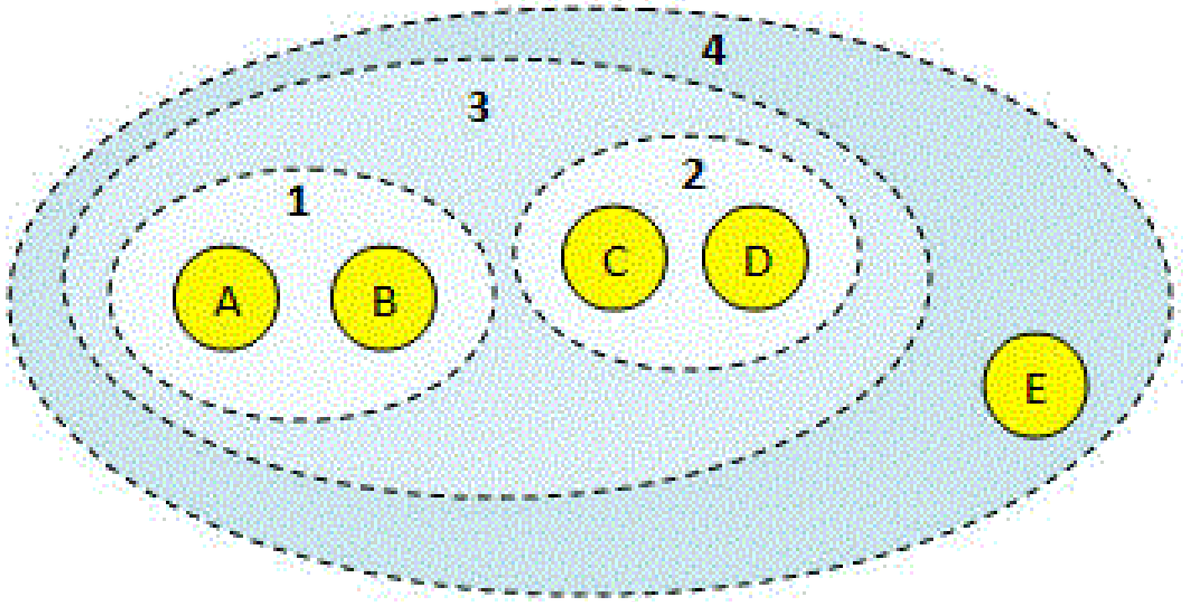
```
from sklearn.cluster import AgglomerativeClustering

# Fit Agglomerative Clustering
n_clusters = 3 # Number of clusters
agg_clustering = AgglomerativeClustering(n_clusters=n_clusters)
labels = agg_clustering.fit_predict(X)
```



TRY THIS (3 CLUSTERS)

```
from sklearn.cluster import AgglomerativeClustering

# Fit Agglomerative Clustering
n_clusters = 3 # Number of clusters
agg_clustering = AgglomerativeClustering(n_clusters=n_clusters)
labels = agg_clustering.fit_predict(X)
```





ASSESSING CLUSTERING

Cluster Evaluation: hard problem

The quality of a clustering is very hard to evaluate because

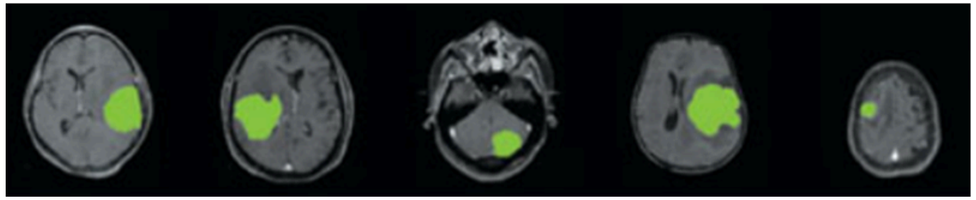
WE DO NOT KNOW THE CORRECT CLUSTERS

GROUND TRUTH

1. Use of ground truth
2. After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements such as precision, recall and F-score.

Input images (original images)

Clustered tumor region



Ground truth for the tumor

EVALUATION BASED ON INTERNAL INFORMATION

1. **Intra-cluster cohesion** (compactness):
   - Cohesion measures how near the data points in a cluster are to the cluster centroid.
   - Sum of squared error (SSE) is a commonly used measure.
2. **Inter-cluster separation** (isolation):
   - Separation means that different cluster centroids should be far away from one another.

In most applications, **expert judgments** are still the key.

THE END

NEXT LECTURE

Artificial Neural Network