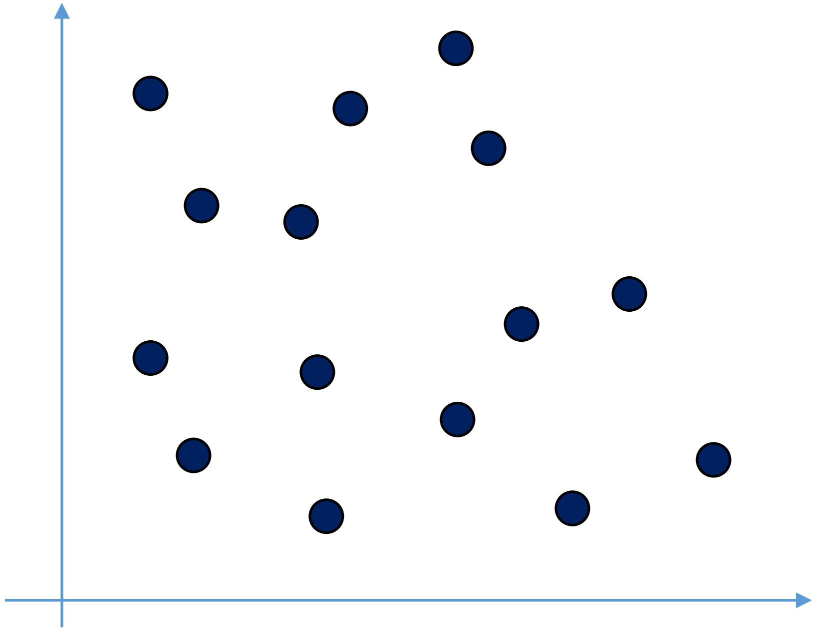
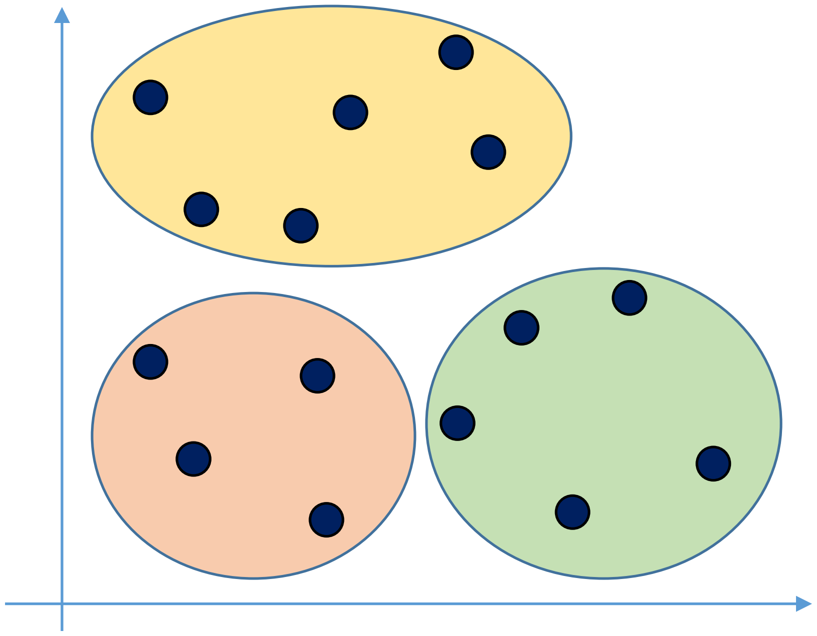
k-Means Clustering

k-Means

- Widely used in classification of data based on the Centroid-based clustering
- A black-box algorithm
- Algorithm breaks the dataset into 'k' different clusters
- Number of clusters to be broken into is specified by the user (Eg. **k=3** breaks dataset into 3 clusters)
- Number of clusters has to be known beforehand



Before clustering



After clustering

k-means algorithm

1. Identify the number of clusters (**k**=*n*) [n >= 2]

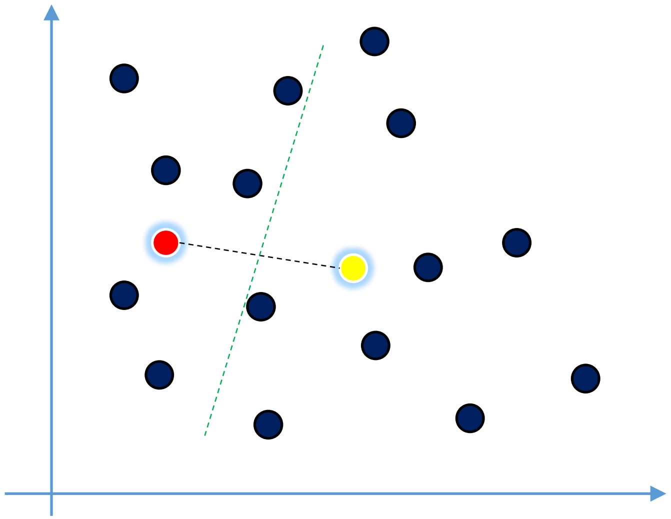
2. Algorithm assigns <k> random values as Centroid values - one for each cluster

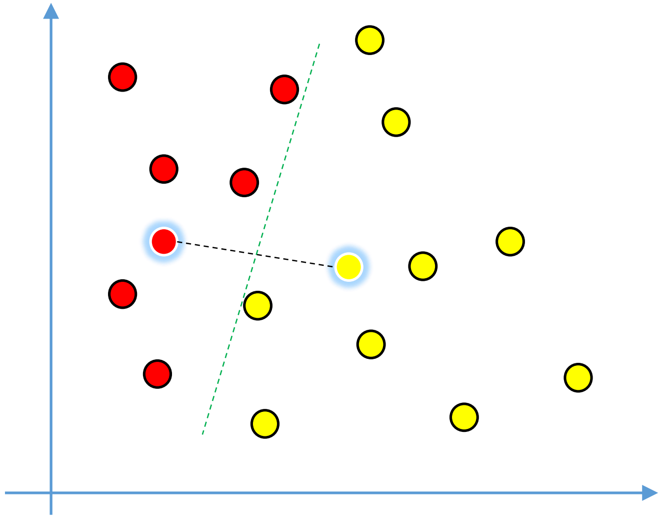
3. Assign every record (observation) to the nearest centroid
   - forms **k**-clusters, each having **n** observations

4. Compute new centroids for each cluster

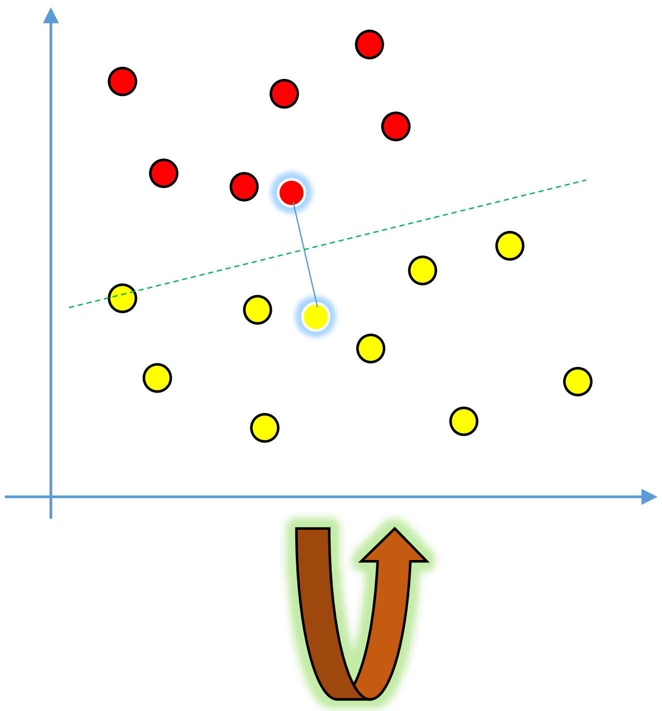
5. Reassign record to the new centroid **(step 3)** and repeat process 4 till no new assignments

6. Build the Model

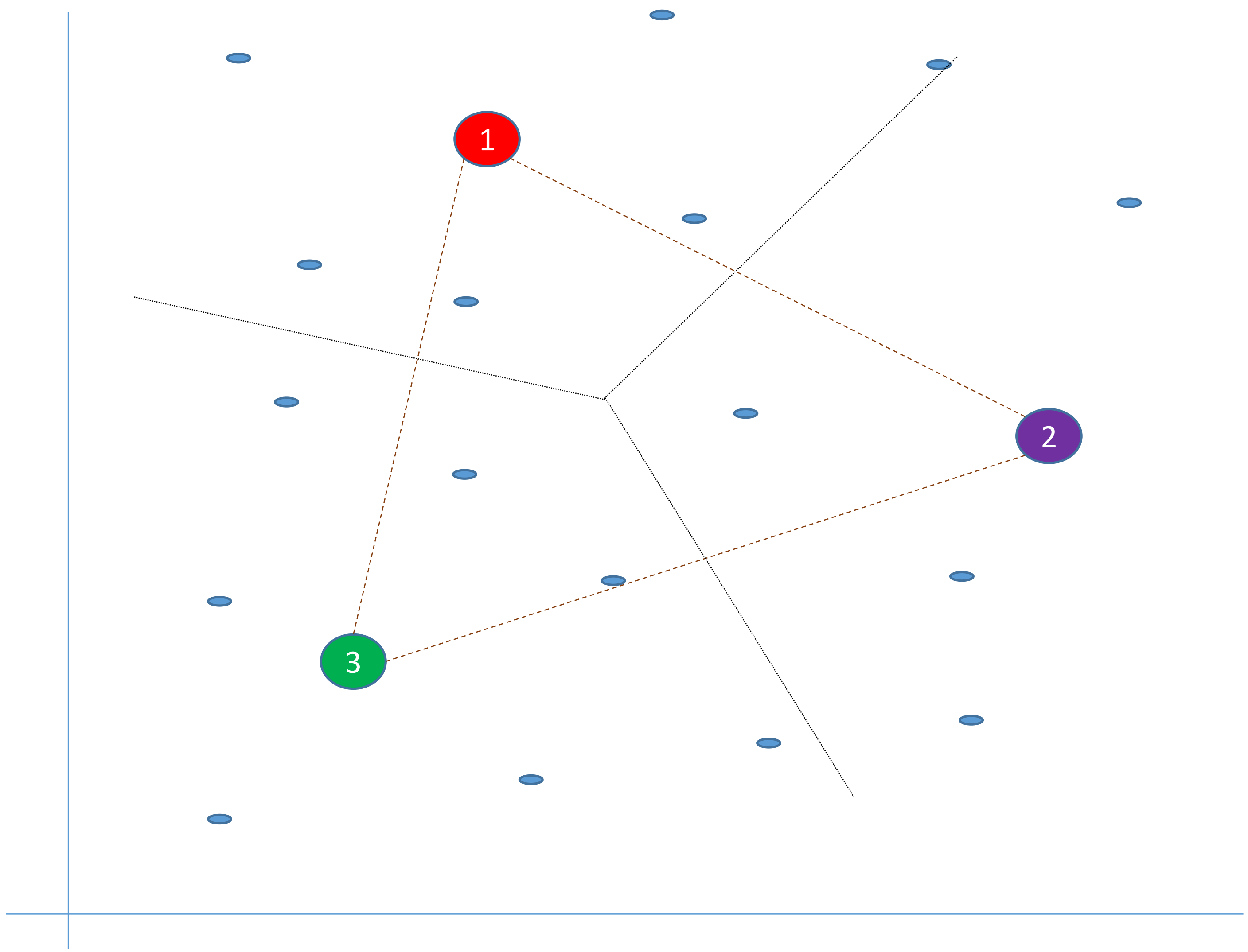




- Algorithm assigns <k> random values in the dataset
- Other records are assigned to one of these seeds based on their proximity to the seeds
  - ❑ join 2 seeds at a time; draw a perpendicular bisector
  - ❑ Every point on the perpendicular bisector is equidistant from the 2 clusters
  - ❑ Points to the left of the bisector are closer to seed on left and vice versa
  - ❑ Observations are classified according to the "area" in which each of them fall under



Algorithm converges



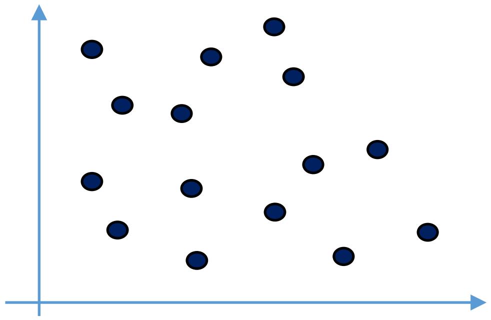
- Algorithm assigns <k> random values in the dataset
- Other records are assigned to one of these seeds based on their proximity to the seeds
  - ❑ join 2 seeds at a time; draw a perpendicular bisector
  - ❑ Every point on the perpendicular bisector is equidistant from the 2 clusters
  - ❑ Points to the left of the bisector are closer to seed on left and vice versa
  - ❑ Observations are classified according to the "area" in which each of them fall under

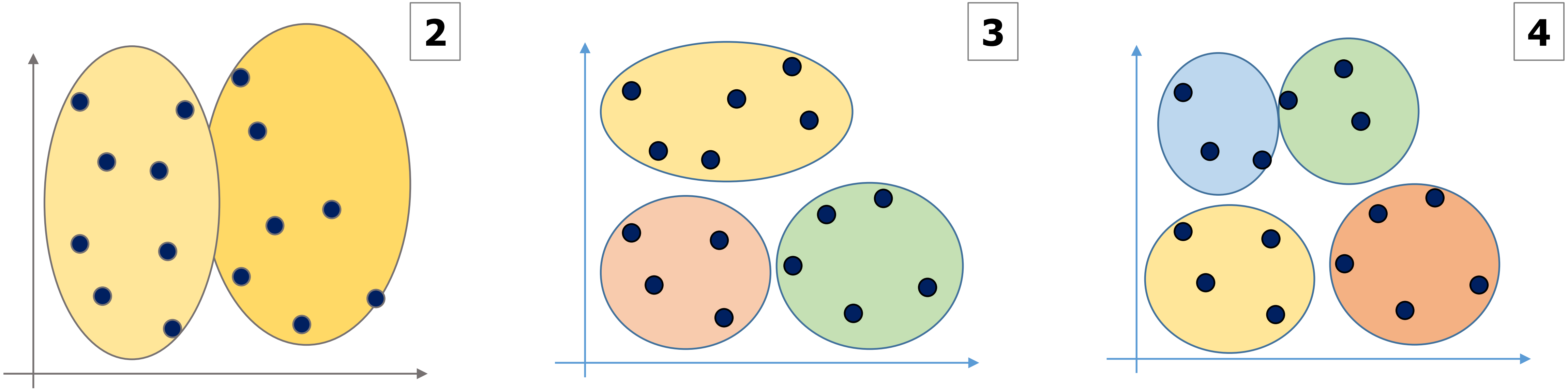
- Identifies centroid of the 3 clusters (by taking average. i.e. moving the red points to a new location)
- Grouping based on minimum distance
- **Repeat process till the algorithm converges to optimum clustering**

Random Initialization trap

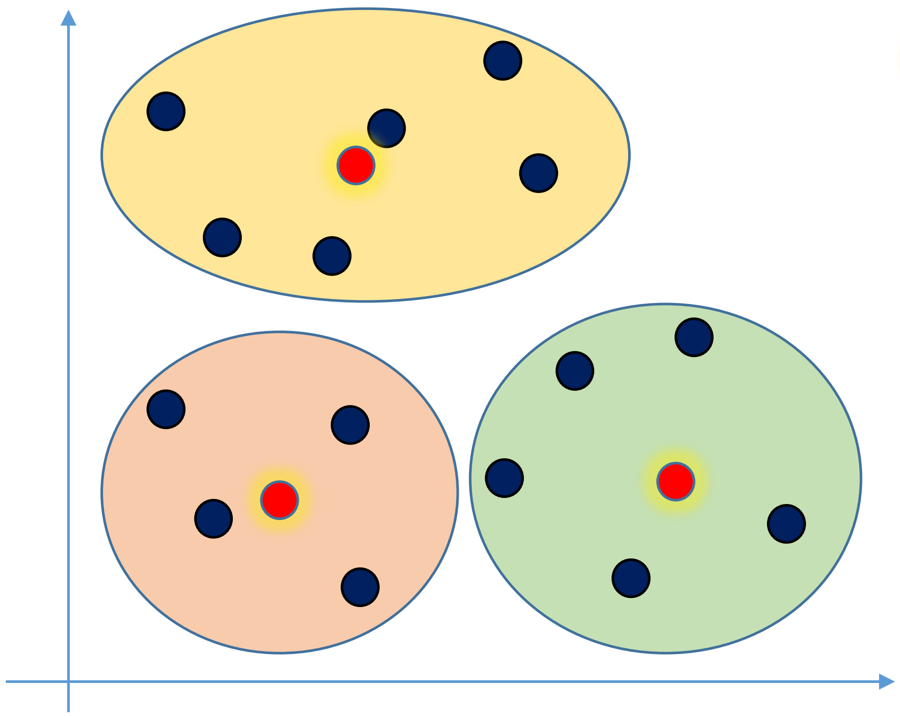
- Random values taken as weights for each ‘k’ cluster
- Observations in the Clusters might change depending upon these random values
  - A cluster **k1** can have more observations
  - A cluster **k1** can have less observations
  - A cluster **k1** having an observation could have moved to another cluster **k2**

Optimum selection of Clusters





Within Cluster Sum of Squares (WCSS)



Element within a cluster (e)

Centroid of cluster (c)

Within Cluster Sum of Squares (WCSS) =

$$\sum_{c} \Sigma_{e_c}\, distance\,(e,c)^2$$

- As the number of clusters increase, Errors decrease

- Optimum cluster is the one that shows <u>less difference</u> in the errors with the previous error component

- Using the Elbow chart, it is easy to determine