CH -10

Unsupervised Learning and Clustering

By:

Arshad Farhad

20177716

Contents

- Supervised vs Unsupervised learning
- Introduction to clustering
- K-means Clustering
- Hierarchical clustering
- Conclusion

Supervised Vs Unsupervised Learning

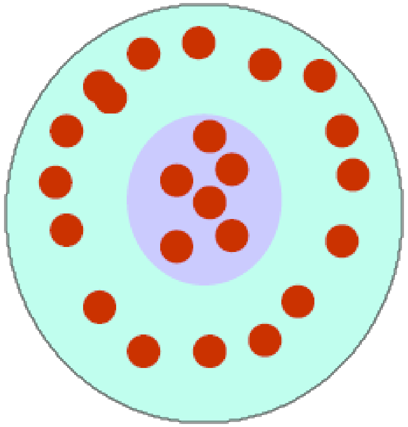
- **Supervised learning** is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

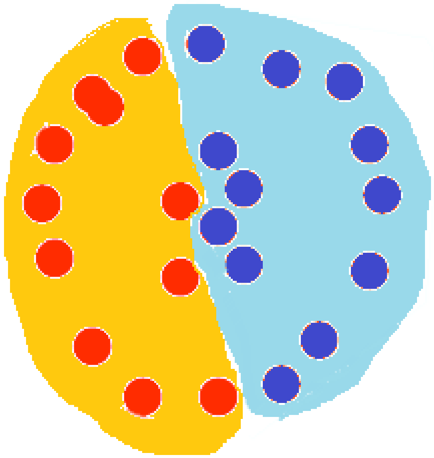
$$Y = f(X)$$

- *The goal* is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data
- **Unsupervised learning** is where you only have input data (X) and no corresponding output variables
- *The goal* for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.
- Unsupervised learning problems can be further grouped into clustering and association problems.
  - **Clustering**
  - **Association**

What is clustering?

- The organization of unlabeled data into similarity groups called clusters.
- A cluster is a collection of data items which are "similar" between them, and "dissimilar" to data items in other clusters.

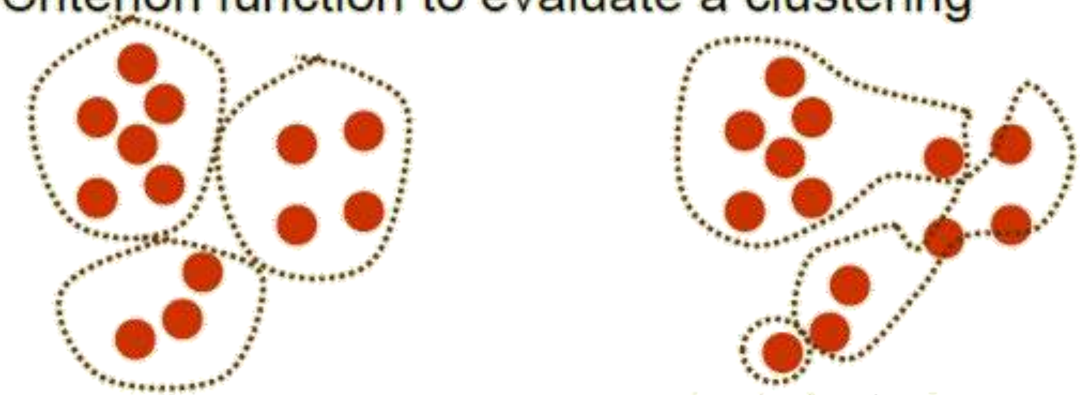




What do we need for clustering?

1. Proximity measure, *either*
   - similarity measure $\boldsymbol{s}(\boldsymbol{x}_i,\boldsymbol{x}_k)$: large if $\boldsymbol{x}_i,\boldsymbol{x}_k$ are similar
   - dissimilarity(or distance) measure $\boldsymbol{d}(\boldsymbol{x}_i,\boldsymbol{x}_k)$: small if $\boldsymbol{x}_i,\boldsymbol{x}_k$ are similar

2. Criterion function to evaluate a clustering



good clustering ***bad clustering***

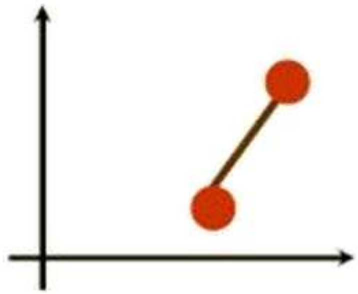
3. Algorithm to compute clustering
   - For example, by optimizing the criterion function

Distance (dissimilarity) measures

- Euclidean distance between points ***i*** and ***j*** is the length of the line segment connecting them
- In Cartesian coordinates, if **i = (i_1, i_2,...i_n)** and **q = (q_1, q_2,...q_n)** then the distance (**d**) from ***i*** to ***j***, or from ***j*** to ***i*** is given by:

- Euclidean distance

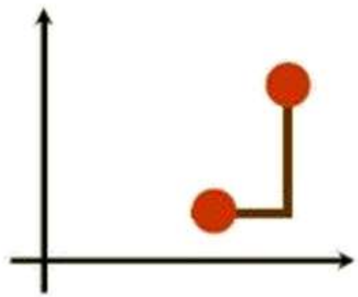
$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left|x_i^{(k)} - x_j^{(k)}\right|$$

  - approximation to Euclidean distance, cheaper to compute



Cluster Evaluation

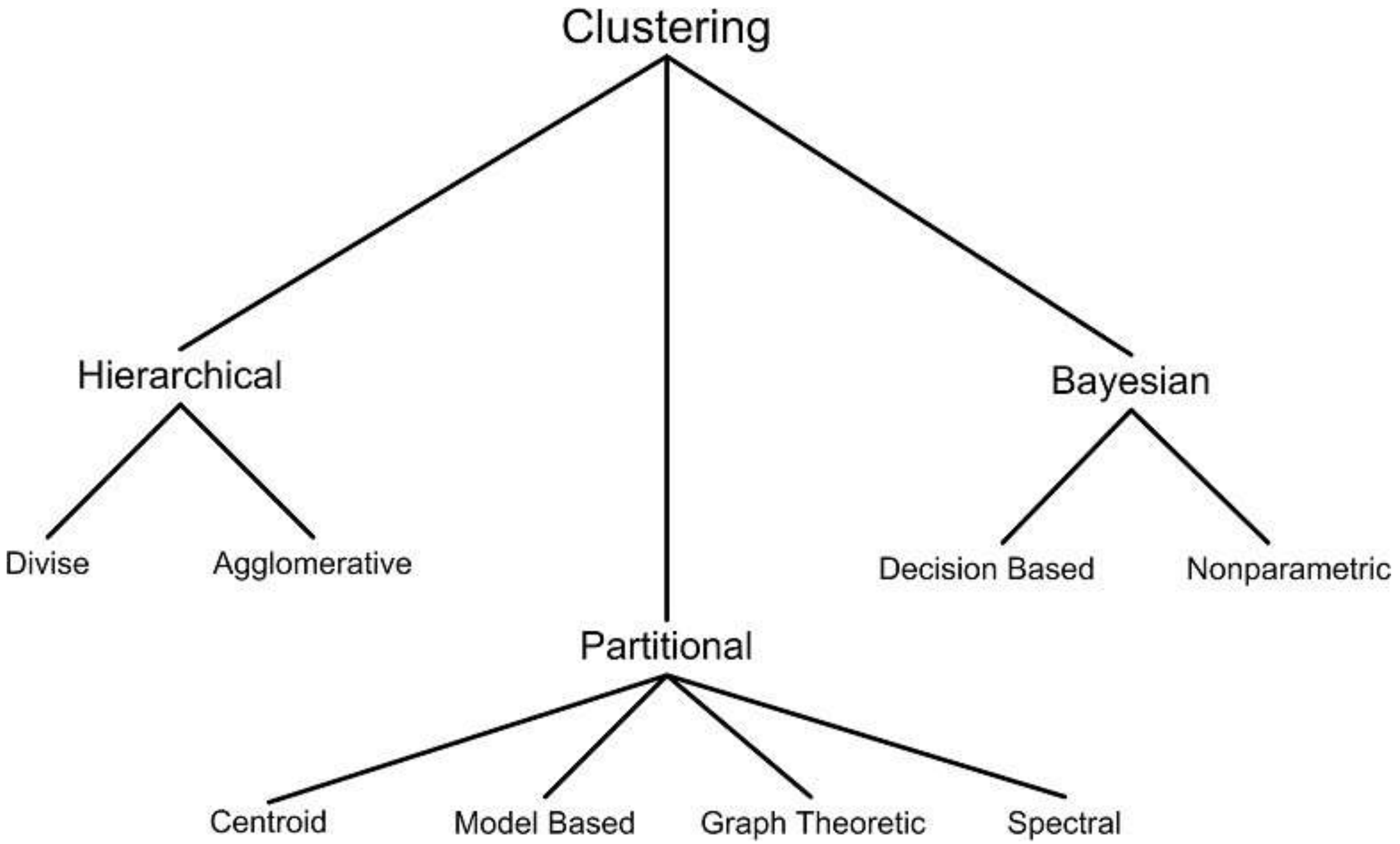
- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- **Inter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far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

How many clusters?

3 clusters or 2 clusters?

- Possible approaches
  1. fix the number of clusters to ***k***
  2. find the best clustering according to the criterion function (number of clusters may vary)

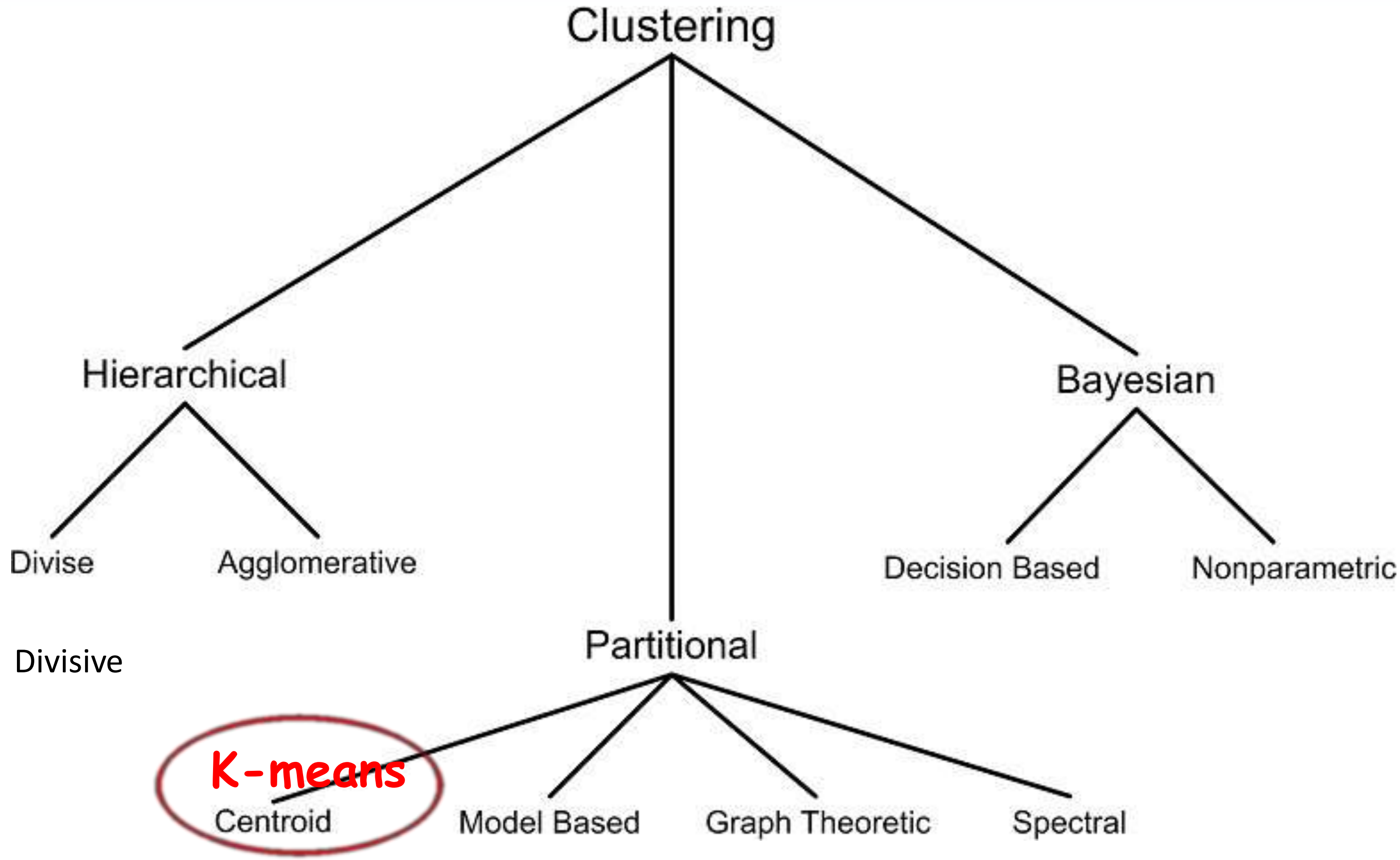
Clustering Techniques



Clustering Techniques

- **Hierarchical** algorithms find successive clusters using previously established clusters. These algorithms can be either agglomerative ("*bottom-up*") or divisive ("*top-down*"):
  1. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters;
  2. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
- **Partitional** algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering.
- **Bayesian** algorithms try to generate a *posteriori distribution* over the collection of all partitions of the data.

Clustering Techniques



K-Means clustering

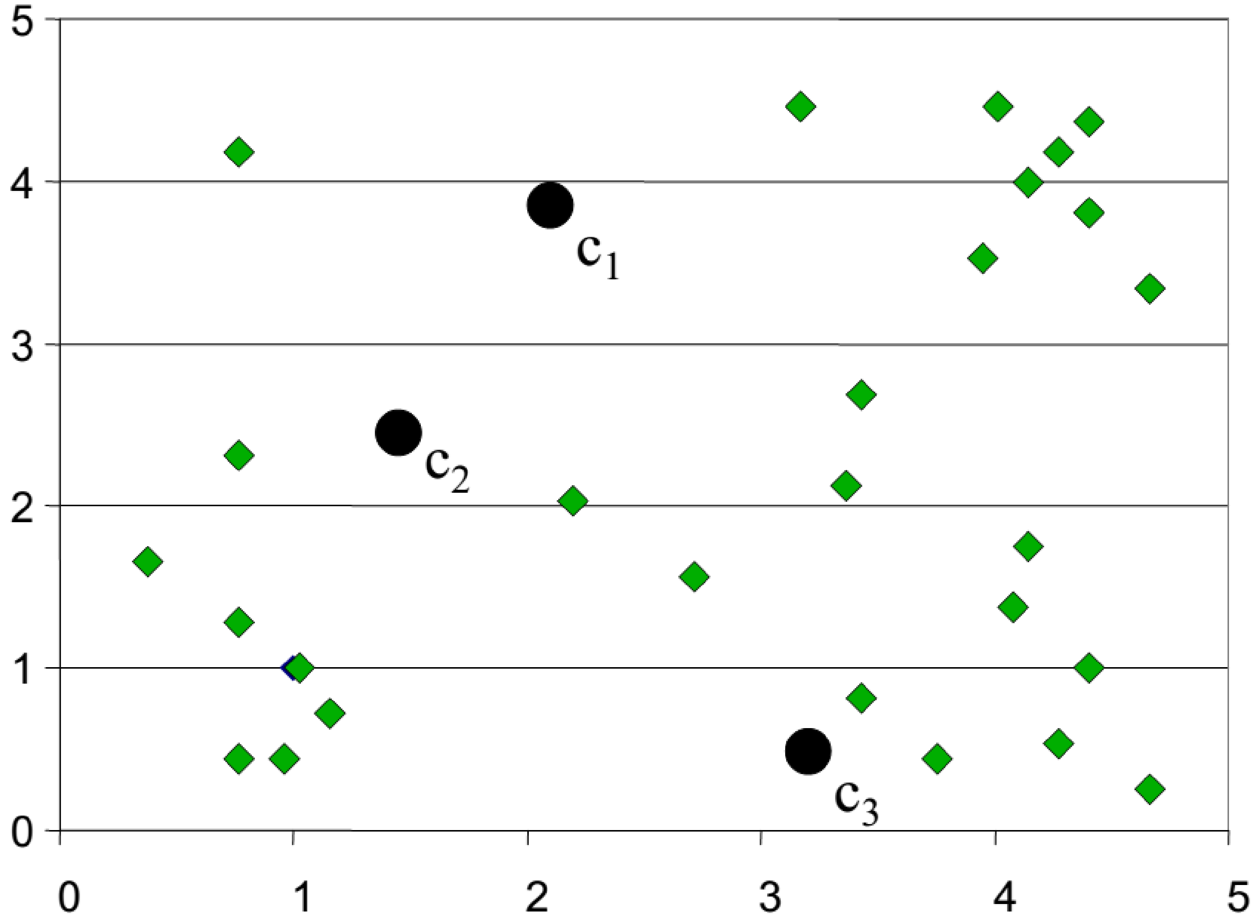
- K-means (MacQueen, 1967) is a partitional clustering algorithm
- The *k*-means algorithm partitions the given data into *k* clusters:
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm

- Given *k*, the *k-means* algorithm works as follows:
  1. Choose *k* (random) 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, repeat steps 2 and 3

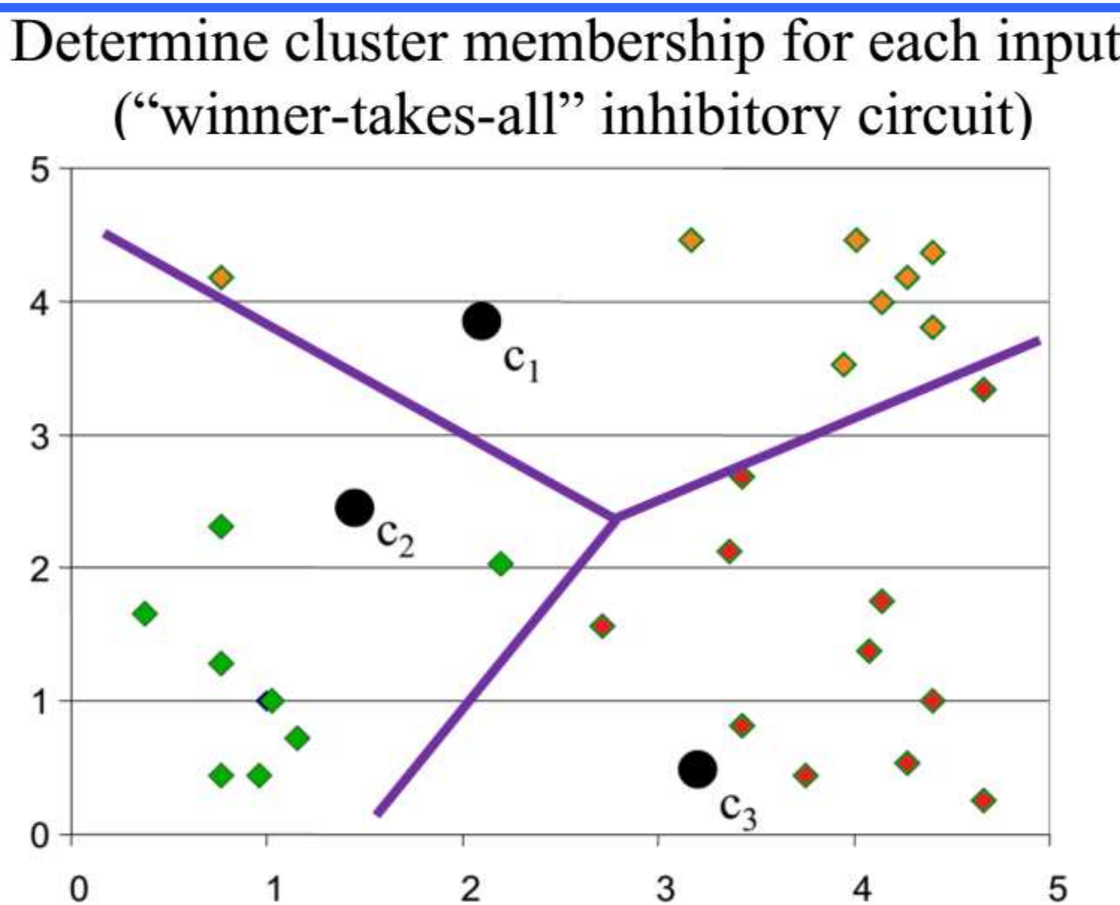
K-means clustering example: step 1

Randomly initialize the cluster centers (synaptic weights)



Choose *k* (random)

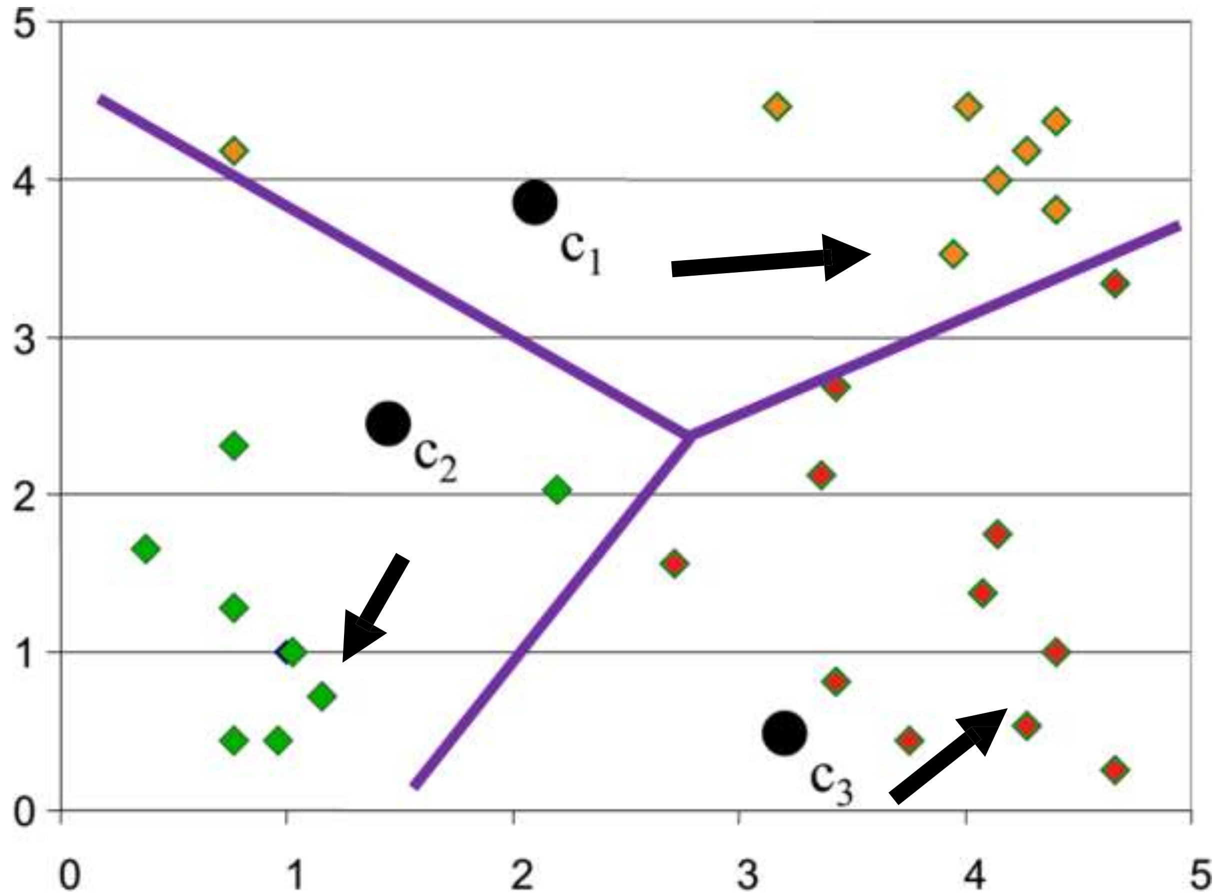
K-means clustering example – step 2



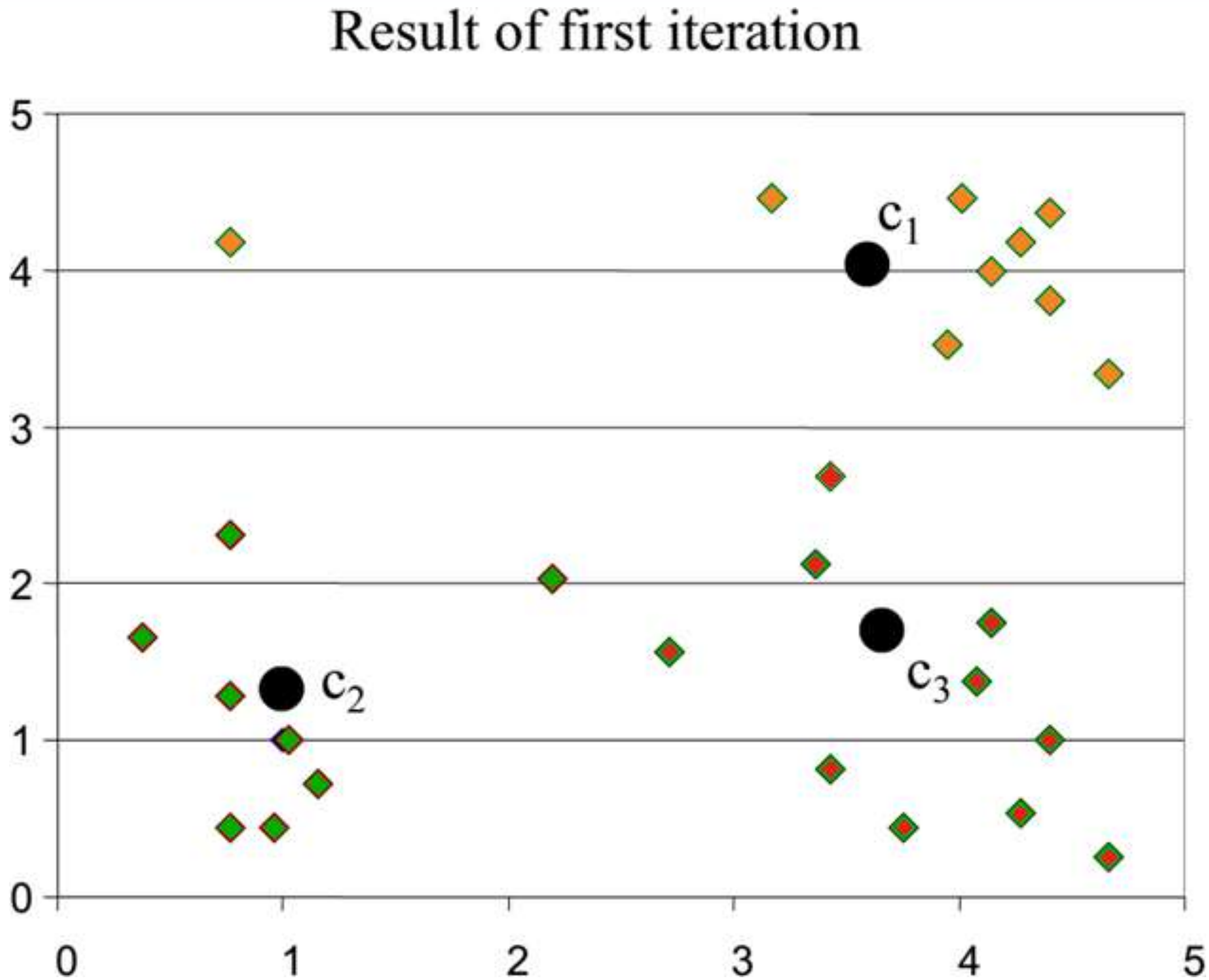
Assign each data point to the closest centroid

K-means clustering example – step 3

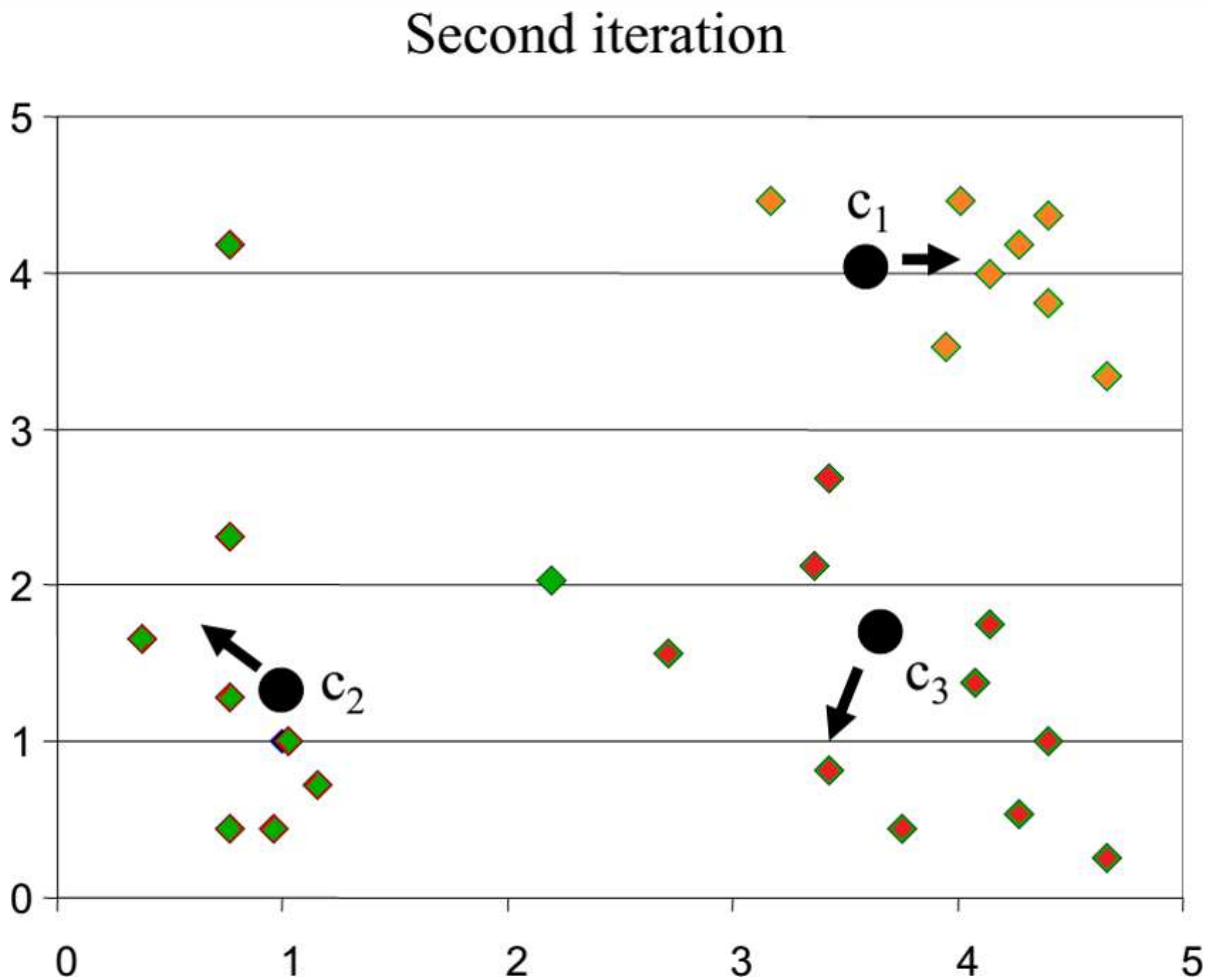
Re-estimate cluster centers (adapt synaptic weights)



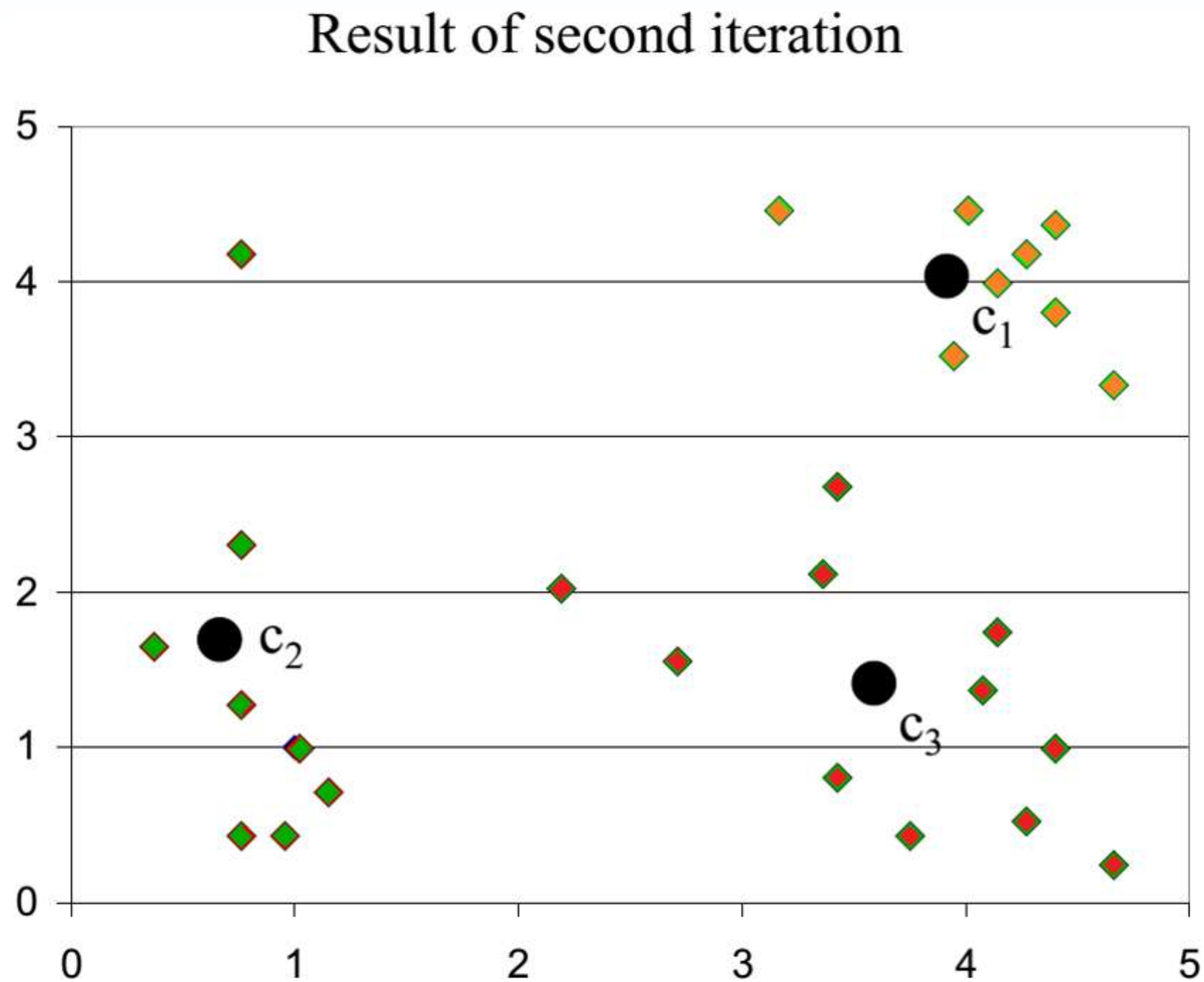
K-means clustering example



K-means clustering example



K-means clustering example



Why use K-means?

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,
  - where n is the number of data points,
  - k is the number of clusters, and
  - t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

Weaknesses of K-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or so me special data points with very different values.

K-means summary

- Despite weaknesses, *k*-means is still the most popular algorithm due to its simplicity and efficiency
- No clear evidence that any other clustering algorithm performs better in general
- Comparing different clustering algorithms is a difficult task. No one knows the correct clusters!

`Thank You!’