Unsupervised Learning

(Example: K-means clustering)

Unsupervised Learning

- In unsupervised learning, data does not have any labels. Unsupervised learning algorithms try to find some structure in the data.

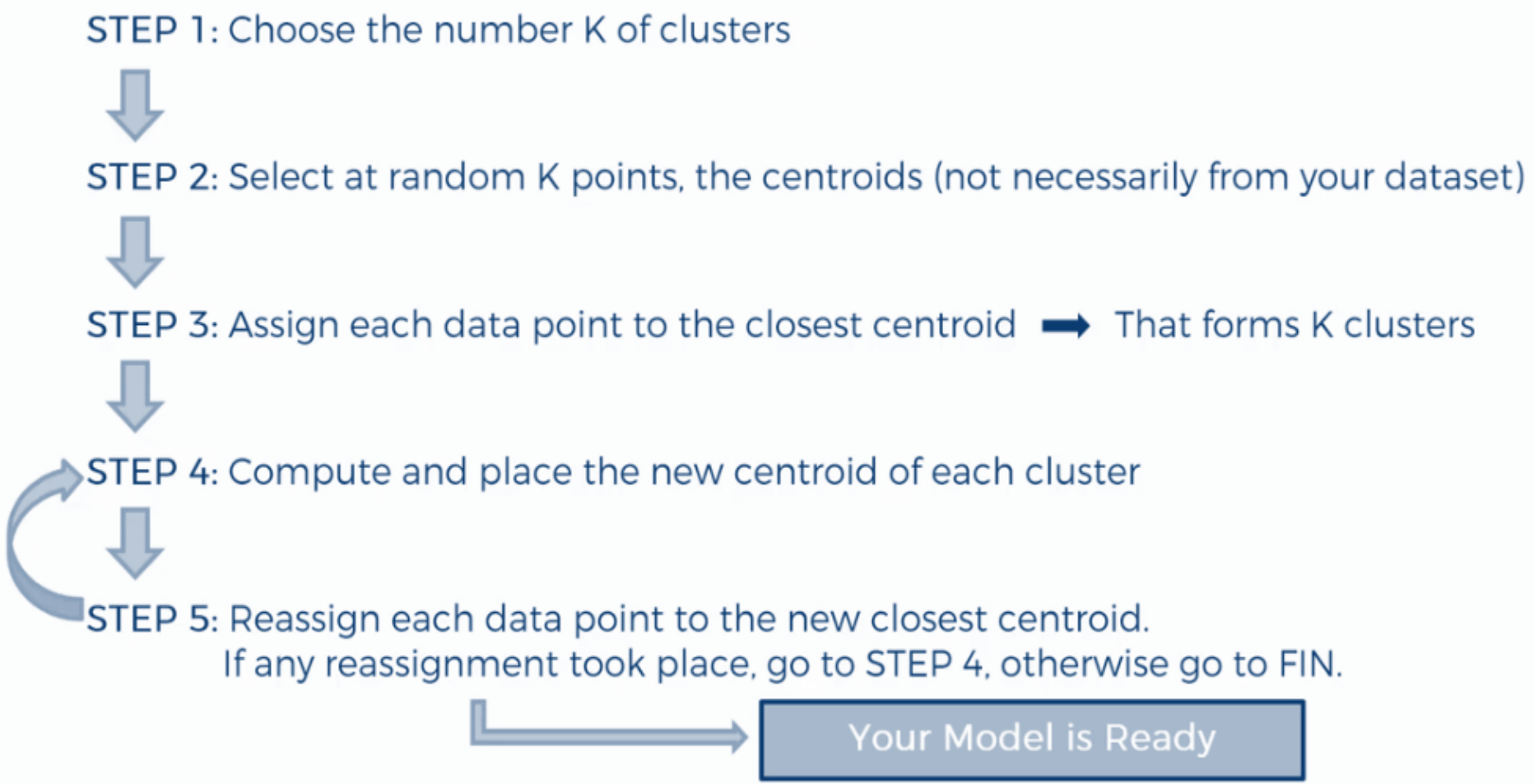
- An example is a clustering algorithm. We don't tell the algorithm in advance anything about the structure of the data; it discovers it on its own by figuring how to group them.

- Some other examples are dimensionality reduction, in which you try to reduce the dimensionality of the data representation, density estimation, in which you estimate the probability distribution of the data, p(x), and feature extraction, in which you try to learn meaningful features automatically.

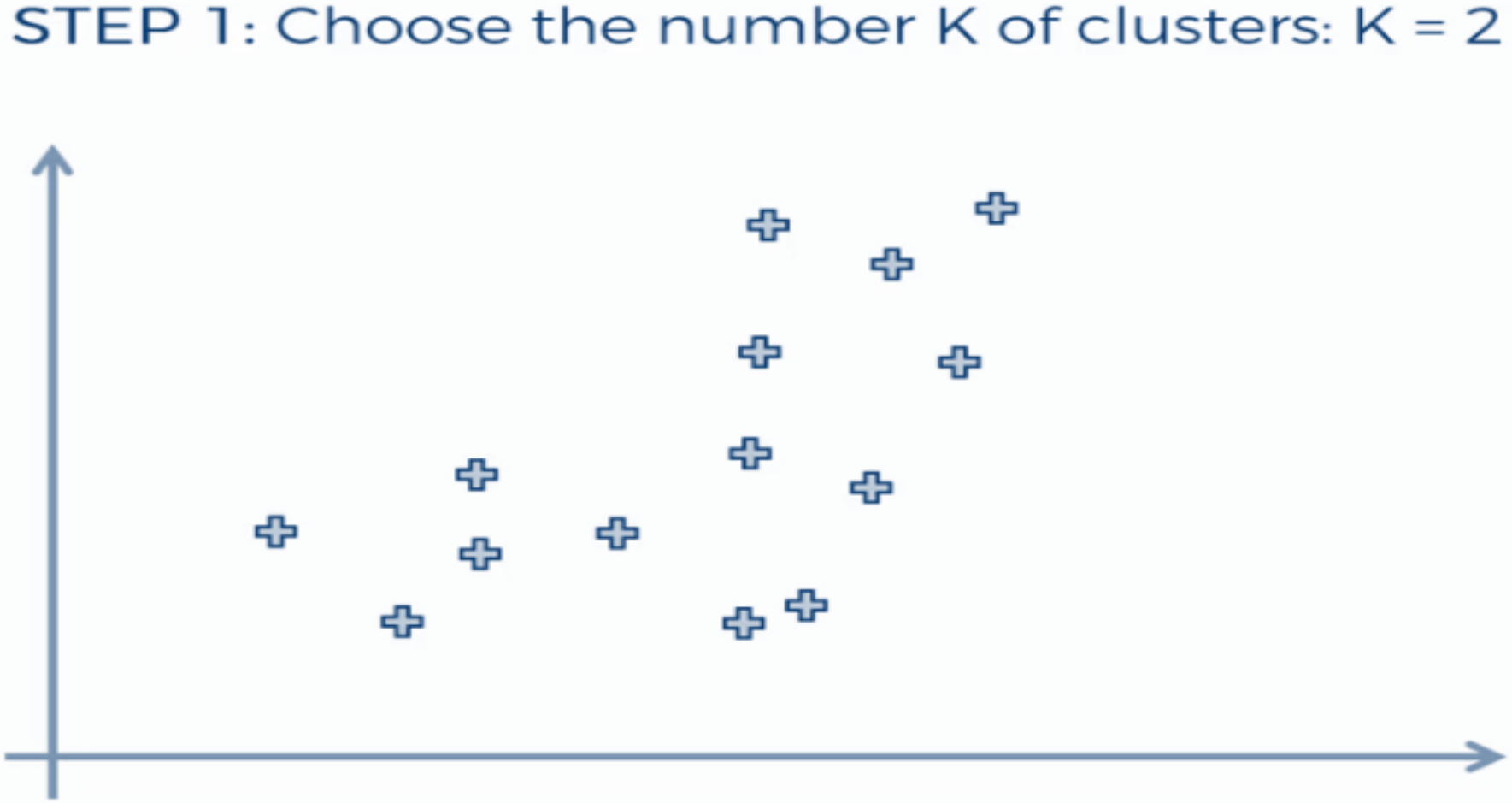
K-means

- Partitional clustering approach
- The algorithm can group your data into k number of categories.
- Number of clusters, K, must be specified
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- The basic algorithm is very simple

K-Means Algorithm

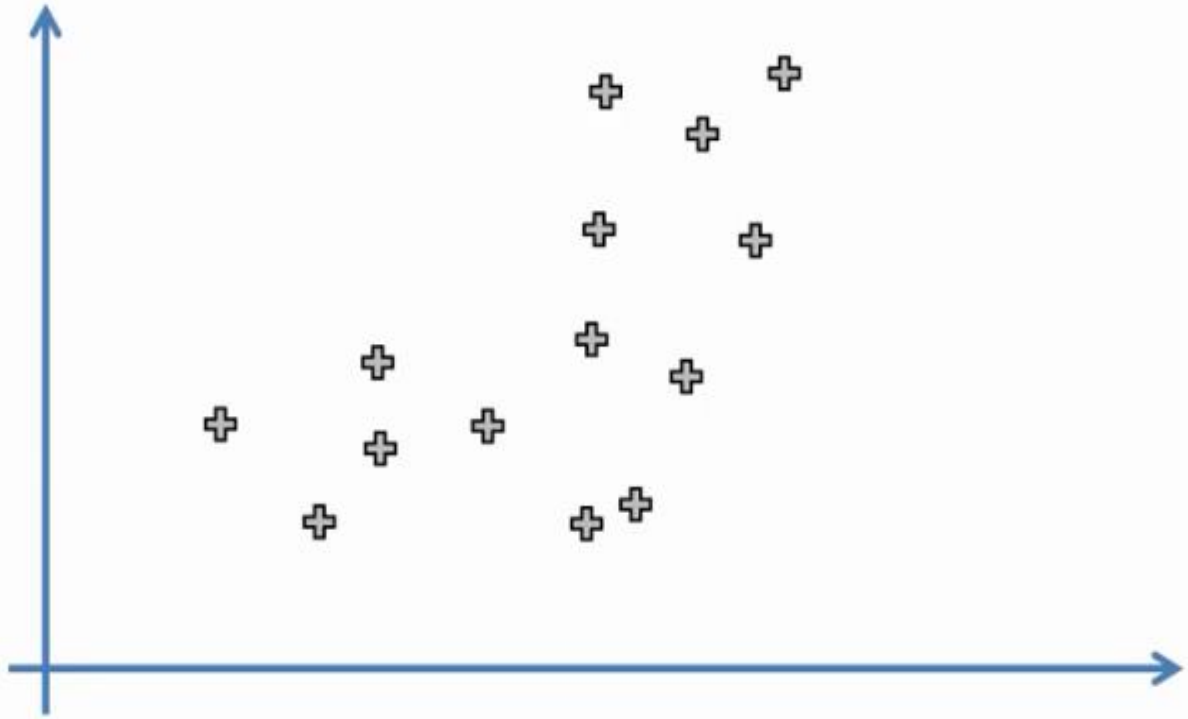


K-Means Algorithm



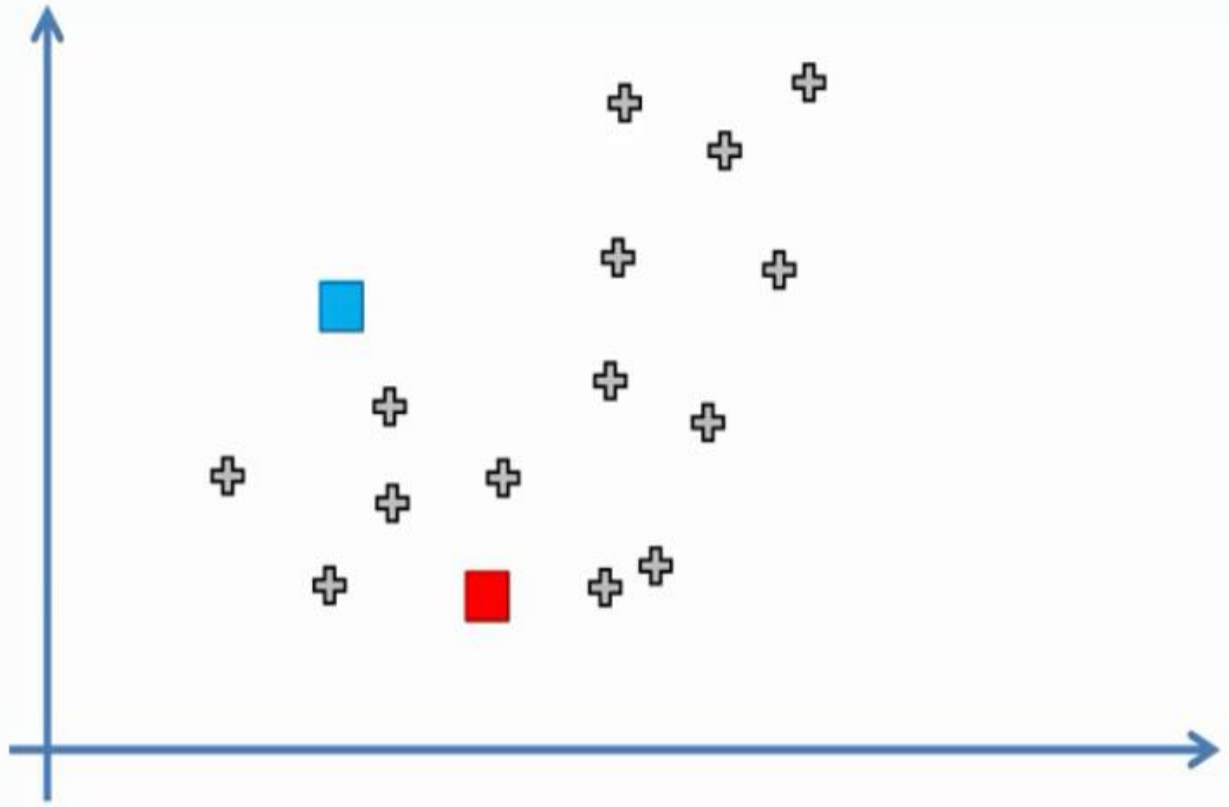
K-Means Algorithm

STEP 2: Select at random K points, the centroids (not necessarily from your dataset)



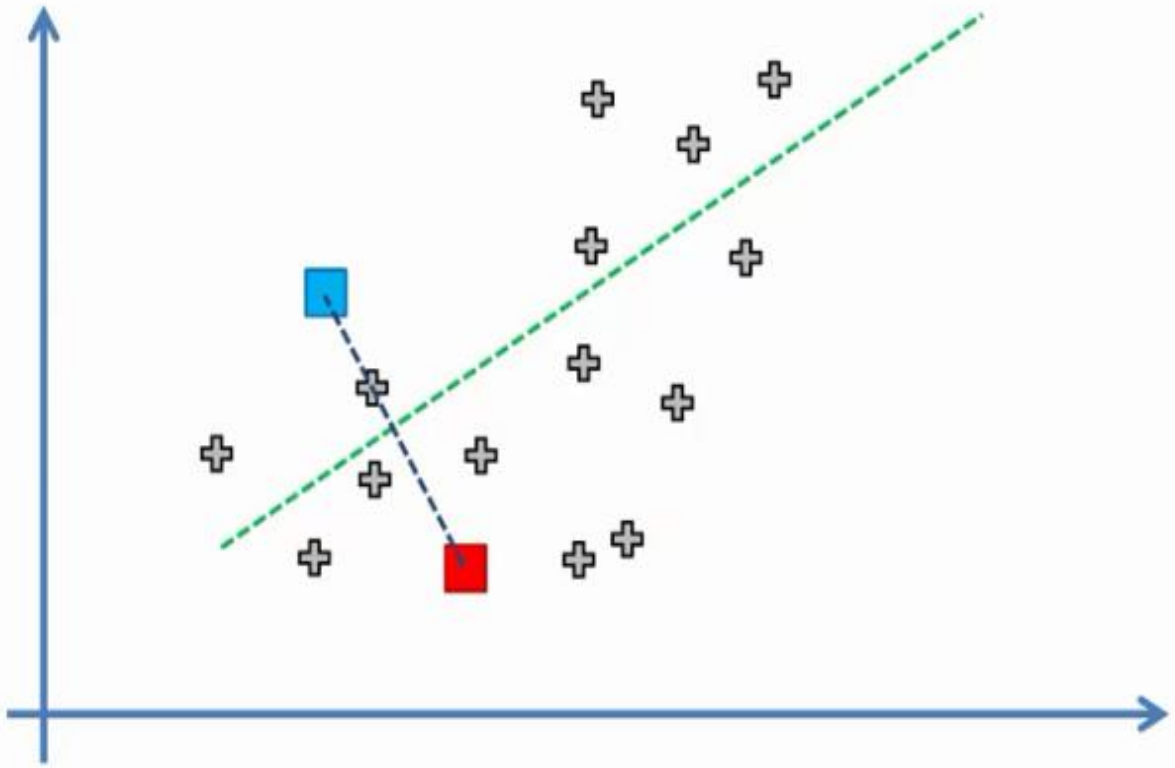
K-Means Algorithm

STEP 2: Select at random K points, the centroids (not necessarily from your dataset)

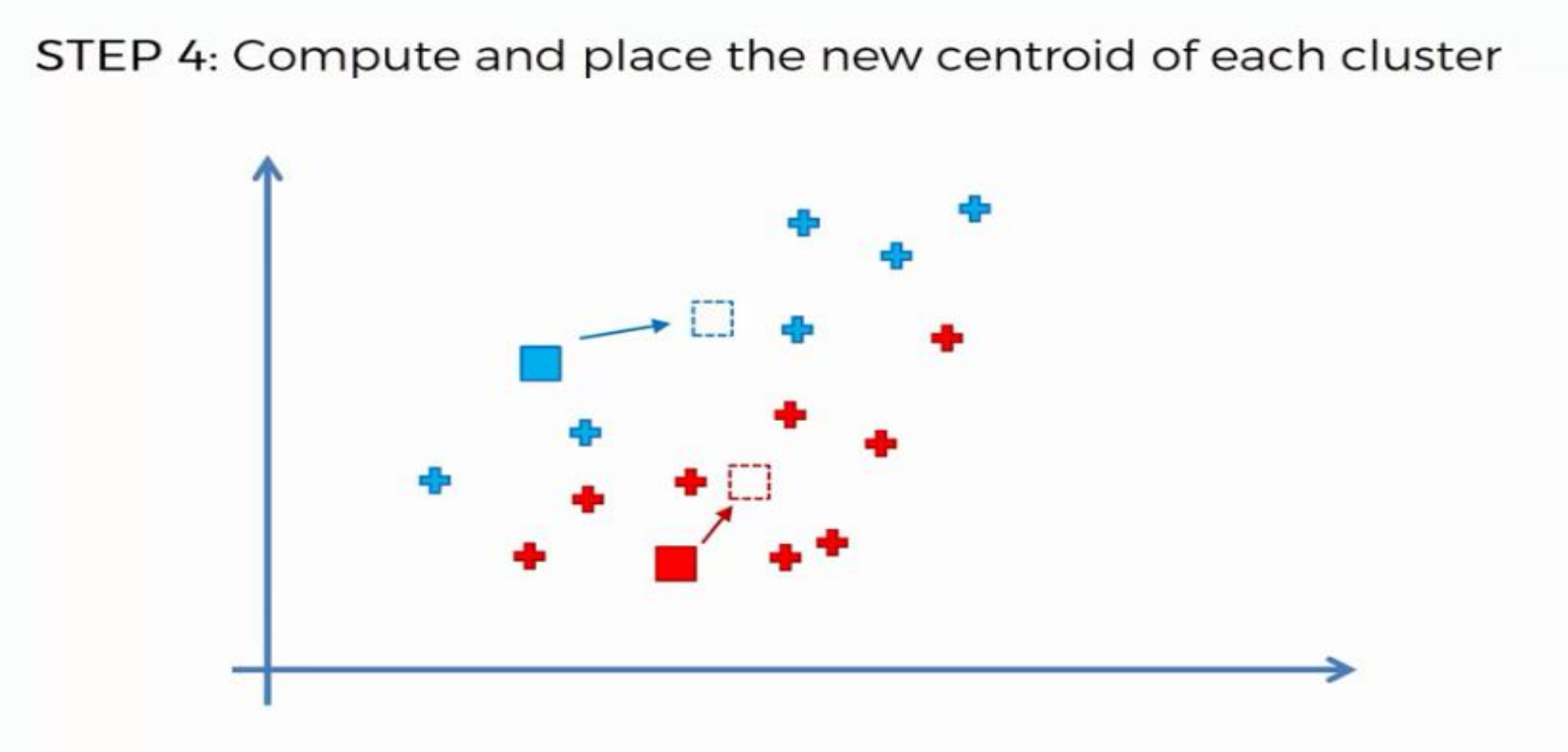


K-Means Algorithm

STEP 3: Assign each data point to the closest centroid ➡ That forms K clusters

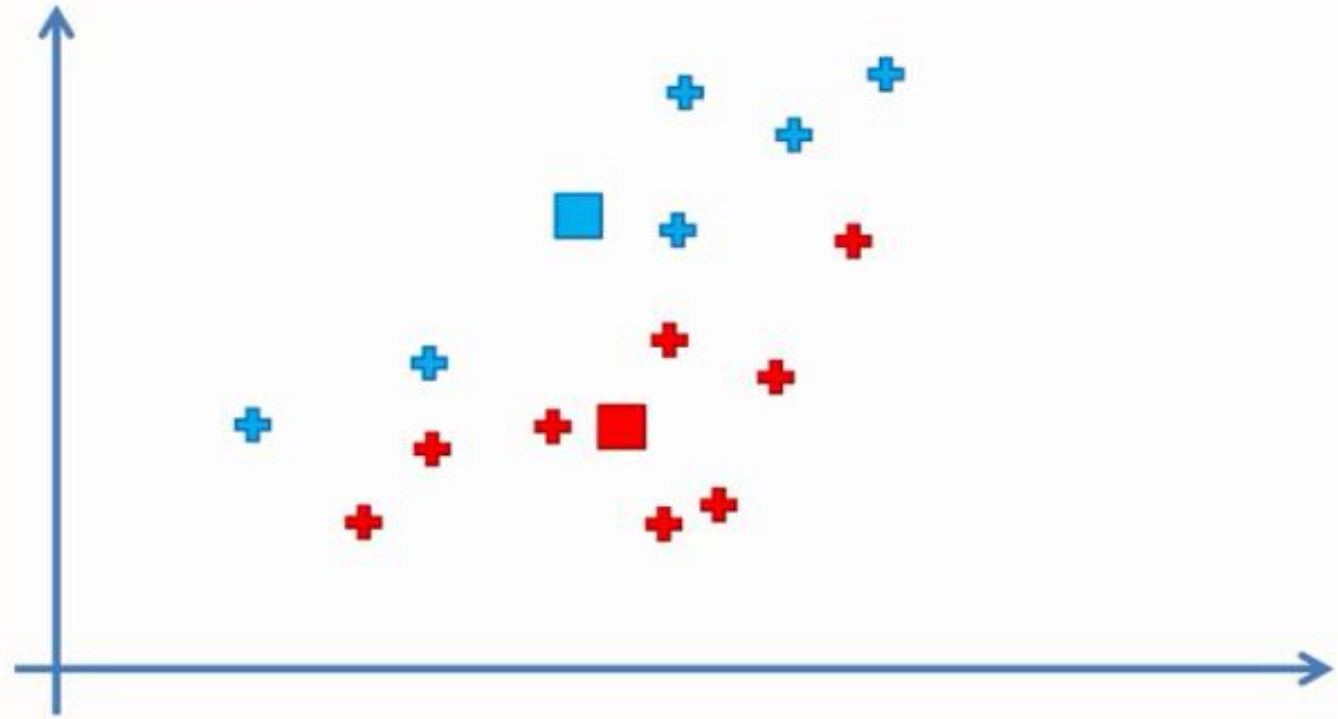


K-Means Algorithm



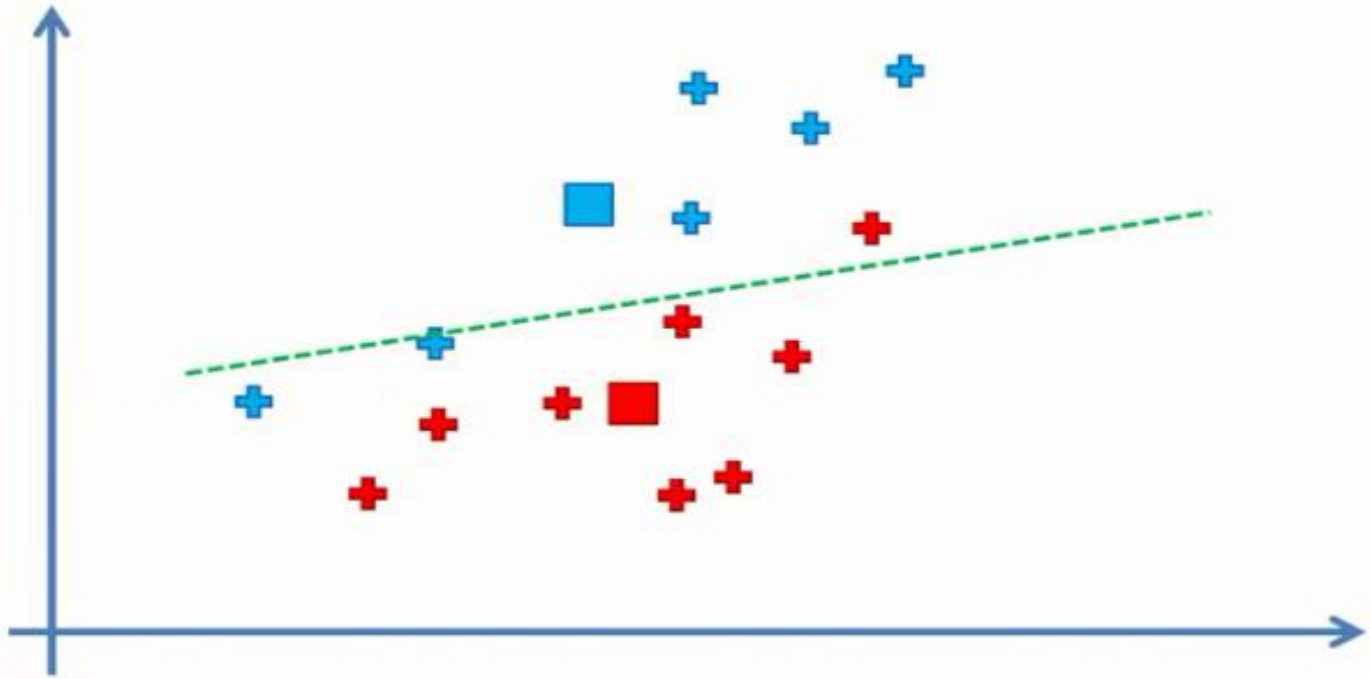
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



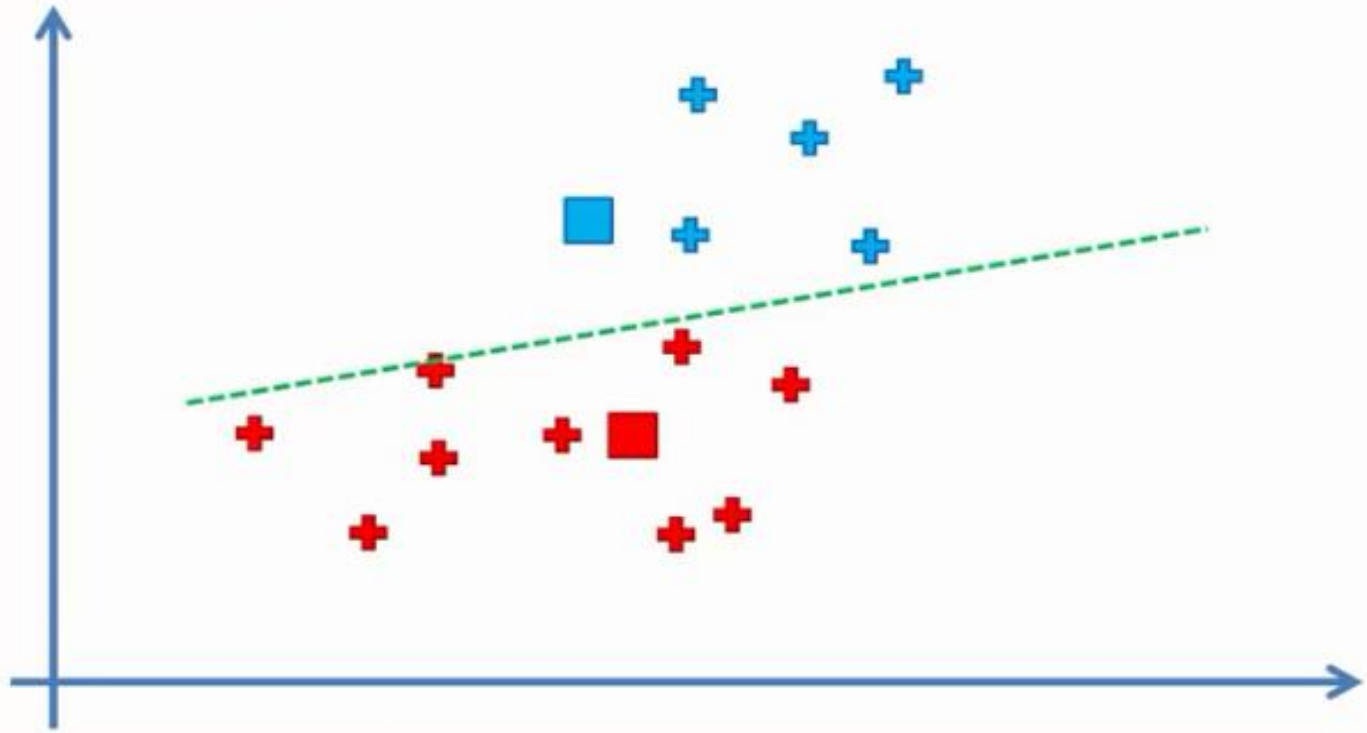
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



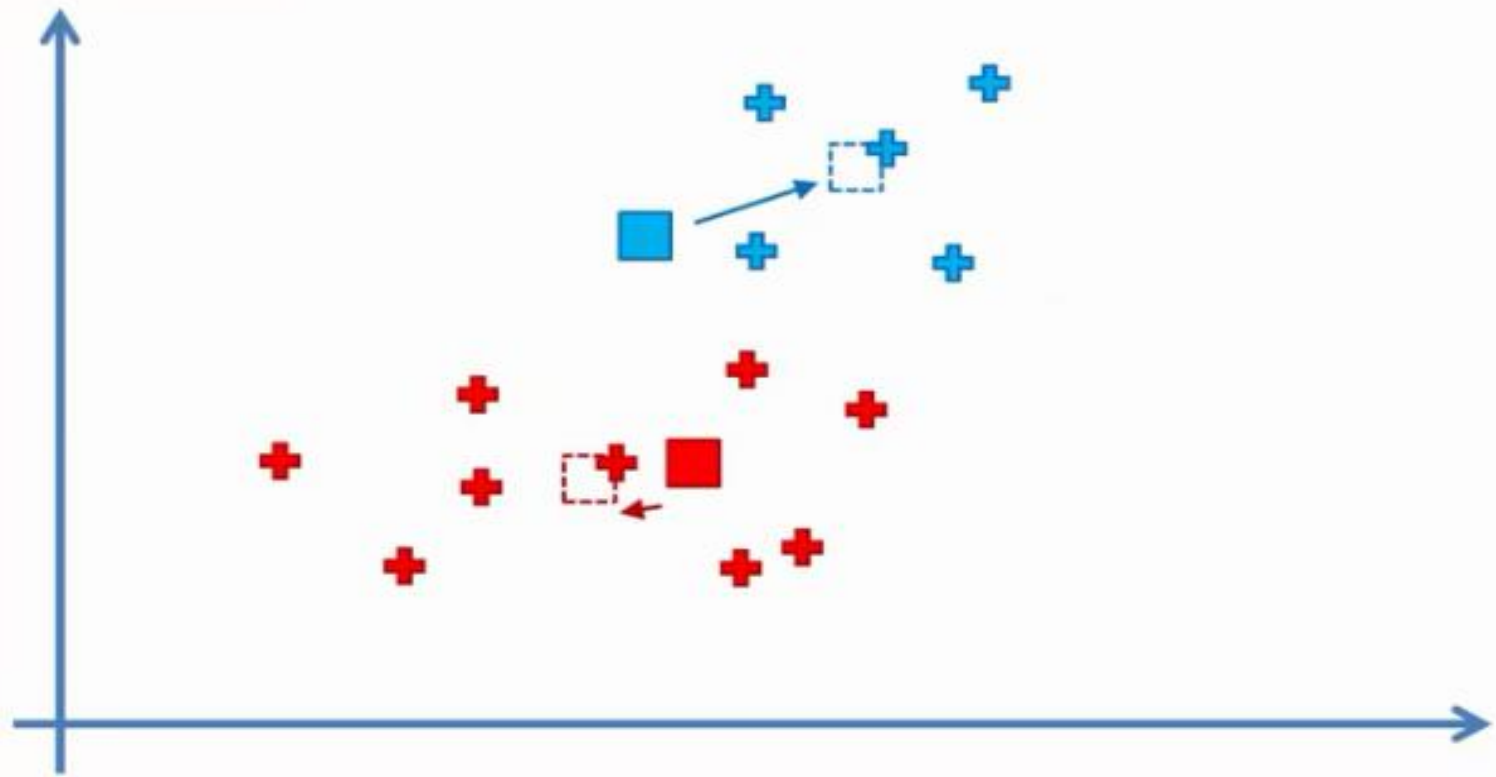
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

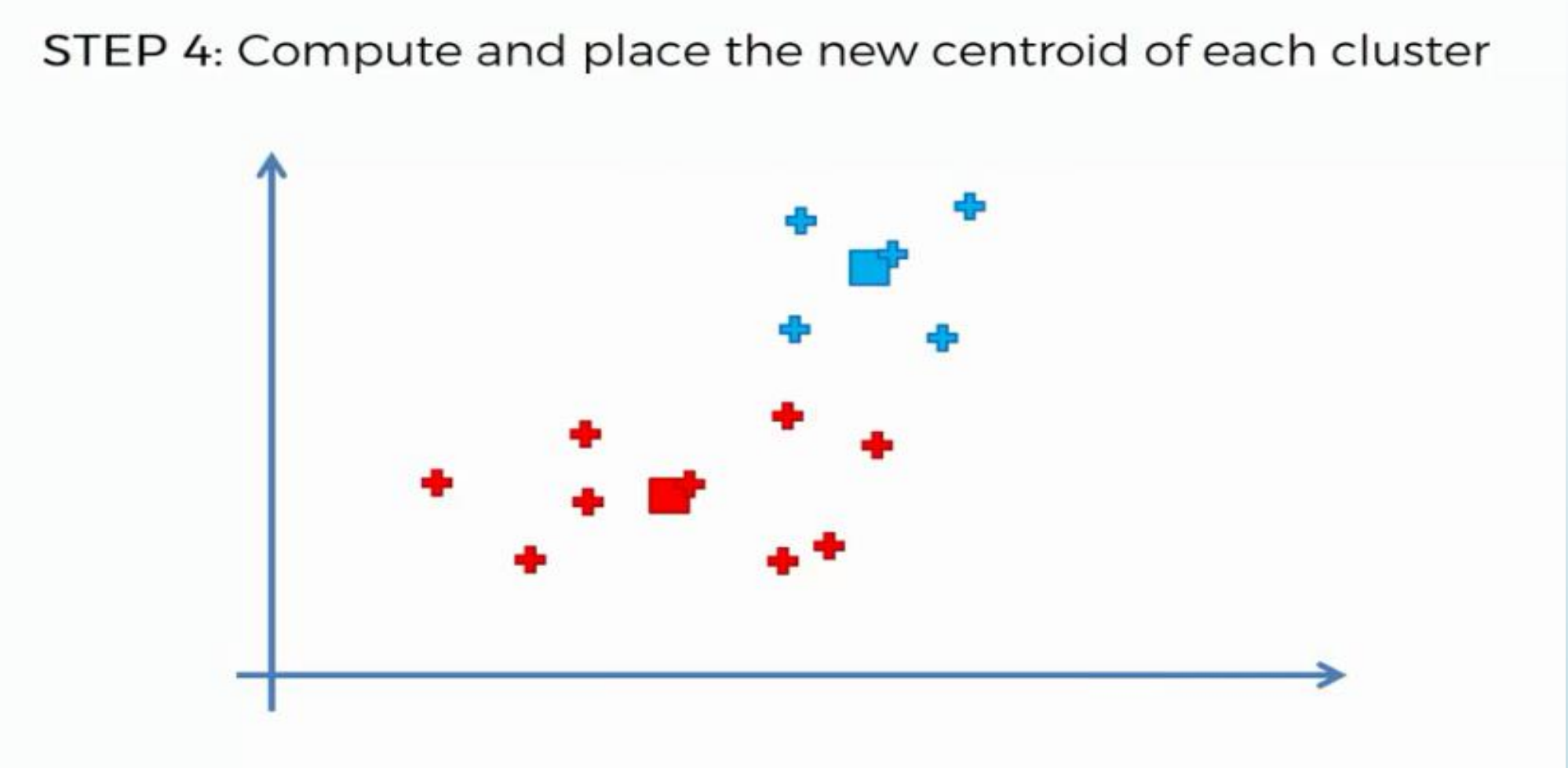


K-Means Algorithm

STEP 4: Compute and place the new centroid of each cluster

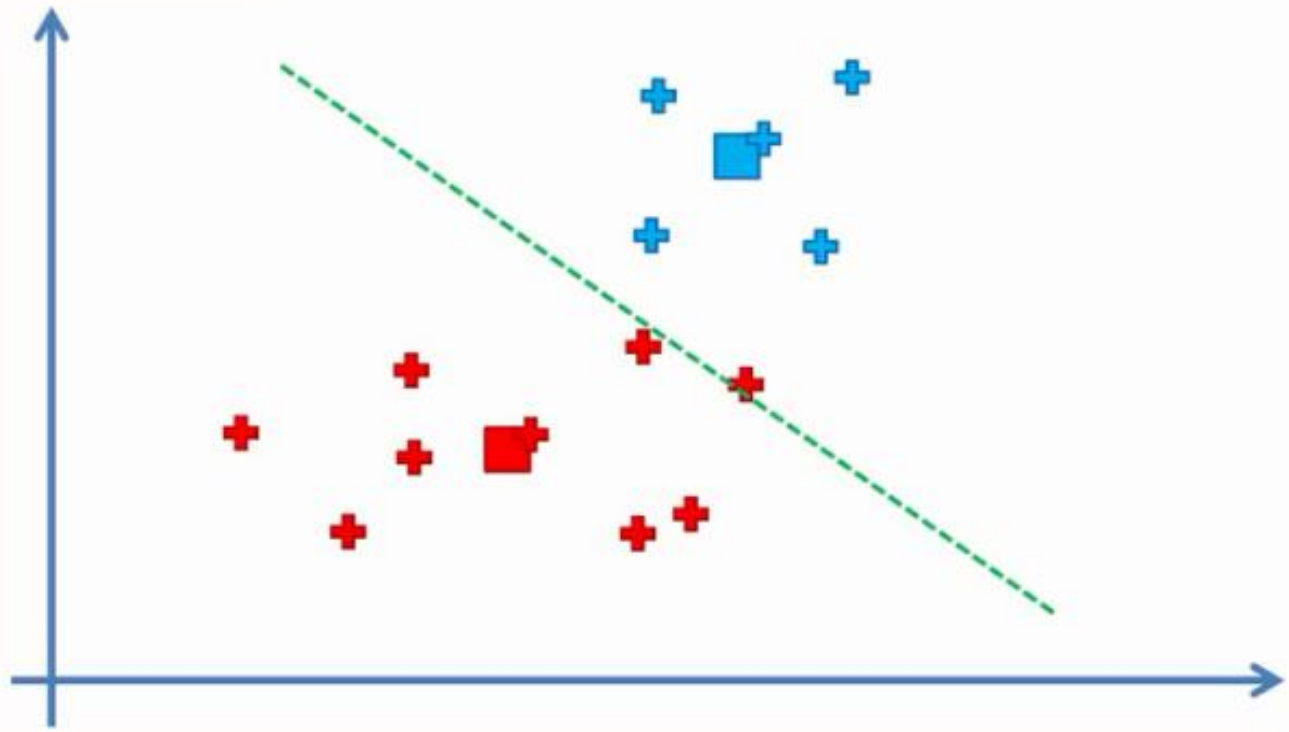


K-Means Algorithm



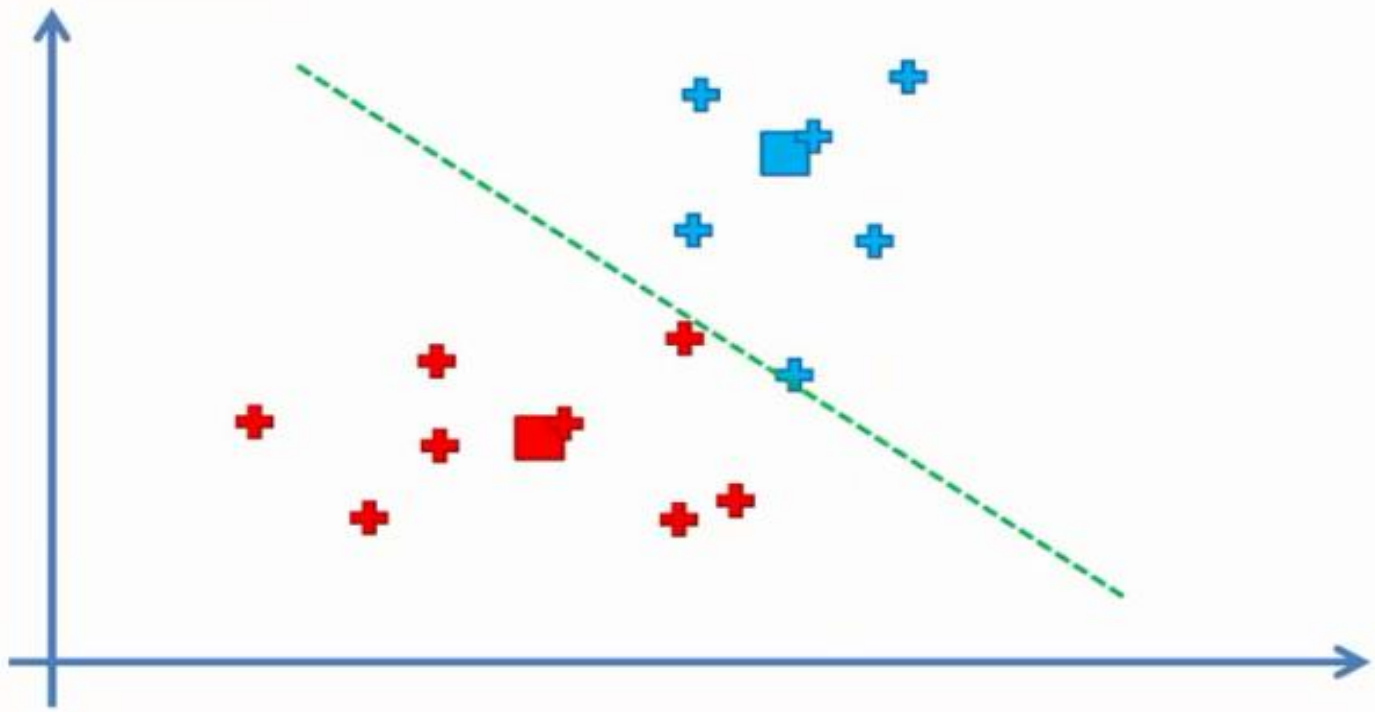
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

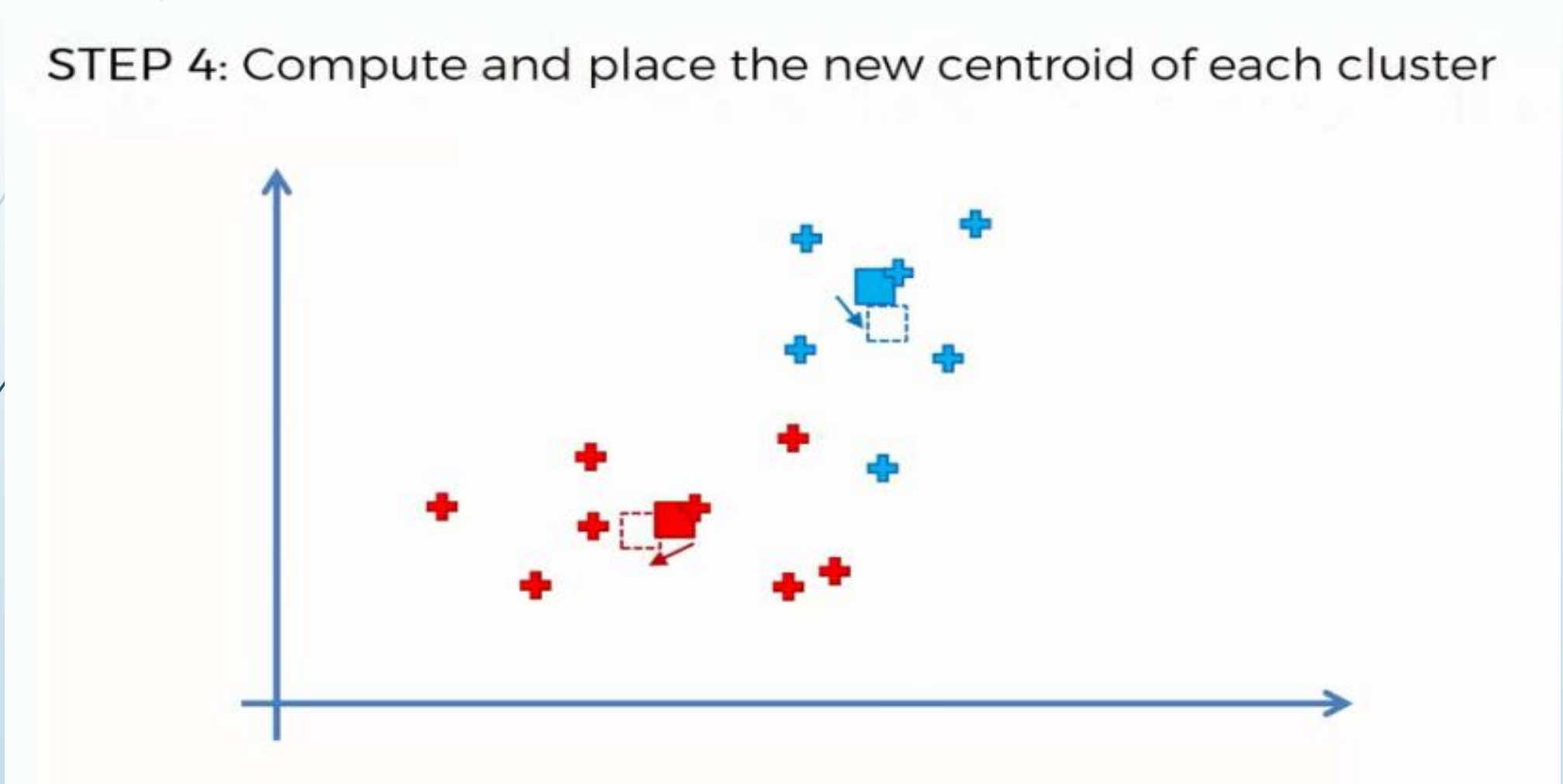


K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.

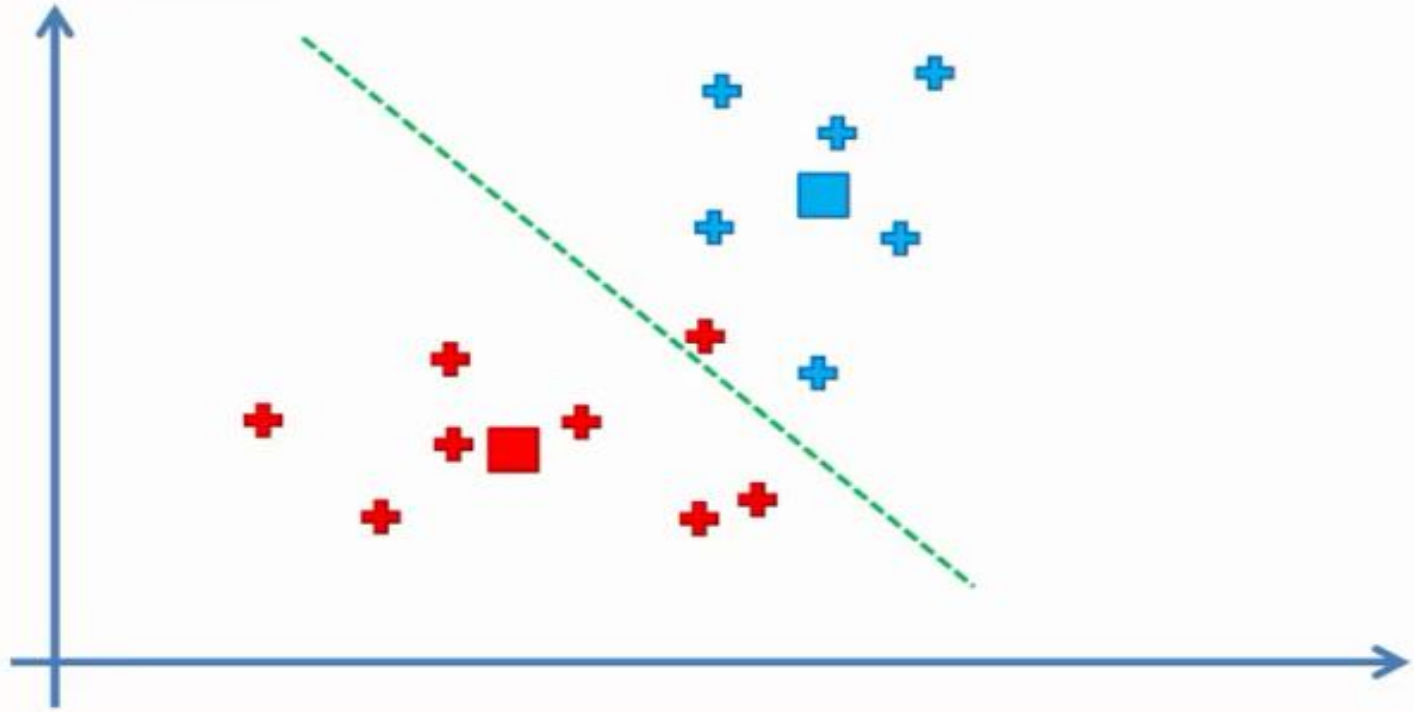


K-Means Algorithm



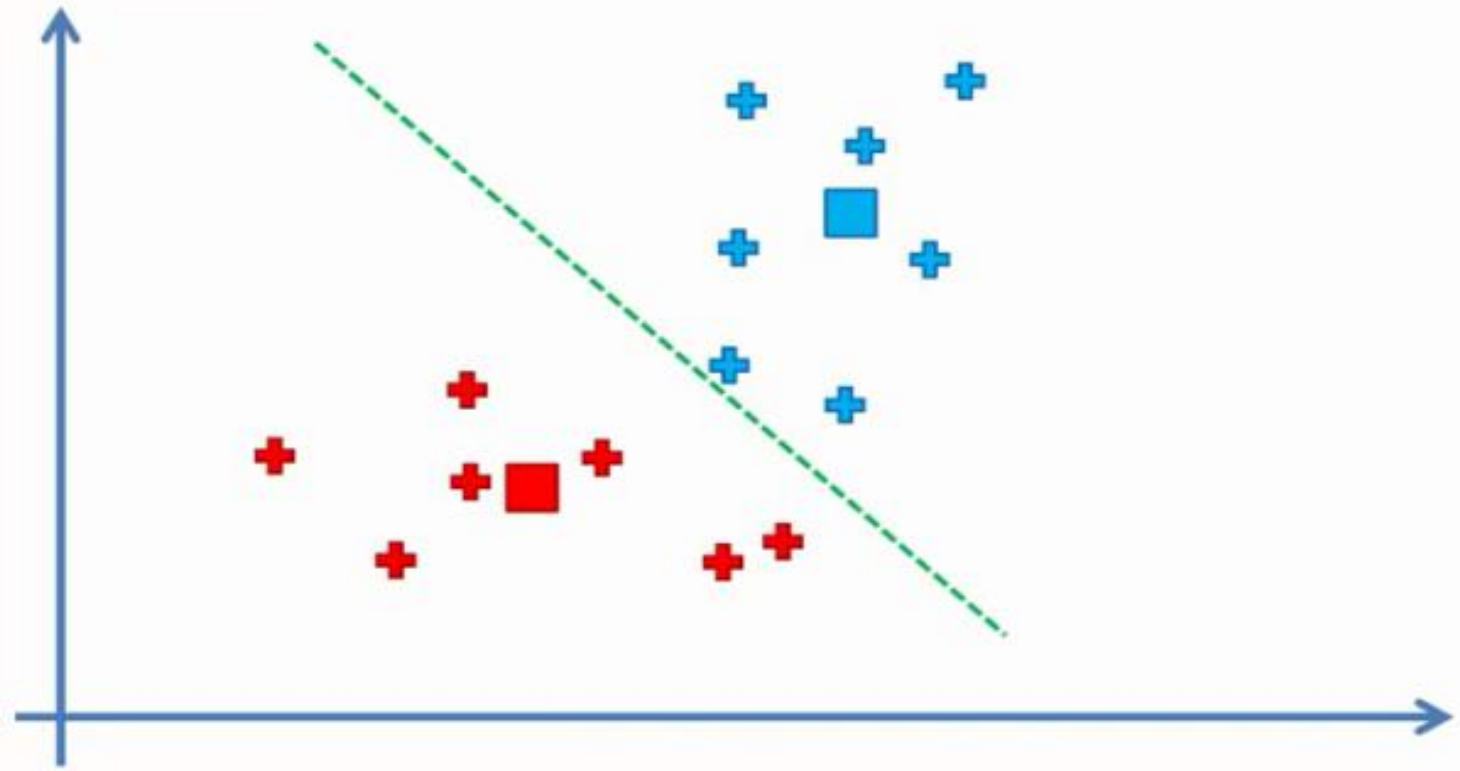
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



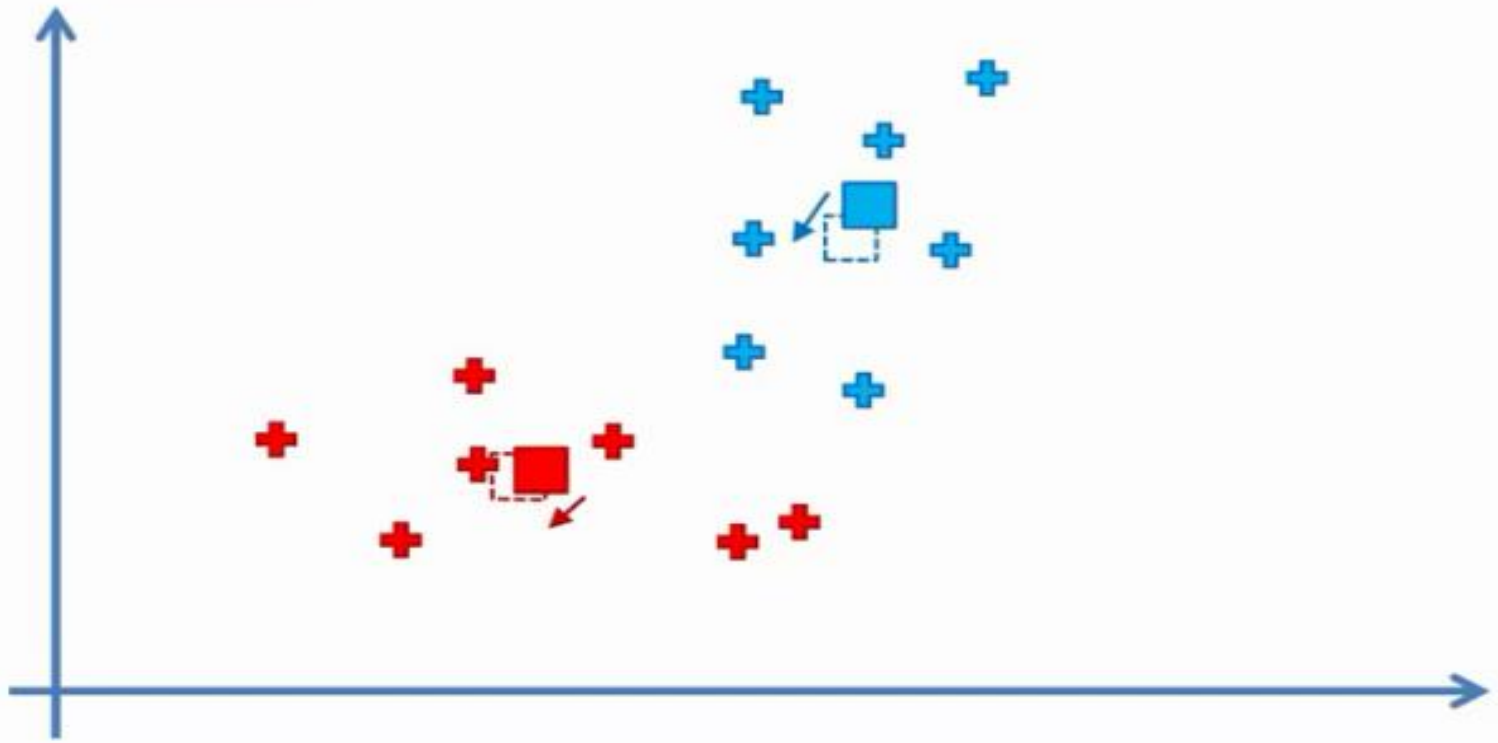
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



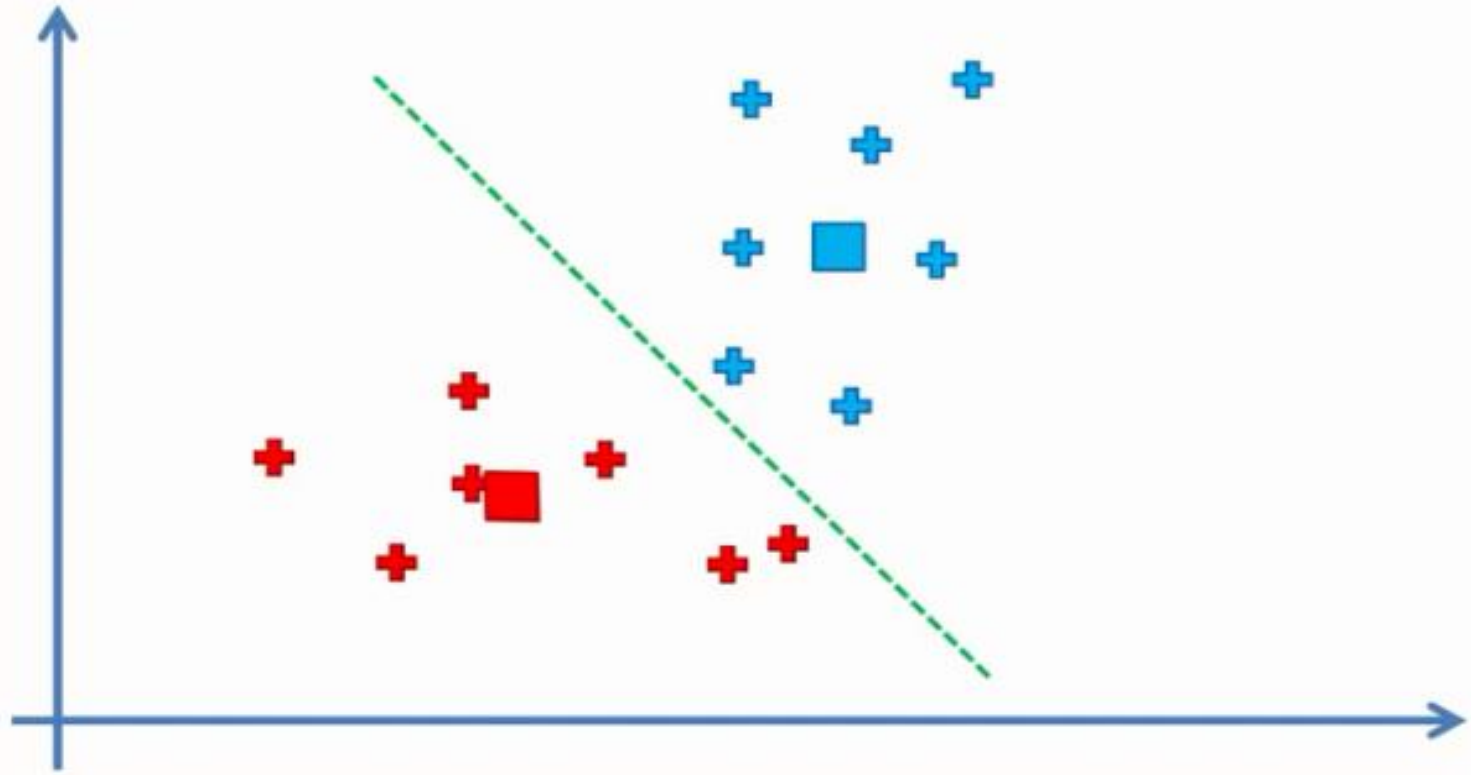
K-Means Algorithm

STEP 4: Compute and place the new centroid of each cluster



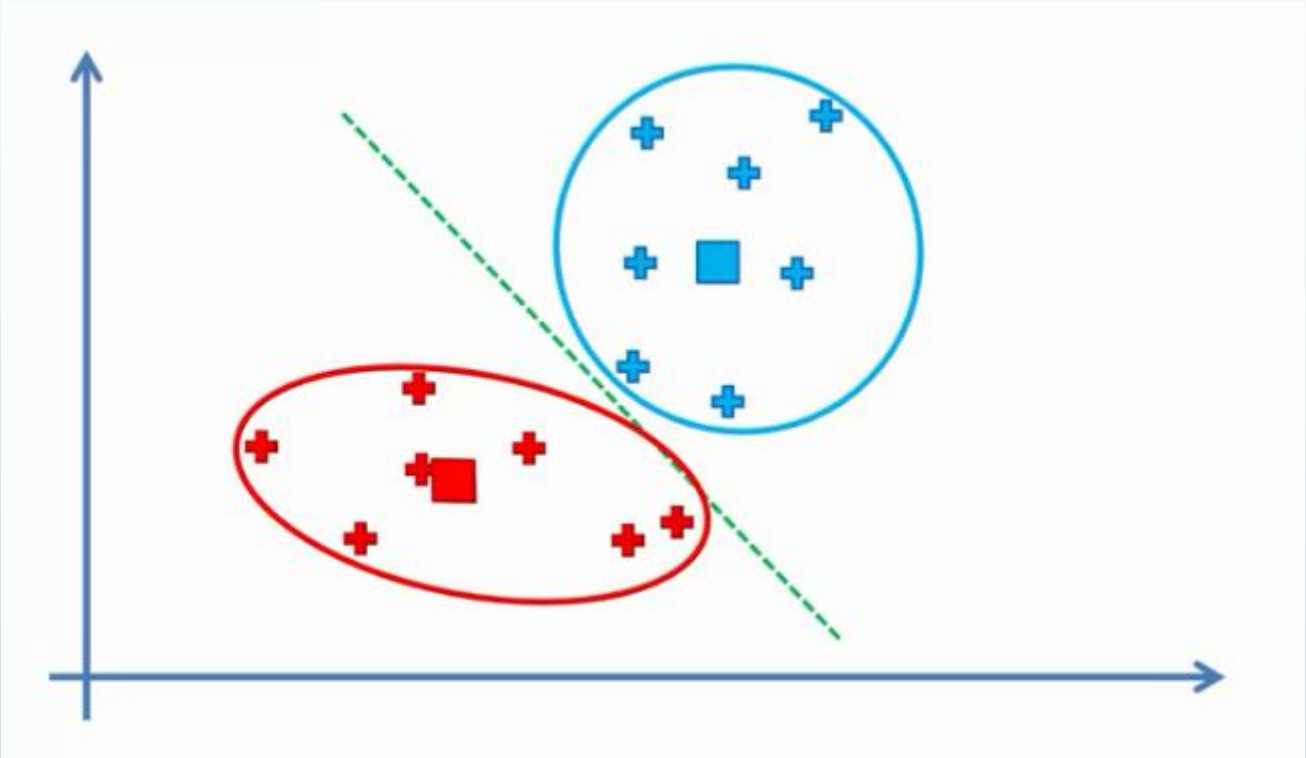
K-Means Algorithm

STEP 5: Reassign each data point to the new closest centroid.
If any reassignment took place, go to STEP 4, otherwise go to FIN.



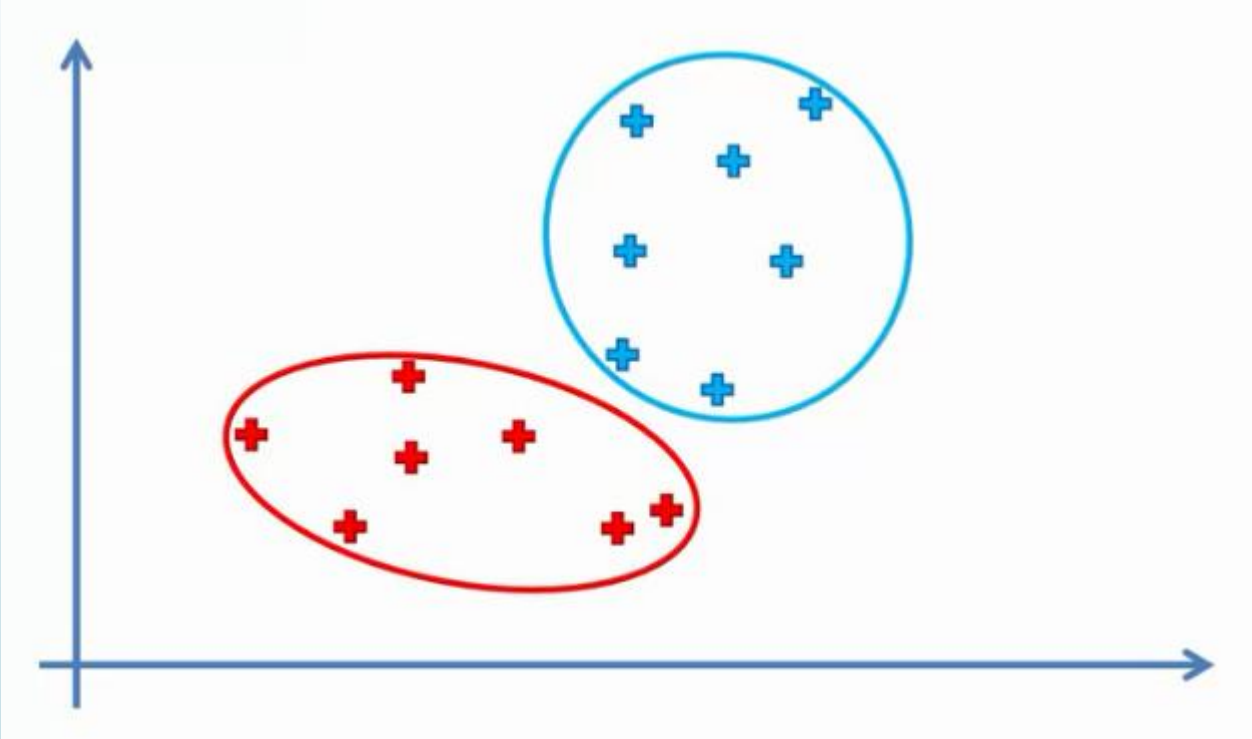
K-Means Algorithm

FIN: Model is ready



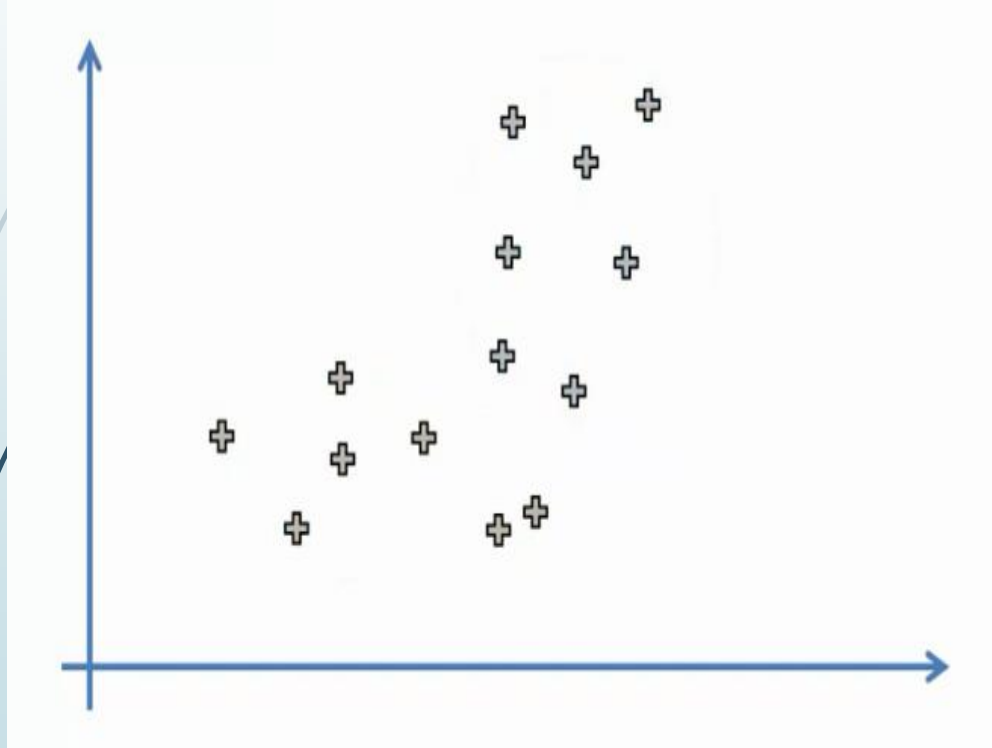
K-Means Algorithm

FIN: Model is ready

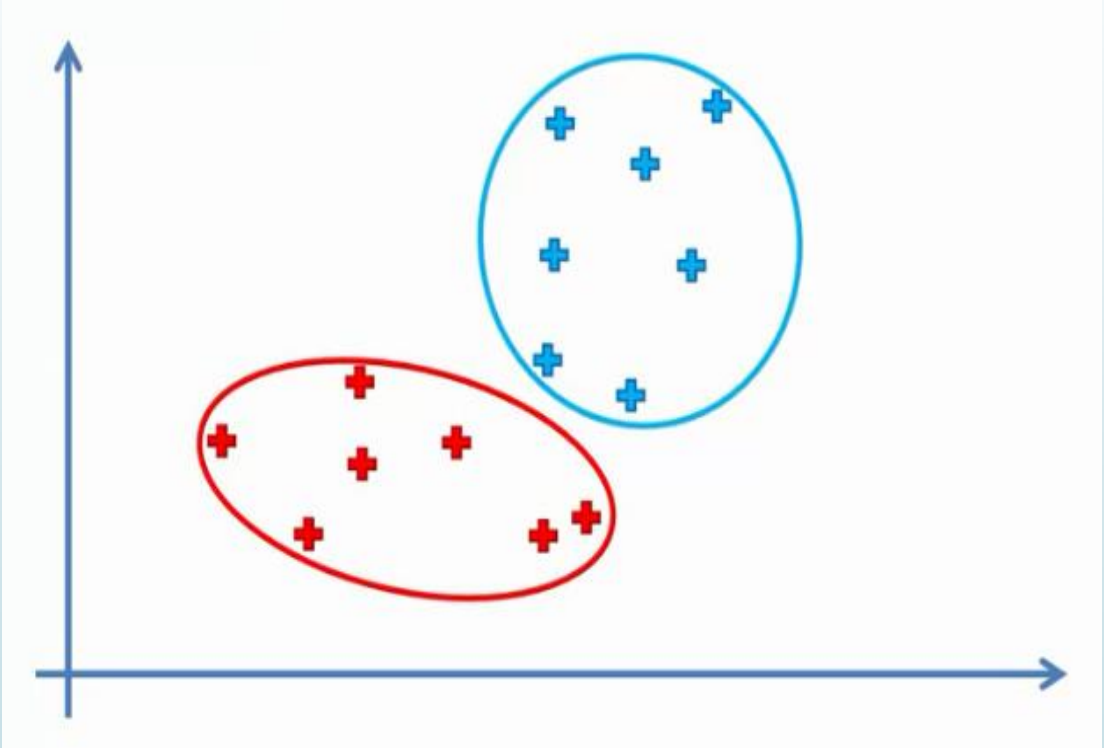


K-Means Algorithm

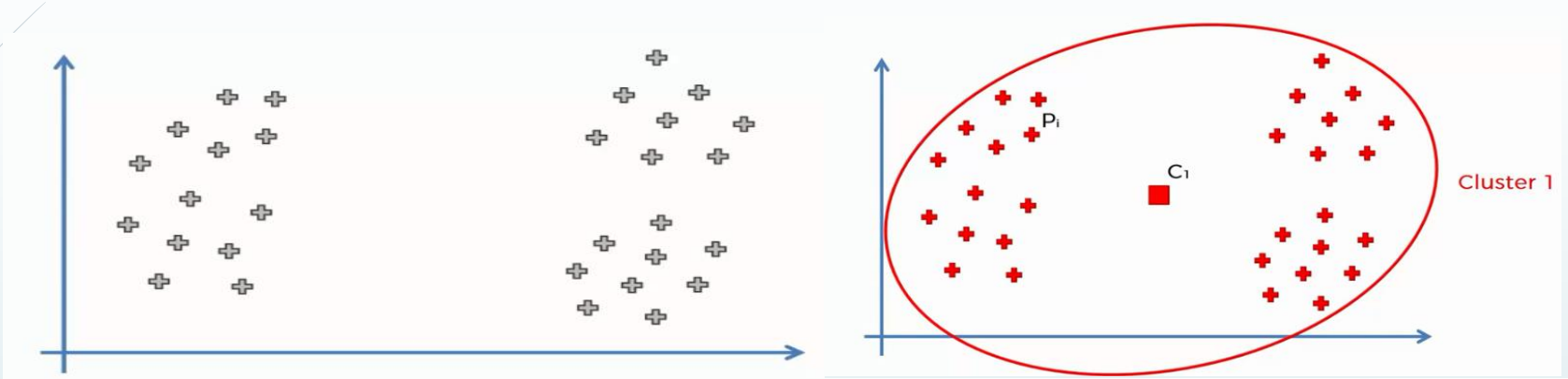
Initial

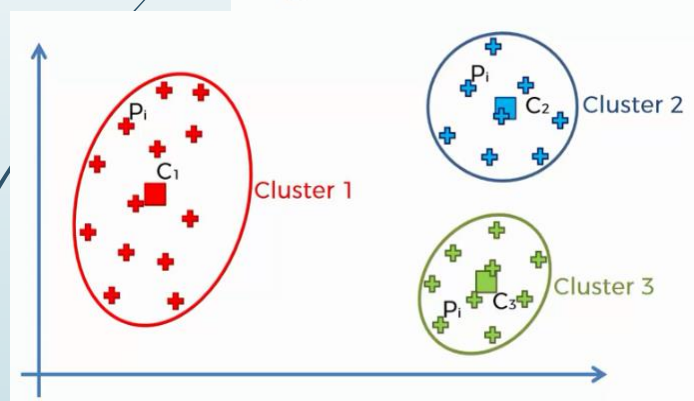


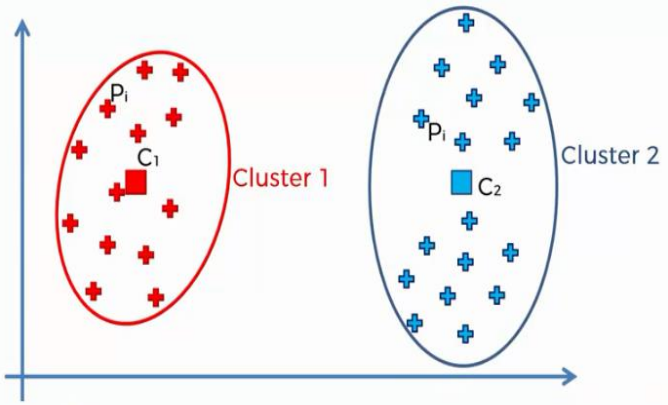
Final

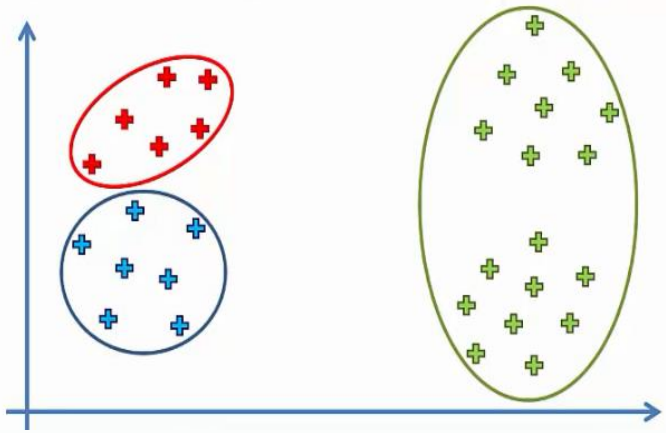


Choosing the Right Number of Cluster









$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

What is WCSS in clustering?

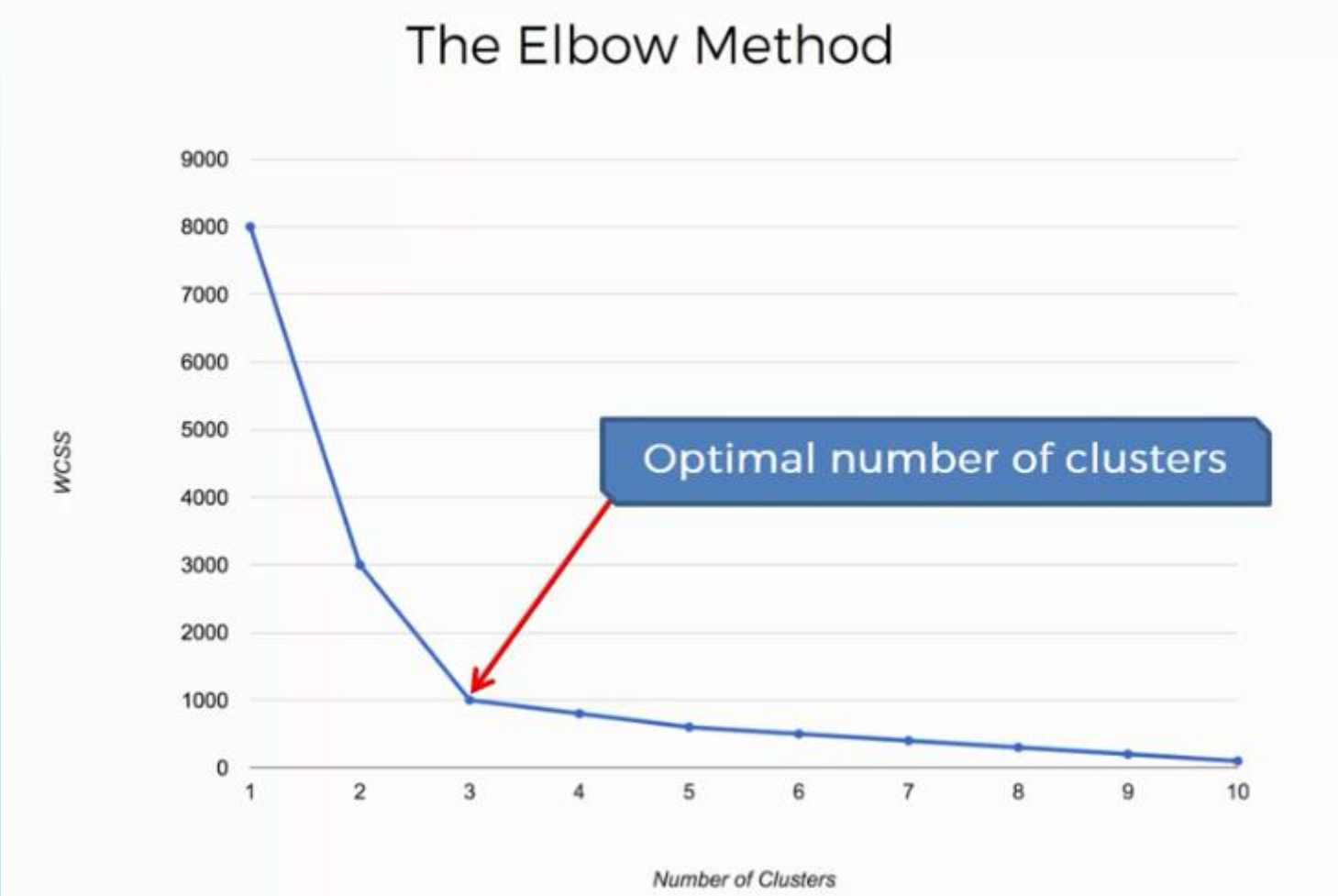
- k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.
- Given a set of observations (x1, x2, …, xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k (≤ n) sets S = {S1, S2, …, Sk} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i-1}^{k} |S_i| \operatorname{Var} S_i$$

where μ_i is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y} \in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

Choosing the Right Number of Cluster



Example

- arousal(-5 to 5) valance

3	**3**
-1	**-4**
2	**3**
0	**-5**

- We know that those data belong to two clusters.
- The question is how to determine which data points belong to cluster 1 and which belong to the other one.

Example

- Initialize the first two centroids

	c1=(3,3)	c2=(2,3)
3	3	
-1	-4	
2	3	
0	-5	

Measuring Distance

- Calculate the distance between two data items.
- Euclidean distance

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

Iteration 1: calculate distances

		c1=(3,3)	c2=(2,3)
3	3	0	1
-1	-4	8.1	7.6
2	3	1	0
0	-5	8.5	8.2

Iteration 1: assign clusters

		c1=(3,3)	c2=(2,3)
3	3	0	1
-1	-4	8.1	7.6
2	3	1	0
0	-5	8.5	8.2

Iteration 1: compute new centroids

		c1=(3,3)	c2=(0.3,-2)
3	3	0	1
-1	-4	8.1	7.6
2	3	1	0
0	-5	8.5	8.2

Iteration 2: calculate distances

		c1=(3,3)	c2=(0.3,-2)
3	3	0	5.7
-1	-4	8.1	2.4
2	3	1	5.3
0	-5	8.5	3.0

Iteration 2: assign clusters

		c1=(3,3)	c2=(0.3,-2)
3	3	0	5.7
-1	-4	8.1	2.4
2	3	1	5.3
0	-5	8.5	3.0

Iteration 2: compute new centroids

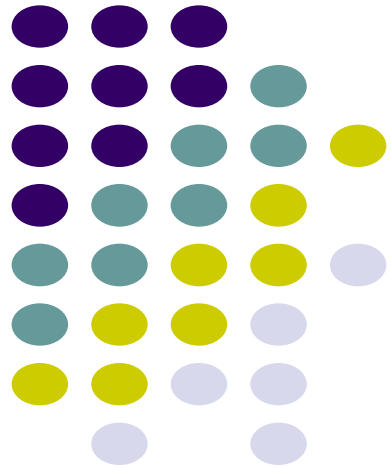
		c1=(2.5,3)	c2=(-0.5,-4.5)
3	3	0	5.7
-1	-4	8.1	2.4
2	3	1	5.3
0	-5	8.5	3.0

Iteration 3

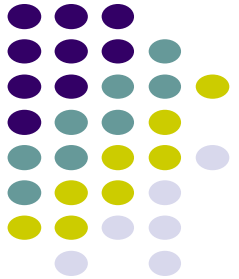
		c1=(2.5,3)	c2=(-0.5,-4.5)
3	3	0.5	8.2
-1	-4	7.8	0.7
2	3	0.5	7.9
0	-5	8.3	0.7

The new centroids will be the same!

K-MEANS CLUSTERING and TF-IDF

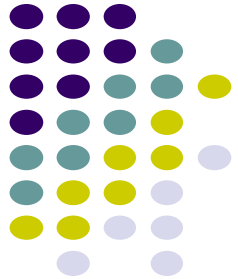


INTRODUCTION-
What is clustering?



- **Clustering** is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

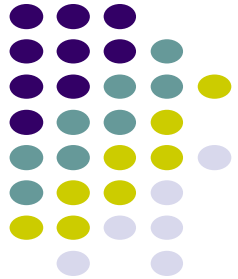
Types of clustering:



1. **Hierarchical algorithms**: these find successive clusters using previously established clusters.
   - Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.
   - Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

2. **Partitional clustering:** Partitional algorithms determine all clusters at once. They include:
   - ***K*-means clustering**
   - Fuzzy *c*-means clustering
   - QT clustering algorithm

Common Distance measures:



- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

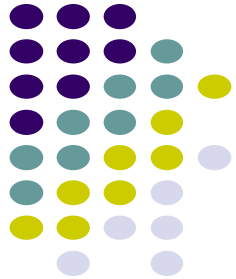
  They include:

1. The Euclidean distance (also called 2-norm distance) is given by:

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}$$

2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

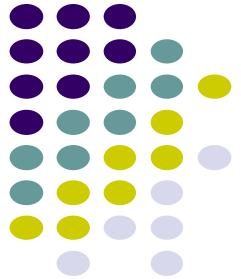
$$|x_1 - x_2| + |y_1 - y_2|.$$



4. Cosine distance: cosine distance is a measure of similarity between two non-zero vectors defined in an inner product space. Cosine similarity is the cosine of the angle between the vectors; that is, it is the dot product of the vectors divided by the product of their lengths

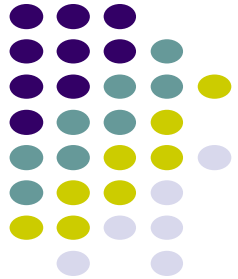
$$\cos(x, y) = \frac{x.y}{||x|| * ||y||}$$

K-MEANS CLUSTERING



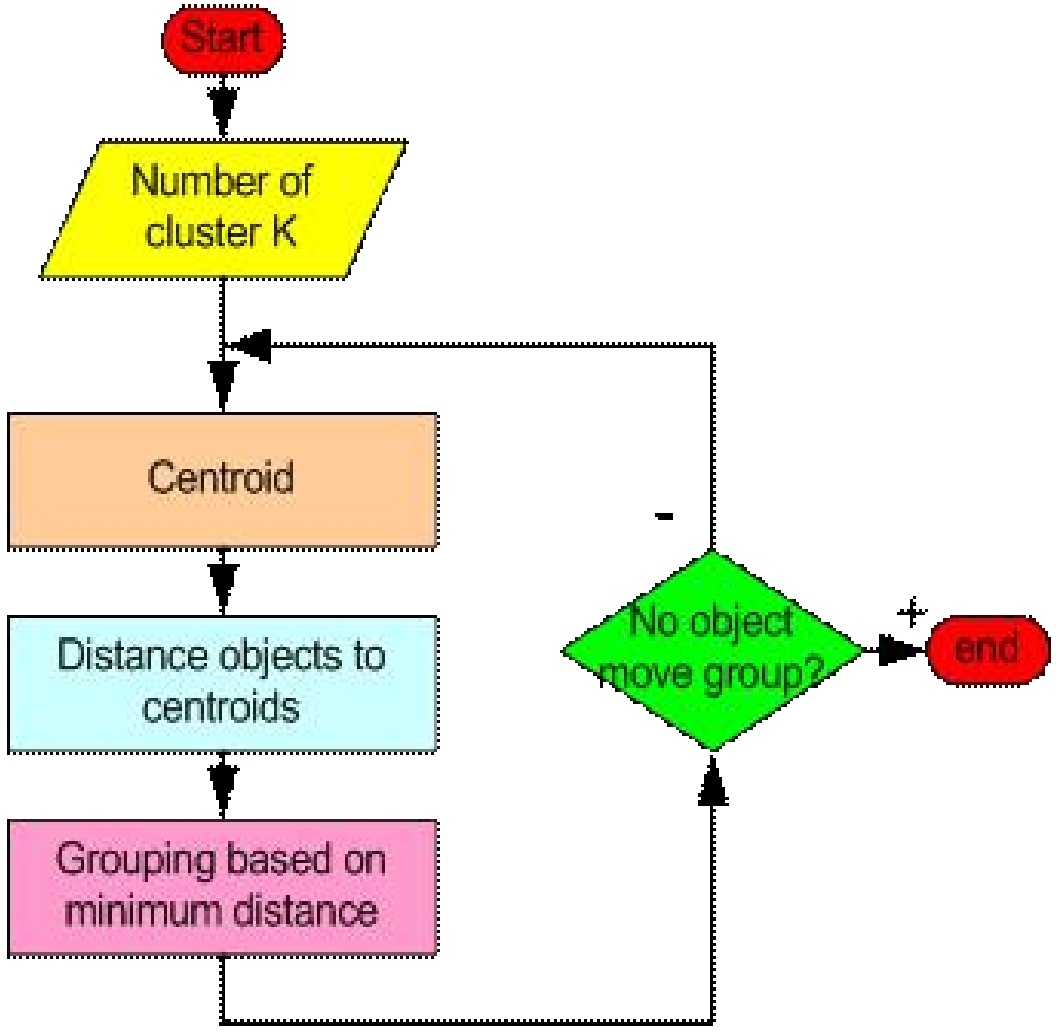
- The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k < n$.
- It assumes that the object attributes form a vector space.
- It is an algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

K-MEANS CLUSTERING

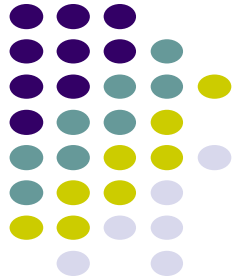


- Simply speaking k-means clustering is an algorithm to classify or group the objects based on attributes/features into K number of group.
- K is a positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

K-Mean Clustering algorithm



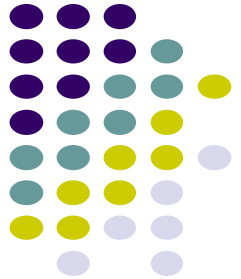
K-Mean Clustering algorithm



- **Step 1:** *Initialization:* Choose the number of clusters, k, that you want to create. Randomly initialize k points in the feature space as the initial centroids.

- **Step 2**: *Assignment:* Assign each data point to the nearest centroid based on a distance metric, often Euclidean distance. Each data point belongs to the cluster with the closest centroid.

K-Mean Clustering algorithm



- **Step 3:** *Update:* Calculate the new centroids for each cluster by taking the mean of all data points assigned to that cluster.

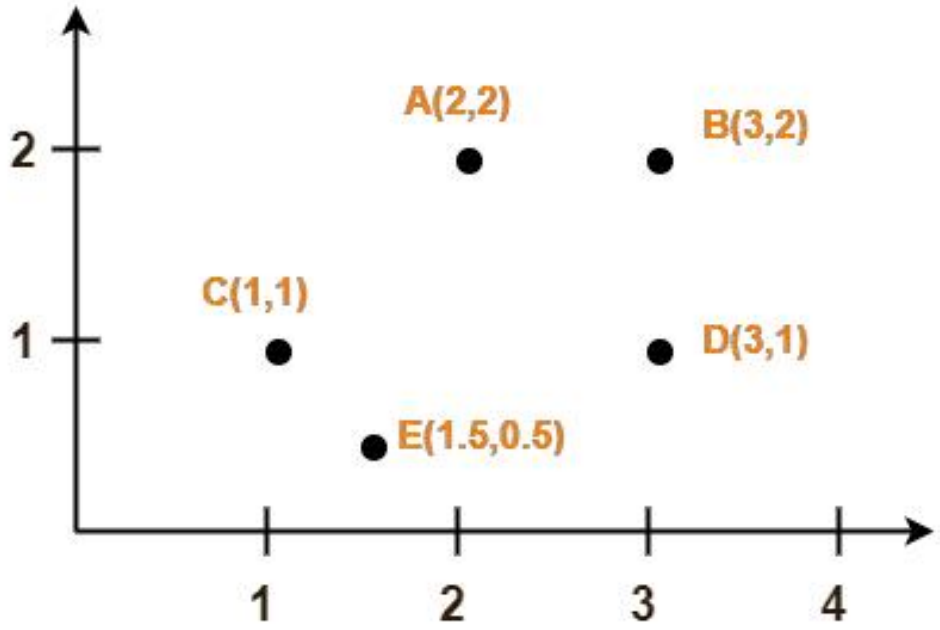
- **Step 4 .** *Iteration:* Repeat steps 2 and 3 until convergence, which occurs when the centroids no longer change significantly or a maximum number of iterations is reached.

- **Step 5.** *Output*: The final output of the k-means algorithm is a set of k clusters, where each data point is assigned to one cluster

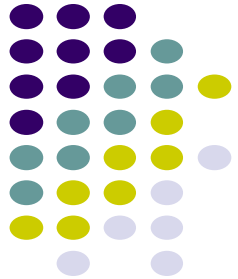
K-Means Clustering Example

Use K-Means Algorithm to create two clusters-

Assume A(2, 2) and C(1, 1) are centers of the two clusters

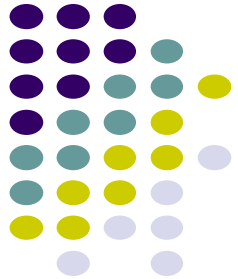


K-Means Clustering Example



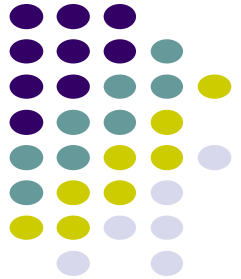
Given Points	Distance from center (2, 2) of Cluster-01	Distance from center (1, 1) of Cluster-02	Point belongs to Cluster
A(2, 2)	0	1.41	C1
B(3, 2)	1	2.24	C1
C(1, 1)	1.41	0	C2
D(3, 1)	1.41	2	C1
E(1.5, 0.5)	1.58	0.71	C2

K-Means Clustering Example



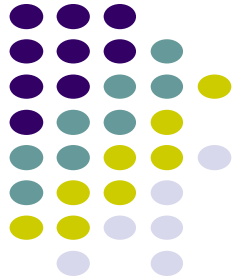
- **Cluster-01:**
  - First cluster contains points-
  - A(2, 2)
  - B(3, 2)
  - D(3, 1)
- **Cluster-02:**
  - Second cluster contains points-
  - C(1, 1)
  - E(1.5, 0.5)

K-Means Clustering Example



- **For Cluster-01:**
  - Center of Cluster-01
  - = ((2 + 3 + 3)/3, (2 + 2 + 1)/3)
  - = (2.67, 1.67)
- **For Cluster-02:**
  - Center of Cluster-02
  - = ((1 + 1.5)/2, (1 + 0.5)/2)
  - = (1.25, 0.75)

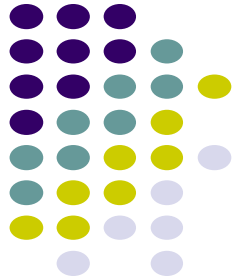
Real-Life Numerical Example of K-Means Clustering



We have 4 medicines as our training data points object and each medicine has 2 attributes. Each attribute represents coordinate of the object. We have to determine which medicines belong to cluster 1 and which medicines belong to the other cluster.

Object	Attribute1(X):weight index	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Weaknesses of K-Mean Clustering

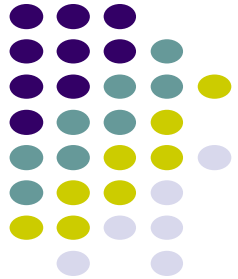


1. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
2. The number of cluster, K, must be determined before hand. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
3. We never know the real cluster, using the same data, because if it is inputted in a different order it may produce different cluster if the number of data is few.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the *local optimum*.

Applications of K-Mean Clustering

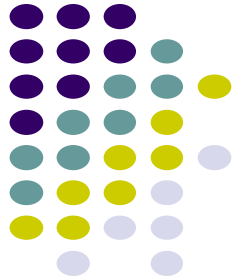
- K-means clustering has various applications, such as customer segmentation, image compression, document clustering, and anomaly detection. However, it has some limitations, such as sensitivity to the initial centroid placement and the requirement to specify the number of clusters beforehand.

CONCLUSION



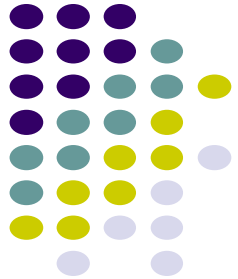
- *K-means algorithm is* useful for undirected knowledge discovery and is relatively simple. K-means has found wide spread usage in lot of fields, ranging from unsupervised learning of neural network, Pattern recognitions, Classification analysis, Artificial intelligence, image processing, machine vision, and many others.

TF-IDF



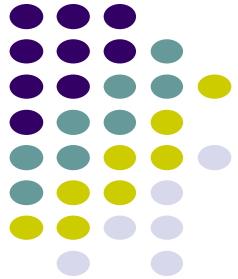
- TF-IDF stands for Term Frequency-Inverse Document Frequency.
- It is a numerical representation used in natural language processing and information retrieval to quantify the importance of a term in a document within a collection of documents.
- TF-IDF is commonly used to extract relevant information and perform text-mining tasks like document similarity, document clustering, and information retrieval.

TF



- Term Frequency (TF): It measures the frequency of a term in a document.
- The idea is that the more times a term appears in a document, the more important it is. TF is calculated as the ratio of the number of occurrences of a term to the total number of terms in the document.
- $TF = \frac{\text{(Number of occurrences of a term in a document)}}{\text{(Total number of terms in the document)}}$
- Logarithmically scaled frequency (e.g. log(1 + raw count))

TF



- Consider a document:
  - *"I love to eat apples. Apples are delicious and healthy."*
- To calculate the term frequency of the term "apples" in this document, we count the number of times it appears. In this case, "apples" occurs twice in the document.
- Total number of terms in the document: 9 Number of occurrences of "apples": 2

- **Term Frequency (TF) for "apples" = 2 / 9 ≈ 0.222**

- Higher TF values suggest that a term is more important or relevant within the document. It's important to note that TF does not consider the frequency of the term in the entire collection of documents; it only focuses on the individual document.

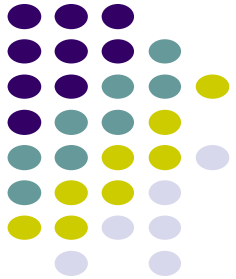
IDF

- Inverse Document Frequency (IDF): It measures the significance of a term across the entire collection of documents.
- The idea is that terms that appear frequently across many documents are less informative compared to terms that appear in a limited number of documents.
- IDF is calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term.
  - $IDF = \log(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ containing\ the\ term})$
- It quantifies how common or rare a term is across the document corpus.

IDF

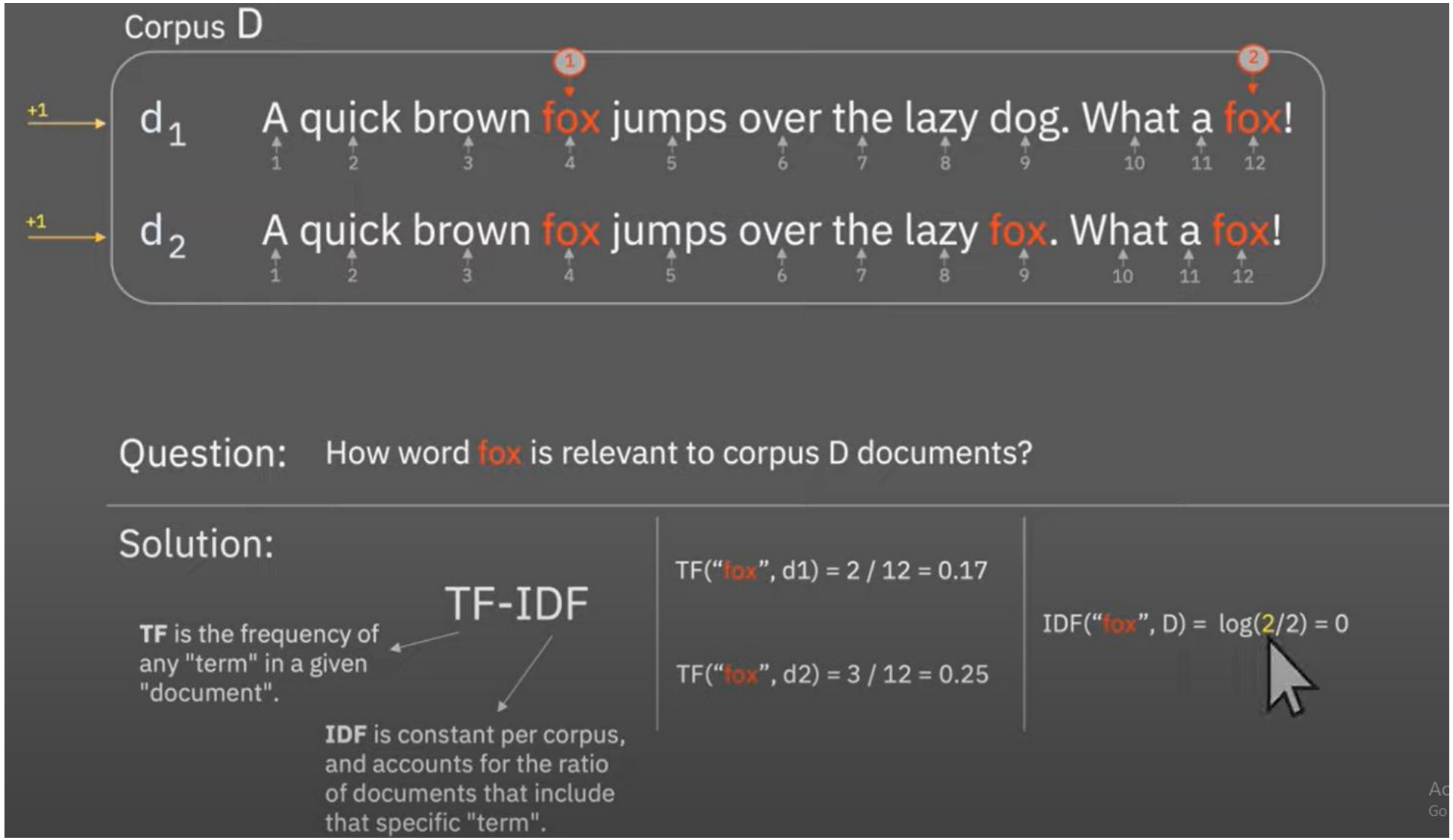
- Consider a collection of five documents:
  - Document 1: "I love to eat apples."
  - Document 2: "Apples are delicious and healthy."
  - Document 3: "I enjoy eating oranges."
  - Document 4: "Oranges are juicy and refreshing."
  - Document 5: "I like to have fruits."
- To calculate the IDF for the term "apples"
  - Total number of documents (N): 5 Number of documents containing "apples": 2
  - $IDF for\ "Apples" = \log(\frac{Total\ number\ of\ documents\ (5)}{Number\ of\ documents\ containing\ the\ term\ (2)}) \approx 0.397$
  - Terms with higher IDF values are considered **more unique or rare** across the document collection

TF-IDF

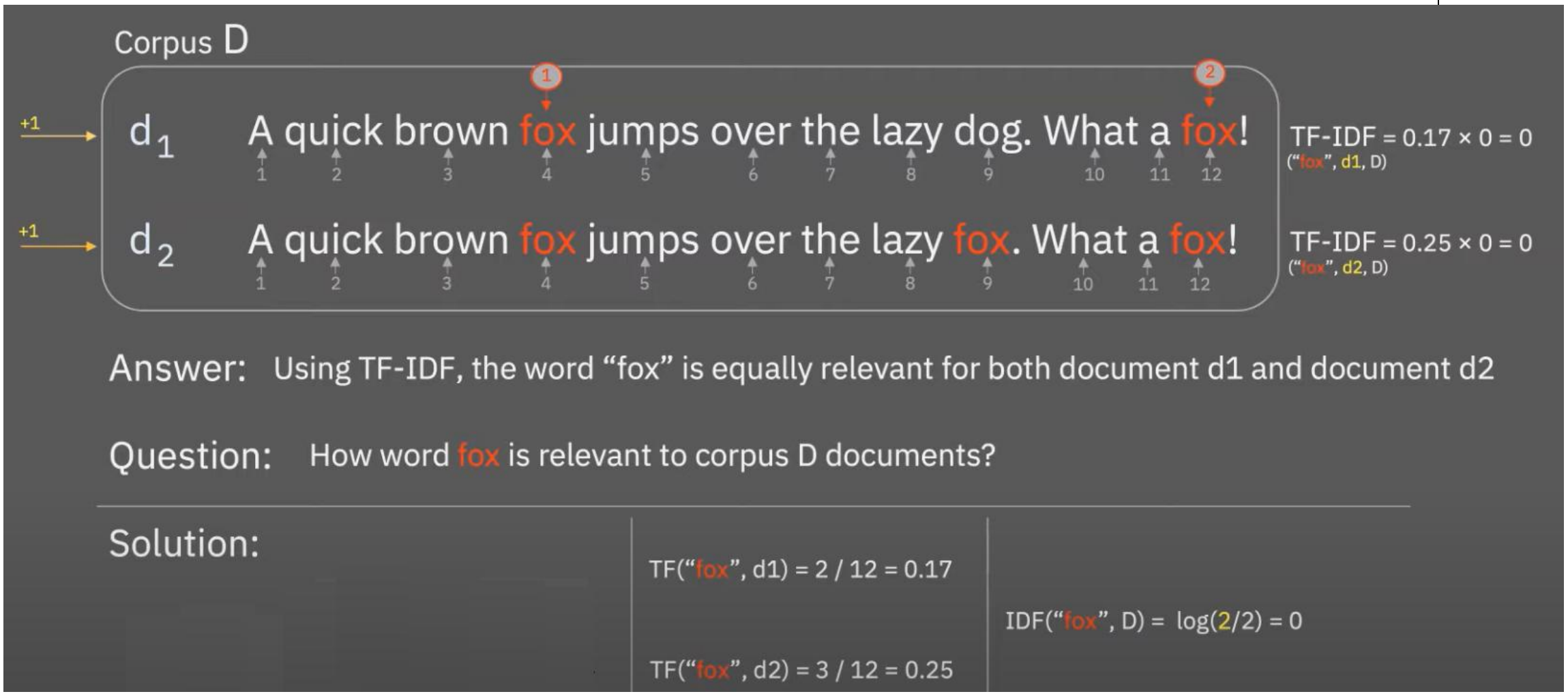


- The TF-IDF score is obtained by multiplying the TF and IDF values for a term in a document.
- This score assigns a higher weight to terms that are frequent within a document but rare across the document collection, emphasizing the importance of those terms in distinguishing the document from others.

TF-IDF (example)



TF-IDF (example)



TF

Document-Term Matrix

	t1	t2	t3	t4	t5	t6
D1	24	21	9	0	0	3
D2	32	10	5	0	3	0
D3	12	16	5	0	0	0
D4	6	7	2	0	0	0
D5	43	31	20	0	3	0
D6	2	0	0	18	7	16
D7	0	0	1	32	12	0
D8	3	0	0	22	4	2
D9	1	0	0	34	27	25
D10	6	0	0	17	4	23

Terms used
In "Database"
Related

t1	database
t2	SQL
t3	index
t4	regression
t5	likelihood
t6	linear

"Regression" Terms

d_{ij} represents number of times that term appears in that document