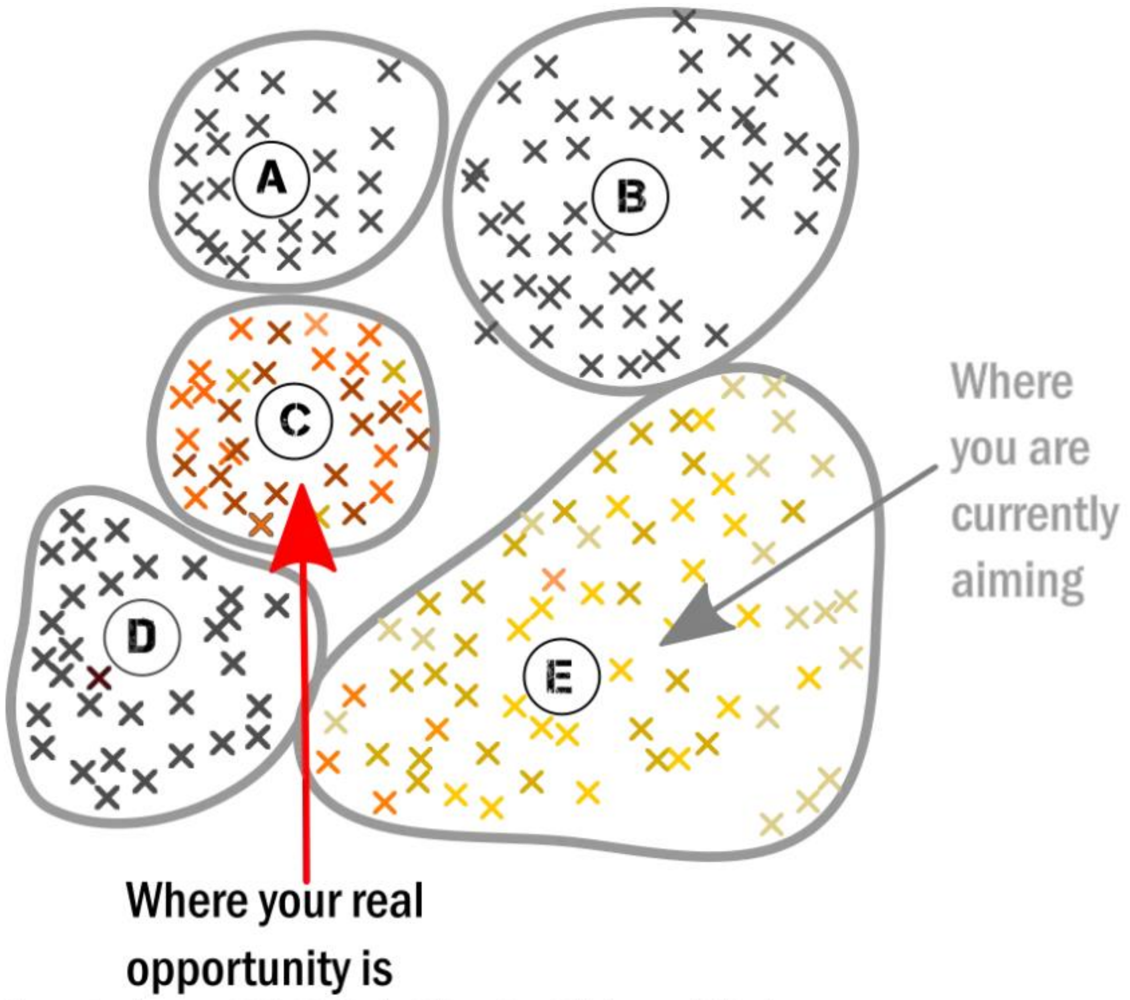
Clustering


A
B
C
D
E
Where you are currently aiming
Where your real opportunity is

Cluster Analyses can Make Marketing Efforts More Efficient and Effective

Clustering

- Clustering is famous task in the area of data mining used for information retrieval, market research, pattern recognition, data analysis, image processing etc..,
- It is a un-supervised learning technique.
  - Forms clusters of similar data automatically,
  - Segments the data so that each training example is assigned to a cluster.
- It is different from classification task – rather than predicting as class, Clustering tries to assign the data to a cluster.

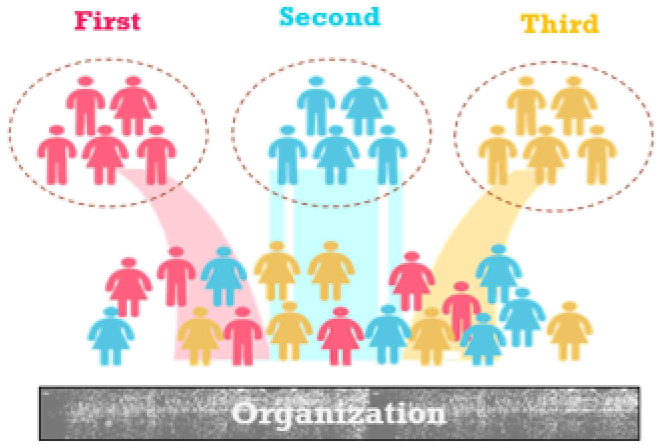
Applications of Clustering

Clustering has a large no. of applications spread across various domains.
Some of the most popular **applications of clustering** are:

- **Recommendation engines**
  - Netflix has used clustering in implementing movie recommendations for its users.
- **Customer segmentation**
  - Clustering to perform Customer Segmentation which helps to target campaigns effectively and help increase the customer engagement across various channels
  - Get customer persona analysis based on various metrics of Recency, Frequency, and Monetary metrics and build an effective User Profile – in-turn this can be used for Customer Loyalty methods to curb customer churn.
- **Search Engine**
  - Search engine can be seen in academics where clustering can help in the associative analysis of various documents – which can be in-turn used in – plagiarism, copyright infringement, patent analysis etc.
- **Image segmentation**
  - Used in image segmentation in bioinformatics where clustering algorithms have proven their worth in detecting cancerous cells from various medical imagery – eliminating the prevalent human errors and other bias.
  - Satellite imagery can be segmented to find suitable and arable lands for agriculture.
- **Anomaly detection**
  - Website network traffic can be divided into various segments and heuristically when we can prioritize the requests and also helps in detecting and preventing malicious activities.

What is Clustering?

- **Problem** : We have a sample of individuals in an organization and want to know the sub groups / sub communities. How do we do it?
- **Solution** :
  - We will classify them based on their interests / income / experience etc.
  - Next, we bring together these individuals that have similar characteristics and group them.
- This process of grouping/organizing similar set of objects/elements(individuals in this case) into classes/ clusters/ segments/partitions is termed as **clustering**.

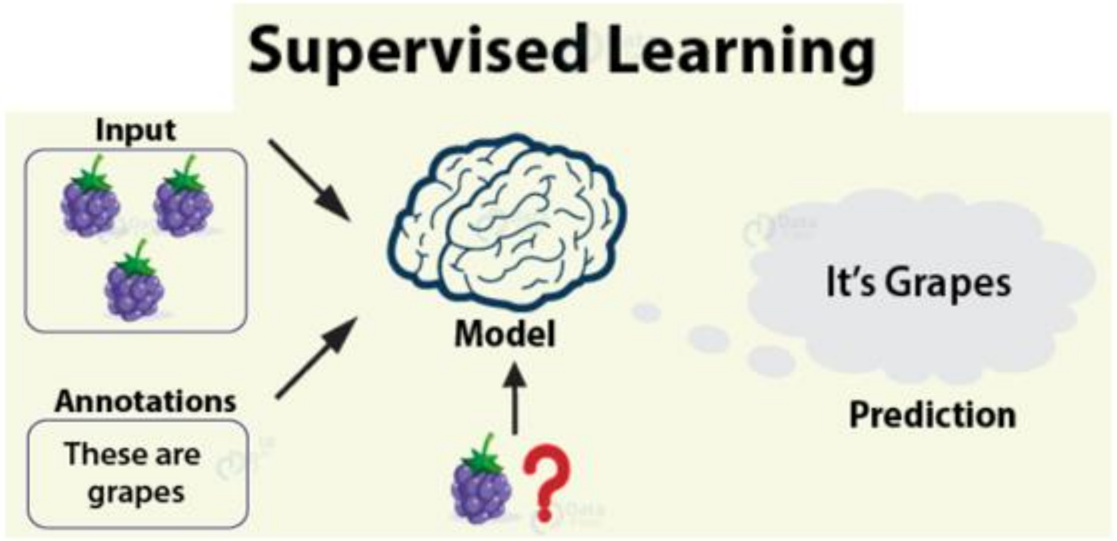


First – Low Income, High Experience
Second – Low Income, Less Experience
Third – High Income, Less Experience

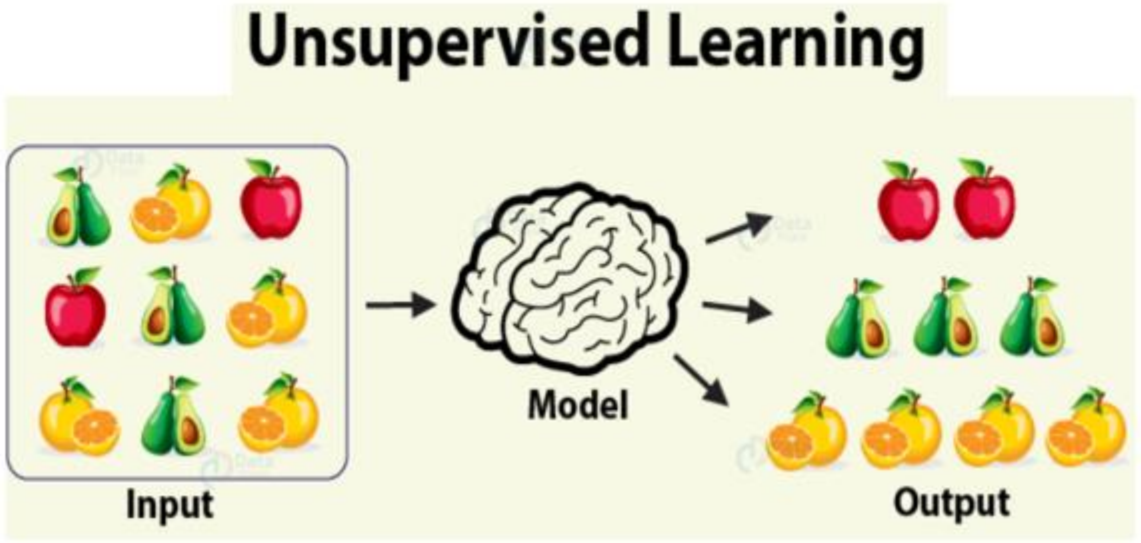
- Attach label to each observation or data points in a set
- This can be said as "unsupervised classification"
- Clustering is alternatively called as "grouping"
- Intuitively, if you would want to assign same label to a data points that are "close" to each other
- Thus, clustering algorithms rely on a distance metric between data points

How is Clustering Unsupervised ?

INNOMATICS RESEARCH LABS



The dataset, in this case is labeled, meaning that the algorithm identifies the features and carries out predictions or classification accordingly.
As we provide it with more examples, algorithm is able to learn so that it can identify the relationships between the two variables such that it can predict a new outcome more accurately.



In the case of unsupervised learning algorithm, the data is not labeled into different classes.

Unsupervised Learning algorithms identify the data based on their patterns, densities, structures, similar segments, and other similar features without any given input-output mapping.

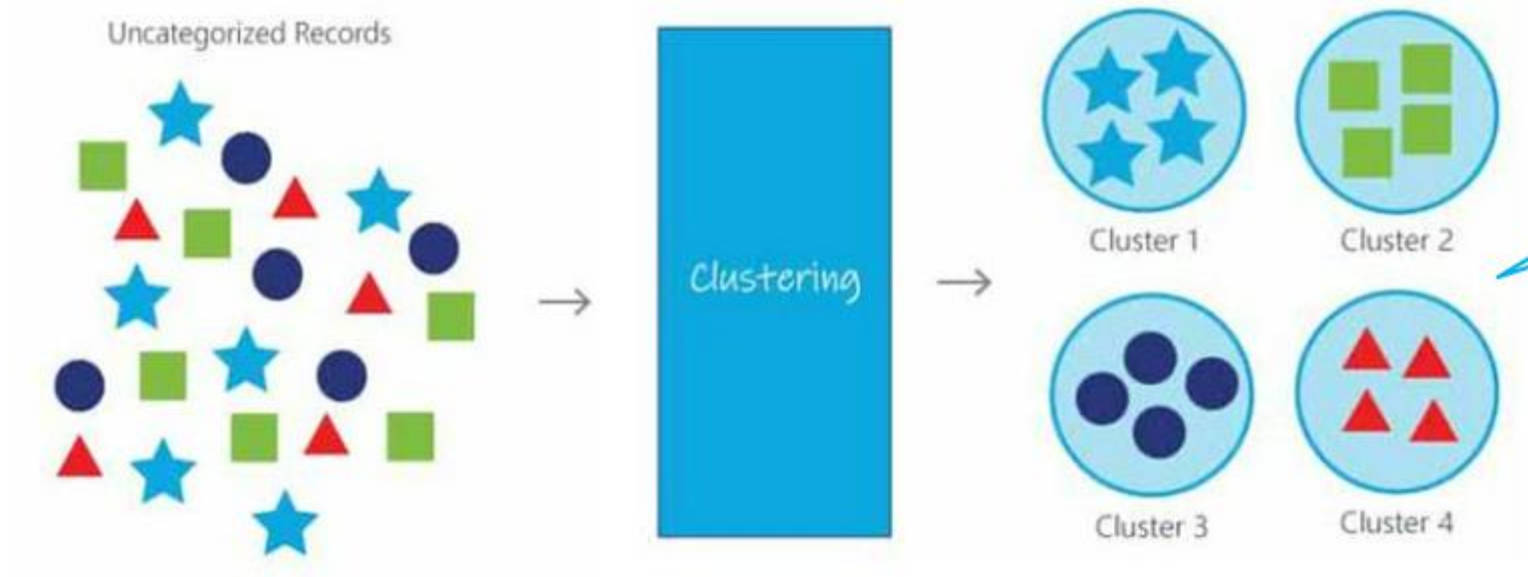
Clustering deals with finding a structure and create groups in a collection of unlabeled data.
Hence it is an **unsupervised learning problem.**
However, it can also be used to improve the accuracy of Supervised Learning models.

Objective of clustering

Primary objective behind clustering

Object in same cluster are similar(Homogeneous) : Intra clustering distance should be minimized.
Objects in different clusters are dissimilar (Heterogeneous) : Inter-clustering should be maximized.

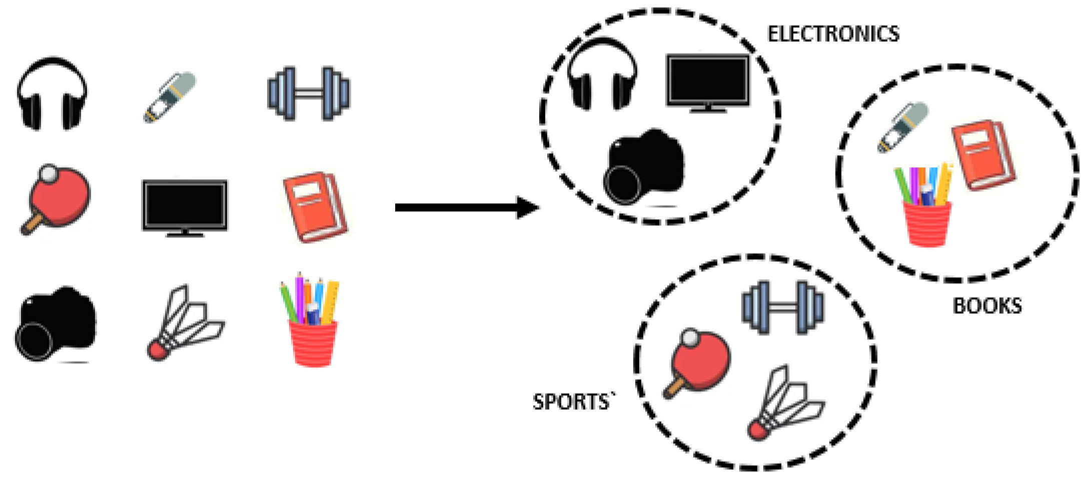
The basic **idea** behind clustering is grouping data objects together.
Such that each individual cluster holds the most similar objects thus can be treated collectively as one.
But as a collection they are sufficiently different from other clusters.

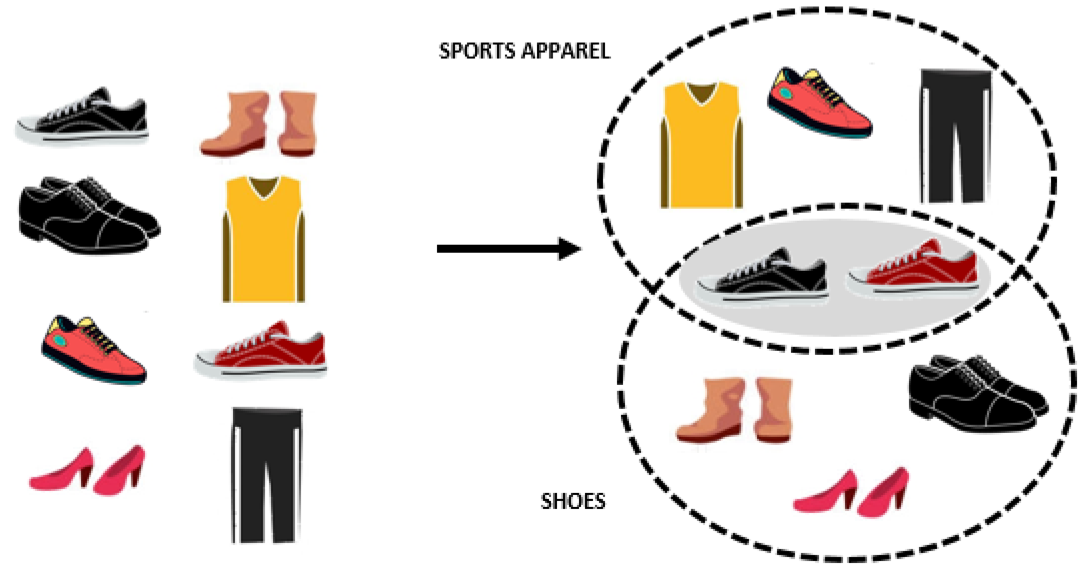


Clusters 1,2,3,4 are **homogenous** within groups as they hold similar objects but **heterogeneous** as each cluster holds different objects.

Broadly speaking, clustering can be divided into two subgroups :

Hard Clustering :	**Soft Clustering :**
▪ Each data point is a member of exactly 1 cluster. ▪ Hard clustering creates different distinct groups. ▪ More common and easier to compute.	▪ A data point can belong to more than 1 clusters : fractional membership. ▪ Overlaps of clusters created with probabilities
▪ If we want to group items in an e-commerce site based on different categories : **Hard Clustering** is done and a category is assigned to each of these items. We can then segment customers on their purchasing habits in each of the groups to understand their preferences & use a separate strategy to scale up the business. ▪ Algorithm : K-means	▪ If we want to create browsable hierarchies for items in an e-commerce site: **Soft Clustering** is done and overlapping of clusters is observed. ▪ Example : Sneakers item can be put in 2 clusters : ▪ (i) Sports apparel and ▪ (ii) Shoes. ▪ Algorithm: Mixture of Gaussians, Fuzzy c-Means





Clustering

- Commonly, to measure similarity or dissimilarity between data points, a distance measure is used.
- Common distance functions are:
  - ❑Euclidean
  - ❑Manhattan
  - ❑Minkowski
- A distance function returns a lower value for pairs of objects that are more similar to one another.

Distance Functions

3 types of distance functions are :

Similar
Not Similar

Euclidean Distance: represents the shortest distance between 2 points
Most ML algorithms including K-means use this distance metric to measure similarity

Euclidean $$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan Distance: is the sum of absolute differences between points across all the dimensions
It is also known as city block distance

Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

Minkowski Distance: is the generalized form of Euclidean & Manhattan distance

Minkowski $$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

Here, q represents the order, if it's 1 will represent **Manhattan Distance** and when the order is 2 will represent **Euclidean Distance**

Here, k is the number of dimensions and xi, yi are data points

INNOMATICS
RESEARCH LABS

Examples of using the distance functions:

1- dimensional (k=1)	
Customer	**Weight**
Cust 1	68
Cust 2	72
Cust 3	100

Which of the two customers are similar ?			
Customer Pair	**Co-Ordinates (x) to (y)**	**Euclidean Distance**	**Manhattan Distance**
Cust 1, Cust 2	x=68, y=72	4 ((72-68)^2)^1/2	4 *(|68-72|)*
Cust 1, Cust 3	x=68, y=100	32 *((68-100)^2)^1/2*	32 *(|68-100|)*
Cust 2, Cust 3	x=72, y=100	28 *((72-100)^2)^1/2*	28 *(|72-100|)*

2- dimensional (k=2)		
Customer	**Weight**	**Age**
Cust 1	68	25
Cust 2	72	70
Cust 3	100	28

Which of the two customers are similar ?			
Customer Pair	**Co-Ordinates (x1,x2) to (y1,y2)**	**Euclidean Distance**	**Manhattan Distance**
Cust 1, Cust 2	(68,25) to (72,70)	45.17742799	49
Cust 1, Cust 3	(68,25) to (100,28)	32.14031736	35
Cust 2, Cust 3	(72,70) to (100,28)	50.47771786	70

Types of Clustering Algorithms

- **Connectivity models (*Hierarchical Clustering*)**
  - As the name suggests, these models are based on the notion that the data points closer in data space exhibit more similarity to each other than the data points lying farther away.
  - These models can follow two approaches.
    - In the first approach, they start with classifying all data points into separate clusters & then aggregating them as the distance decreases. (***Agglomerative or Bottom-Up***)
    - In the second approach, all data points are classified as a single cluster and then partitioned as the distance increases. (***Divisive or Top-down***)

  These models are very easy to interpret but lacks scalability for handling big datasets.
  Examples of these models are **hierarchical clustering algorithm and its variants**

  Centroid models (*Partitional Clustering*)
  - These are iterative clustering algorithms in which the notion of similarity is derived by the closeness of a data point to the centroid of the clusters. **K-Means clustering algorithm** is a popular algorithm that falls into this category. In these models, the no. of clusters required at the end have to be mentioned beforehand, which makes it important to have prior knowledge of the dataset.
  - These models run iteratively to find the local optima.

  Density models (*Density based Clustering*)
  - These models search the data space for areas of varied density of data points in the data space. It isolates various different density regions and assign the data points within these regions in the same cluster.
  - Popular examples of density models: **DBSCAN**

K-Means

- K-Means clustering intends to partition **n** objects into **k** clusters in which each object belongs to the cluster with the nearest mean.
- This method produces exactly **k** different clusters of greatest possible distinction.
- The best number of clusters **k** leading to the greatest separation (distance) is not known as a priori and must be computed from the data.

K-Means

- The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:
- Also, called as Within sum of squares (WSS).

number of clusters number of cases

centroid for cluster *j*

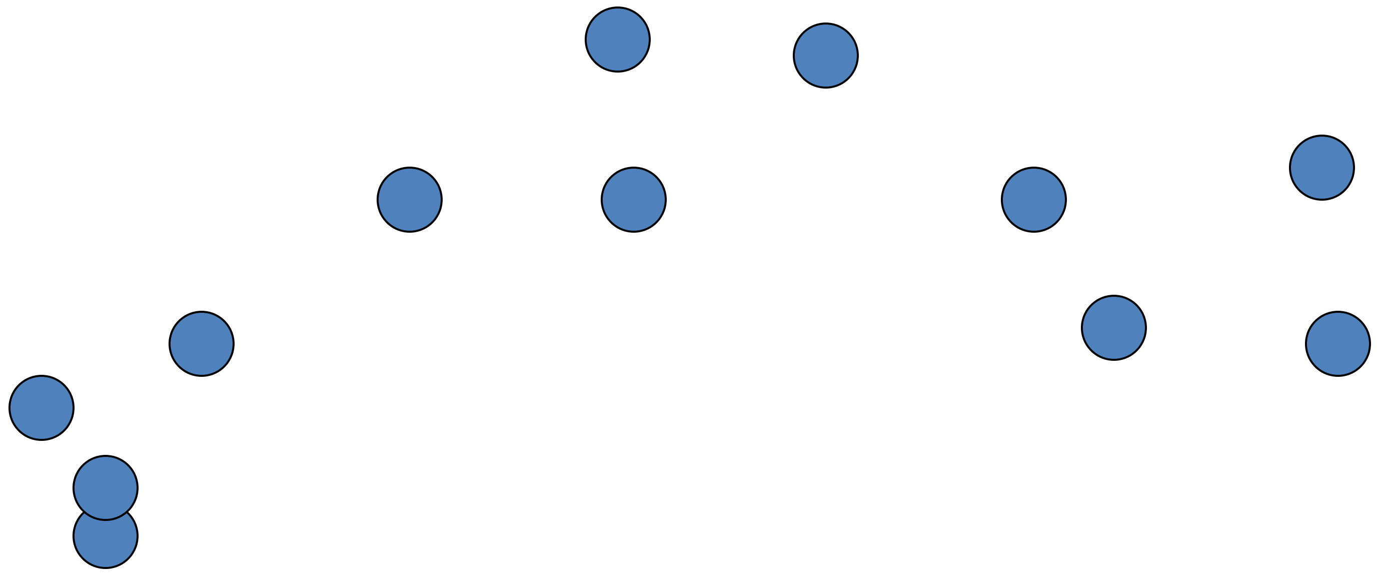
case *i*

$$\text{objective function} \leftarrow J = \sum_{j=1}^{k}\sum_{i=1}^{n}\underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

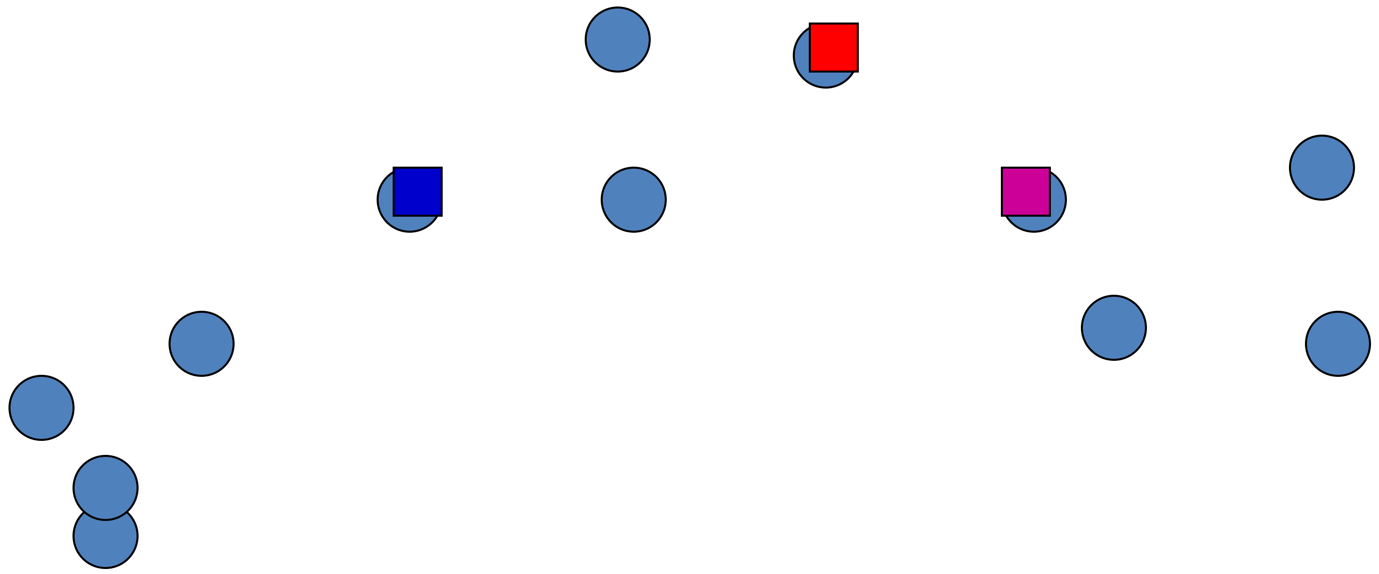
K-Means

- Steps involved in K-Means:
  1. Clusters the data into **k** groups where **k** is predefined.
  2. Select **k** points at random as cluster centers.
  3. Assign objects to their closest cluster center according to the Euclidean distance function.
  4. Calculate the centroid or mean of all objects in each cluster.
  5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

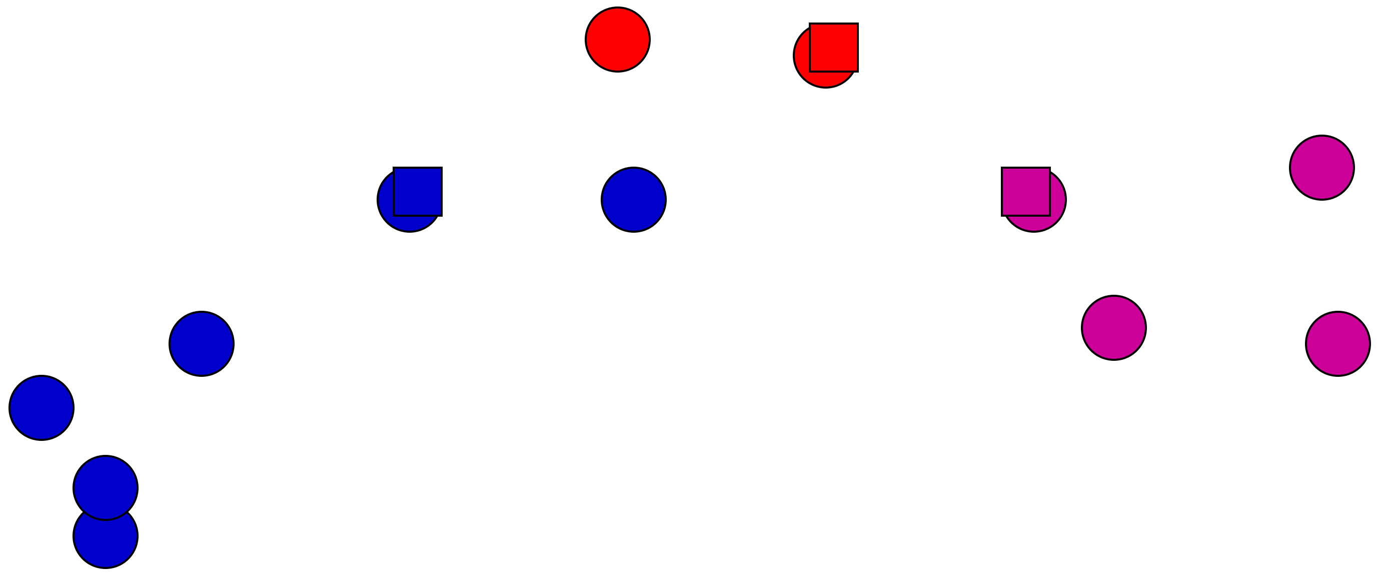
K-means: example



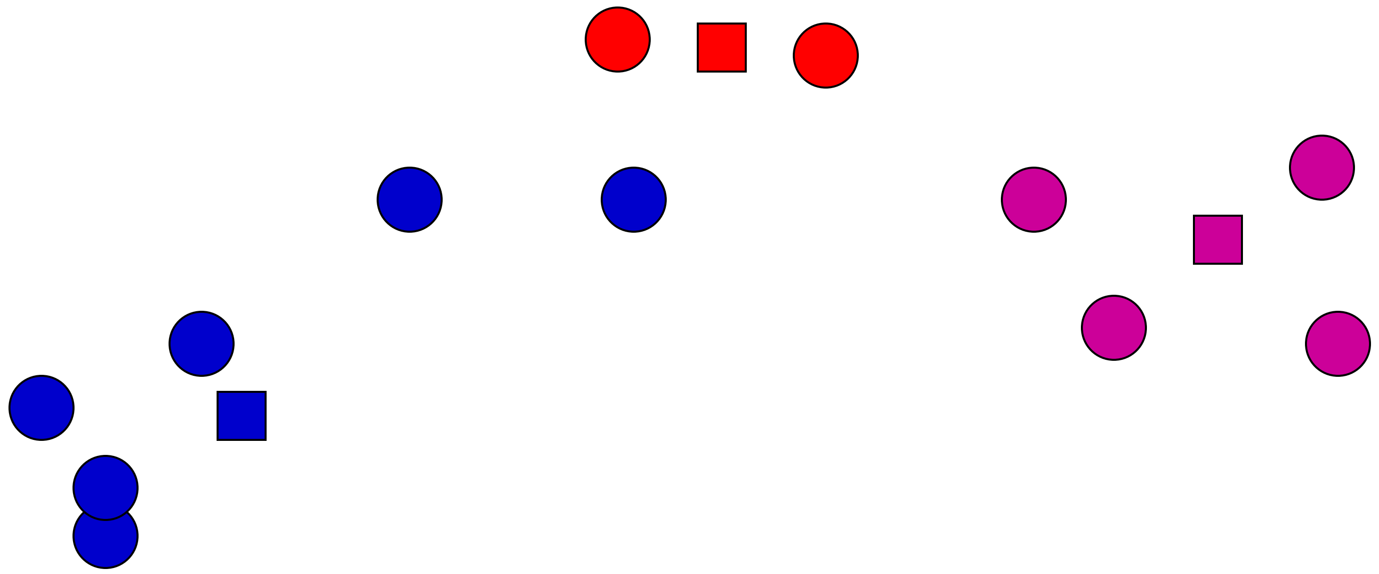
K-means: Initialize centers randomly



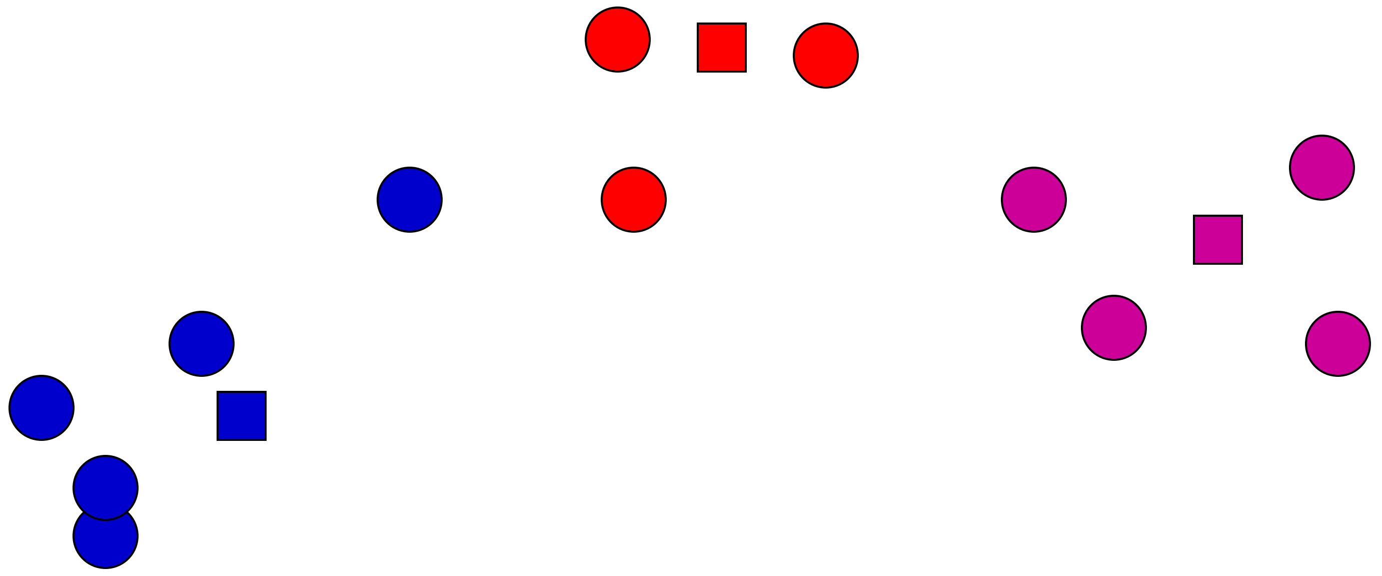
K-means: assign points to nearest center



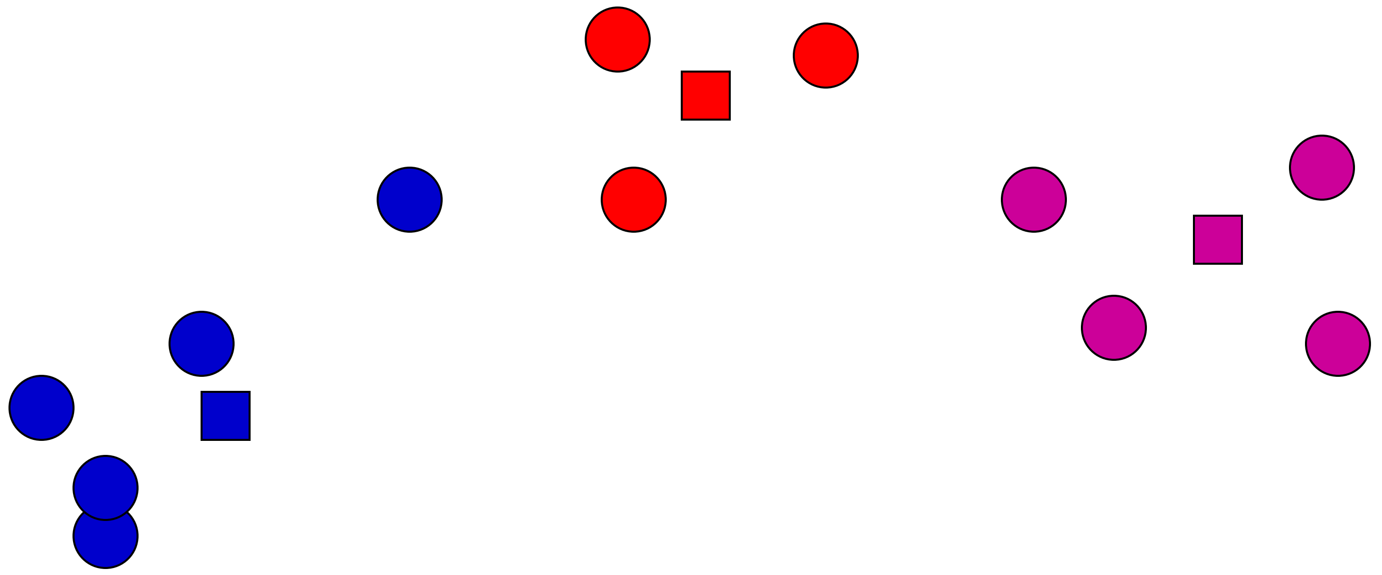
K-means: readjust centers



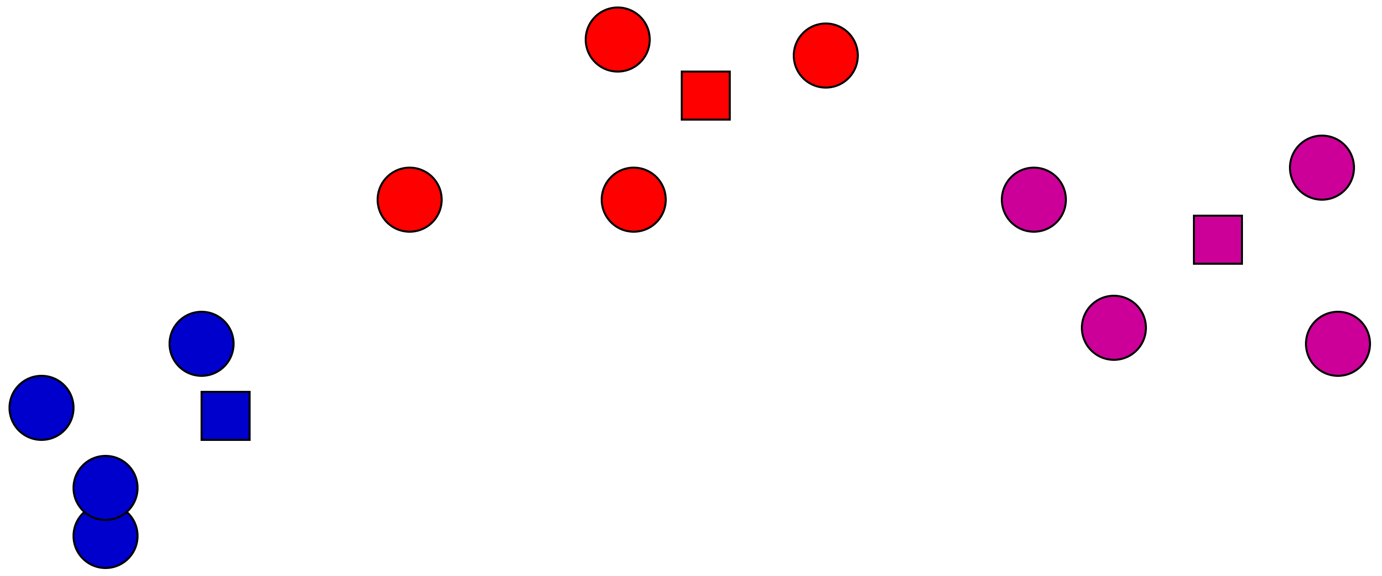
K-means: assign points to nearest center



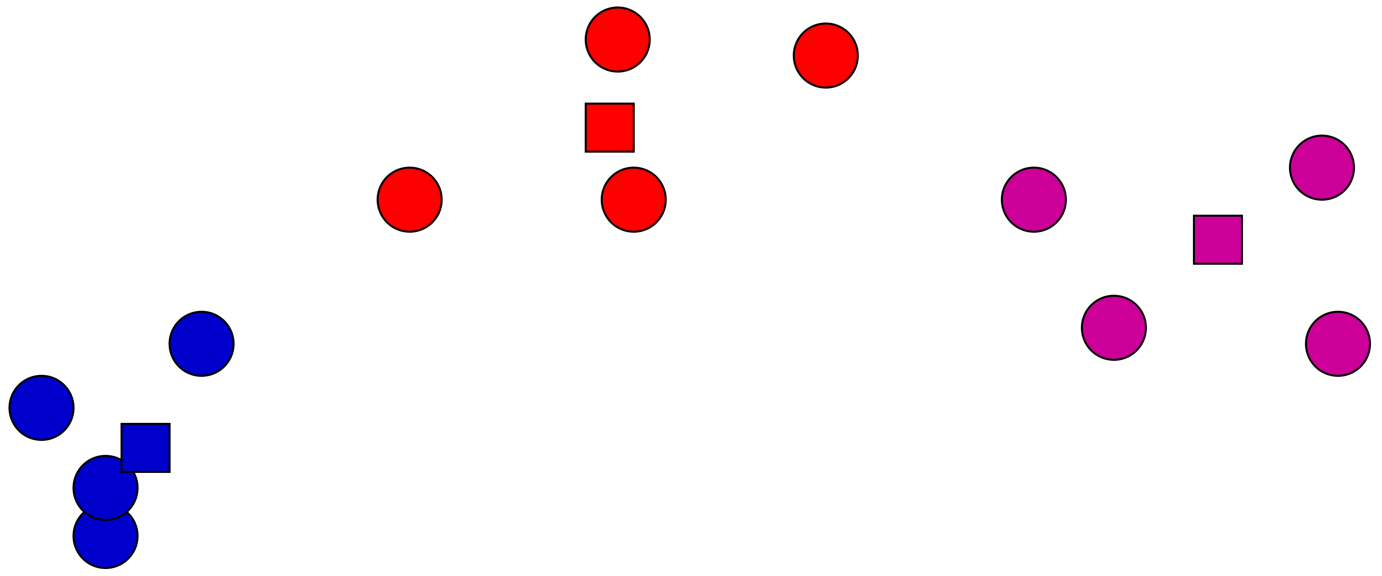
K-means: readjust centers



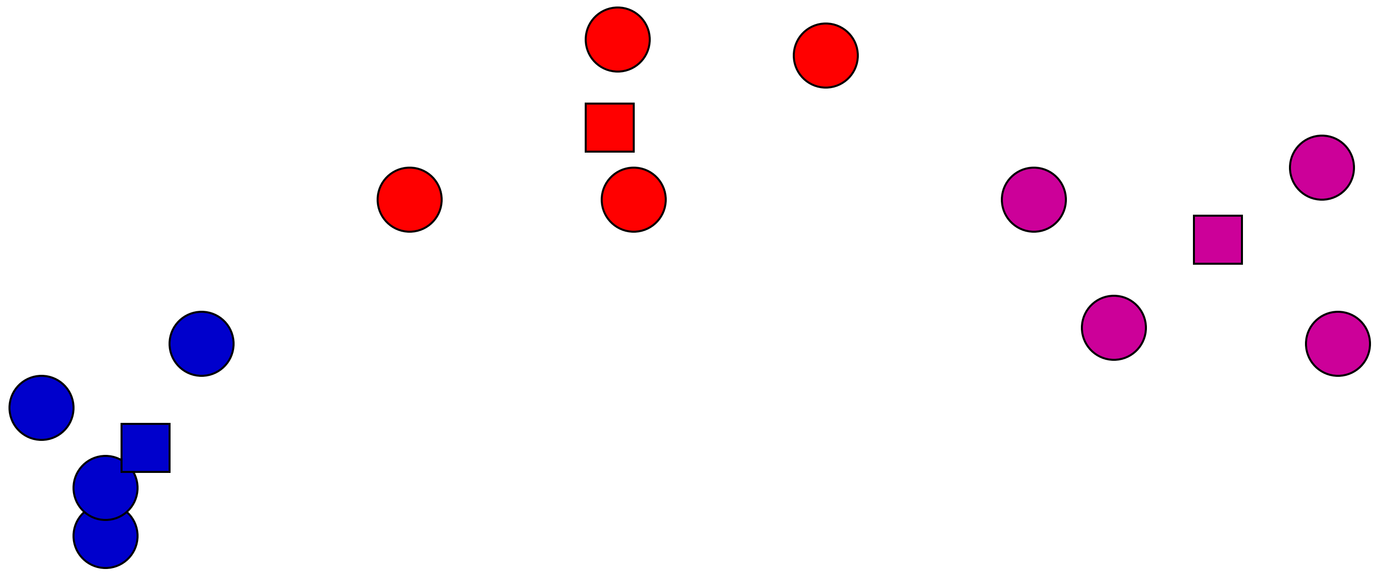
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center



No changes: Done

K-means- Summary

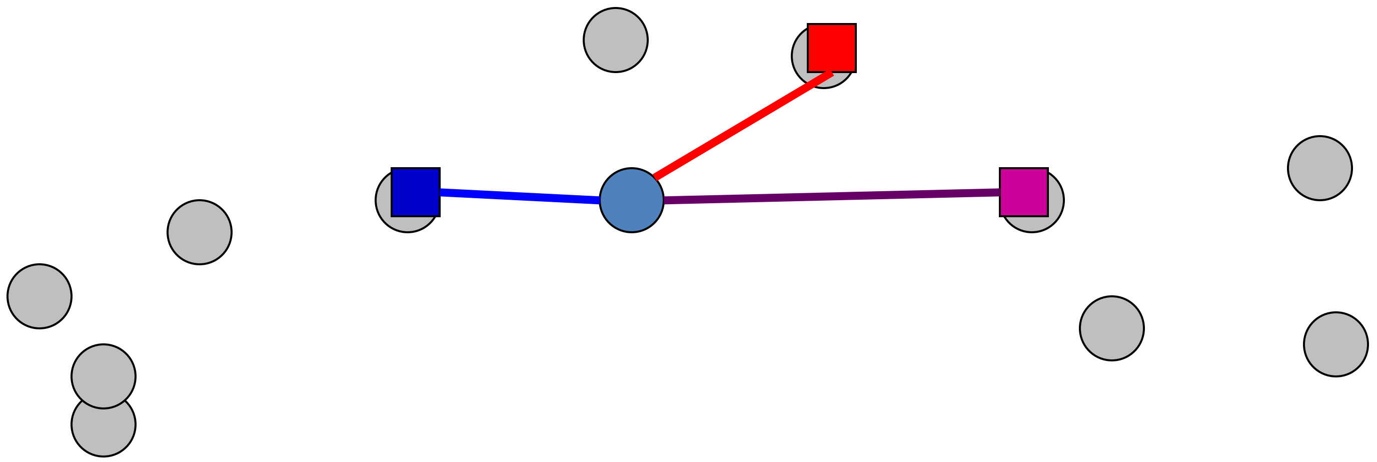
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

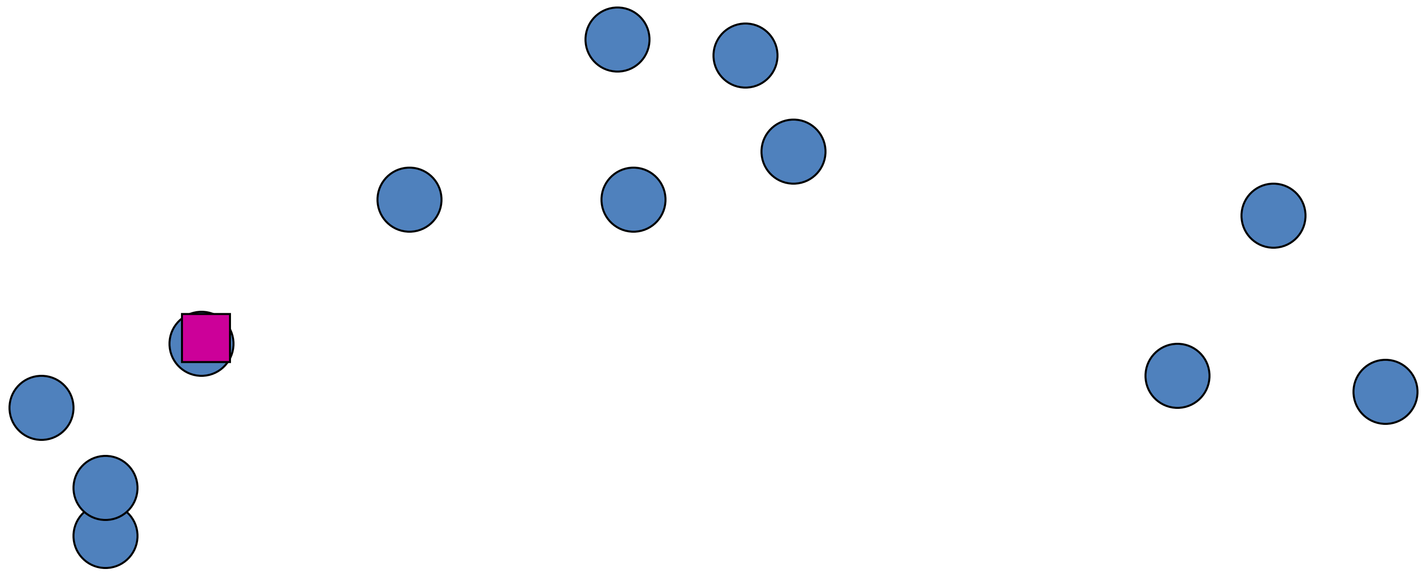
  - get distance to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



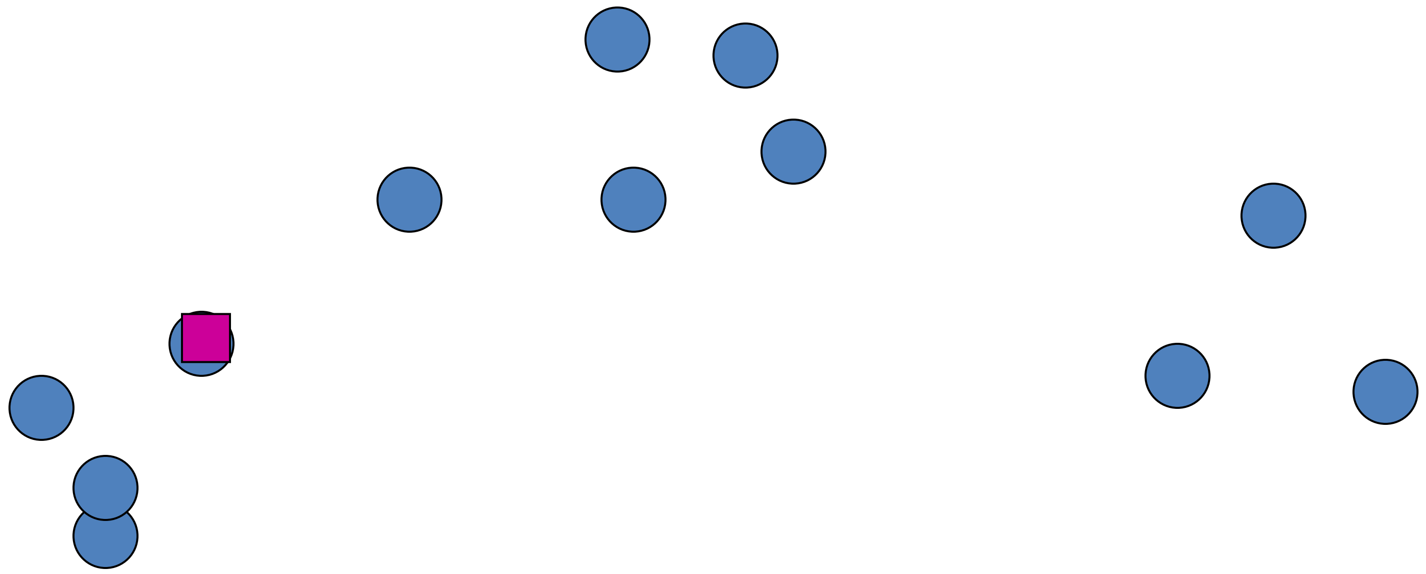
Initializing Centroids

K-means: Initialize farthest from centers



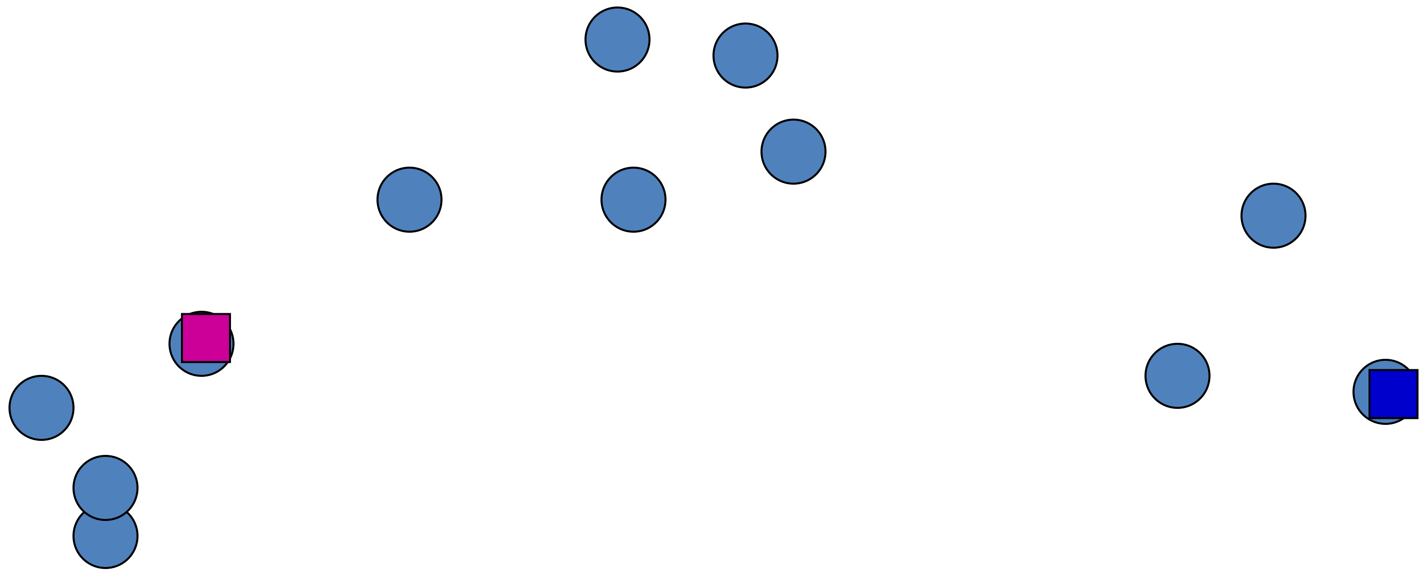
Pick a random point for the first center

K-means: Initialize farthest from centers



What point will be chosen next?

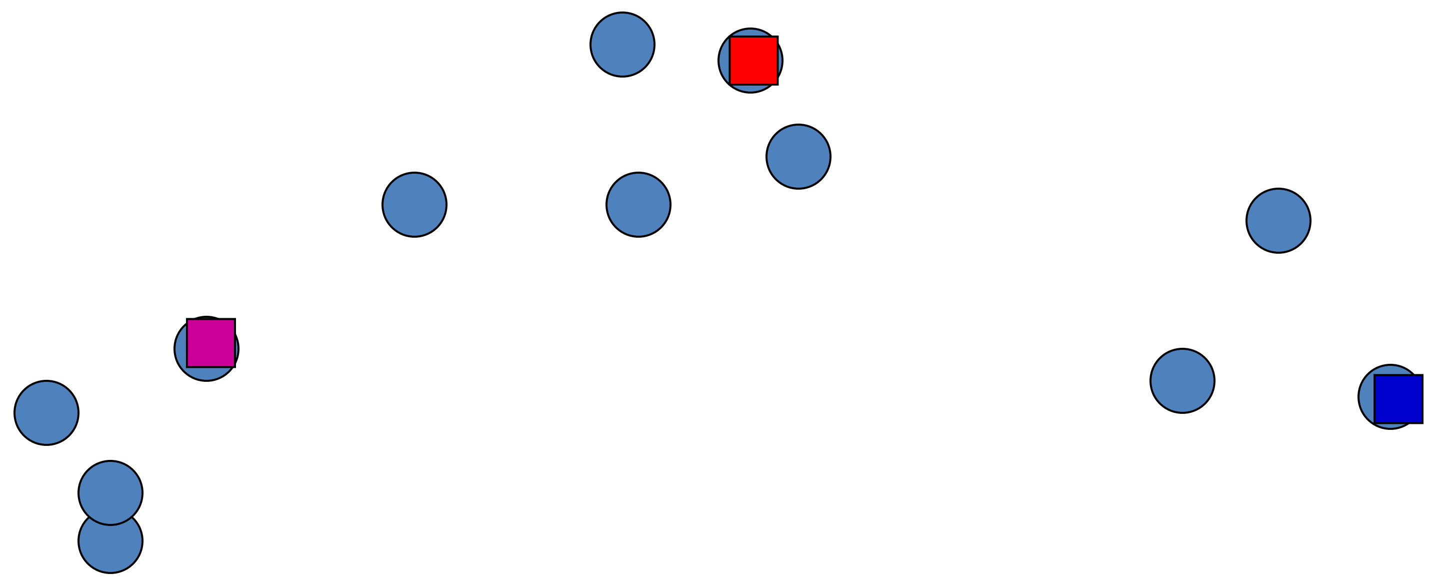
K-means: Initialize farthest from centers



Furthest point from center

What point will be chosen next?

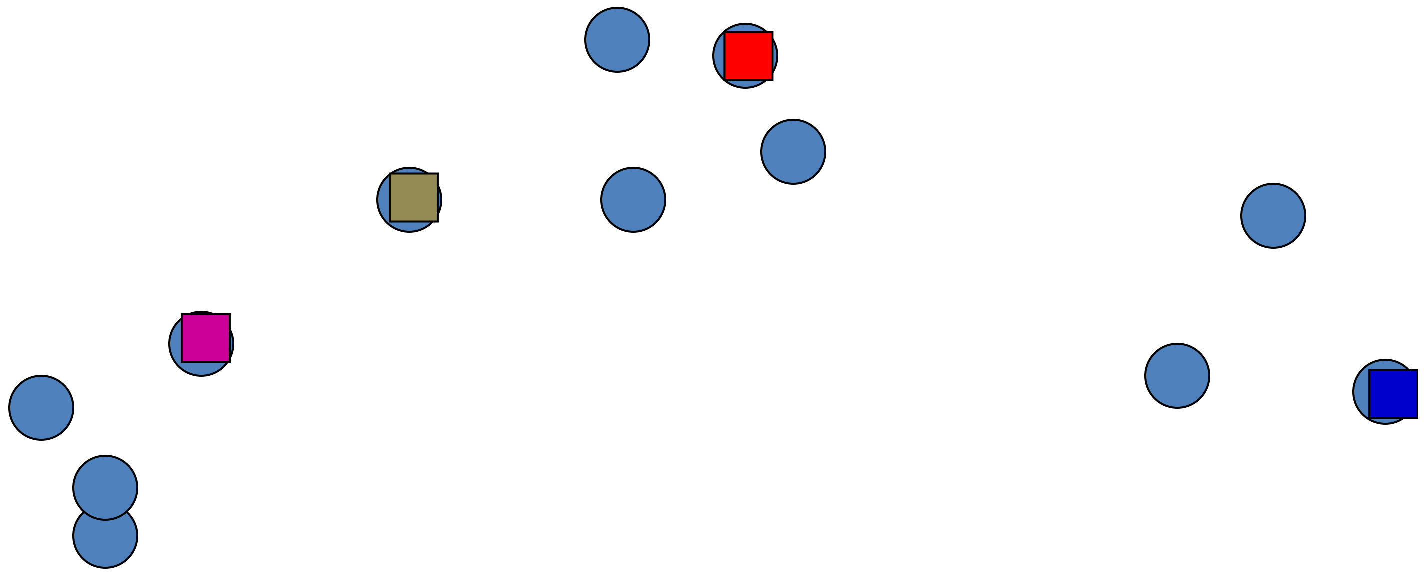
K-means: Initialize farthest from centers



Furthest point from center

What point will be chosen next?

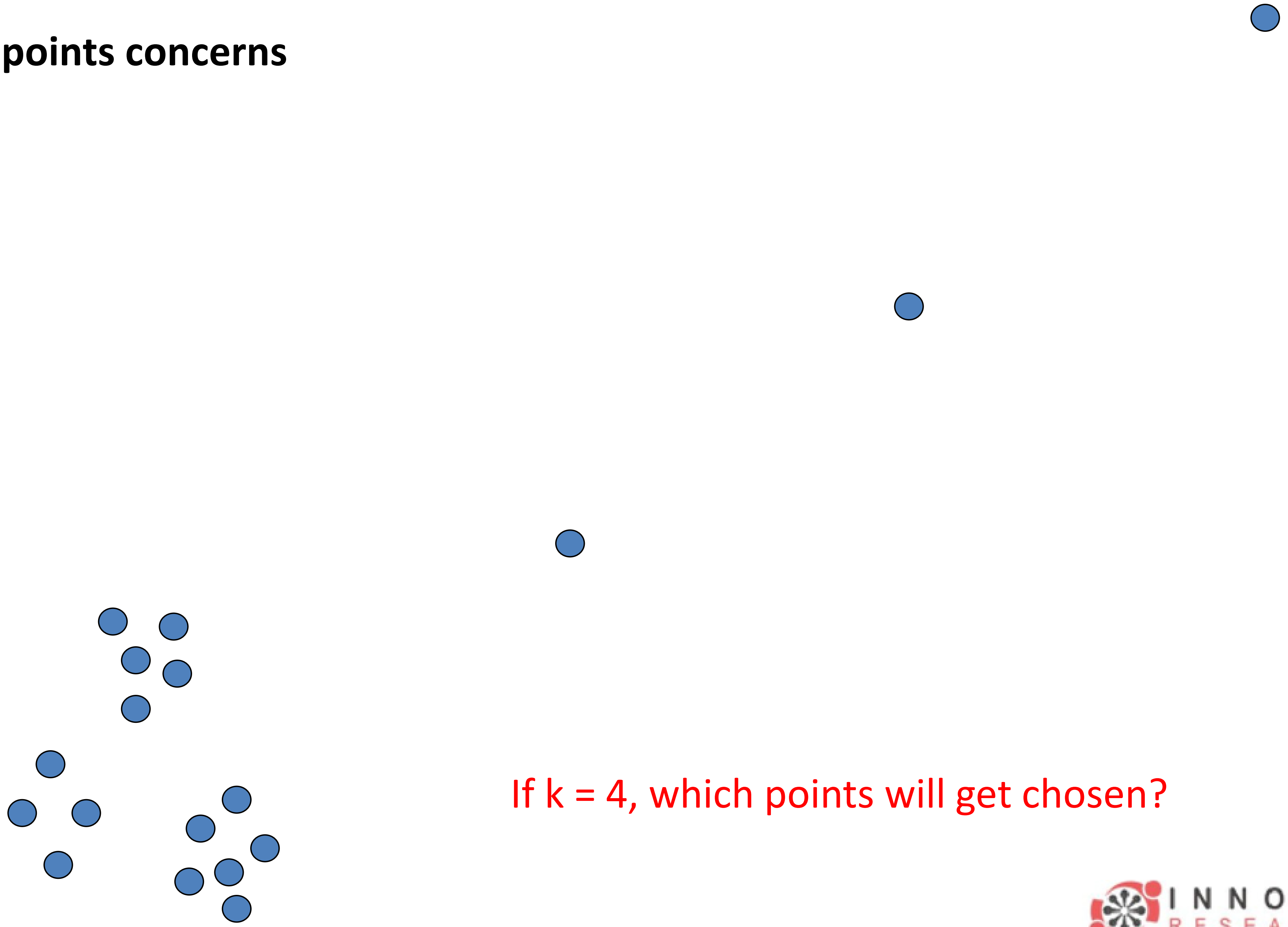
K-means: Initialize farthest from centers



Furthest point from center

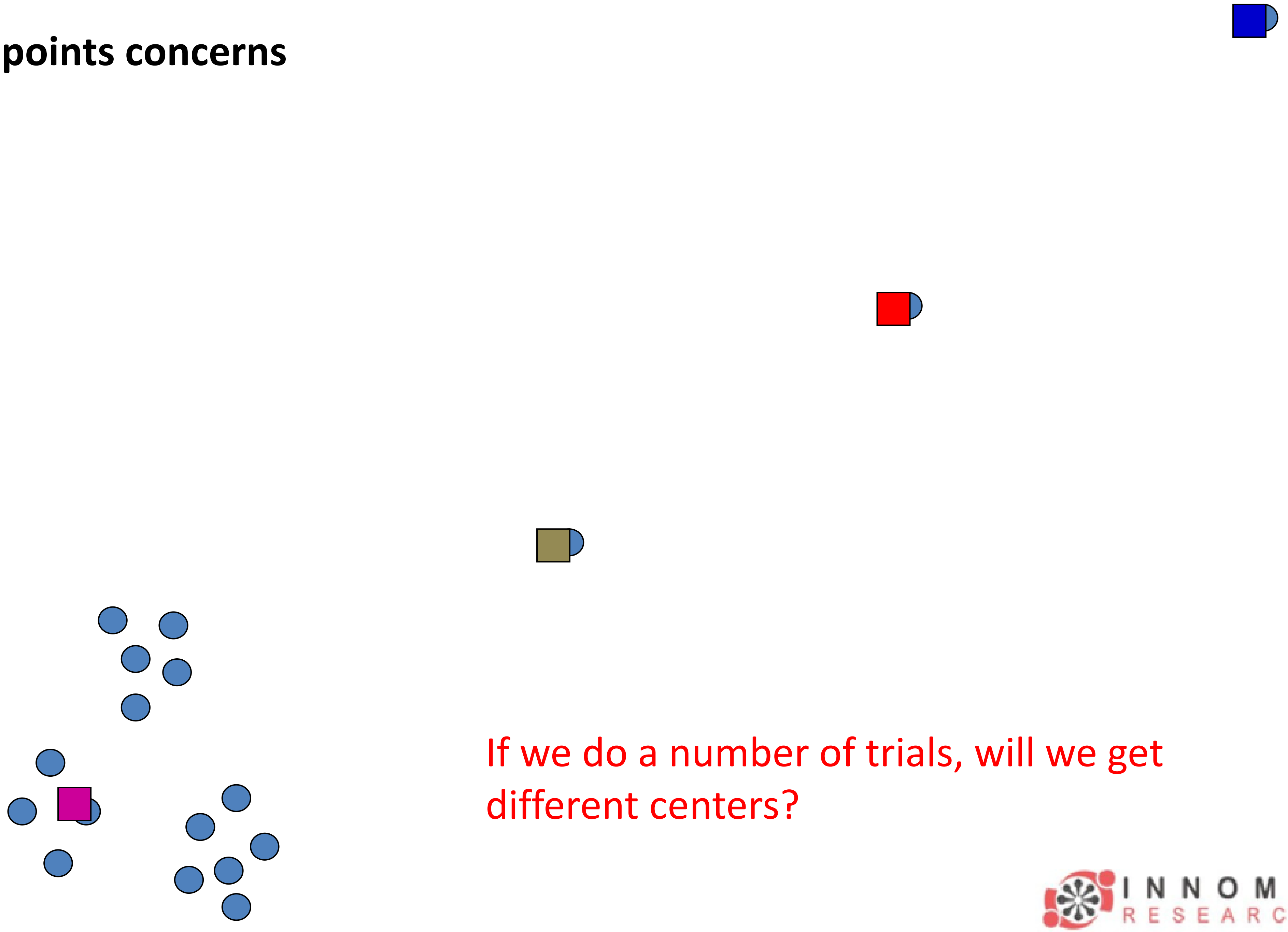
Any issues/concerns with this approach?

Farthest points concerns



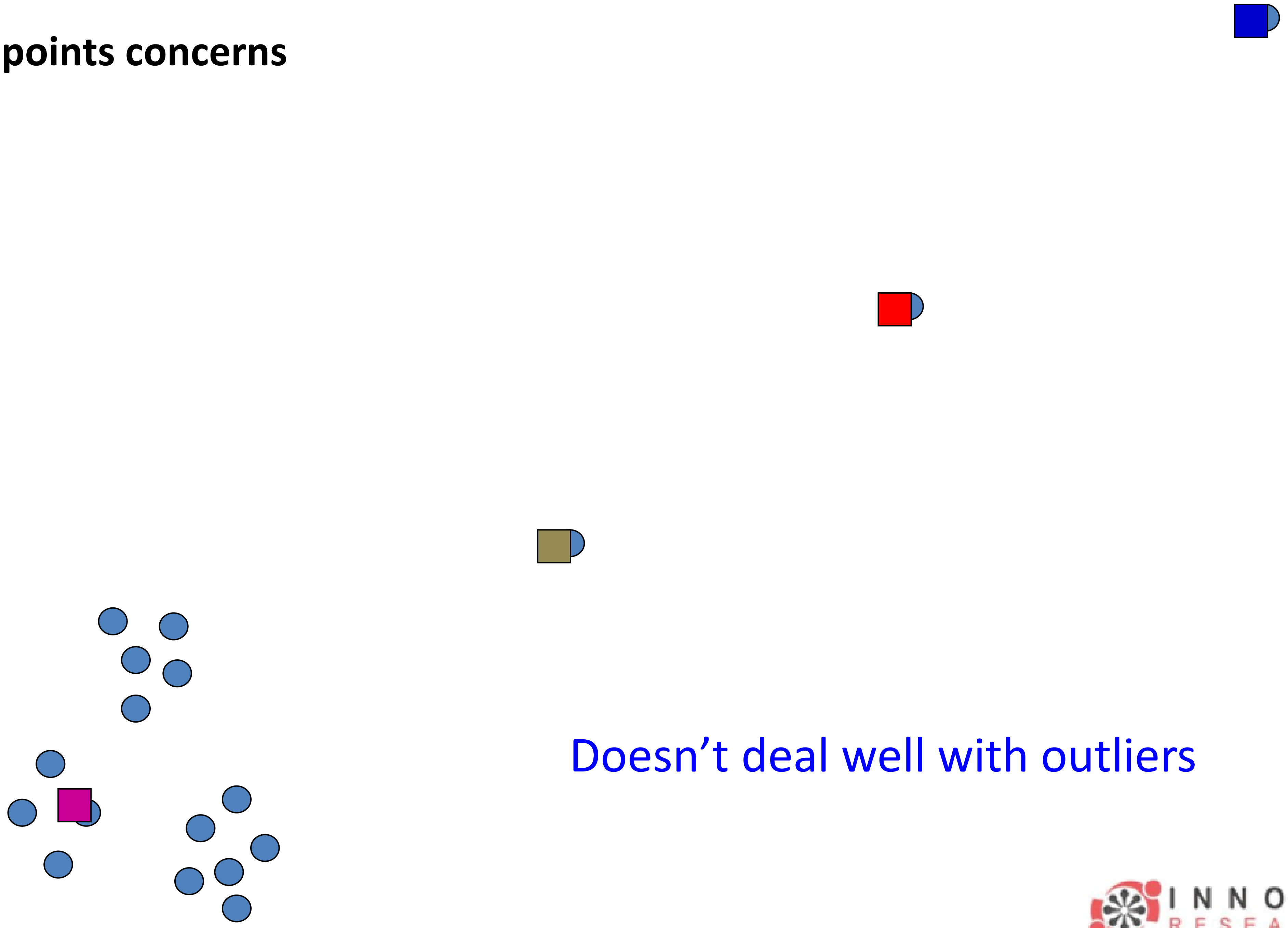
If k = 4, which points will get chosen?

Farthest points concerns



If we do a number of trials, will we get different centers?

Farthest points concerns



Doesn’t deal well with outliers

K-means++

Choose Initial Cluster Centroids for K-Means Clustering

In some cases, if the initialization of clusters is not appropriate, K-Means can result in arbitrarily bad clusters.

This is where K-Means++ helps. **It specifies a procedure to initialize the cluster centers before moving forward with the standard k-means clustering algorithm.**

The 1st centre is selected as a random case from the dataset. The 2nd centre is selected also randomly, but the probability of selection of a case is proportional to the distance (square Euclidean) of it to that (1st) centre. The 3rd centre is selected also randomly with the probability of selection proportional to the distance of a case to the nearest of those two centers, and so on

```
$\mu_1$ = pick random point
for k = 2 to **K**:
    for i = 1 to **N**:
        $s_i = \min d(x_i, \mu_{1\ldots k-1})$ // smallest distance to any center
$\mu_k$ = randomly pick point ***proportionate*** to ***s***
```

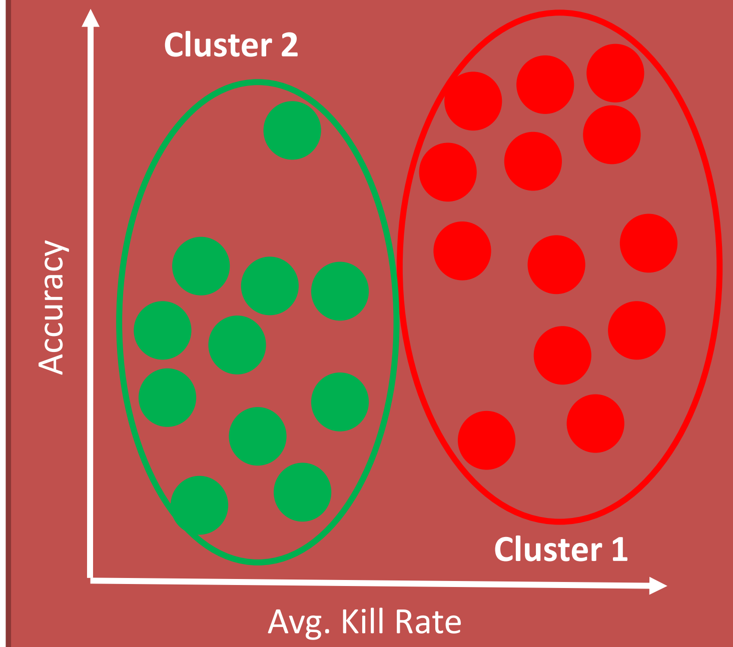
- Makes it possible to select other points
  - if #points >> #outliers, we will pick good points

The **steps to initialize the centroids using K-Means++** are:

1. The first cluster is chosen uniformly at random from the data points that we want to cluster. This is similar to what we do in K-Means, but instead of randomly picking all the centroids, we just pick one centroid here
2. Next, we compute the distance (D(x)) of each data point (x) from the cluster center that has already been chosen
3. Then, choose the new cluster center from the data points with the probability of x being proportional to (D(x))2
4. We then repeat steps 2 and 3 until *k* clusters have been chosen

How K-means Work (interactive example)

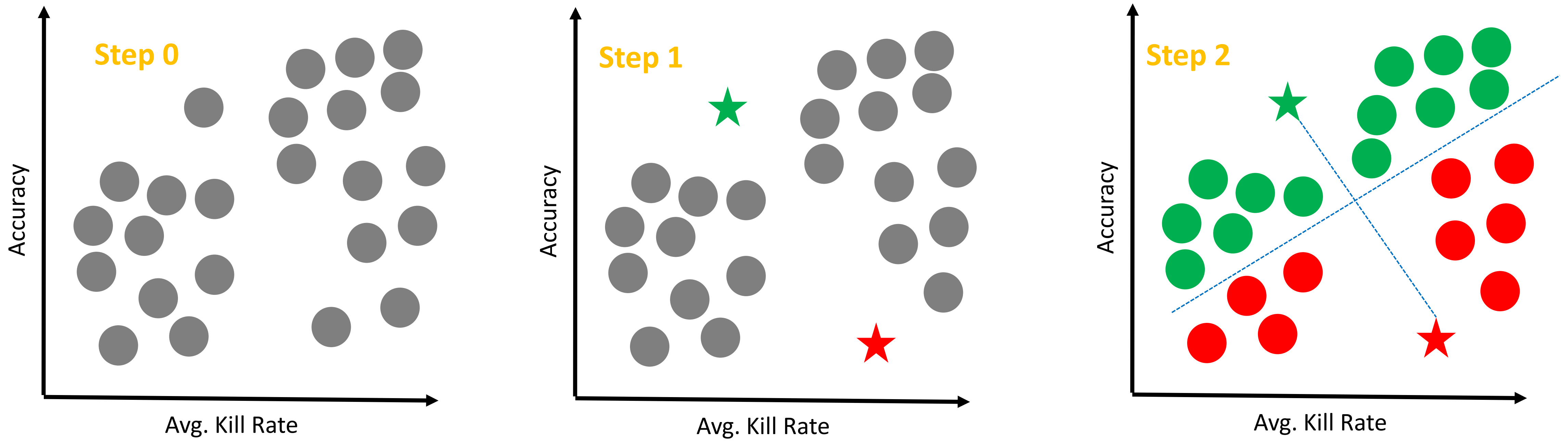
When you are playing **PUB-G**, you will be facing extremely remarkable players at random even if you are an novice player . There is a huge gap in skill level when starting out. Now, what if an algorithm could help cluster the players based on skill levels ?
Let's understand the concept by : **Average kills per game** vs **Accuracy** plot and split the players into **2 classes** : **Amateur** and **Skilled**



- Here, we can observe players having high accuracy and high kill rates (Cluster 1) must be skilled players &
- Players with low accuracy and low kills (Custer 2) must be novice or amateur.

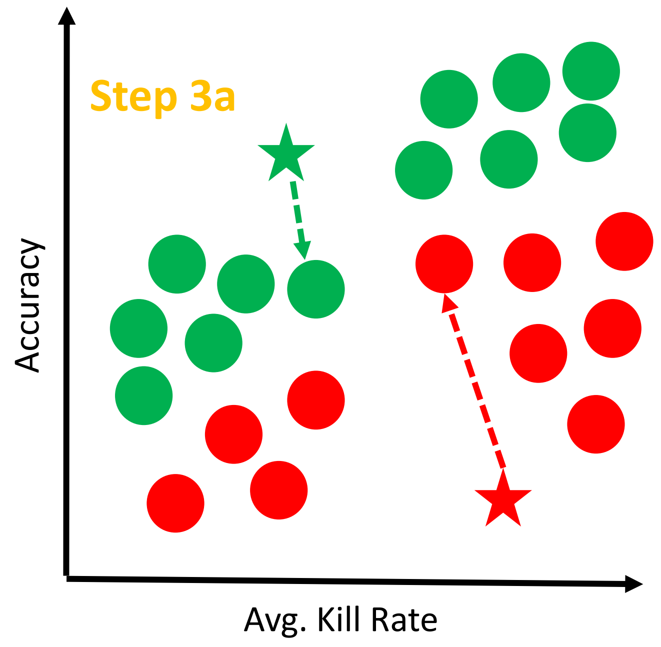
Can we determine this using K-means algorithm ? Let's see in the next slide…

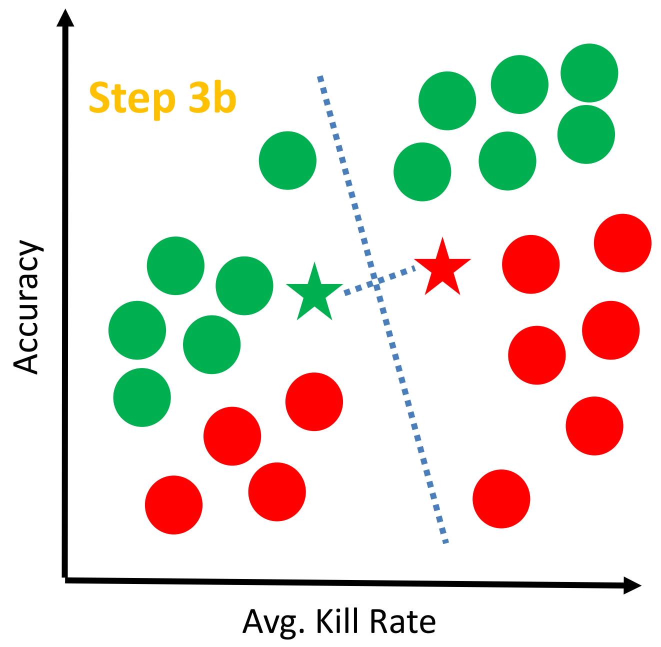
K in K-means denotes number of clusters.
We use K=2 here for amateur and skilled classes.

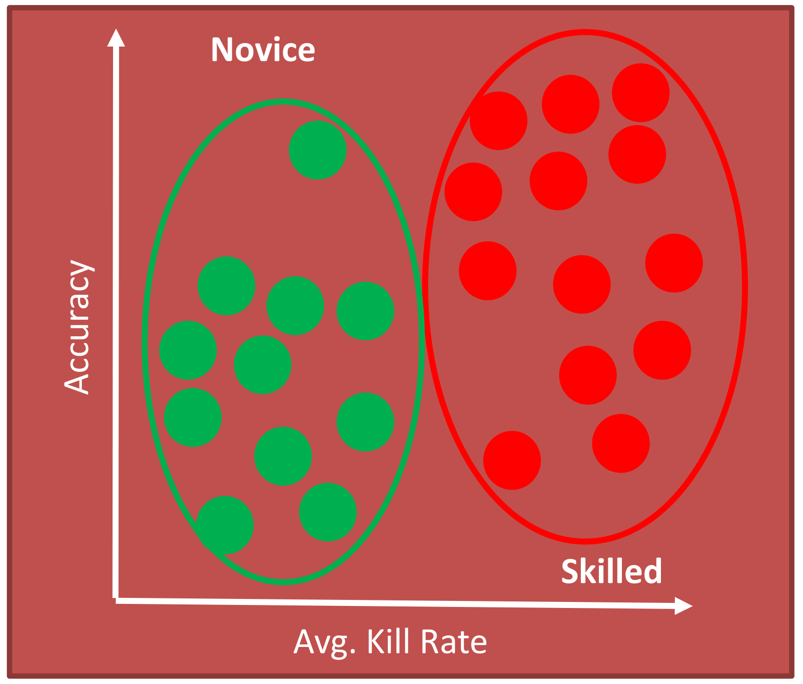


<u>Step 1</u>: Select at random K points, the centroids. Here, there are 2 clusters so we would plot 2 centroids randomly (denoted by red and green stars) as randomly selected data points. (Note : It is not necessary that we have to choose the centroids from the dataset)

<u>Step 2:</u> Assign each data point to the closest centroid. This will help us form clusters. An easy way is to draw a perpendicular line at mid point between red and the green cluster centers joining the centroids. The points that are lying on this perpendicular line would be equidistant from both the centroids.
The points that lie above the perpendicular line are those points which are closer to the green centroid and thus, these points should be colored as **green** and similarly for **red** cluster







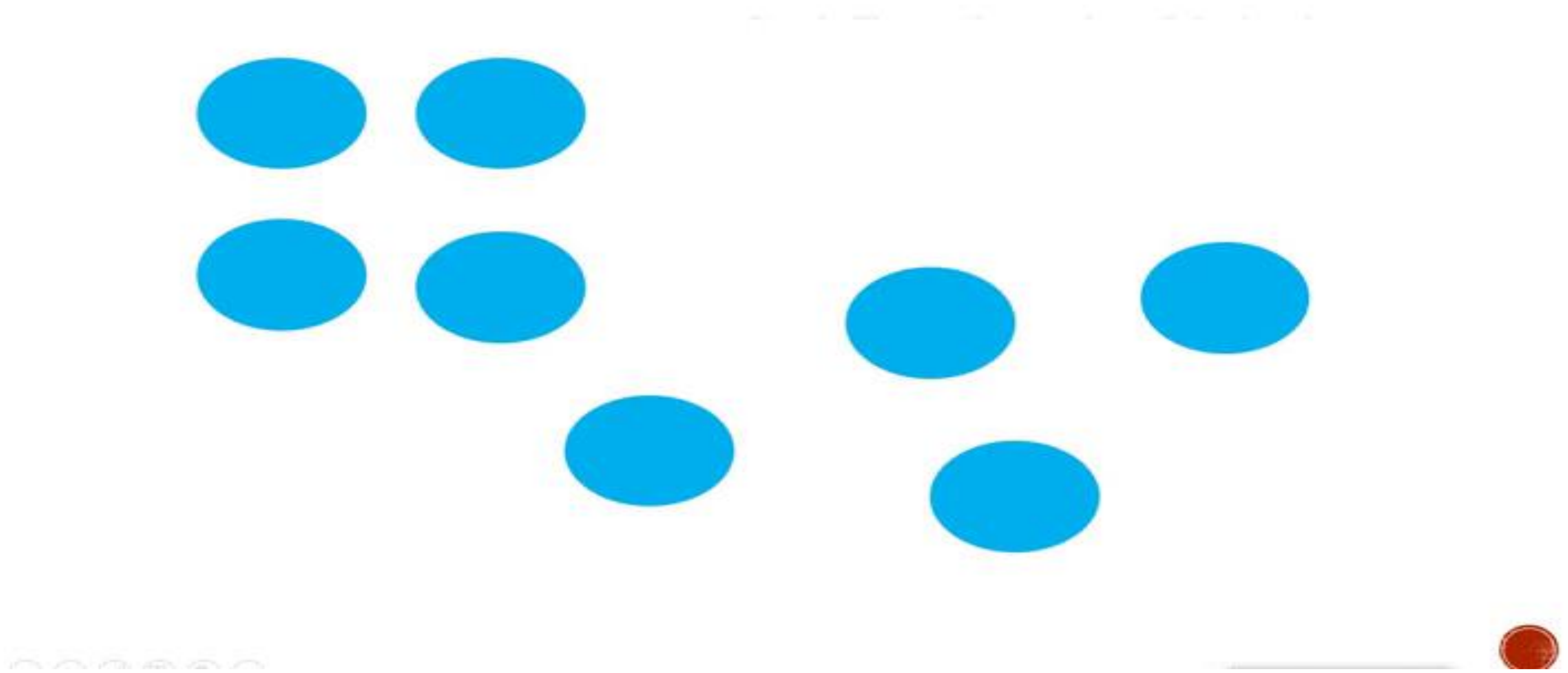
Step 3 a): Once we have assigned each data point to the clusters, we are going to compute the new centroids for each cluster and then going to shift the centroids corresponding to each cluster.

Step 3 b): Assign each data point to the closest centroid by drawing a perpendicular line. Here, we see that after shifting the centroid there are few green points that has now come under the area of the red centroid and vice versa. So, now, we are going to assign the new colors to such data points.

Step 4 : Repeat Steps 3a-3b until convergence of the clusters formed that are inseparable post all iterations
(Novice and Skilled players are now shown as 2 different clusters)

How k-means actually work?(Animated Video)

Note - Play in slideshow mode

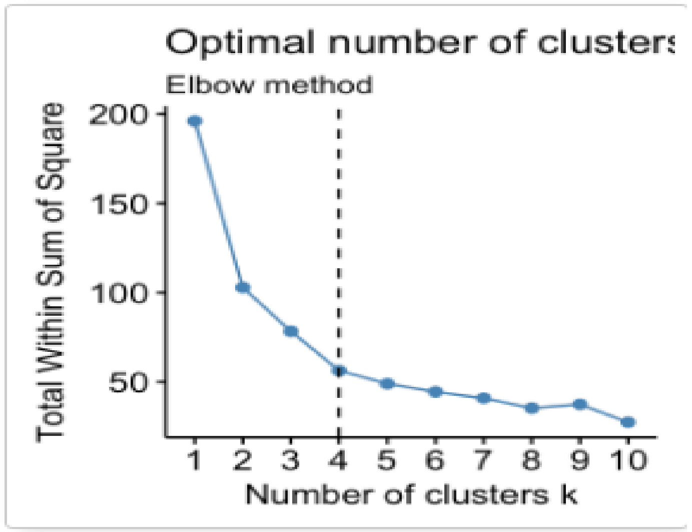


Finding the Number of cluster in K means

1. Elbow method
2. Silhouette method:
3. Gap statistic method

Elbow method:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (wss).
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters



*Calculate the **Within-Cluster-Sum of Squared** Errors (WSS) for **different values of k**, and choose the k for which WSS becomes first starts to diminish. In the plot of WSS-versus-k, this is visible as an **elbow.***

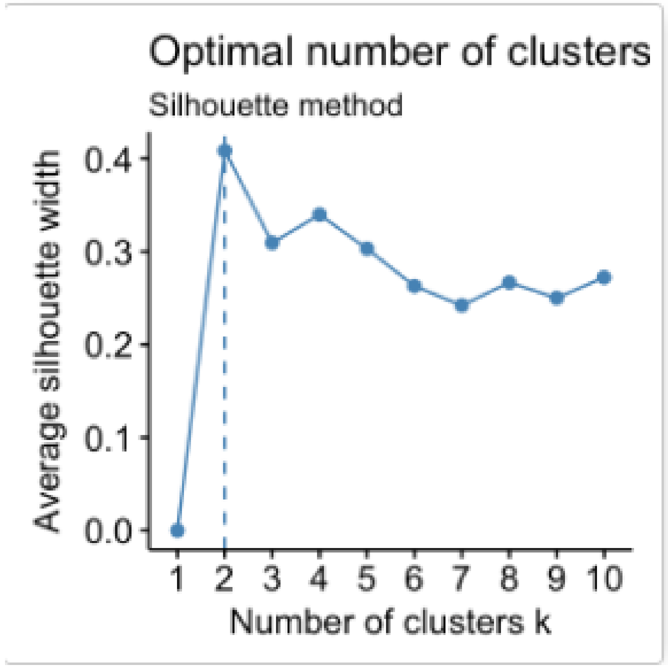
The Squared Error for each point is the square of the distance of the point from its representation i.e. its predicted cluster center.
The **WSS** score is the sum of these Squared Errors for all the points.
Any distance metric like the Euclidean Distance or the Manhattan Distance can be used.

Finding the Number of cluster in K means

Silhouette method : *The silhouette value measures how similar a point is to its own cluster (cohesion) compared to other clusters (separation).*

The range of the Silhouette value is between +1 and -1. A **high value is desirable** and indicates that the point is placed in the correct cluster. If many points have a negative Silhouette value, it may indicate that we have created too many or too few clusters.

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the average silhouette of observations (*avg.sil*).
3. Plot the curve of *avg.sil* according to the number of clusters k.
4. The location of the maximum is considered as the appropriate number of clusters



The Silhouette Value ***s(i)*** for each data point ***i*** is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

and

$$s(i) = 0, \text{ if } |C_i| = 1$$

Source: Wikipedia

ai measures similarity (point i to it's own cluster) ,
bi measures dissimilarity of i from points in other clusters

Summary of finding optimal value of K
(*How do you decide the number of clusters***)**

The Elbow Method is more of a decision rule, while the Silhouette is a metric used for validation while clustering. Thus, it can be used in combination with the Elbow Method.

Therefore, the **Elbow Method** and the **Silhouette Method** are not alternatives to each other for finding the optimal K. Rather they are tools to be used together for a more confident decision.

Comments on the *K-Means* Method

<u>Strength</u>

Relatively efficient: $O(tkn)$, where n is # objects, k is # clusters, and t is # iterations. Normally, k, $t << n$.

Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*

<u>Weakness</u>

Applicable only when *mean* is defined, then what about categorical data?

Need to specify *k,* the *number* of clusters, in advance

Unable to handle noisy data and *outliers*

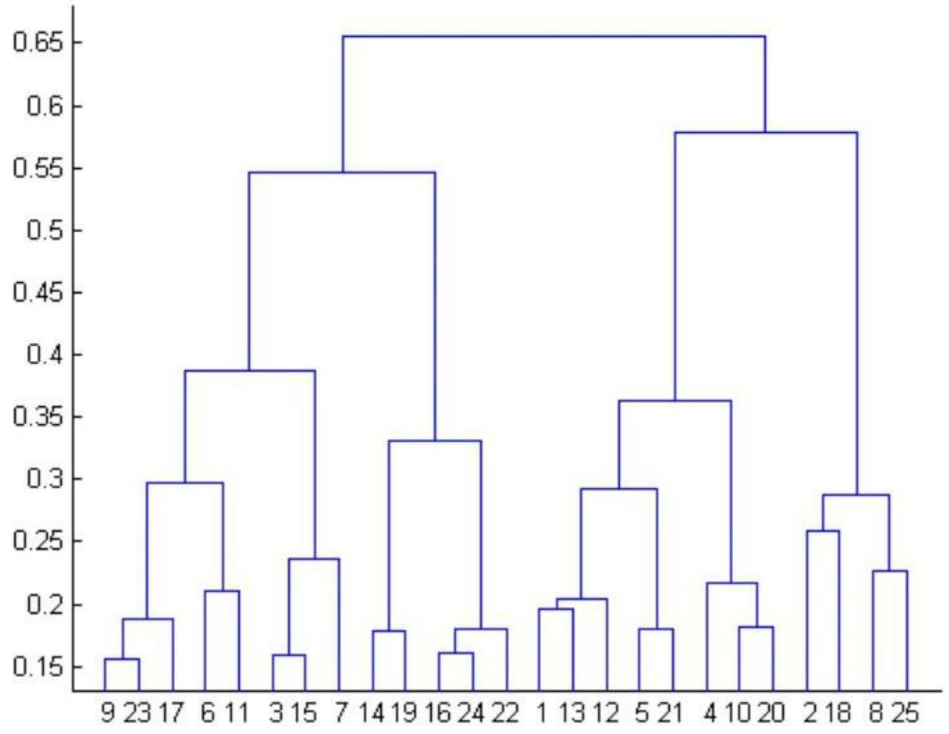
Not suitable to discover clusters with *non-convex shapes*

Hierarchical Clustering

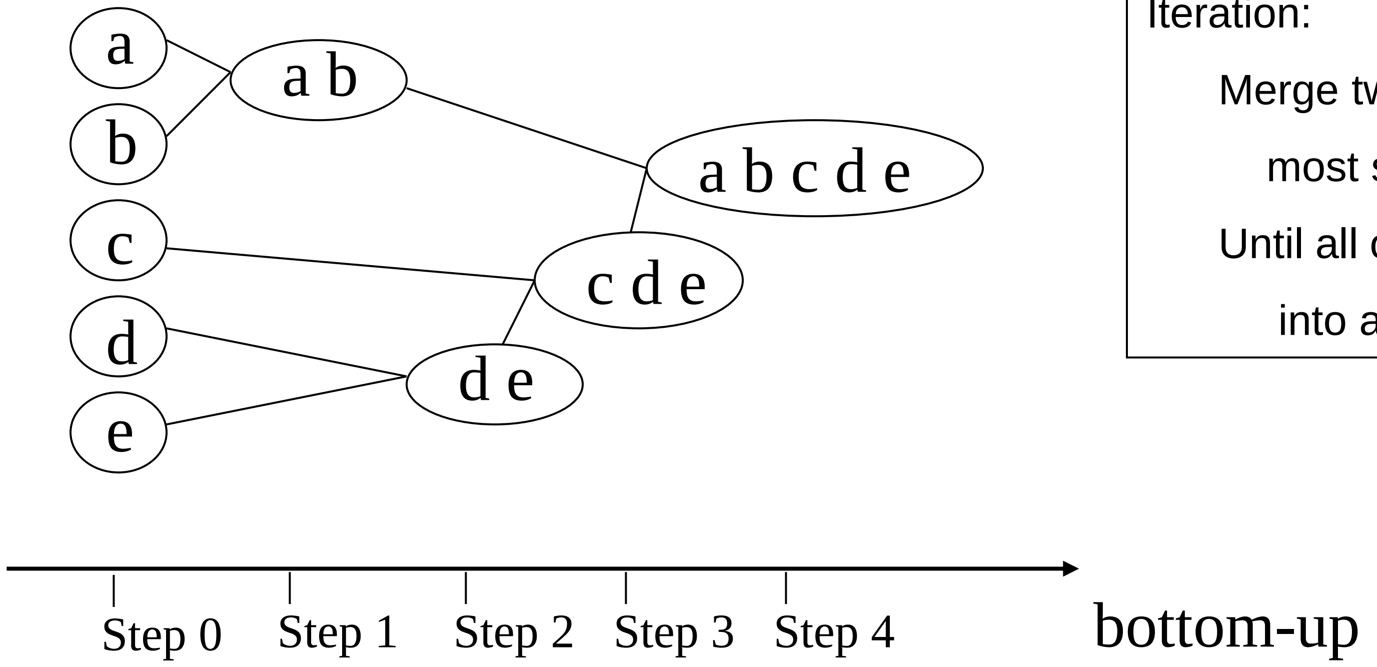
- Hierarchical clustering involves creating clusters that have a predetermined ordering from top to bottom.
- There are two types of hierarchical clustering –
  - ❑Agglomerative
  - ❑Divisive
- Hierarchical clustering is mostly used when the application requires a hierarchy, e.g creation of a taxonomy.
- However, they are expensive in terms of their computational and storage requirements.

Hierarchical Clustering

- The results of hierarchical clustering can be shown using **dendrogram**.



Agglomerative approach



Initialization:

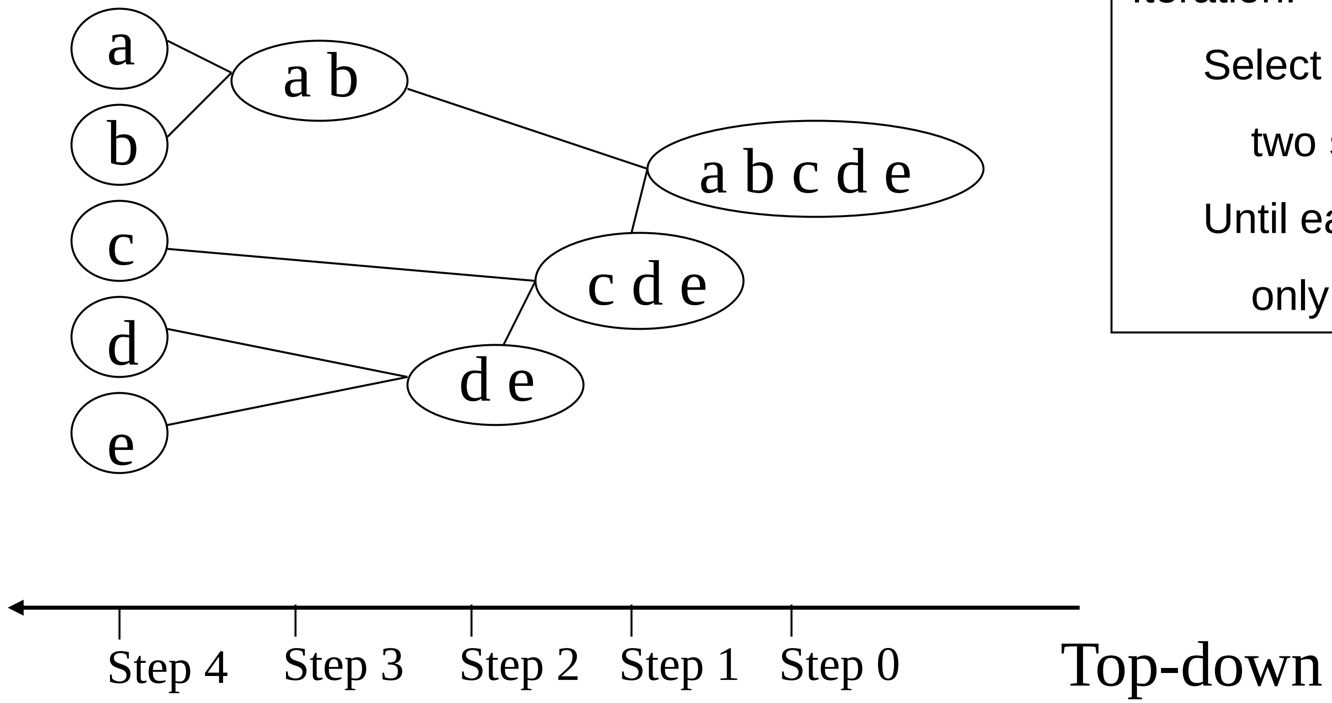
Each object is a cluster

Iteration:

Merge two clusters which are most similar to each other;

Until all objects are merged into a single cluster

Divisive Approaches



Initialization:

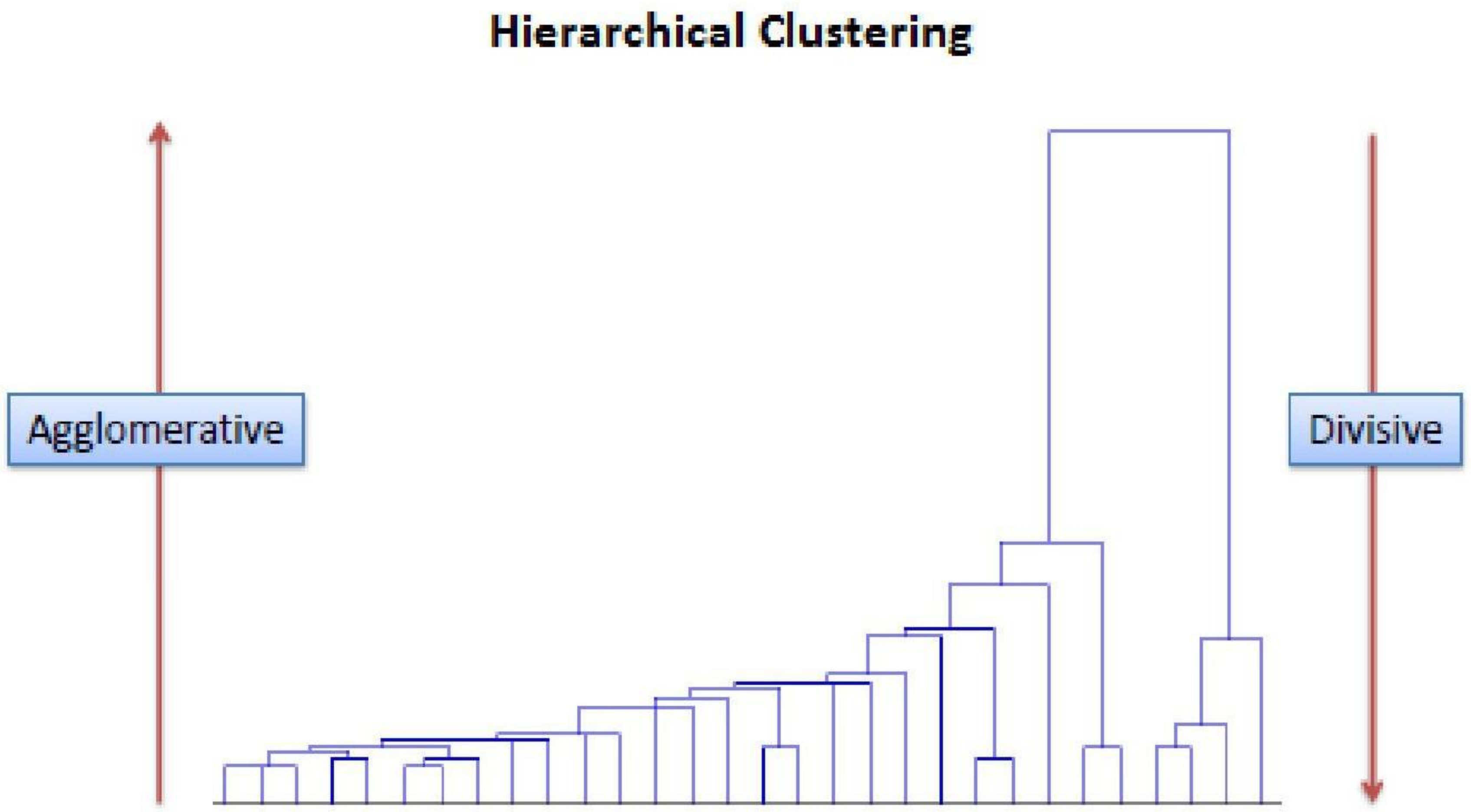
All objects stay in one cluster

Iteration:

Select a cluster and split it into two sub clusters

Until each leaf cluster contains only one object

Hierarchical Clustering



Intuitive understanding of Agglomerative Hierarchical clustering

Suppose a teacher wants to divide her students into different groups. She has the marks scored by each student in an assignment and based on these marks, she wants to segment them into groups. There's no fixed target here as to how many groups to have. Since the teacher does not know what type of students should be assigned to which group, it cannot be solved as a supervised learning problem. So, we will try to apply hierarchical clustering here and segment the students into different groups.
Sample of 5 students :

Student ID	Marks
1	10
2	7
3	28
4	20
5	25

First, we will create a proximity matrix which tells us the distance between each of these points (Student ID's given). Since we are calculating the distance of each point from each of the other points, we will get a square matrix of shape n X n (where n is the number of observations).
Let's make the 5 x 5 proximity matrix for our example: (How did we create this ?)
The **diagonal elements of this matrix will always be 0** as the distance of a point with itself is always 0. We will use the "Euclidean distance formula" to calculate the rest of the distances. So, let's say we want to calculate the distance between point 1 and 2 (For student ID: 1 the value is 10 and for student ID : 2 the value is 7 ("refer to the previous slide for the table) √(10-7)^2 = √9 = 3 (hence: you see below for ID 1 and column 2 we have the distance to be 3) and so on for all the values , they are filled using the Euclidean distance formula.

Original Table

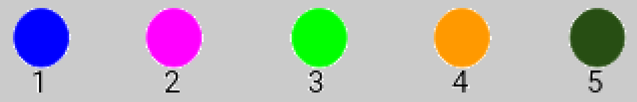
Student ID	Marks
1	10
2	7
3	28
4	20
5	25

Proximity Table

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

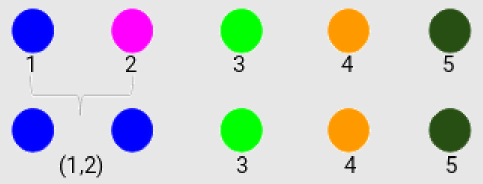
Steps to perform agglomerative hierarchical clustering

Step 1: First, we assign all the points to an individual cluster. 5 clusters for 5 data points



Step 2: Next, we will look at the smallest distance in the proximity matrix and merge the points with the smallest distance. We then update the proximity matrix:

Here, the smallest distance is 3 and hence we will merge point 1 and 2



Proximity Table :

ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Step 3: Let's look at the updated original table and updated proximity table. Here : Instead of the maximum, we can also take the minimum value or the average values as well. Now, we will again calculate the proximity matrix for these clusters

Original Table :

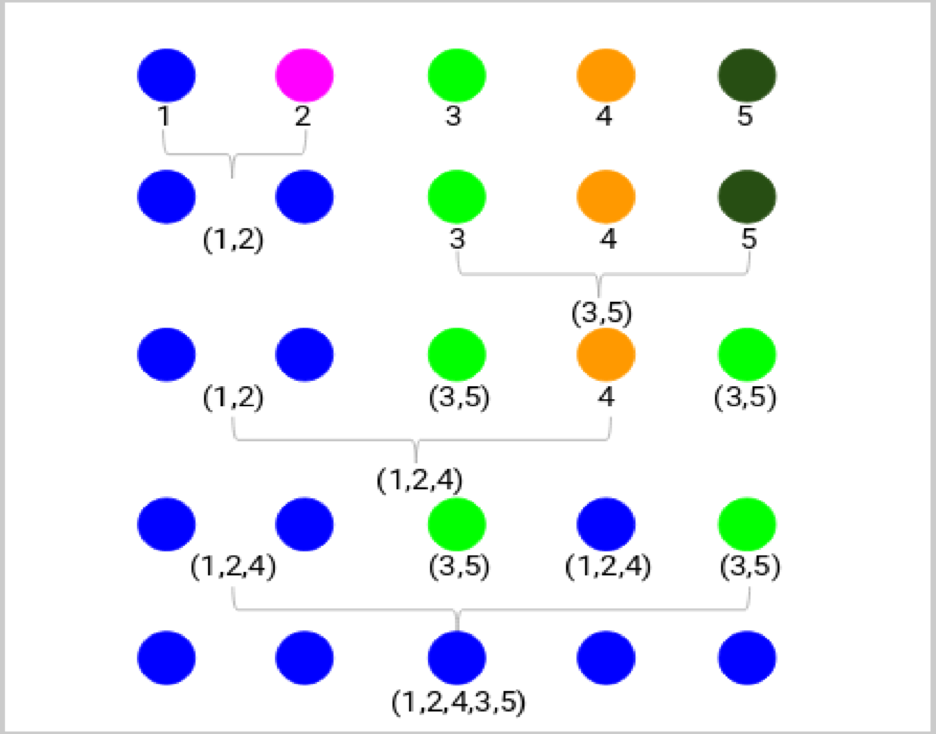
Student ID	Marks
(1,2)	10 Max(10,7)
3	28
4	20
5	25

Proximity Table :

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

Step 4 : Repeat steps 2 and 3 until only a single cluster is left .

We started with 5 clusters and finally have a single cluster. This is how **agglomerative hierarchical clustering** works.

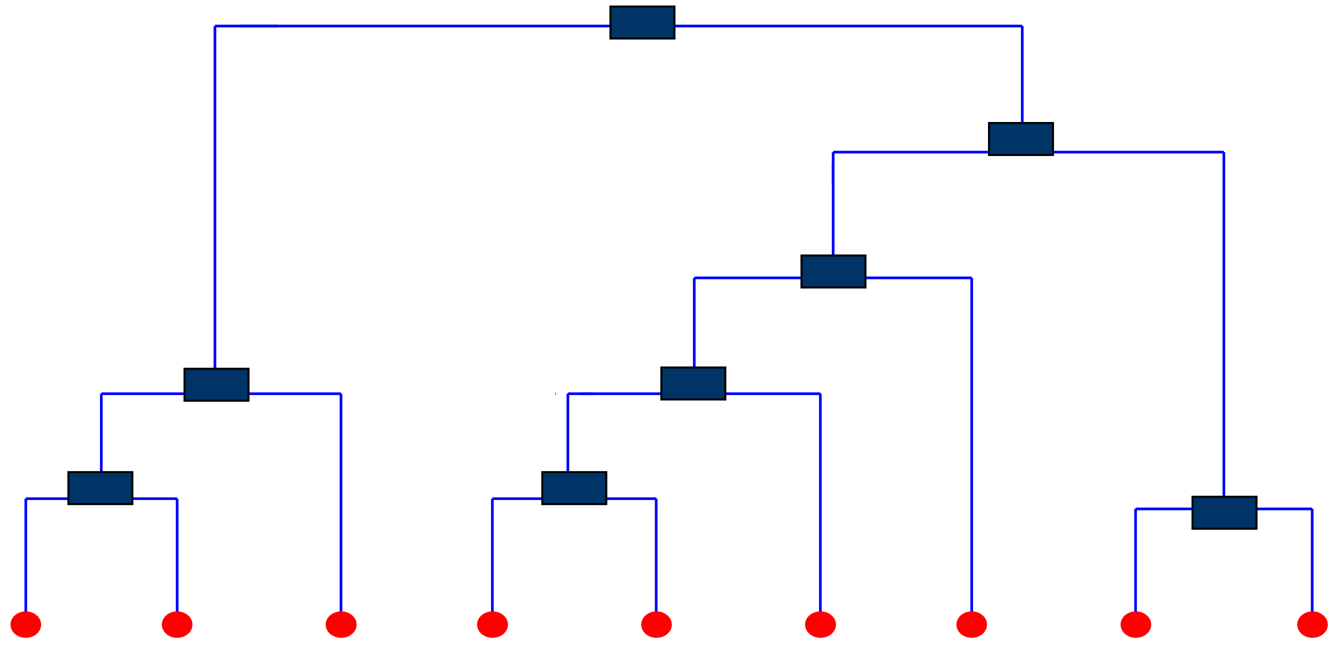


Dendrogram

Dendrogram (A *tree graph).* A graphical device for displaying clustering results.

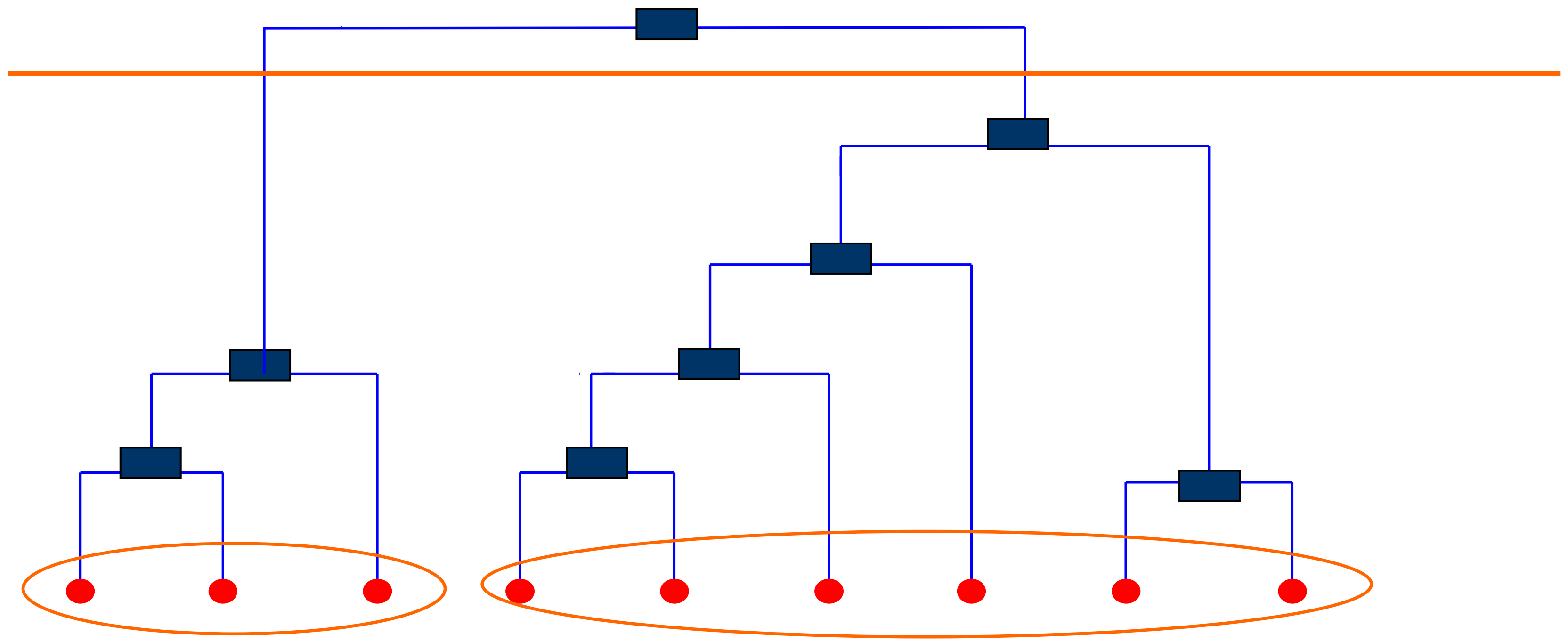
-Vertical lines represent clusters that are joined together.

-The position of the line on the scale indicates distances at which clusters were joined.

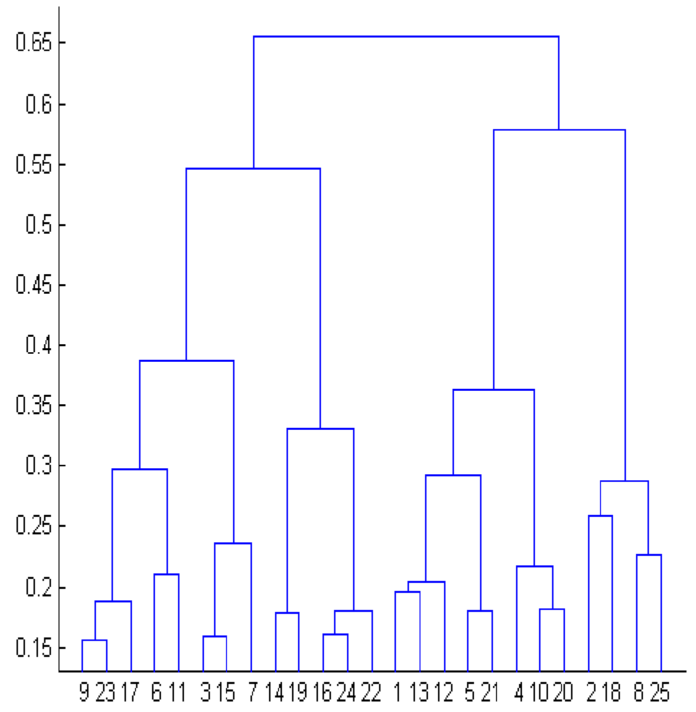
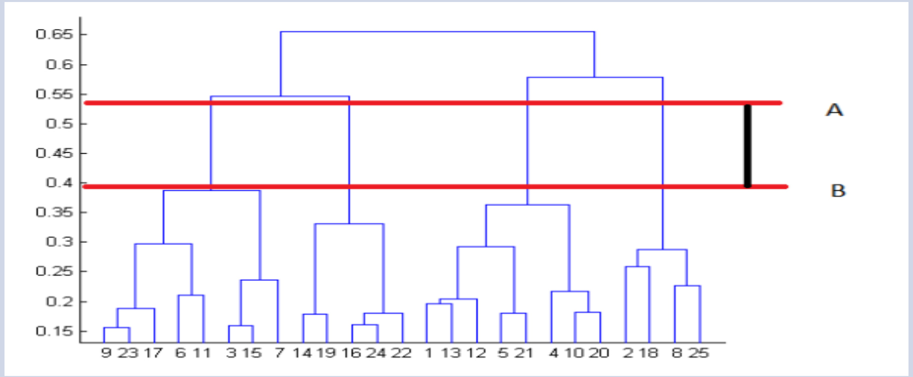


Dendrogram

A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster

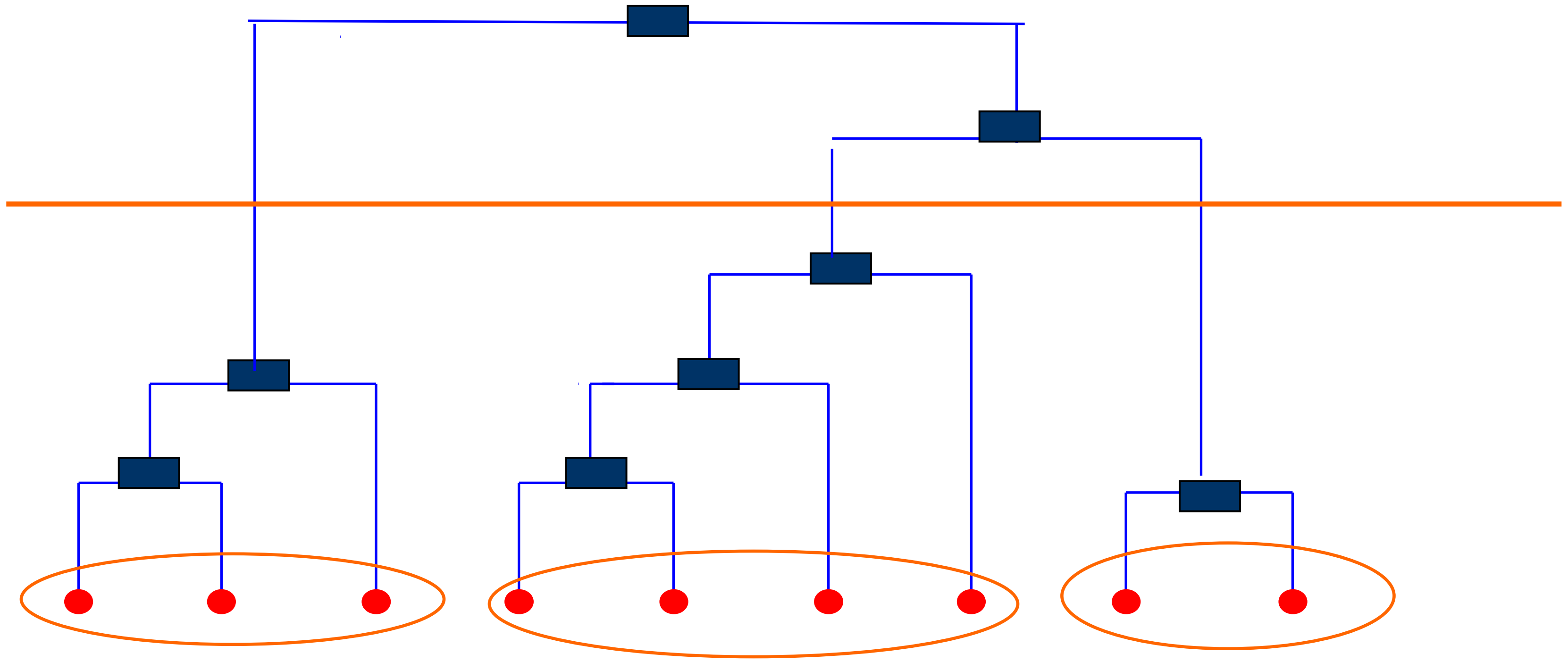


Example

Question	Answer
What is the most appropriate no. of clusters for the data points represented by the following dendrogram: 	The decision of the no. of clusters that can best depict different groups can be chosen by observing the Dendrogram. **The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold.** In the example, since the red line intersects 4 vertical lines, we will have 4 clusters. **Hence : Answer is 4** as the red horizontal line in the dendrogram below covers maximum vertical distance AB. 

Dendrogram

A clustering of the data objects is obtained by cutting the *dendrogram* at the desired level, then each connected component forms a cluster



Agglomerative clustering algorithm Summary

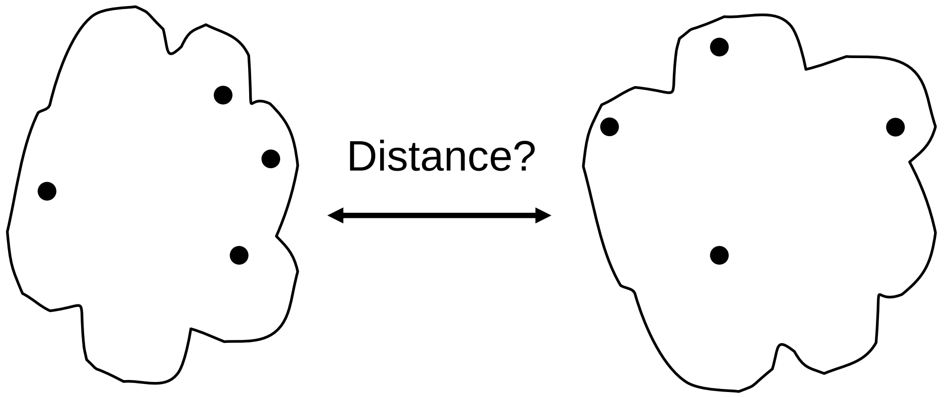
Most popular hierarchical clustering technique

Basic algorithm

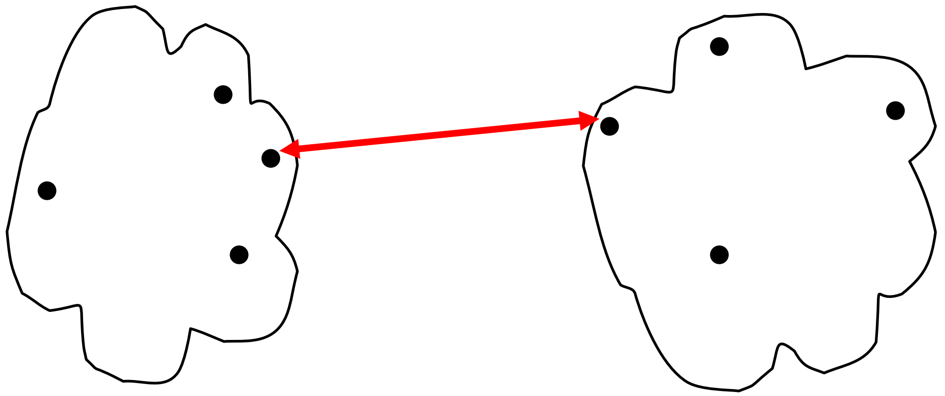
```
1. Compute the distance matrix between the input data points
2. Let each data point be a cluster
3. Repeat
4.         Merge the two closest clusters
5.         Update the distance matrix
6. Until only a single cluster remains
```

How to measure the distance between clusters?

1. Single-link
2. Complete-link
3. Average-link
4. Centroid distance



Single-Link



Single-link

Complete-link

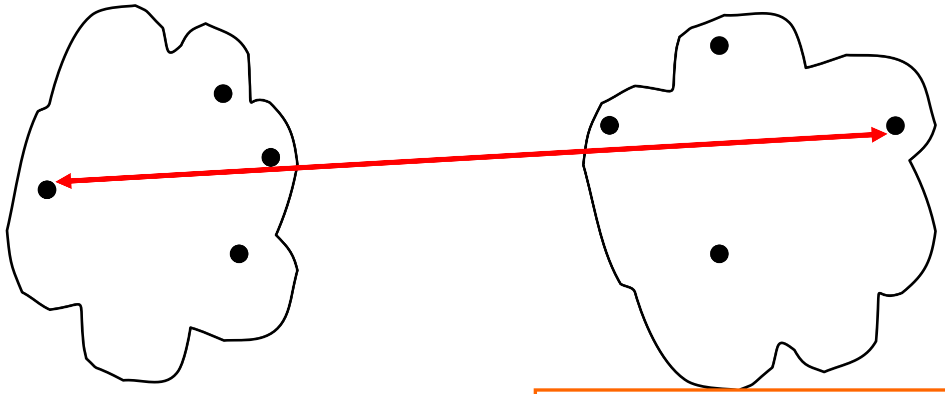
Average-link

Centroid distance

$$\boxed{d_{\min}(C_i, C_j) = \min_{p \in C_i, q \in C_j} d(p, q)}$$

The distance between two clusters is represented by the distance of the *closest pair of data objects* belonging to different clusters.

Complete-Link



Single-link

Complete-link

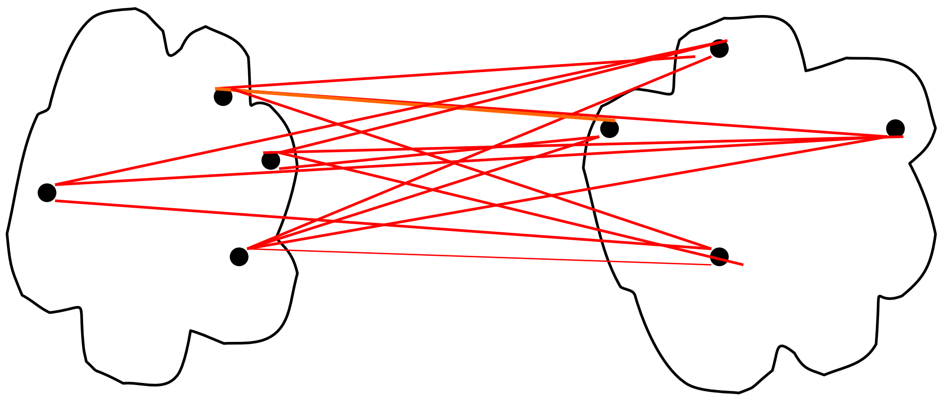
Average-link

Centroid distance

$$d_{\min}(C_i, C_j) = \max_{p \in C_i, q \in C_j} d(p, q)$$

The distance between two clusters is represented by the distance of the *farthest pair of data objects* belonging to different clusters.

Average Link



Single-link

Complete-link

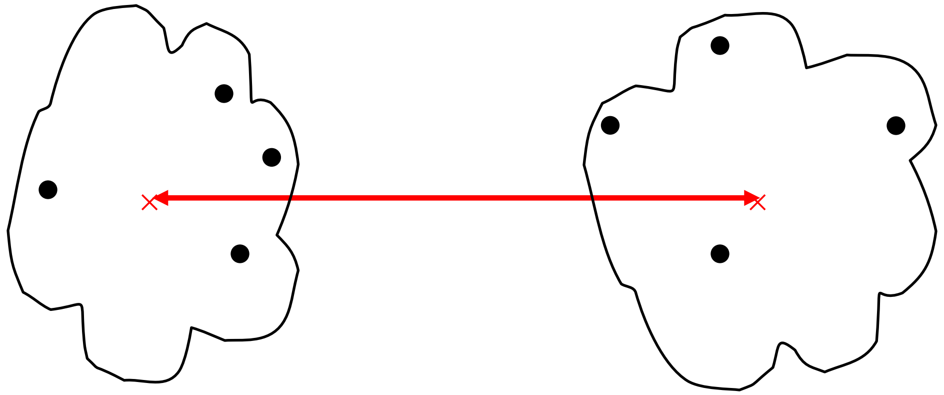
Average-link

Centroid distance

$$\boxed{d_{\min}(C_i, C_j) = \underset{p \in C_i, q \in C_j}{avg} d(p,q)}$$

The distance between two clusters is represented by the *average* distance of *all pairs of data objects* belonging to different clusters.

Centroid Distance



m_i, m_j are the means of C_i, C_j,

Single-link

Complete-link

Average-link

Centroid distance

$$\boxed{d_{mean}(C_i, C_j) = d(m_i, m_j)}$$

The distance between two clusters is represented by the distance between *the means of the cluters.*

Which Distance Measure is Better

- Each method has both advantages and disadvantages; application-dependent, single-link and complete-link are the most common methods
- **Single-link**
  - Can find irregular-shaped clusters
  - Sensitive to outliers, suffers the so-called chaining effects
- **Complete-link, Average-link, and Centroid distance**
  - Robust to outliers
  - Tend to break large clusters
  - Prefer spherical clusters