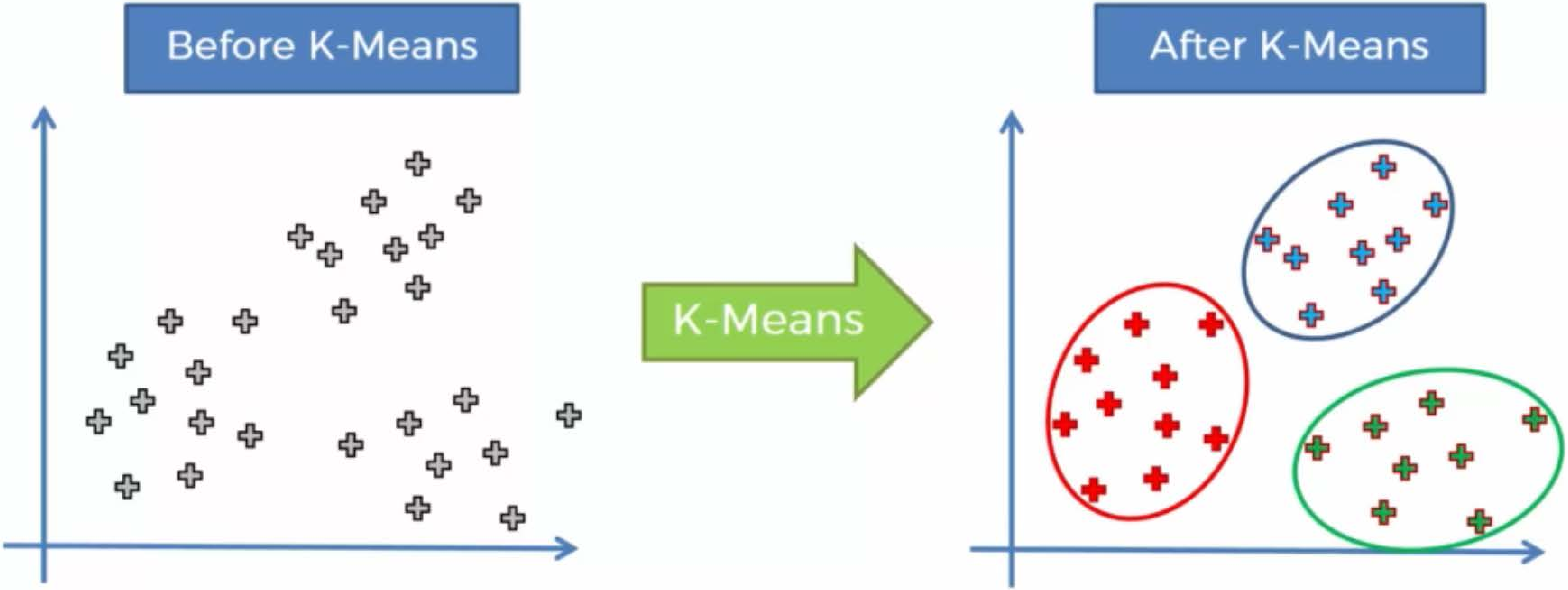
K-Means Intuition: Understanding K-Means

What K-Means does for you



How did it do that ?

STEP 1: Choose the number K of clusters

⬇

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

⬇

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

⬇

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

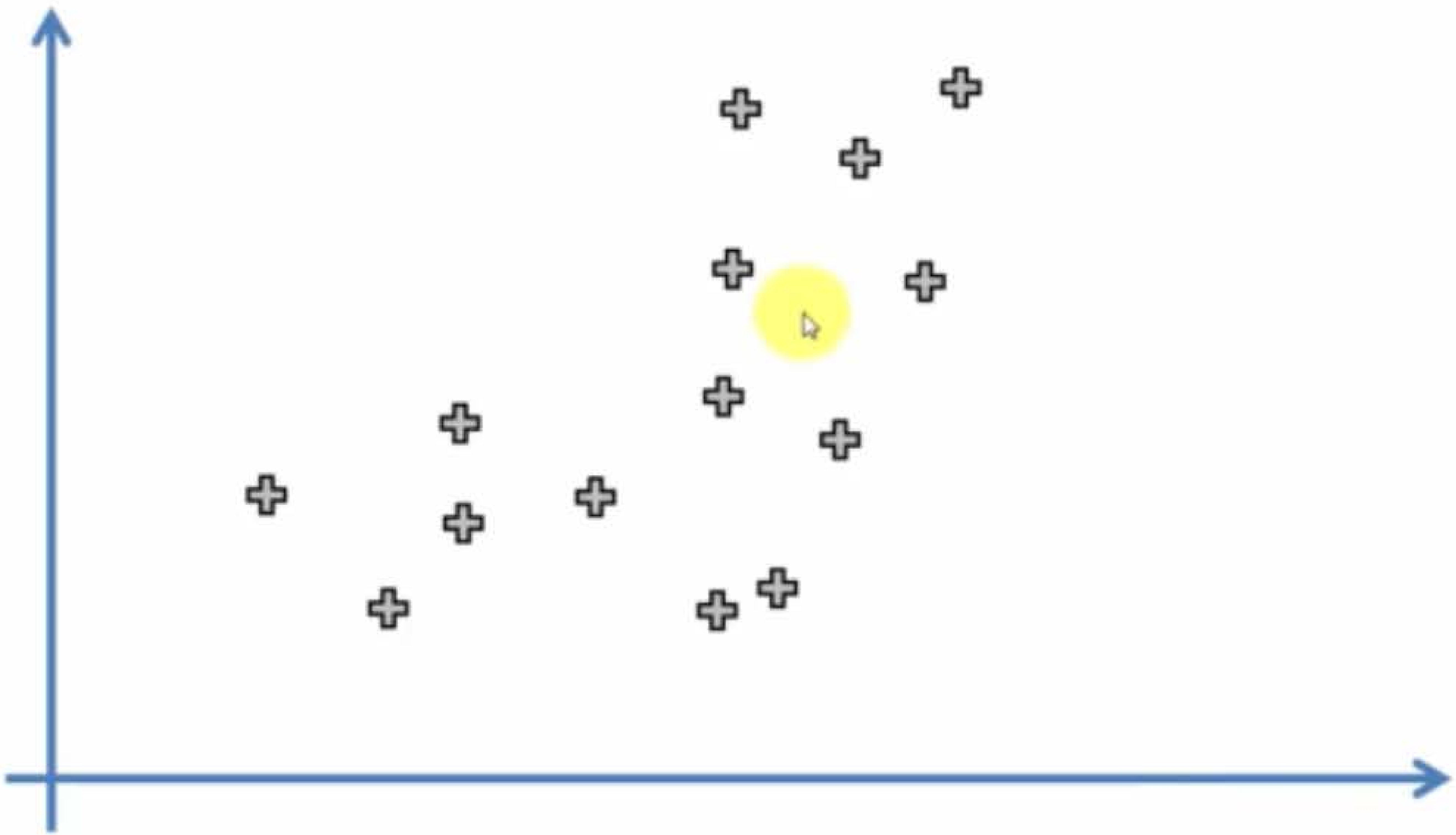
⬇

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

➡ **Your Model is Ready**

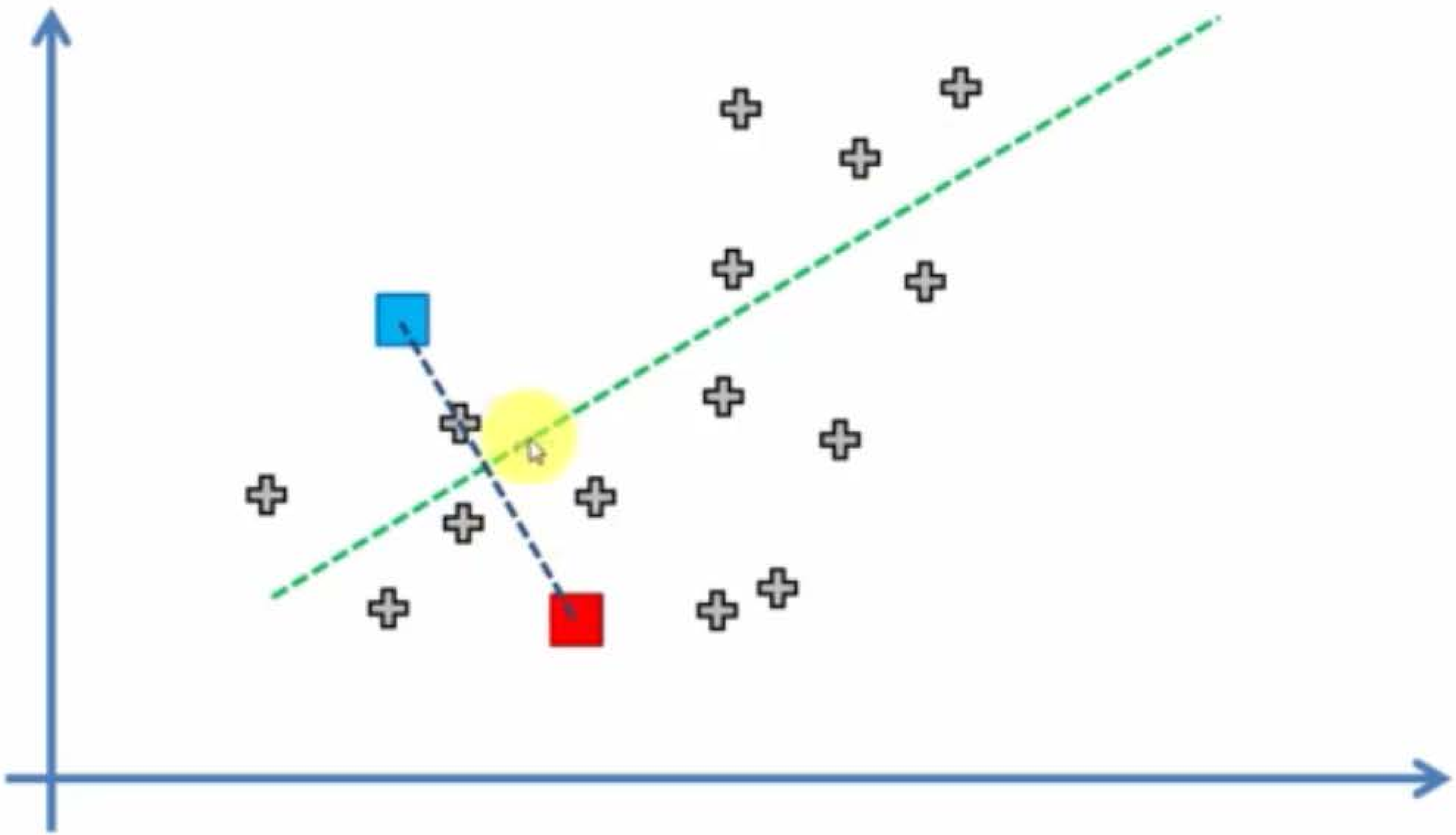
K-Means algorithm

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



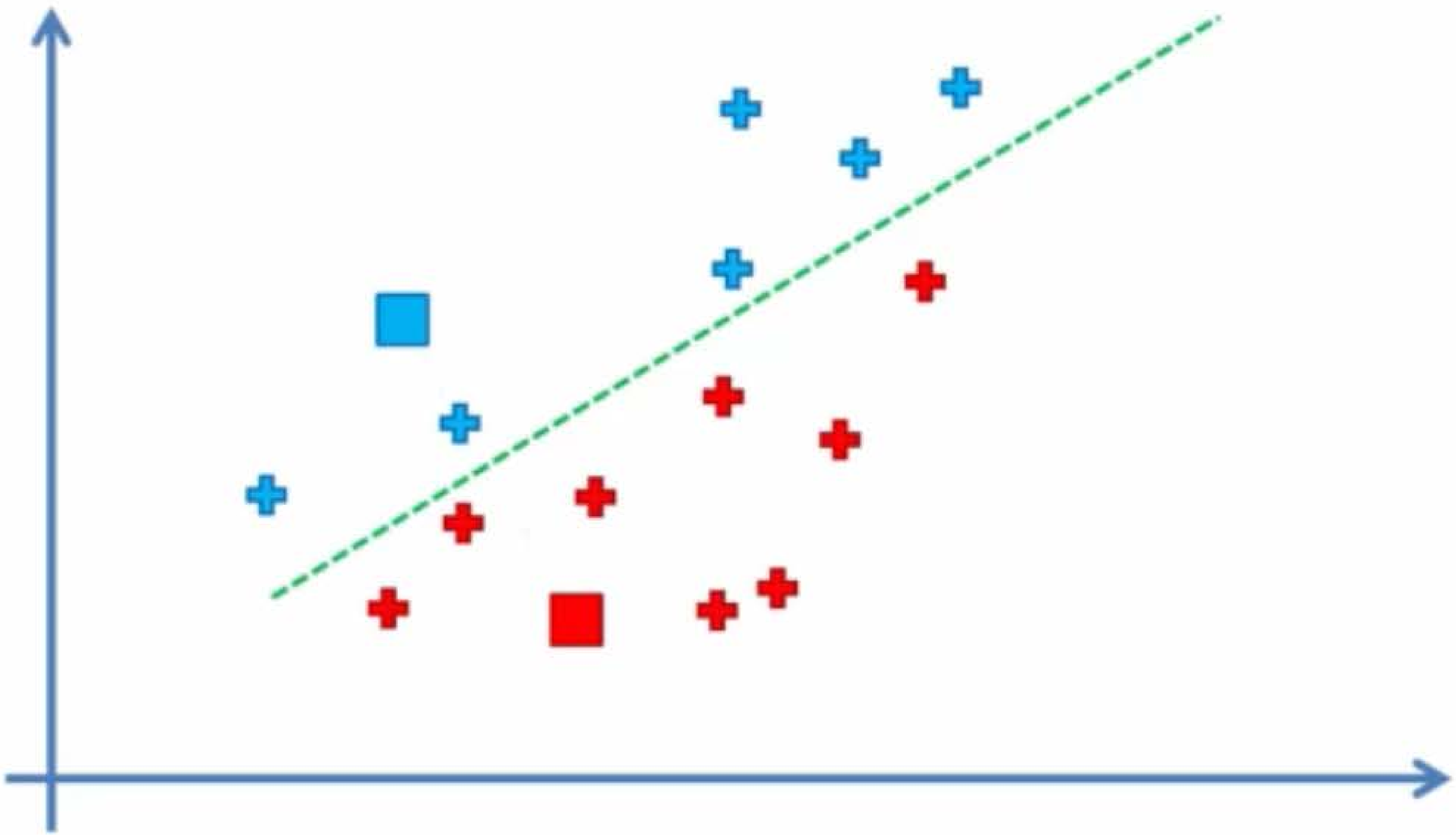
K-Means algorithm

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



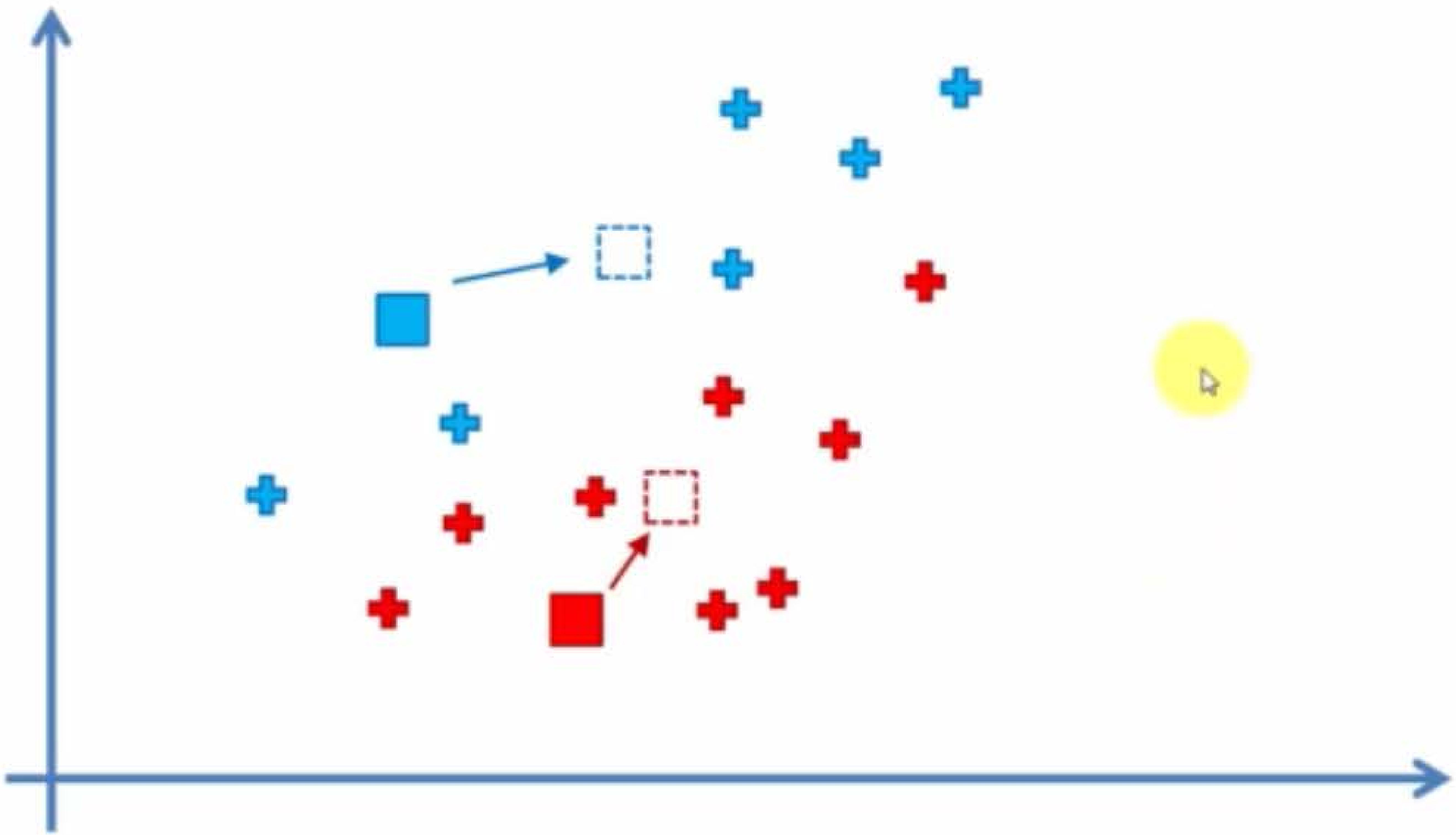
K-Means algorithm

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



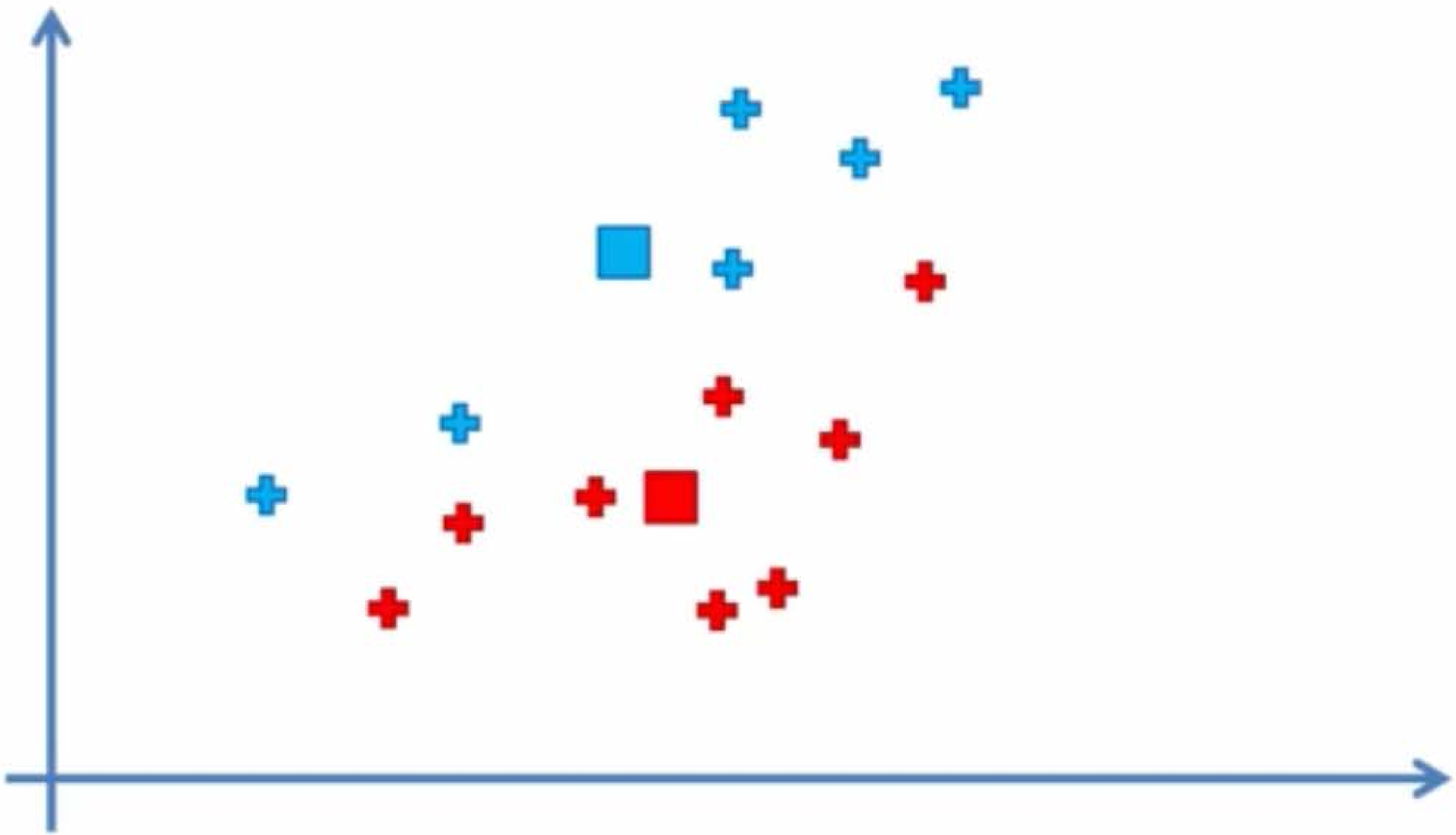
K-Means algorithm

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



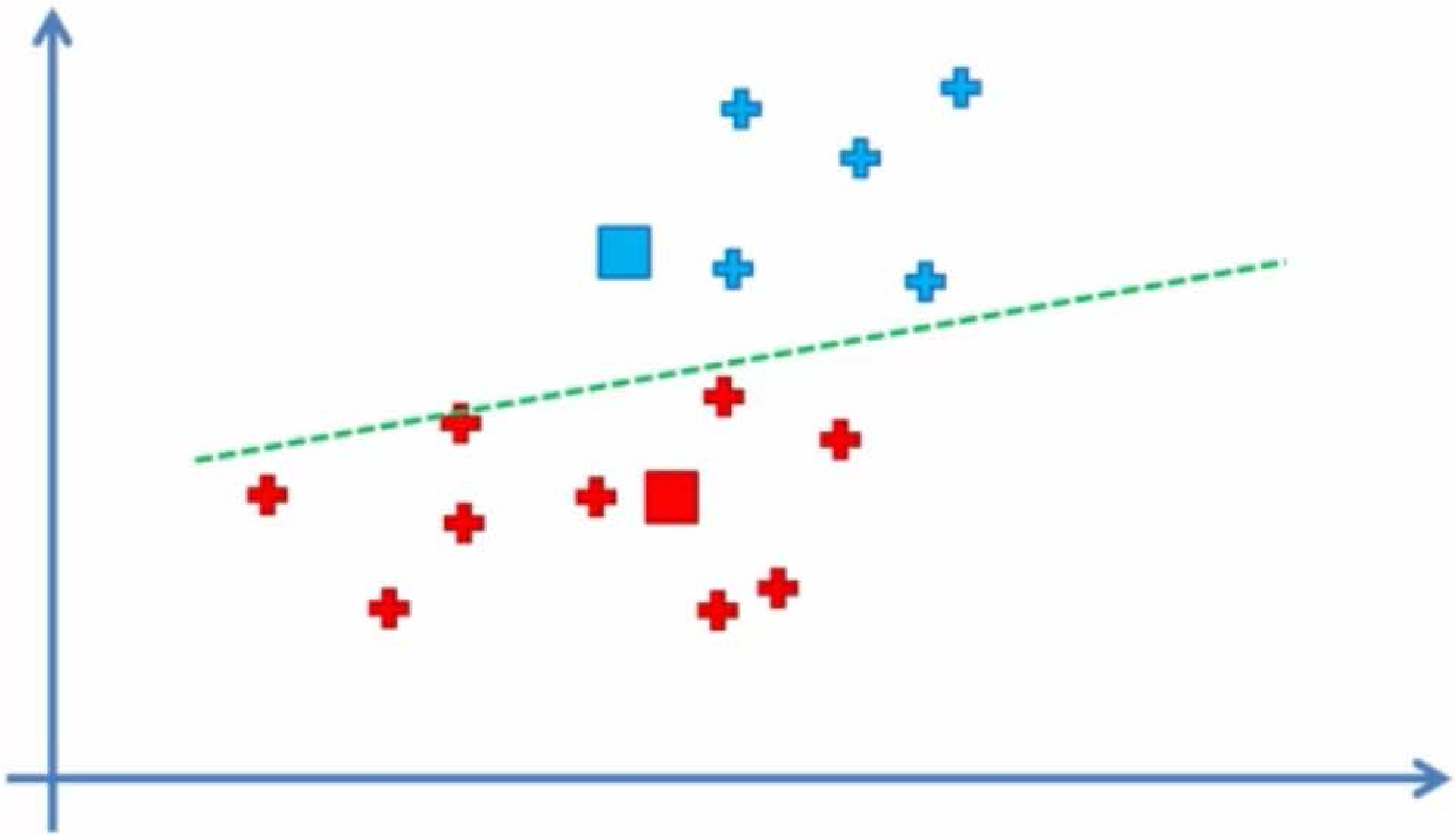
K-Means algorithm

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



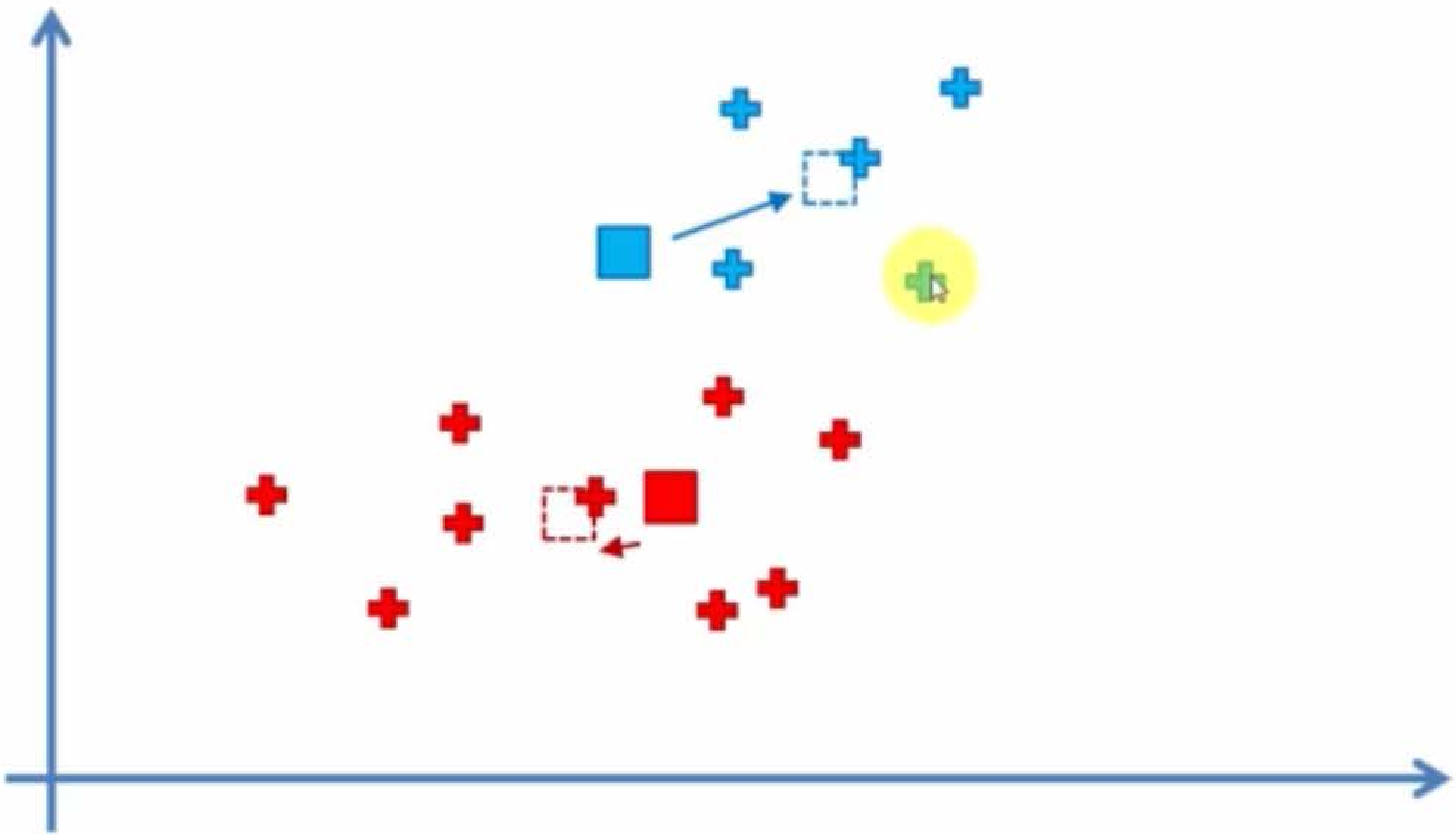
K-Means algorithm

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



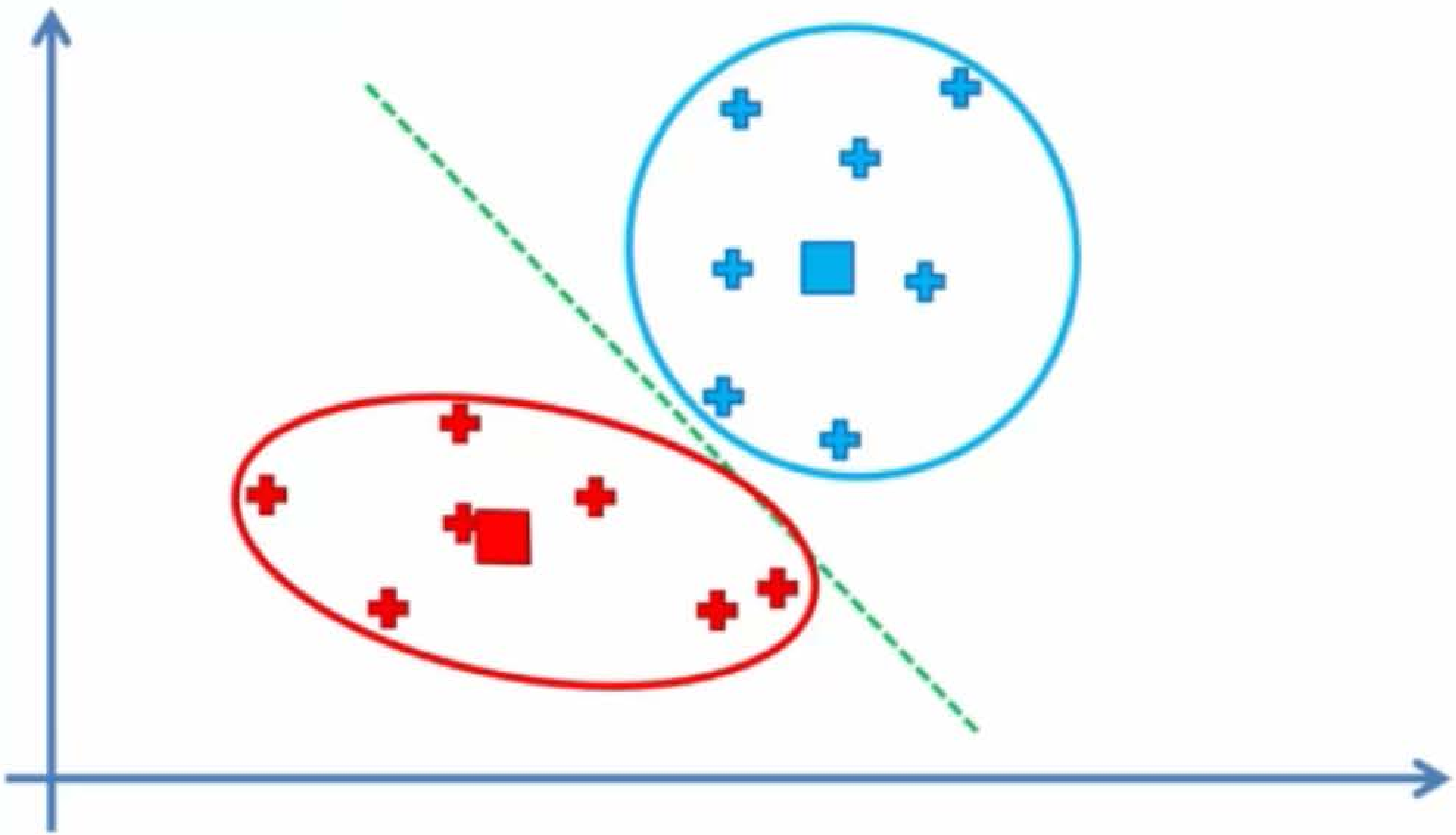
K-Means algorithm

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



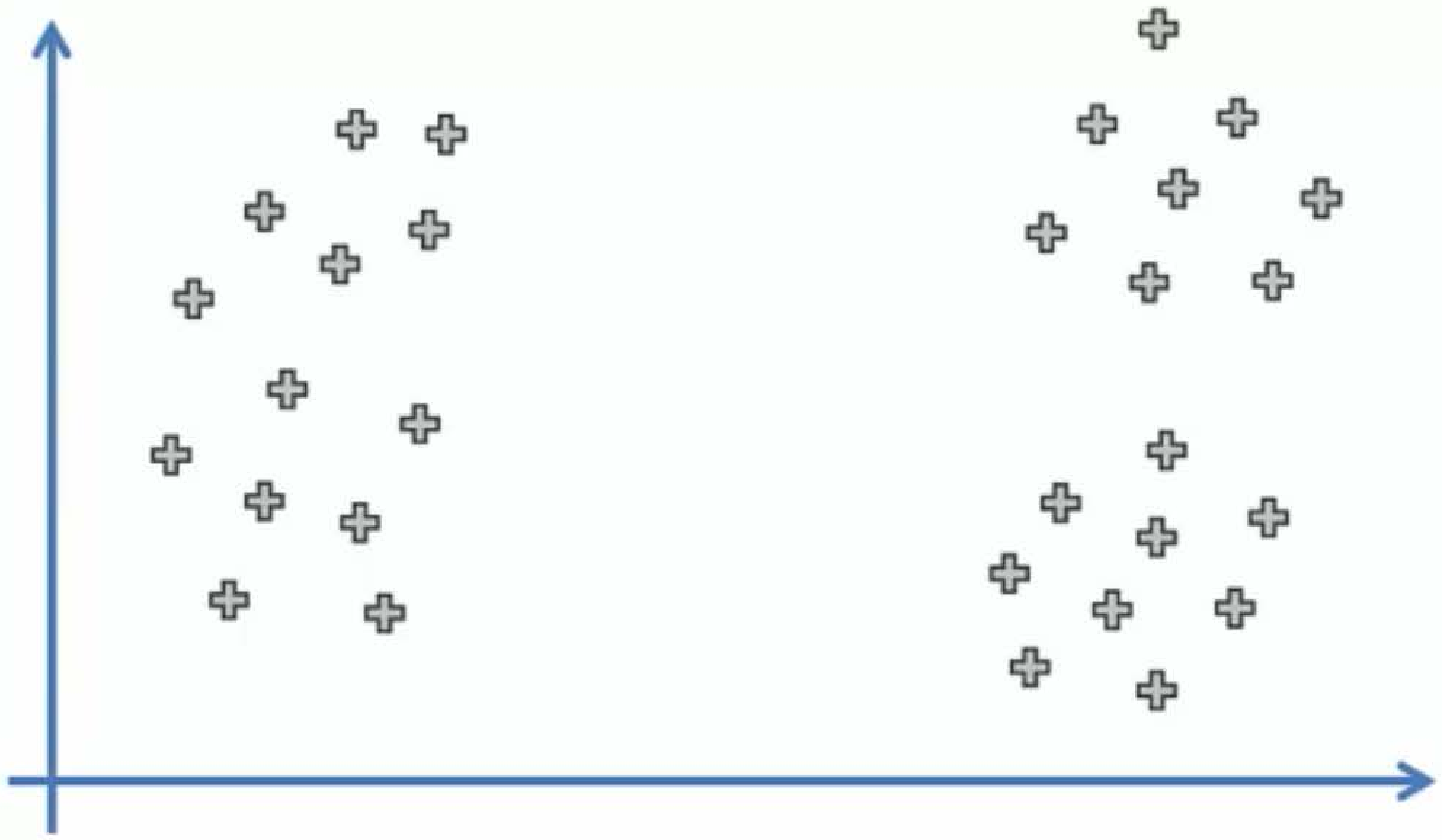
K-Means algorithm

FIN: Your Model Is Ready



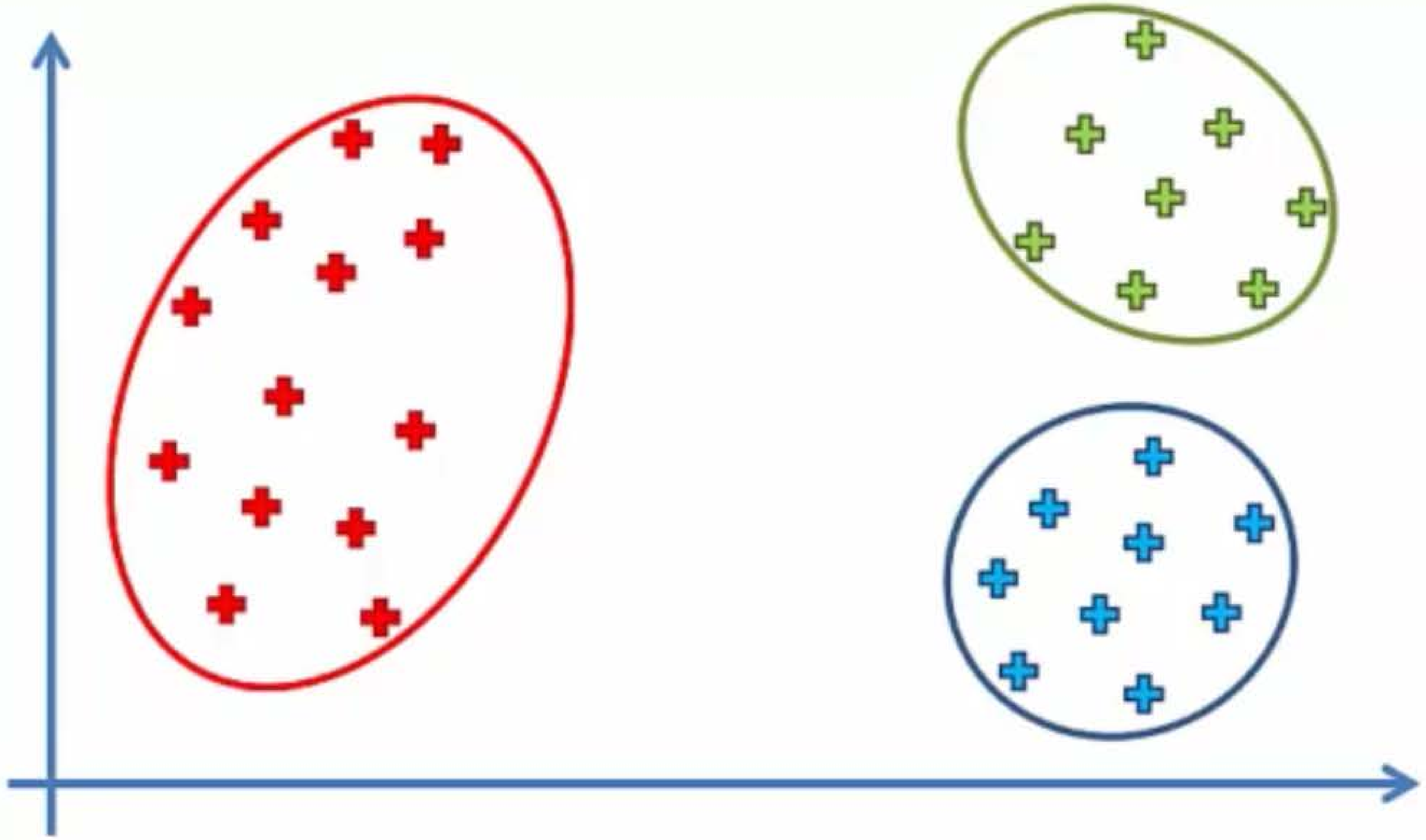
K-Means Intuition: Random Initialization Trap

Random Initialization Trap



If we choose K = 3 clusters...

Random Initialization Trap



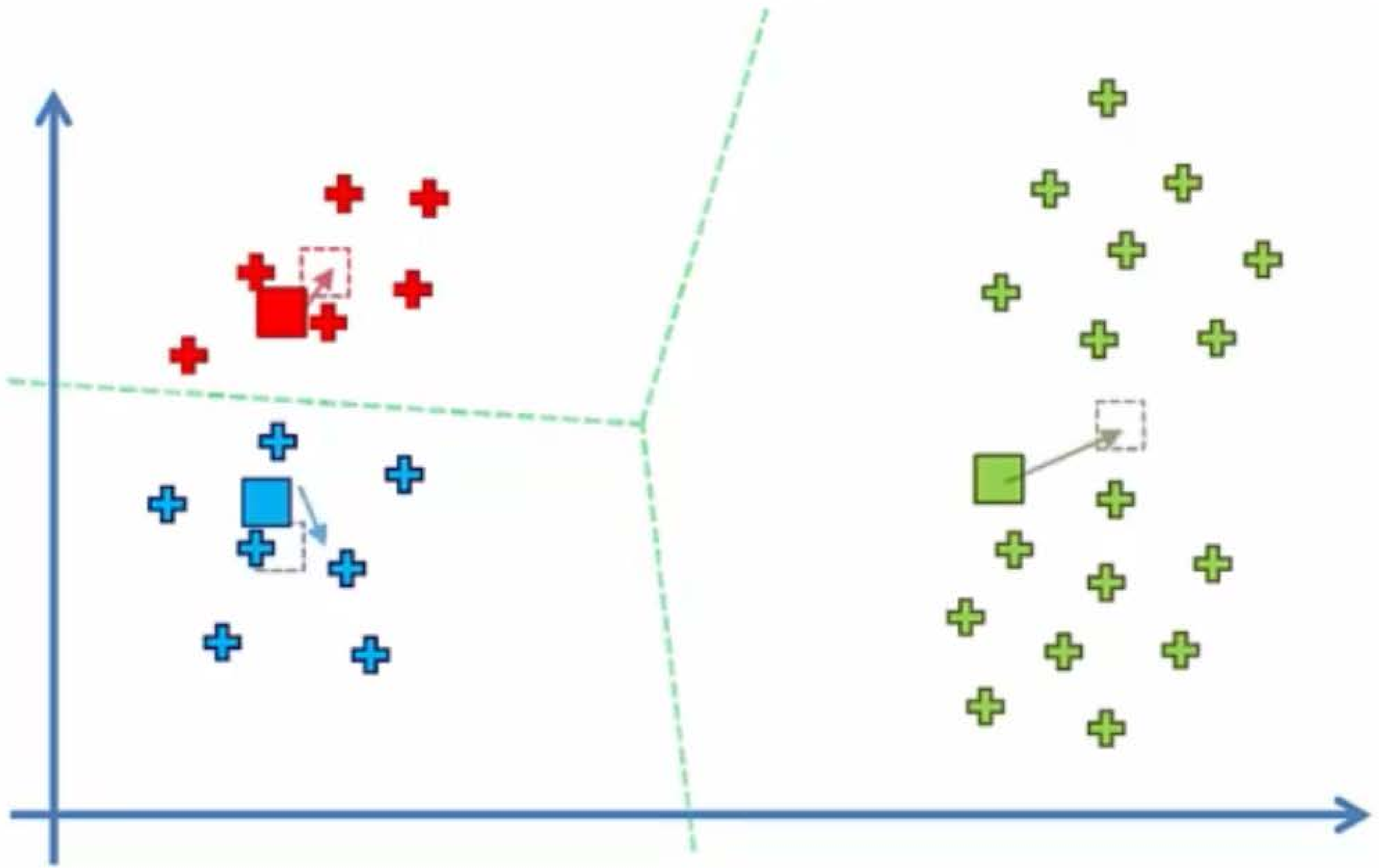
...the following three clusters

Random Initialization Trap

But what would happen if we had a bad random initialisation ?

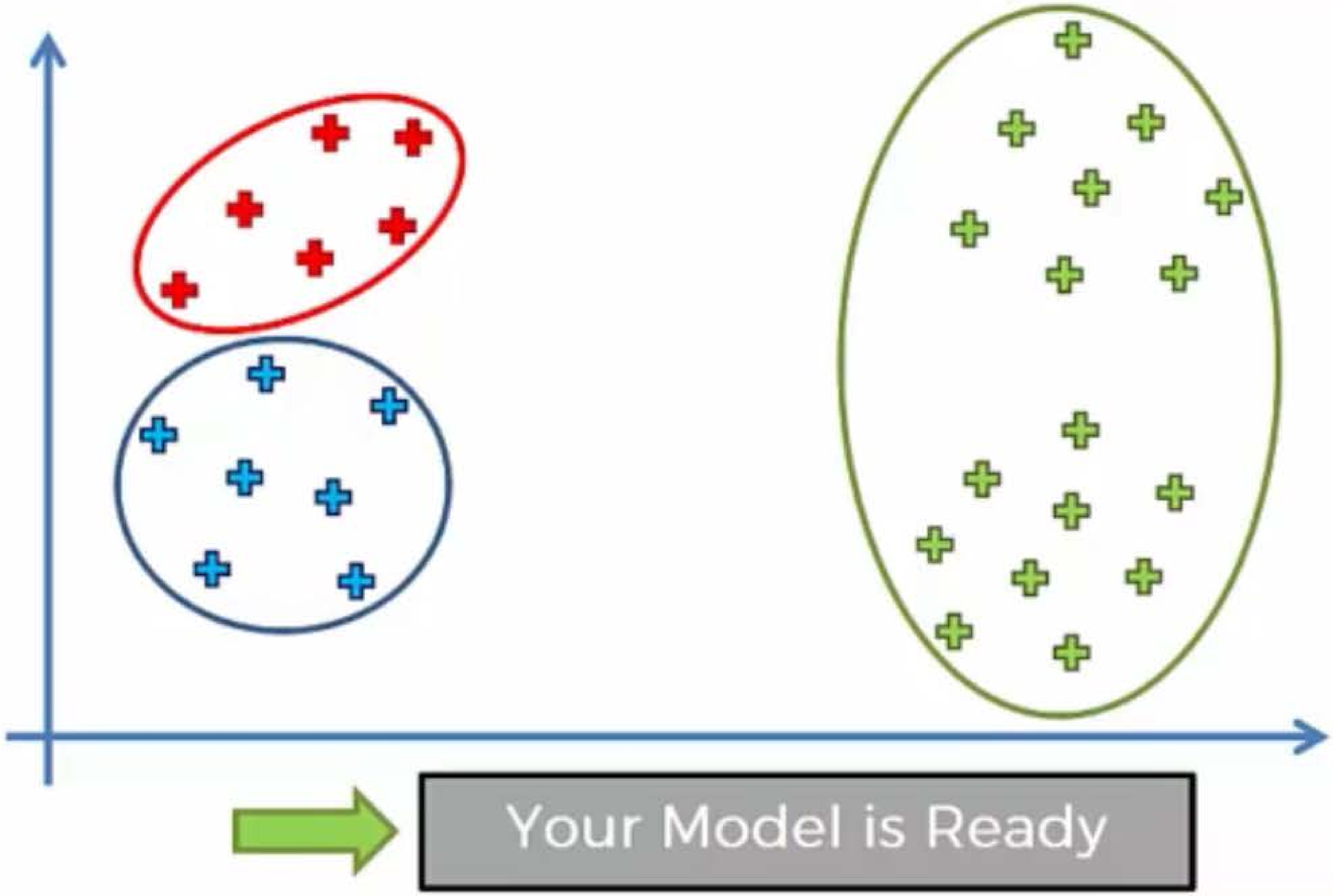
Random Initialization Trap

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



Random Initialization Trap

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

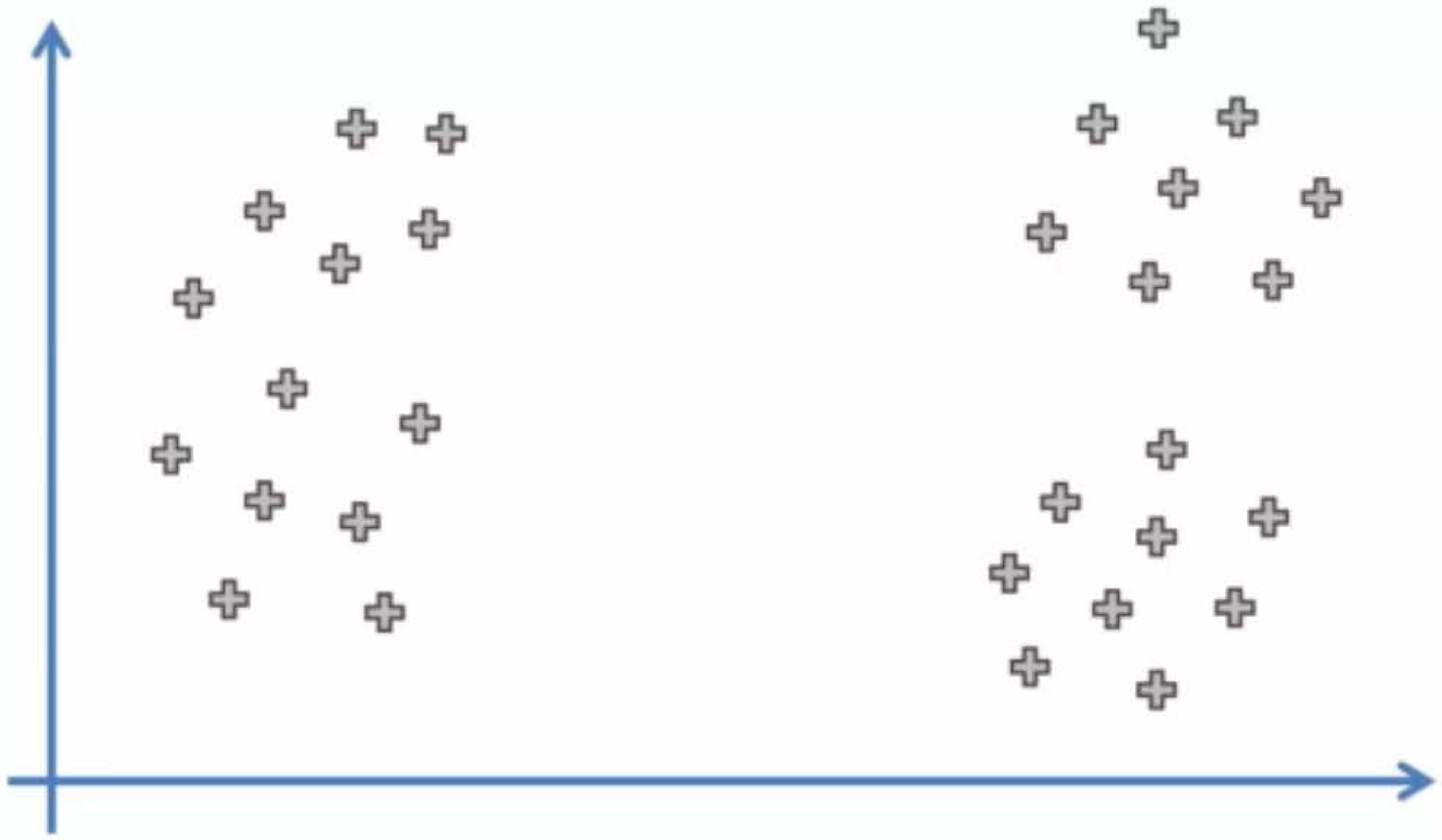


Random Initialization Trap

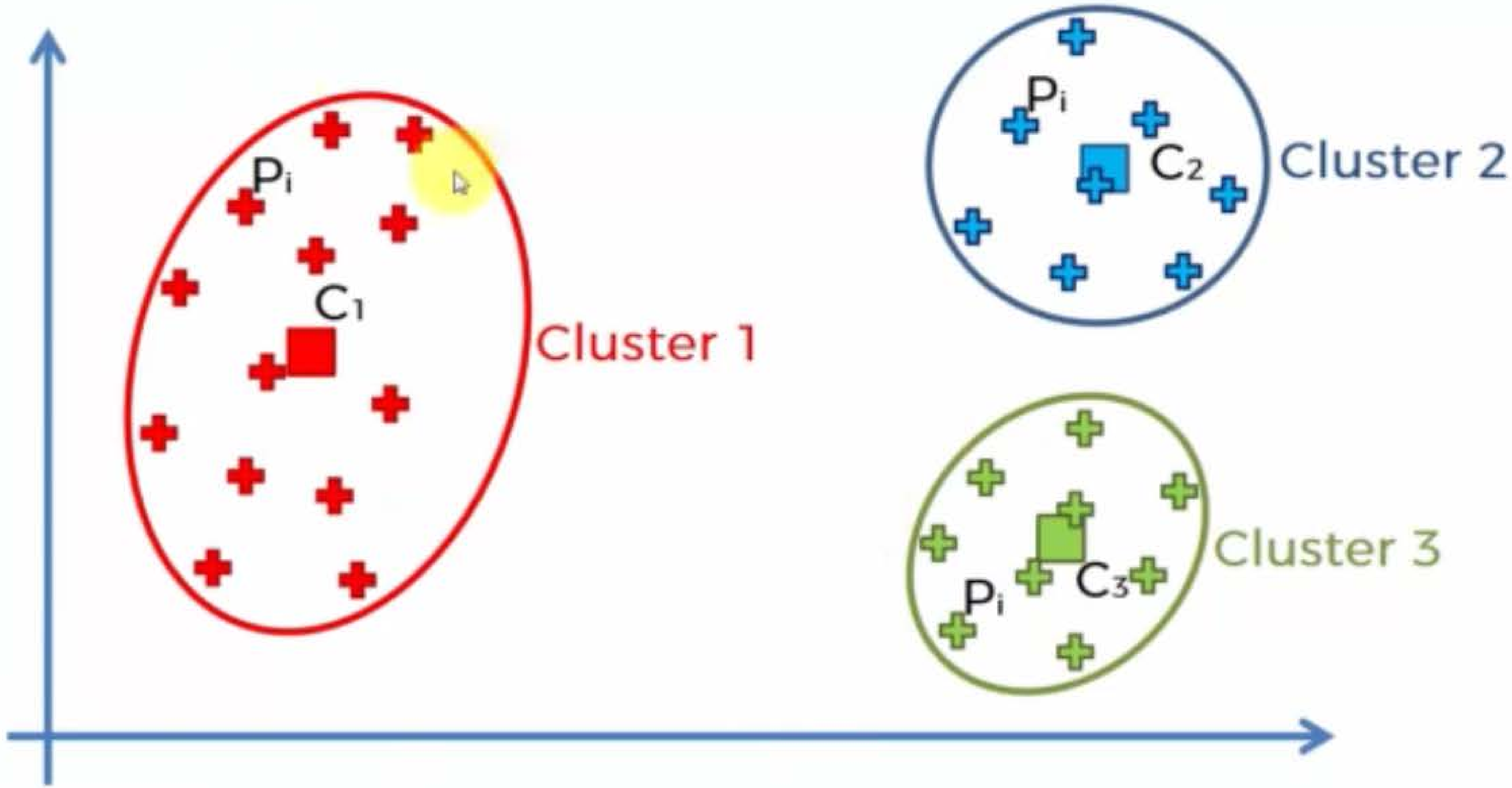
Solution → K-Means++

K-Means Intuition: Choosing the right number of clusters

Choosing the right number of clusters

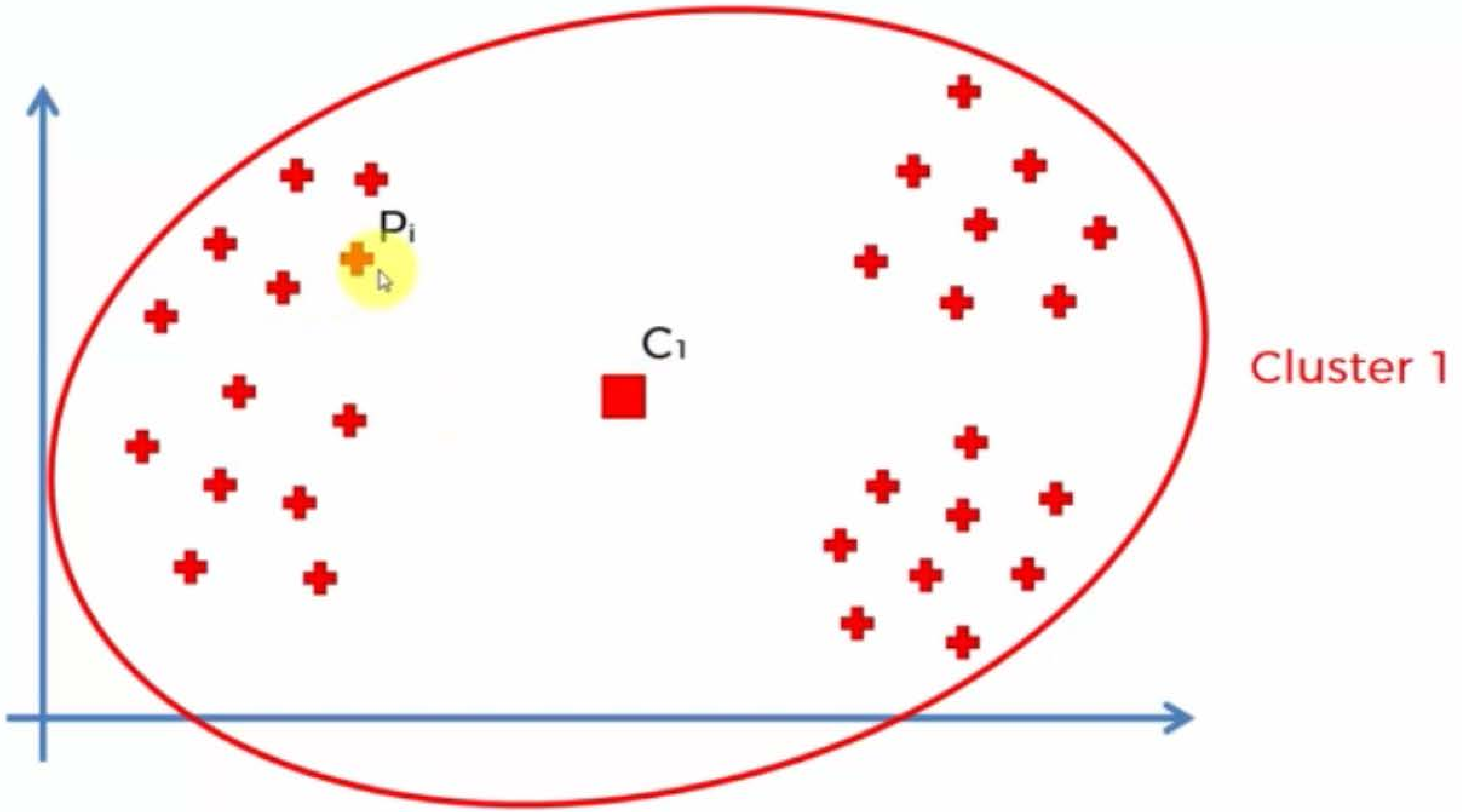


Choosing the right number of clusters



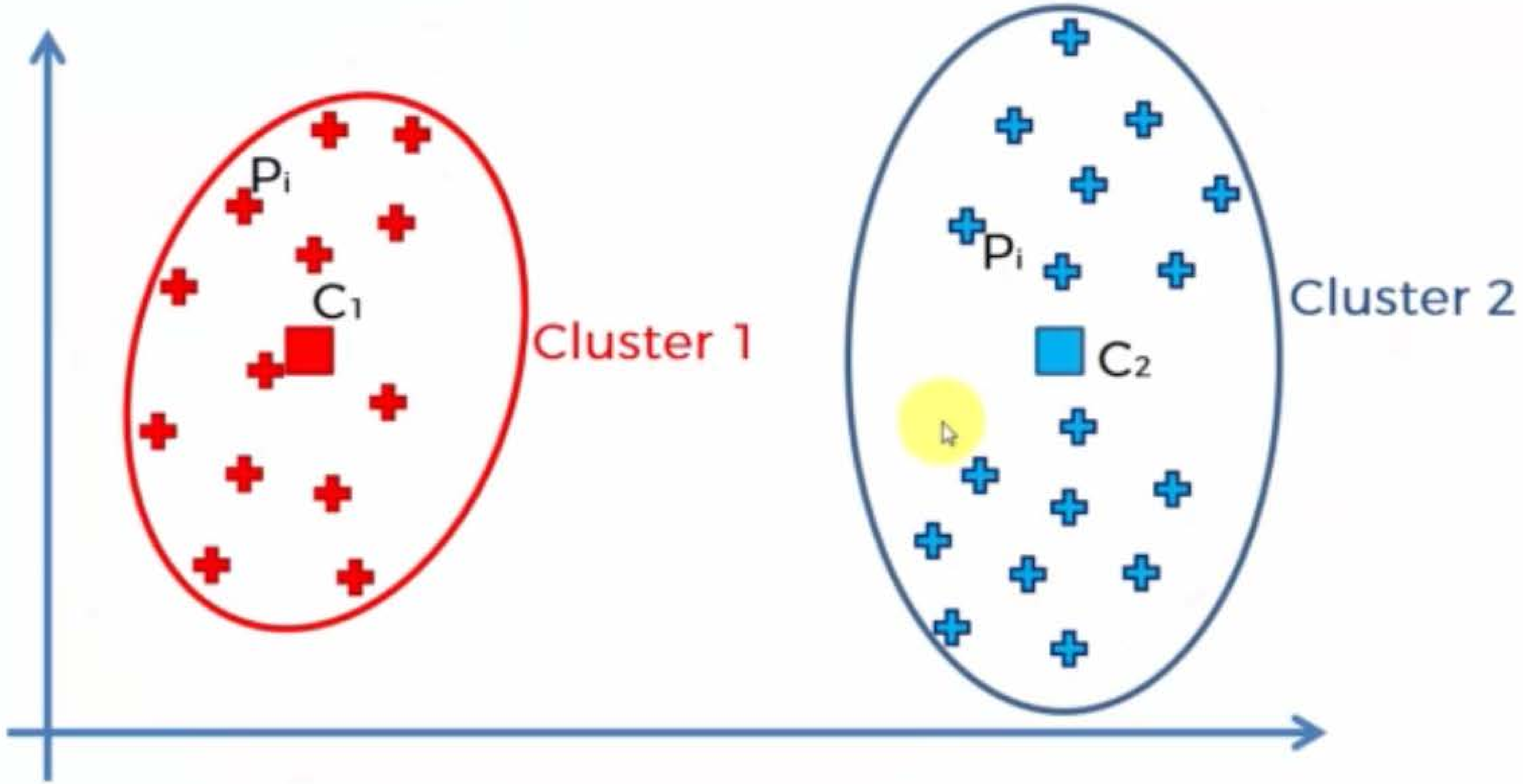
$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

Choosing the right number of clusters



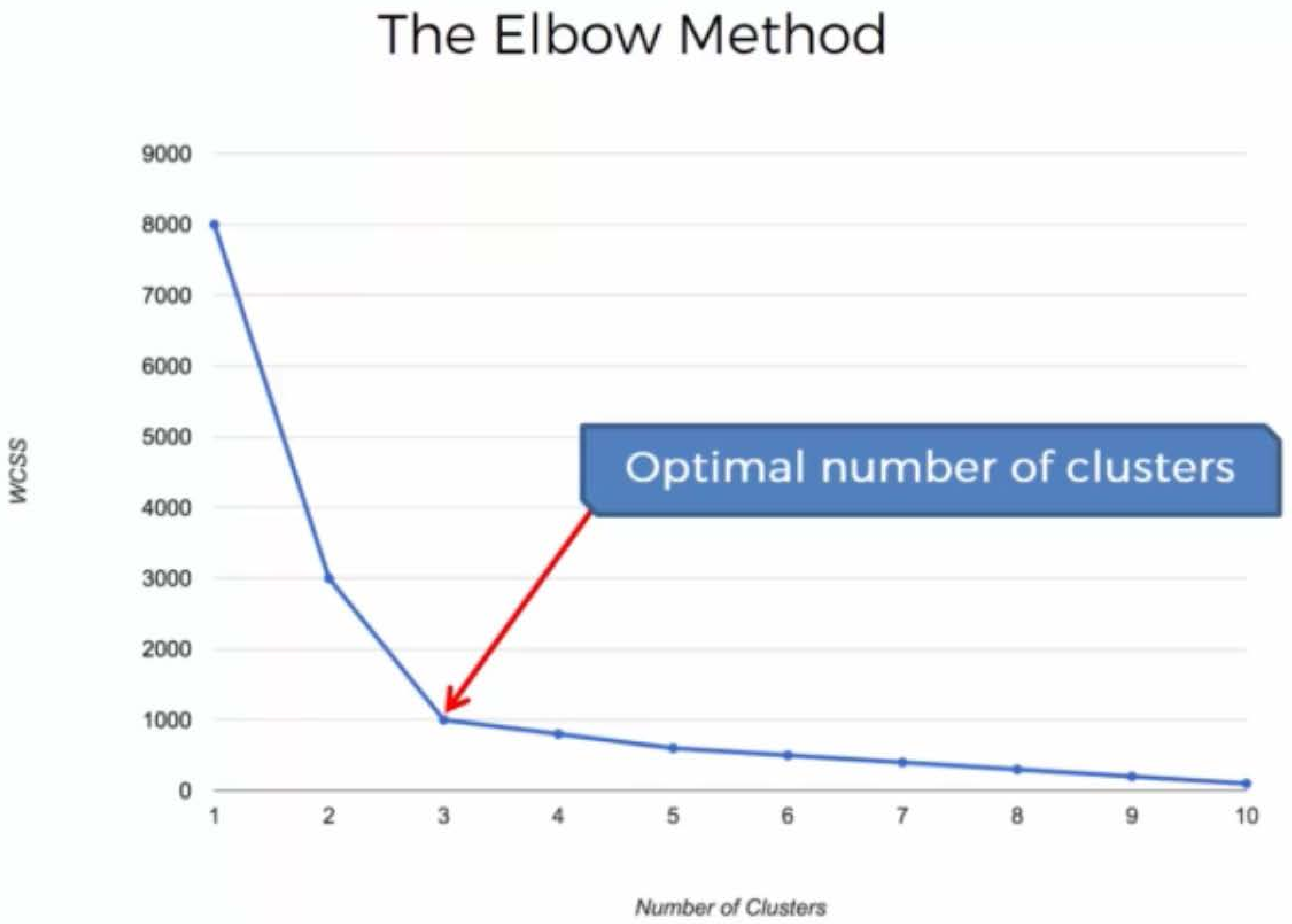
$$\mathrm{WCSS} = \sum_{P_i \text{ in Cluster } 1} \mathrm{distance}(P_i, C_1)^2$$

Choosing the right number of clusters



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

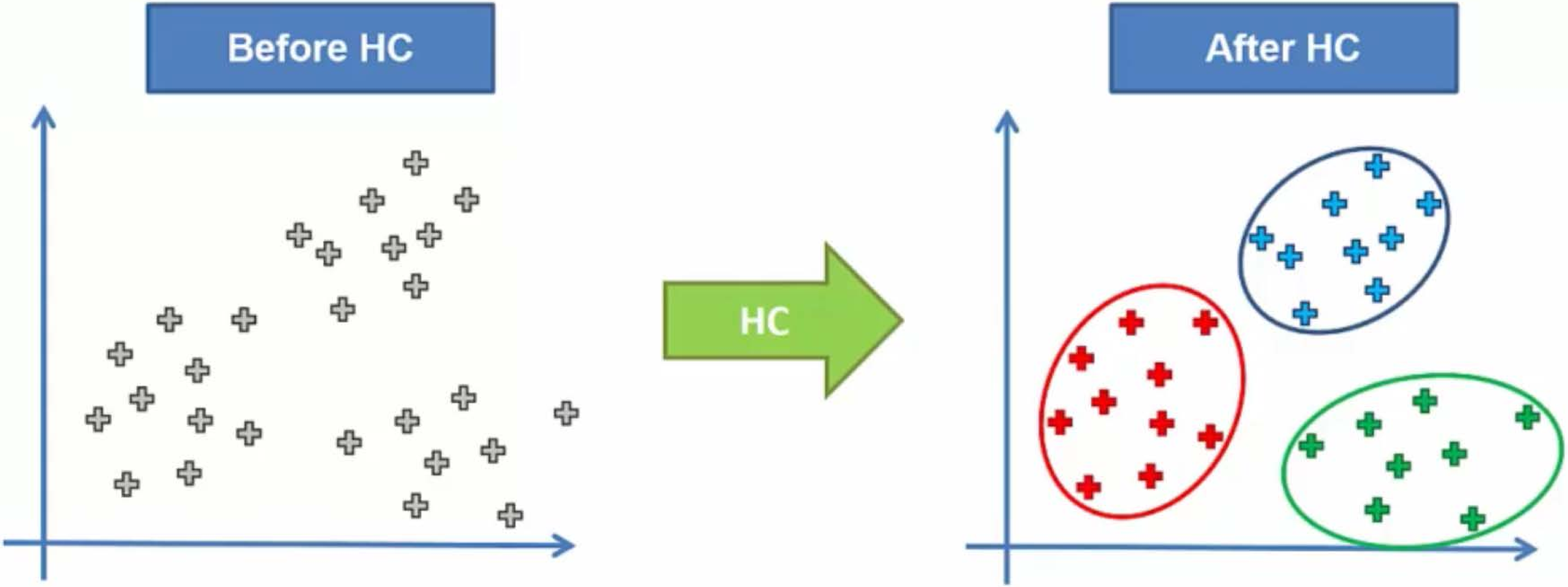
Choosing the right number of clusters



HC Intuition: Understanding HC

Hierarchical Clustering performs better than K-Means on large datasets: False

What HC does for you



Same as K-Means but different process

NOTE:
Agglomerative
&
Divisive

Agglomerative HC

STEP 1: Make each data point a single-point cluster → That forms N clusters

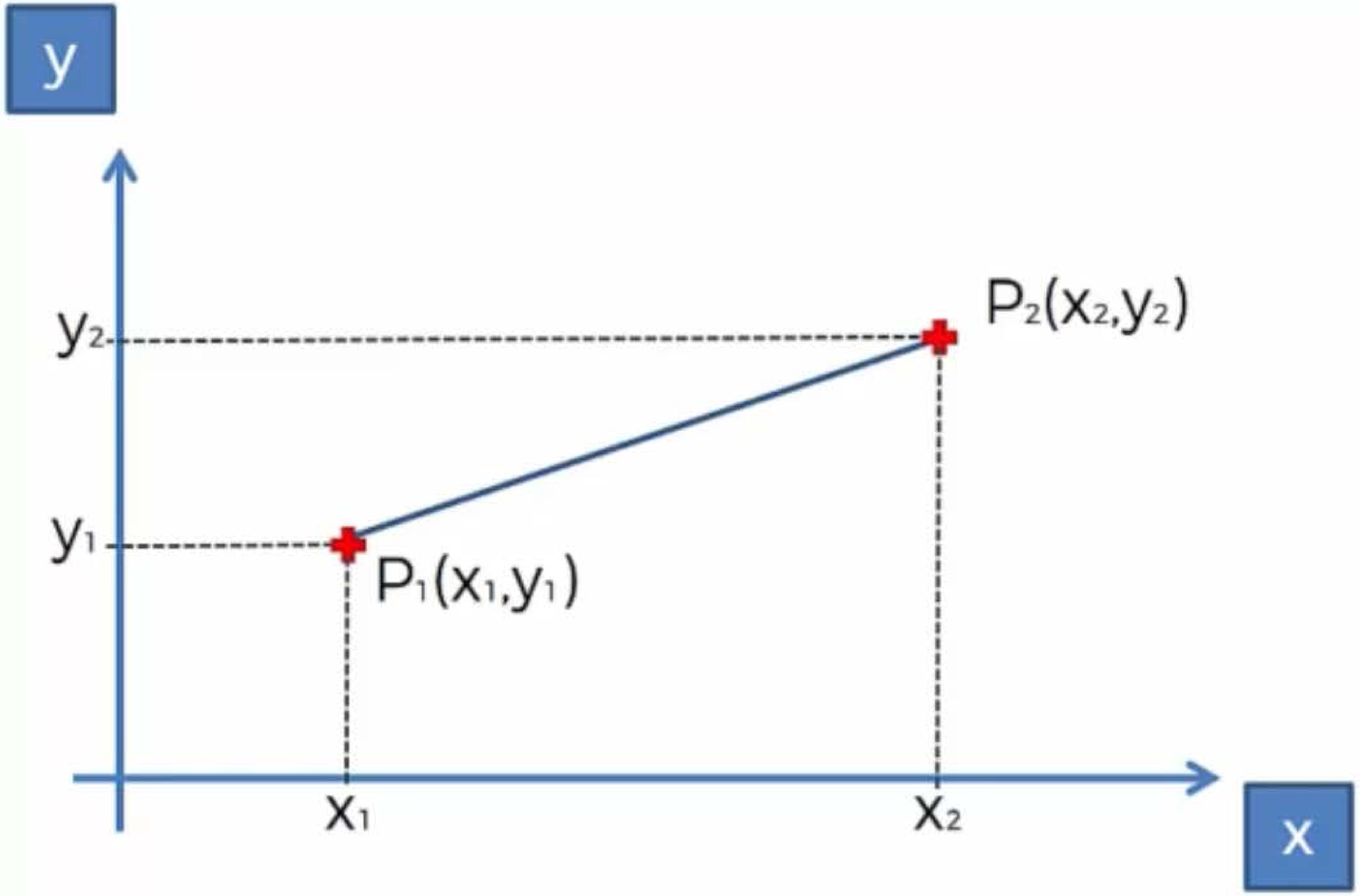
STEP 2: Take the two closest data points and make them one cluster → That forms N-1 clusters

STEP 3: Take the two <u>closest clusters</u> and make them one cluster → That forms N - 2 clusters

STEP 4: Repeat **STEP 3** until there is only one cluster

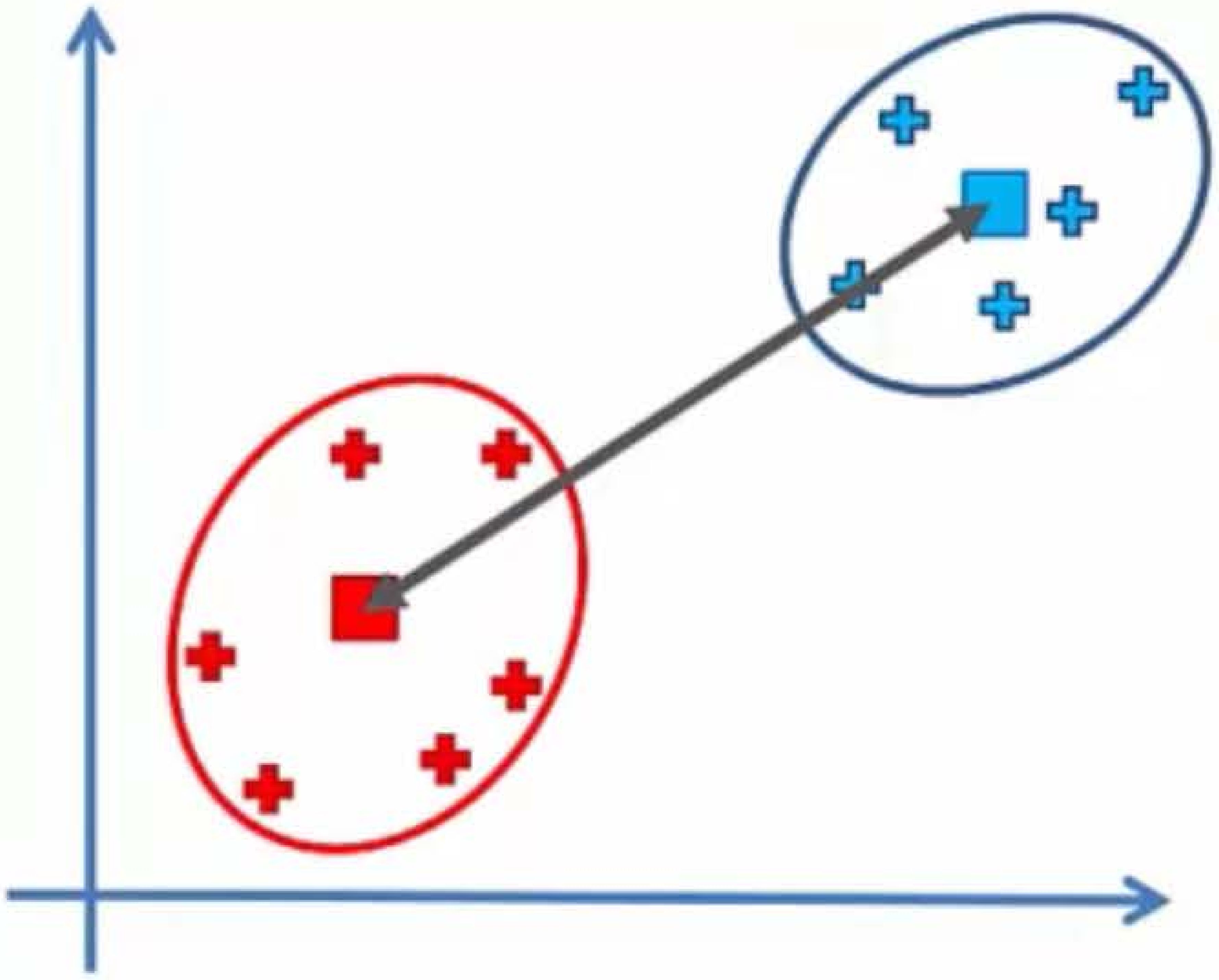
FIN

Euclidean Distance



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Distance Between Clusters

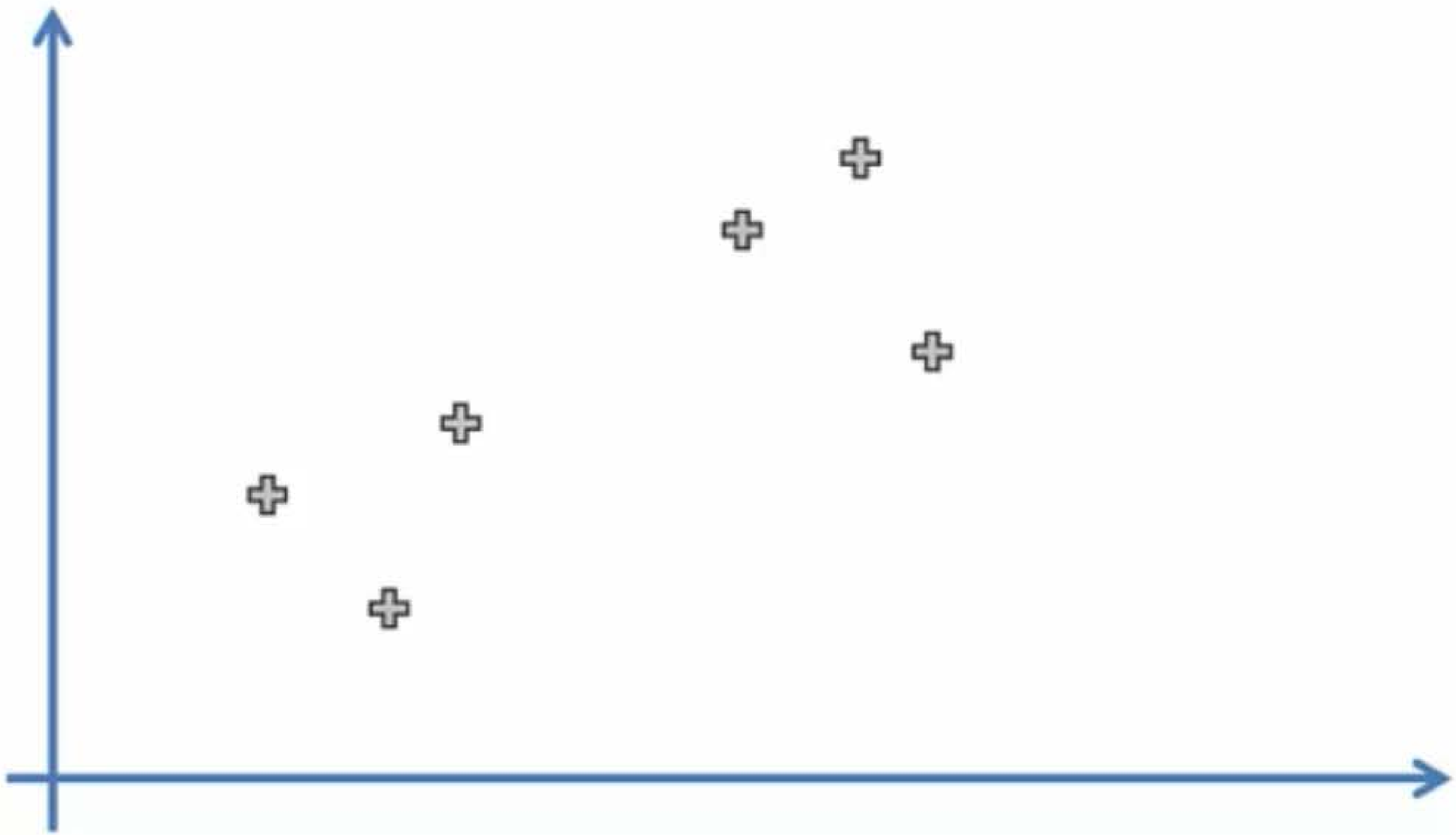


Distance Between Two Clusters:

- Option 1: Closest Points
- Option 2: Furthest Points
- Option 3: Average Distance
- Option 4: Distance Between Centroids

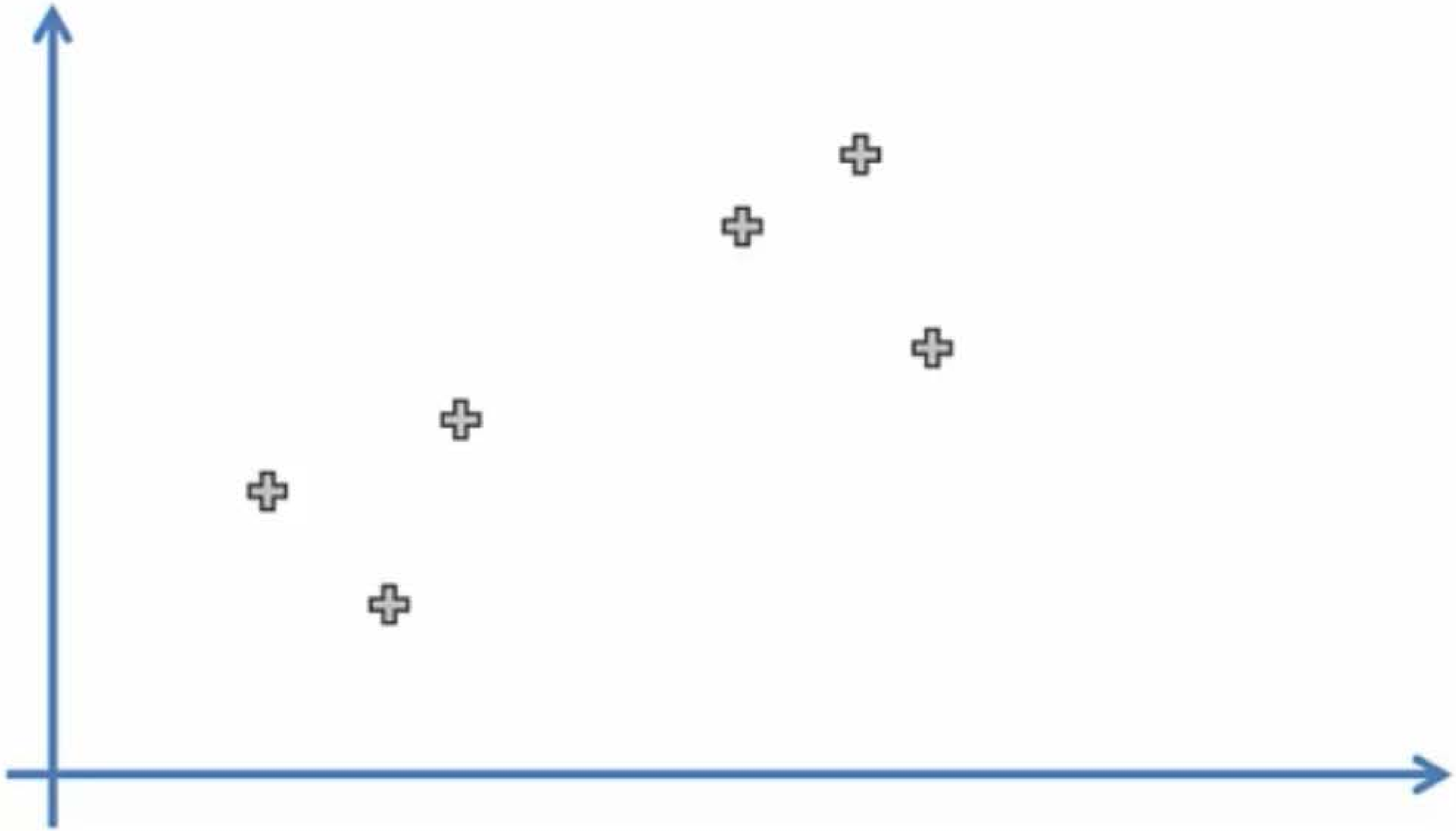
Agglomerative HC

Consider the following dataset of N = 6 data points



Agglomerative HC

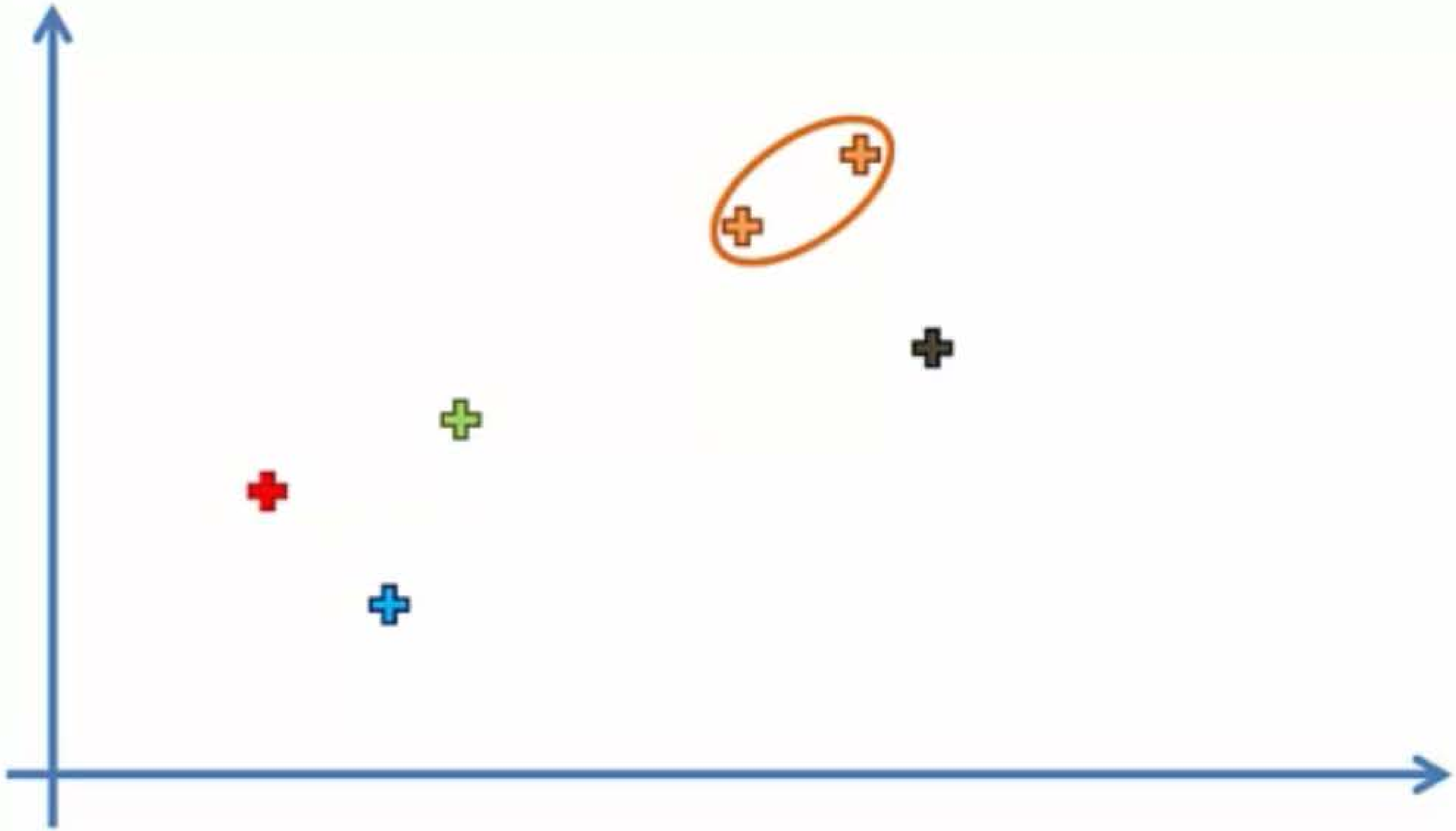
STEP 1: Make each data point a single-point cluster ➡ That forms 6 clusters



Agglomerative HC

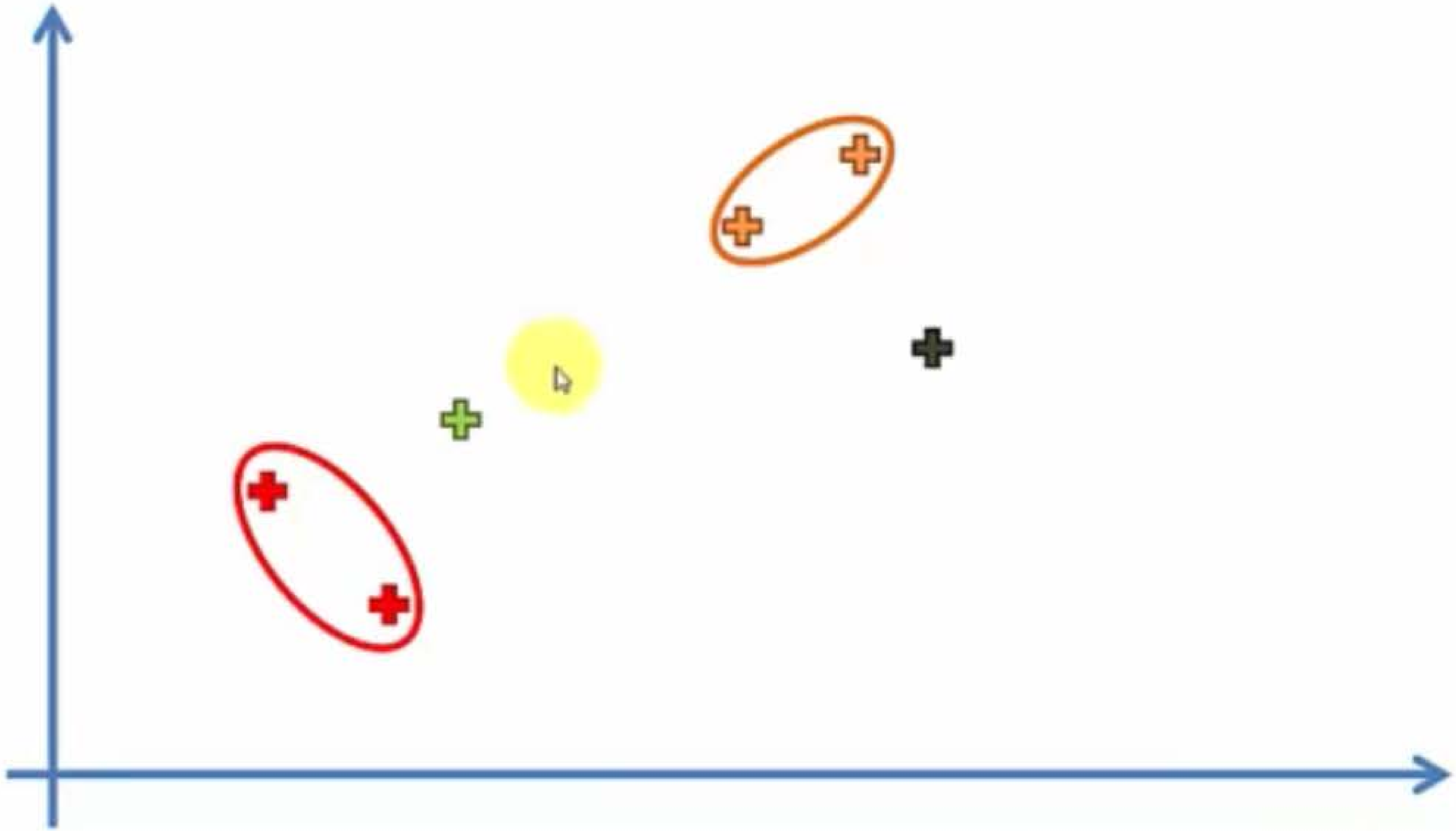
STEP 2: Take the two closest data points and make them one cluster

➡ That forms 5 clusters



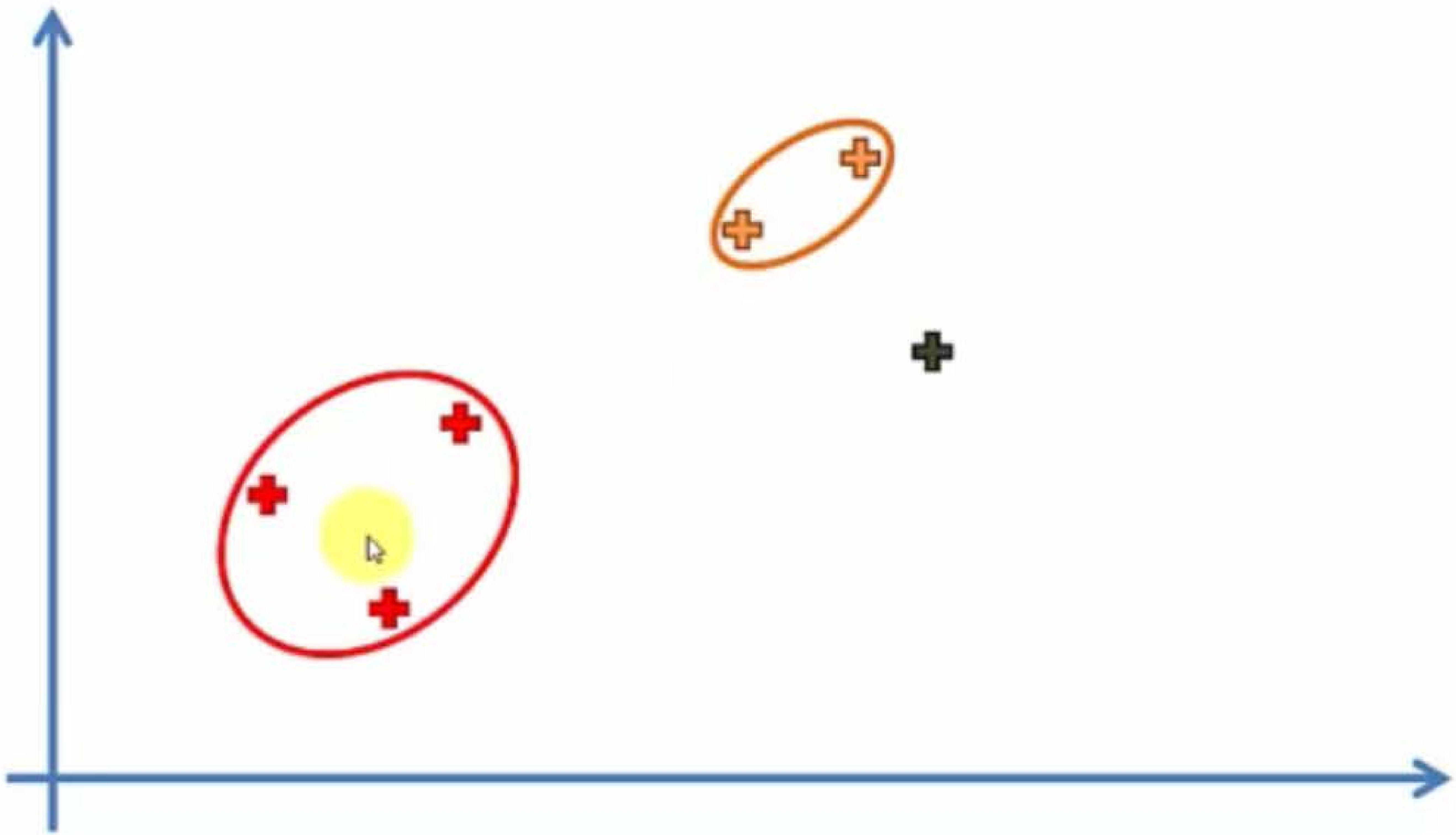
Agglomerative HC

STEP 3: Take the two closest clusters and make them one cluster
➡ That forms 4 clusters



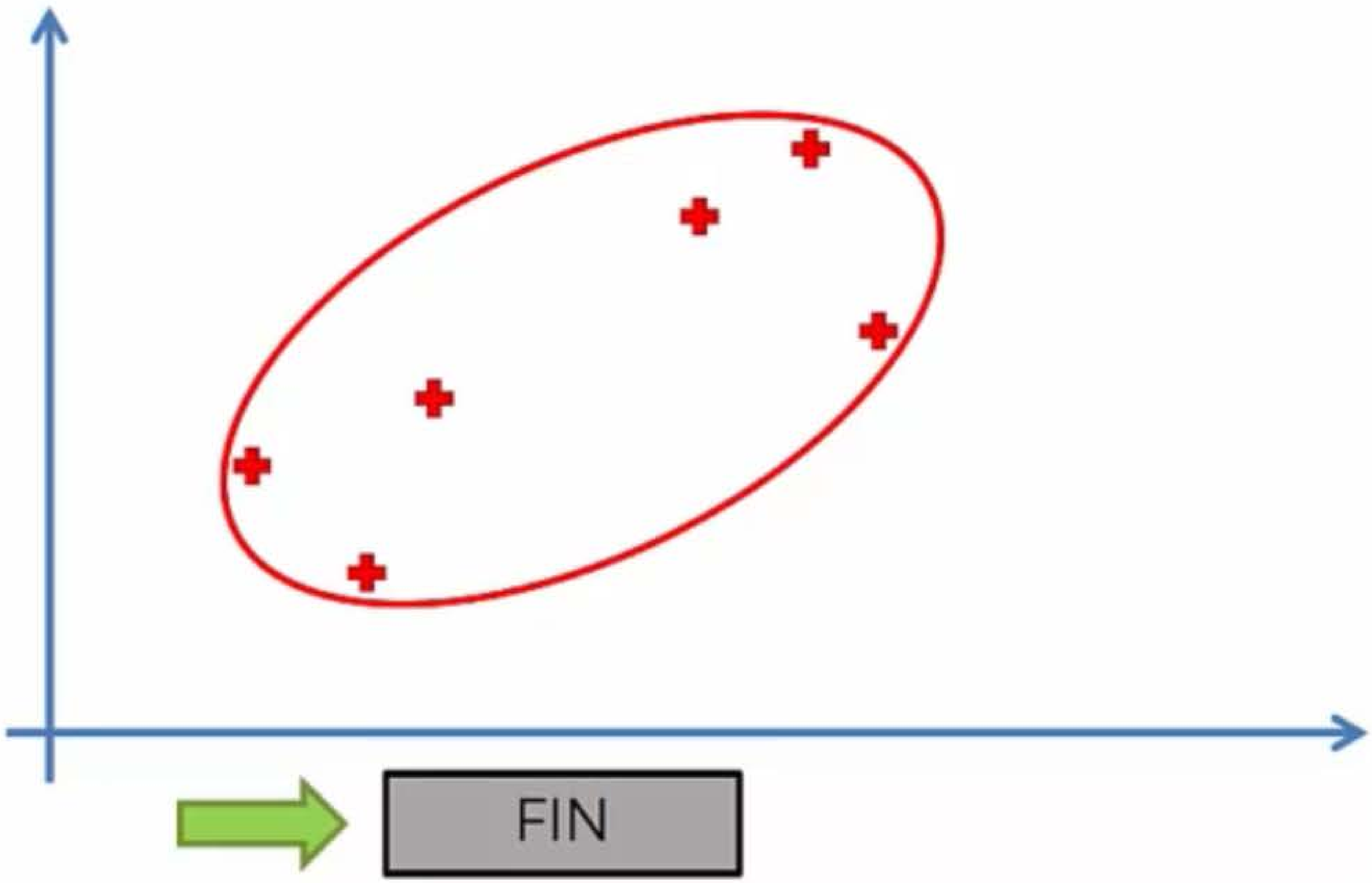
Agglomerative HC

STEP 4: Repeat STEP 3 until there is only one cluster



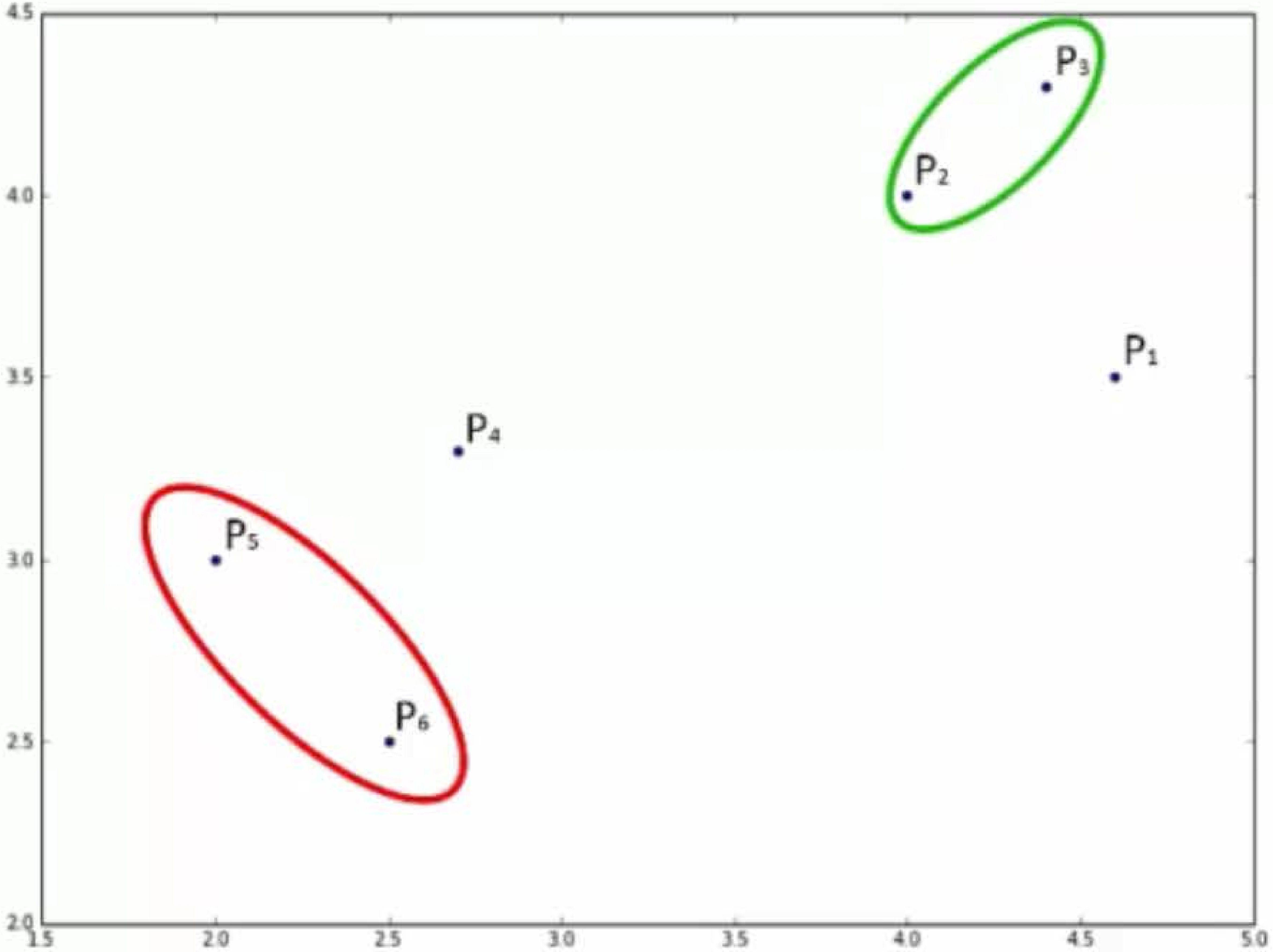
Agglomerative HC

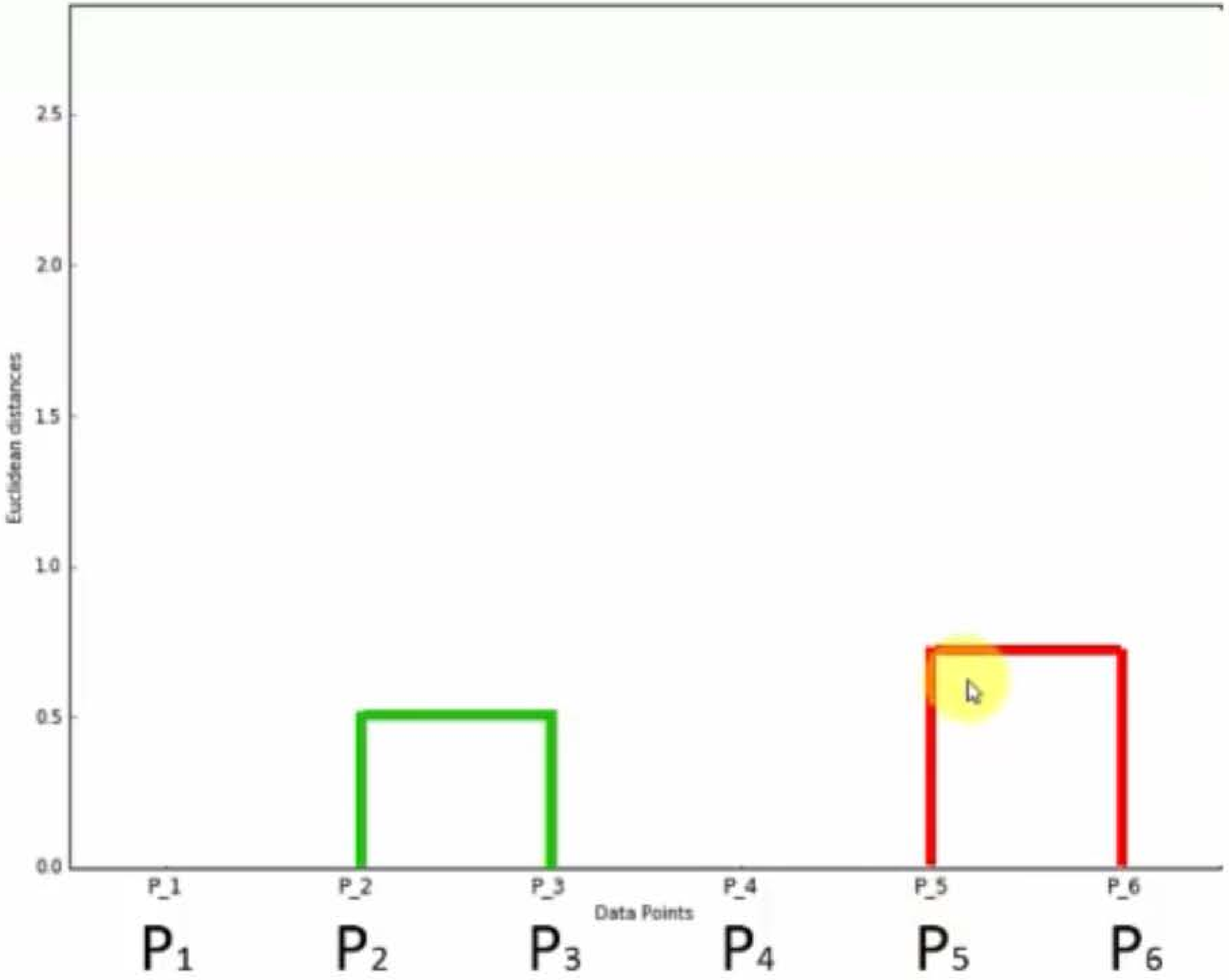
STEP 4: Repeat STEP 3 until there is only one cluster



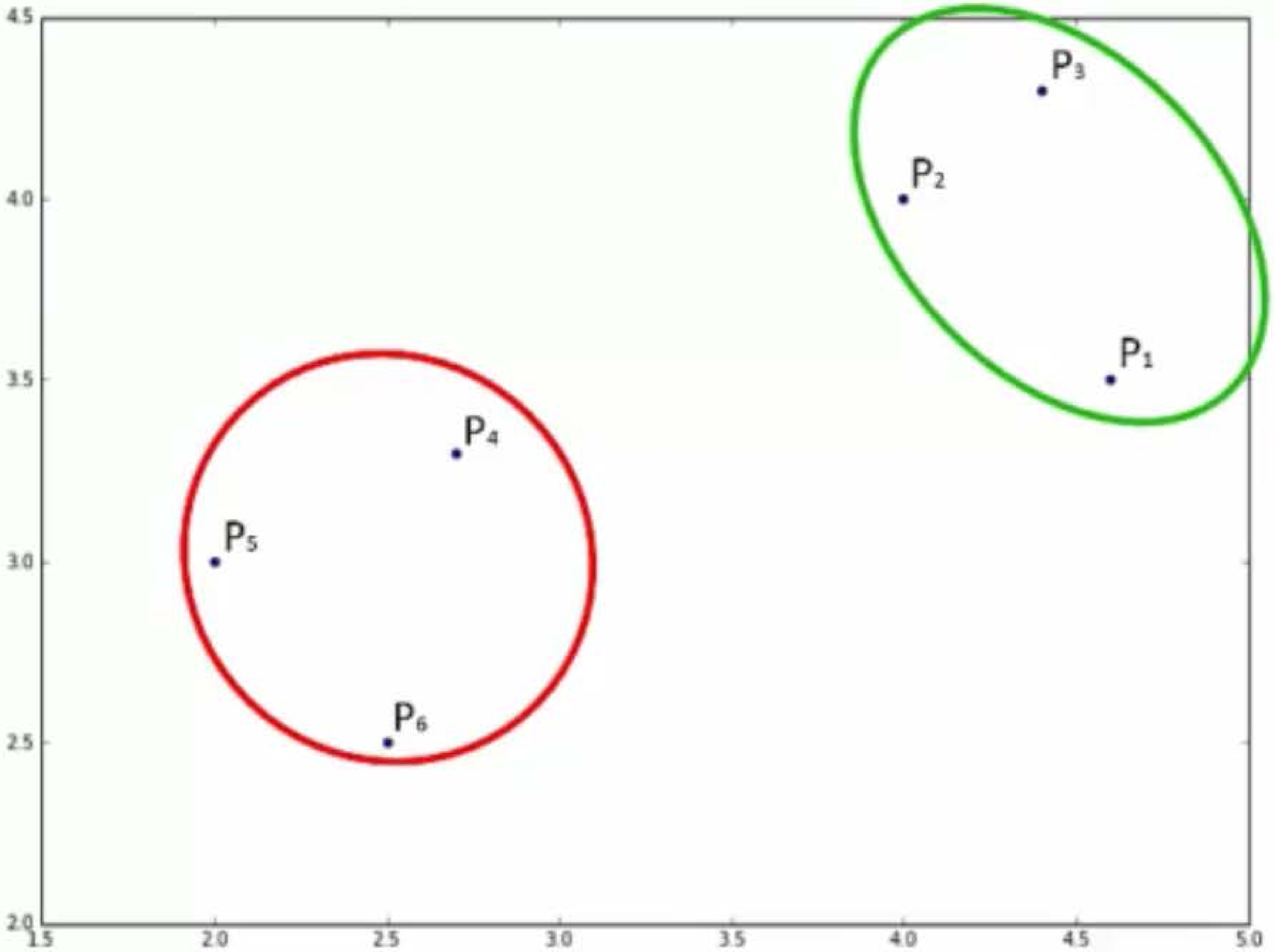
HC Intuition:
How Do Dendograms Work?

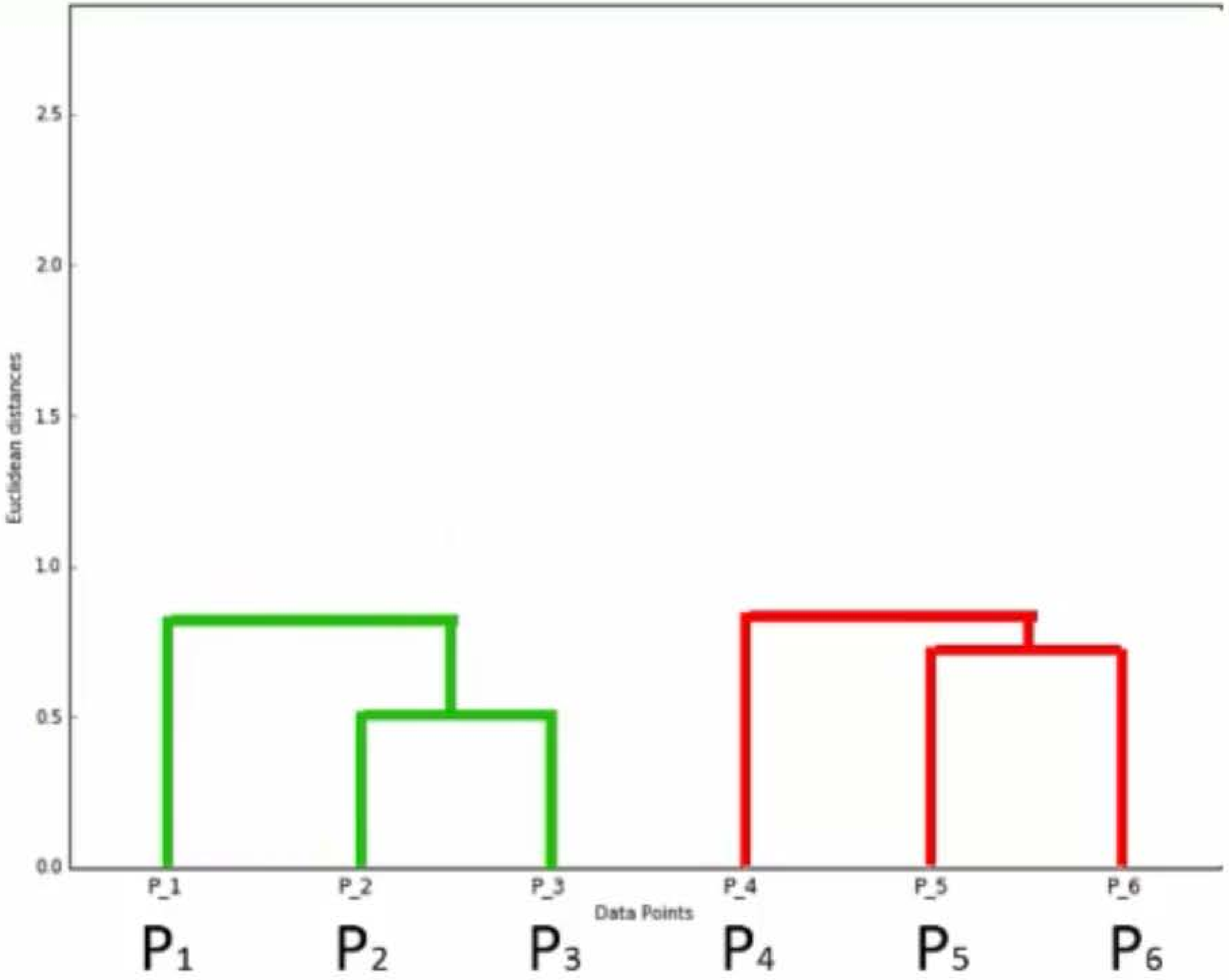
How Do Dendograms Work?





How Do Dendograms Work?

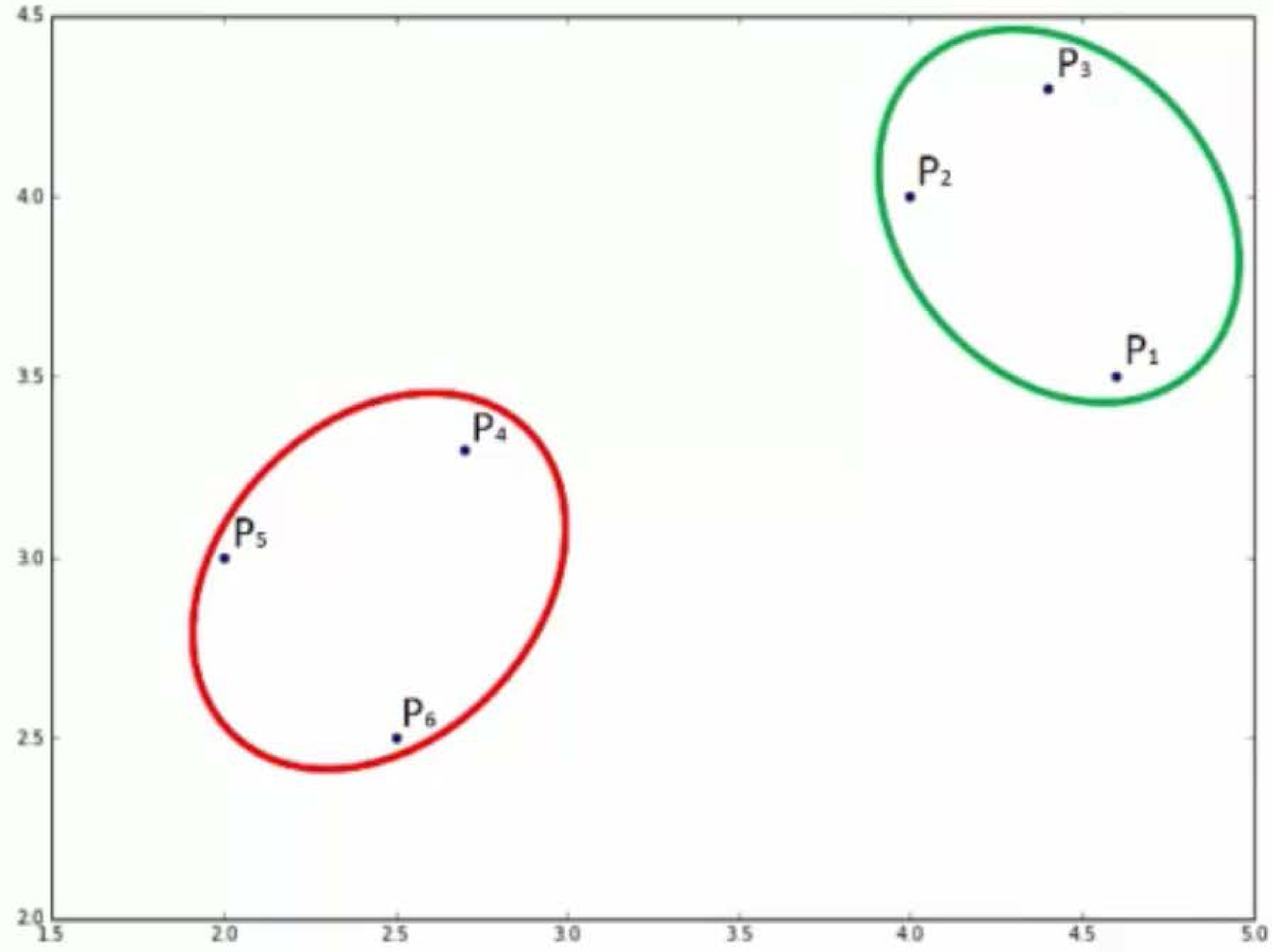


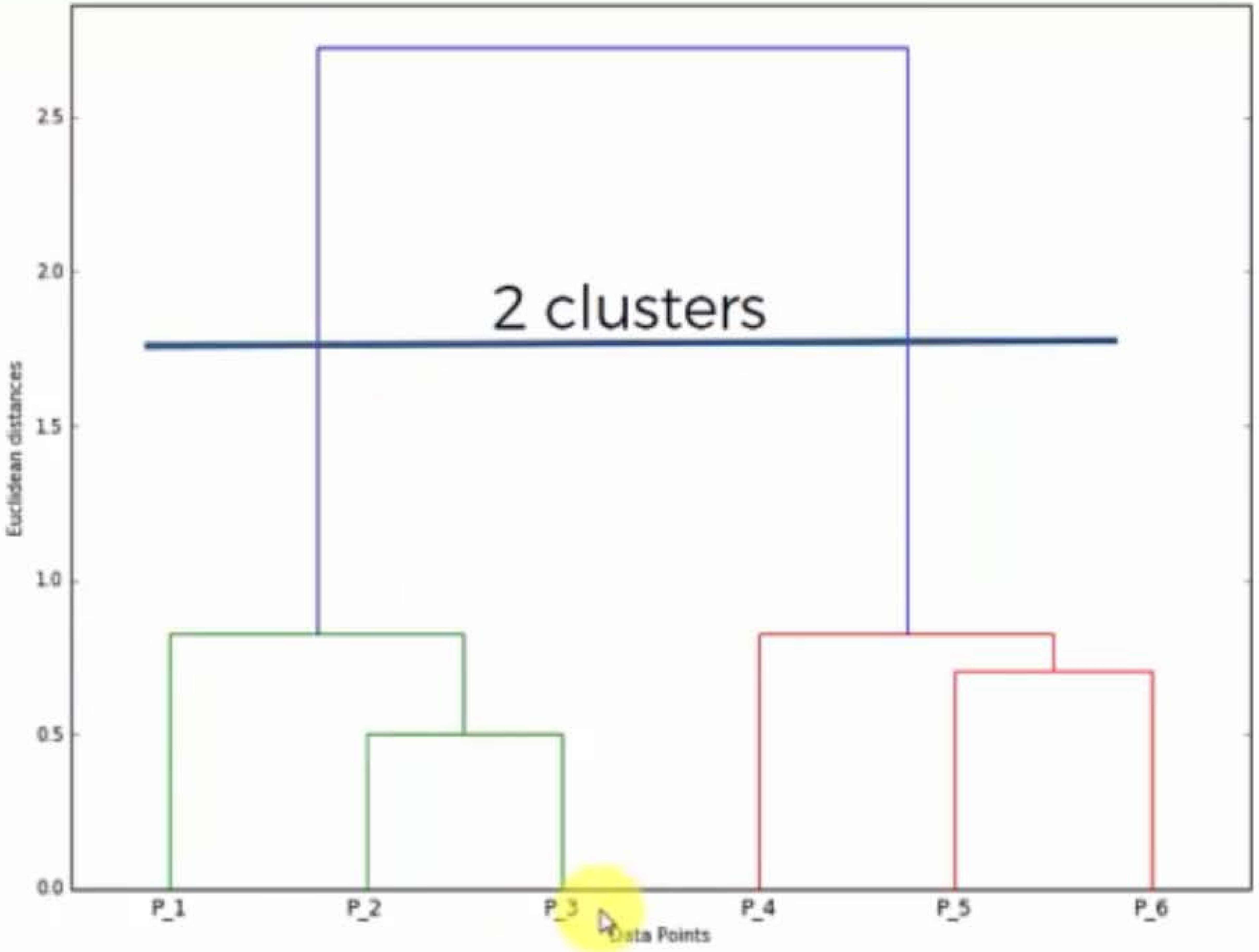


How Do Dendograms Work?

HC Intuition: Using Dendrograms

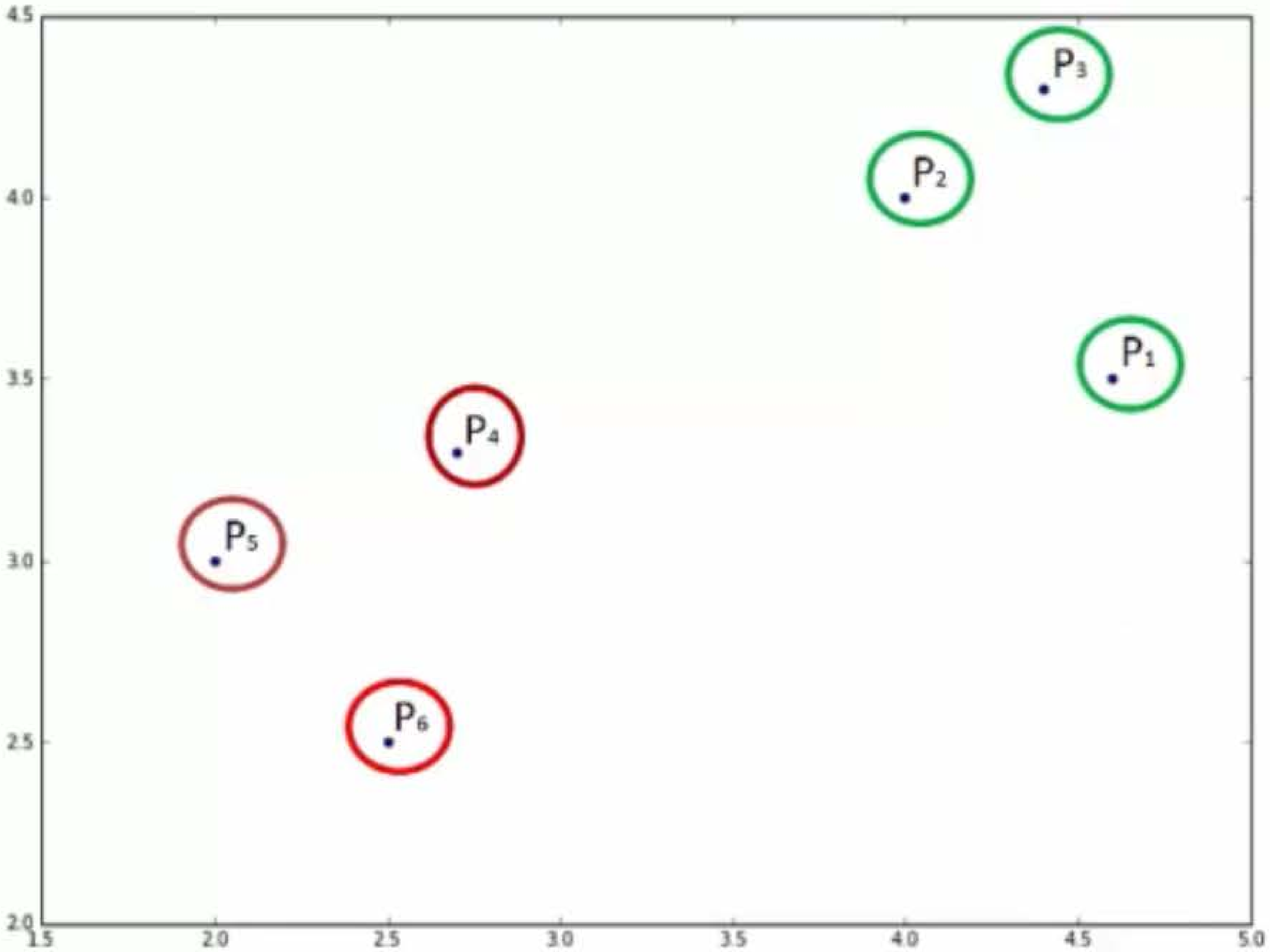
Dendrograms – Two Clusters

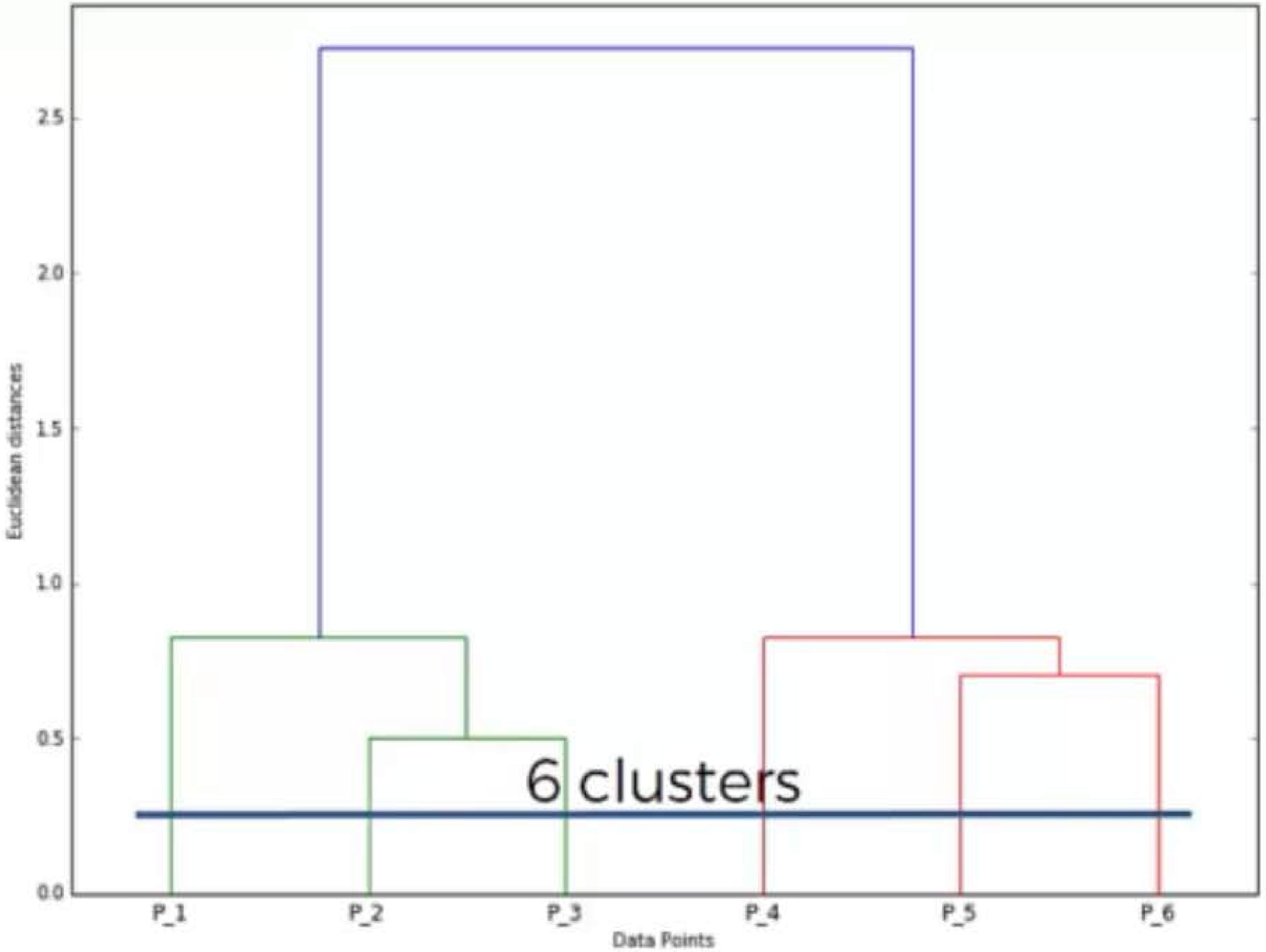




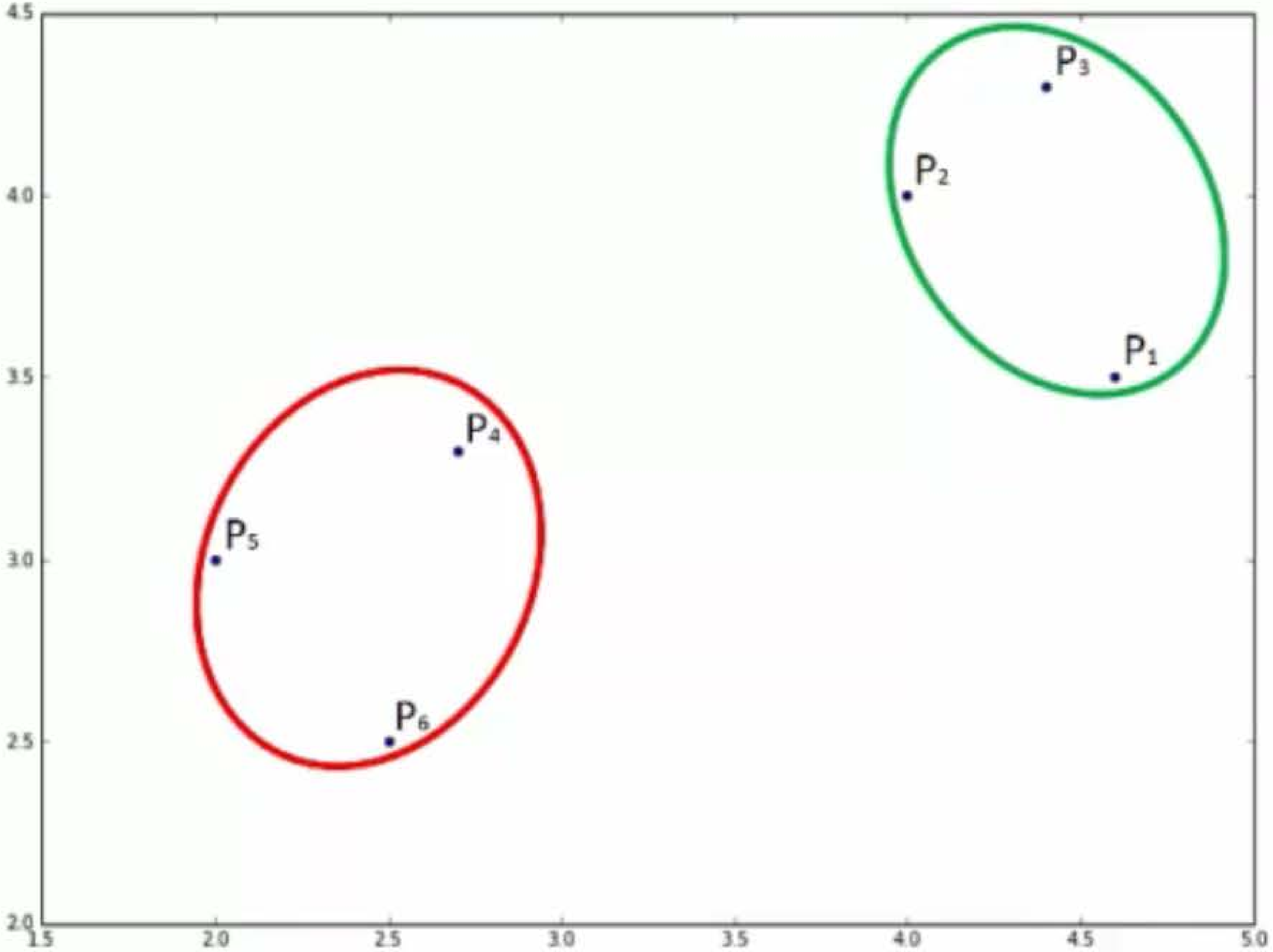
Dendrograms – Four Clusters

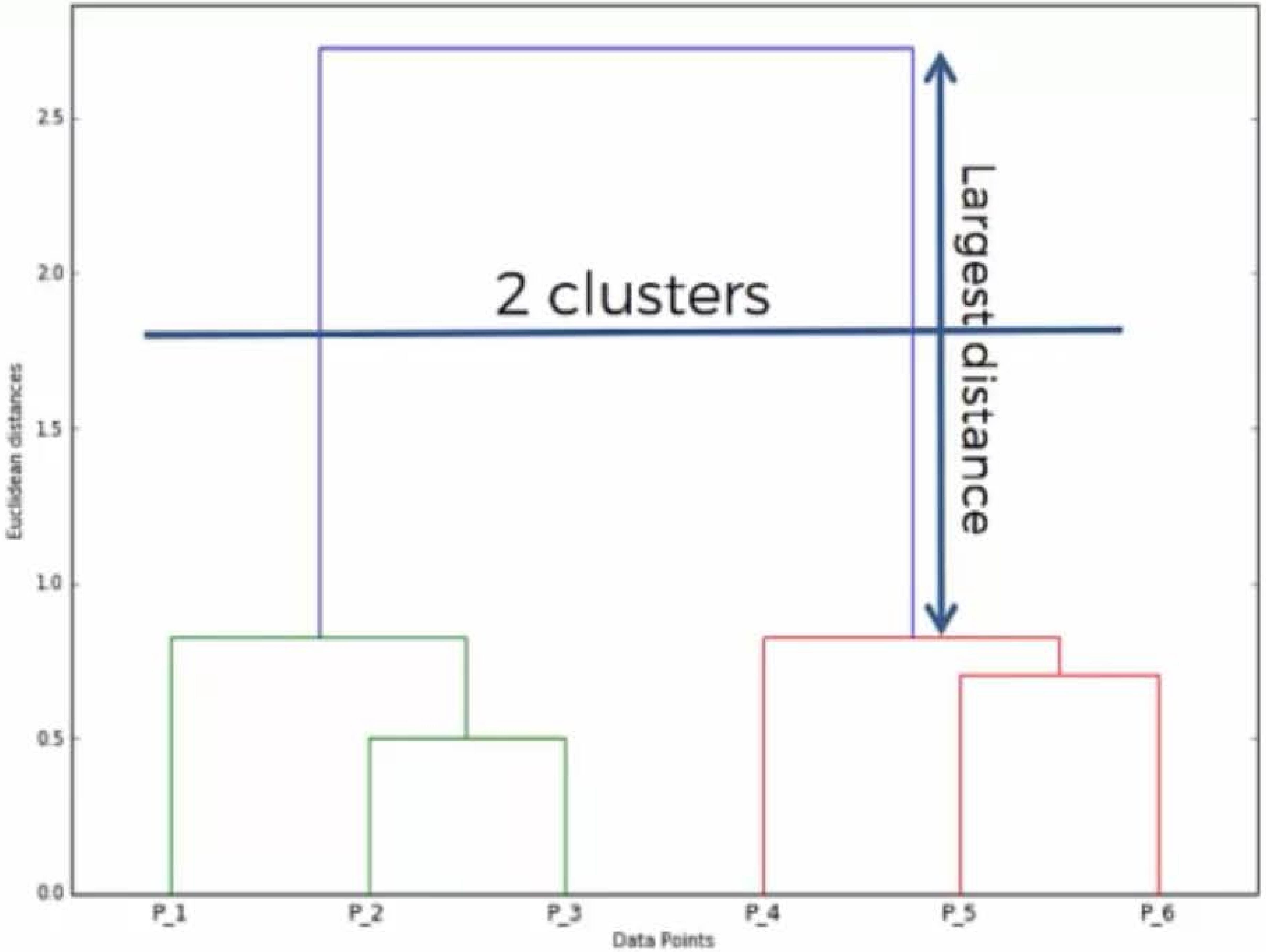
Dendrograms – Six Clusters



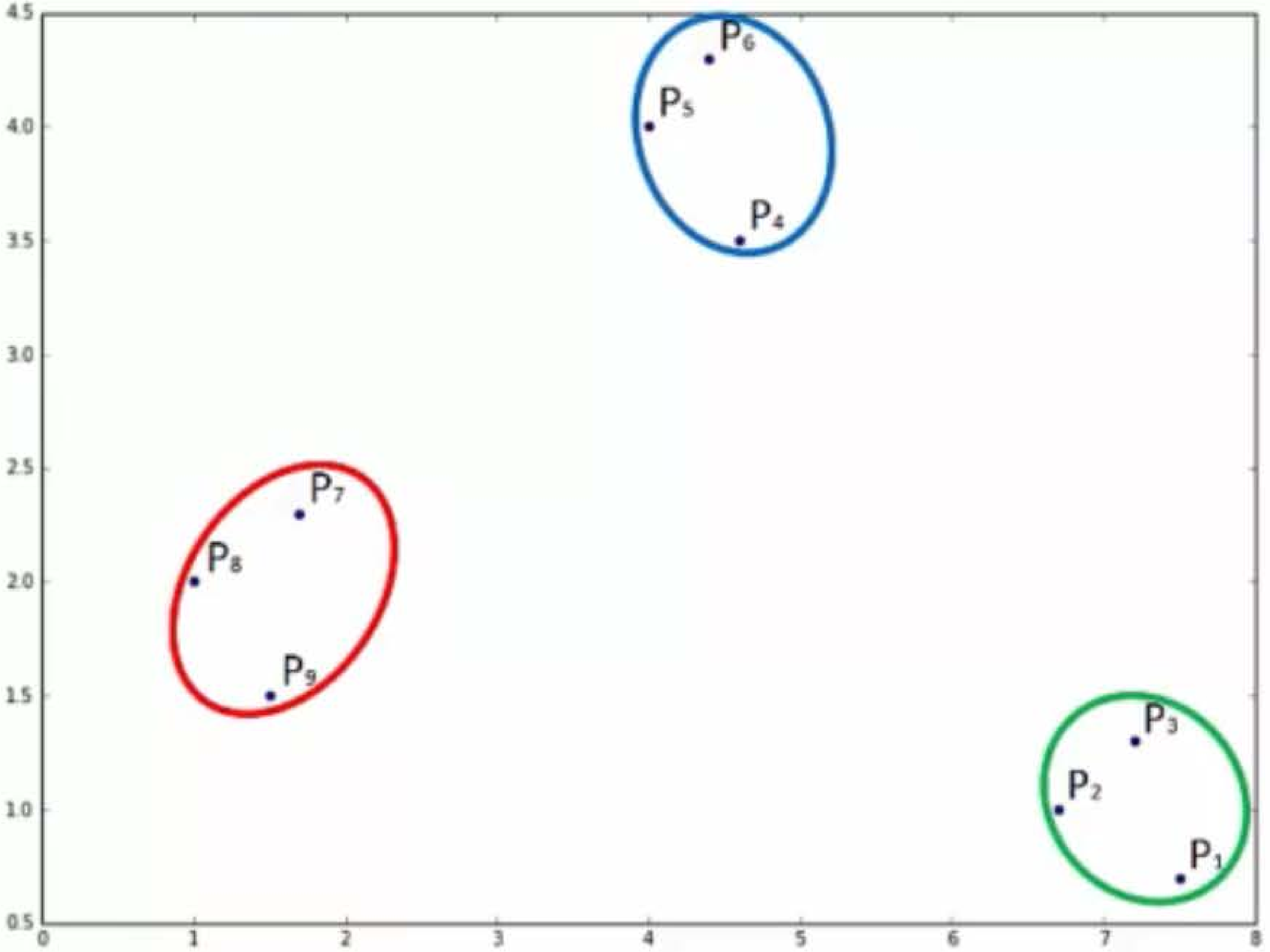


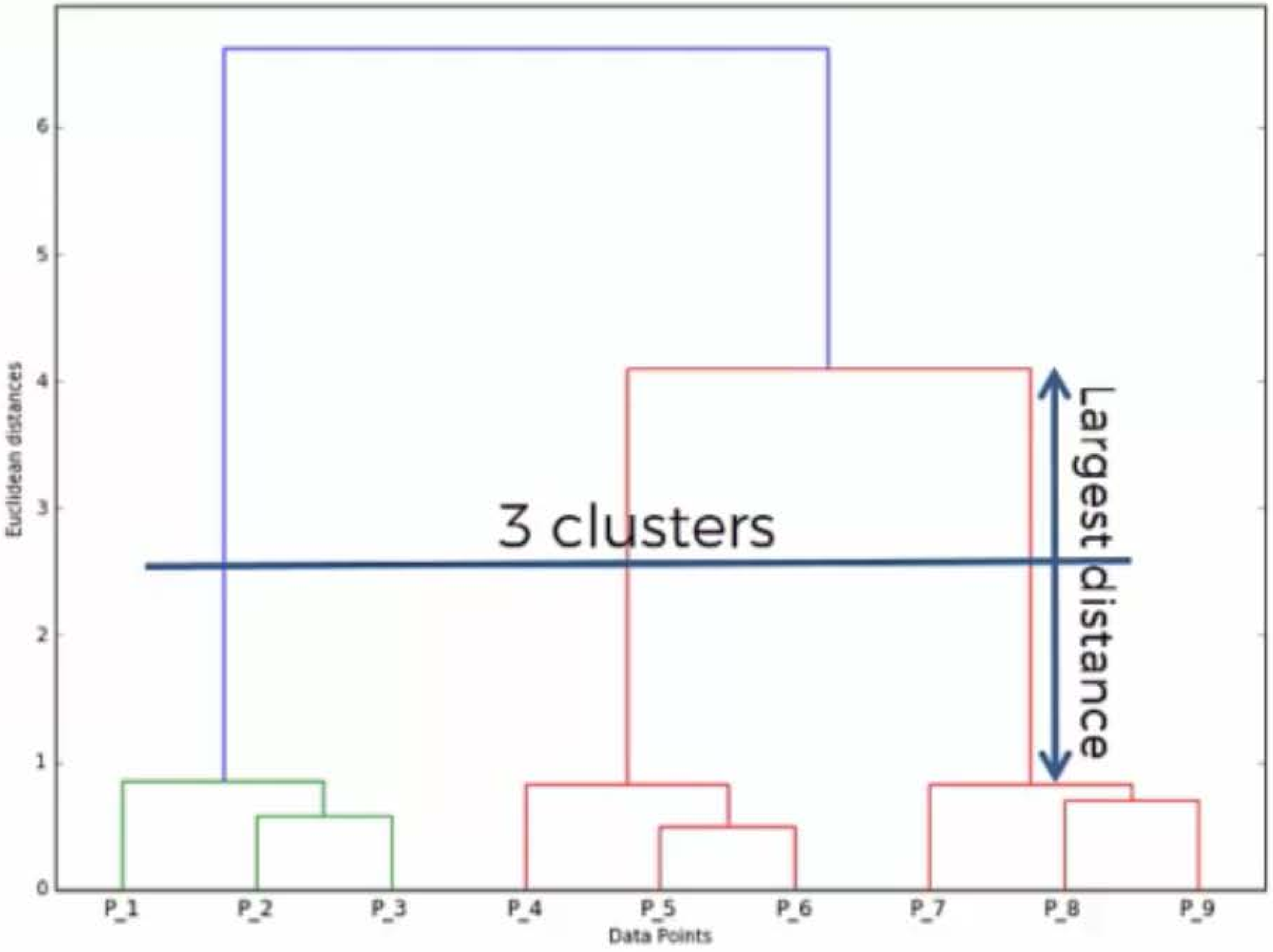
Dendrograms – Optimal # of Clusters





Dendrograms – Knowledge Test





jupyter hierarchical_clustering (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3

Run Markdown

Hierarchical Clustering

Importing the libraries

In [0]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the dataset

In [0]:
```
dataset = pd.read_csv('Mall_Customers.csv')
X = dataset.iloc[:, [3, 4]].values
```

Using the dendrogram to find the optimal number of clusters

In [3]:
```
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Customers')
plt.ylabel('Euclidean distances')
plt.show()
```