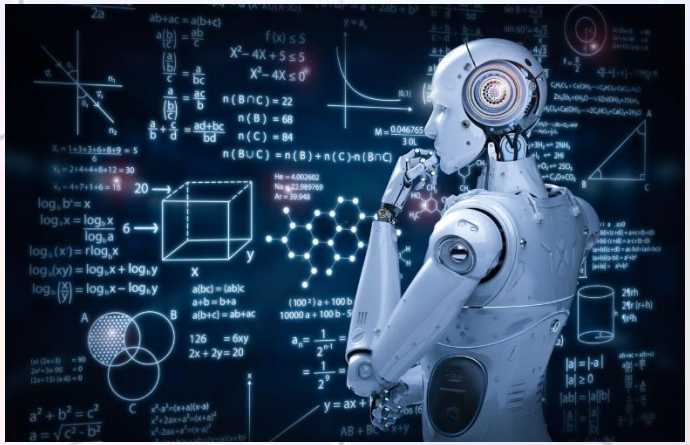
Machine Learning

GTU#3170724

B.E - Semester VII



Unit 8: Unsupervised Learning

Clustering k-means

Lecture#3

Instructor:

Munira Topia

Computer Engineering Department

L.J. Institutes of Engineering and Technology

Outline

K-means Clustering

Clustering

Unsupervised learning

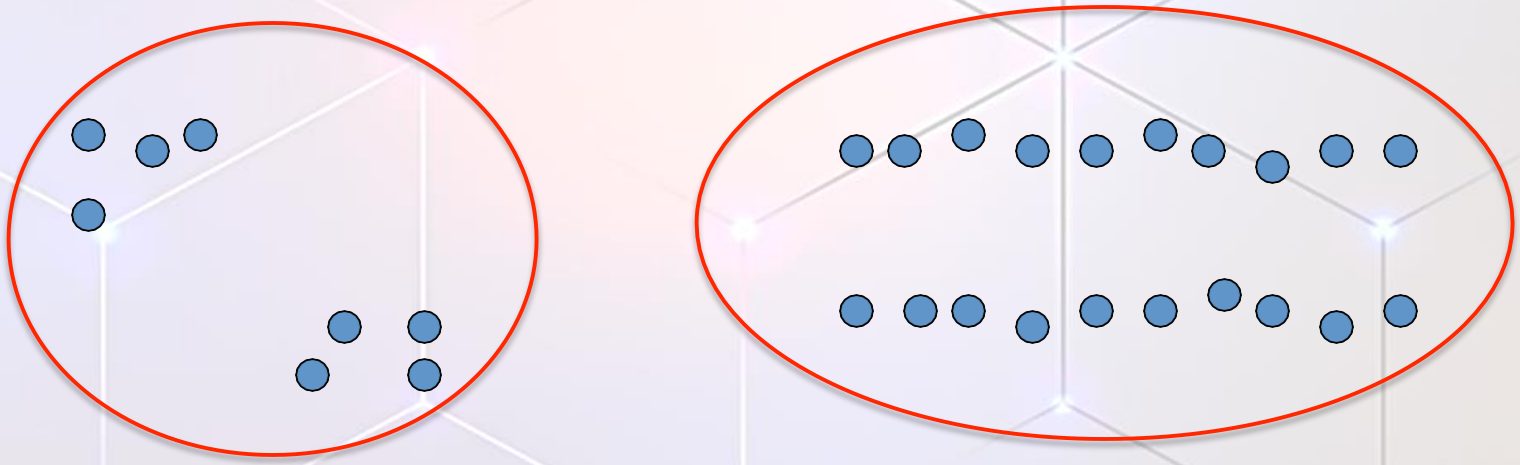
- Requires data, but no labels
- Detect patterns e.g. in Group emails or search results
- Customer shopping patterns
- Regions of images

Useful when don’t know what you’re looking for
But: can get gibberish

Clustering: Basic Idea

Group together similar instances

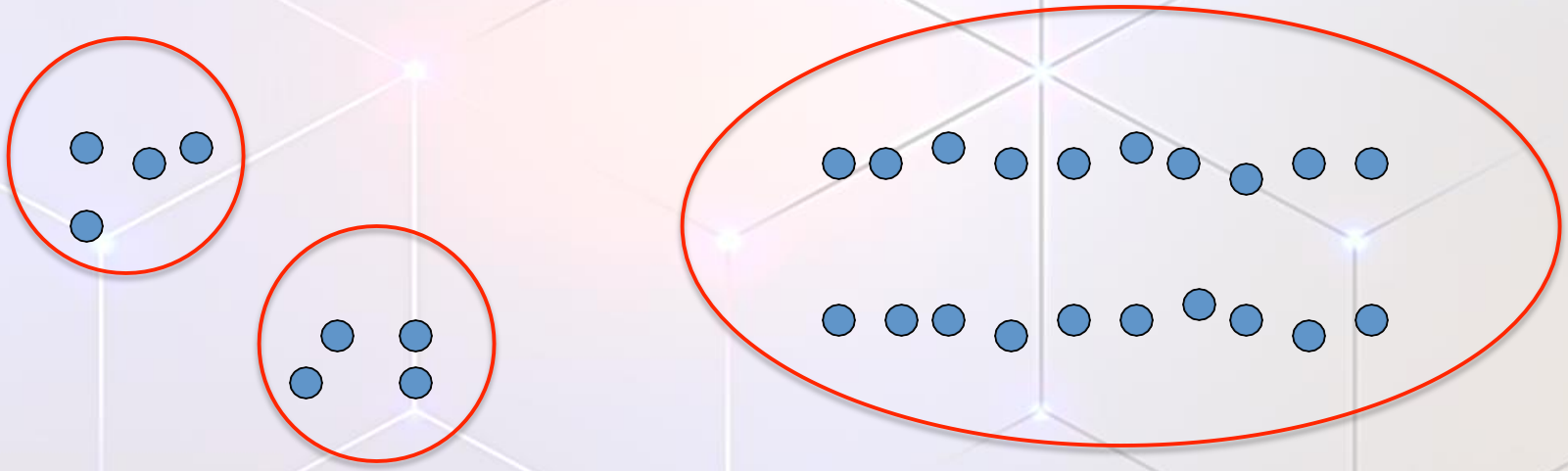
Example: 2D point patterns



Clustering: Basic Idea

Group together similar instances

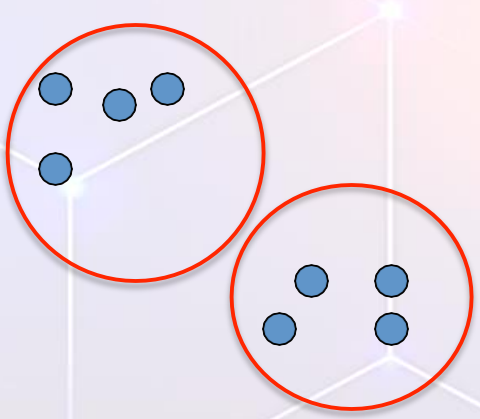
Example: 2D point patterns

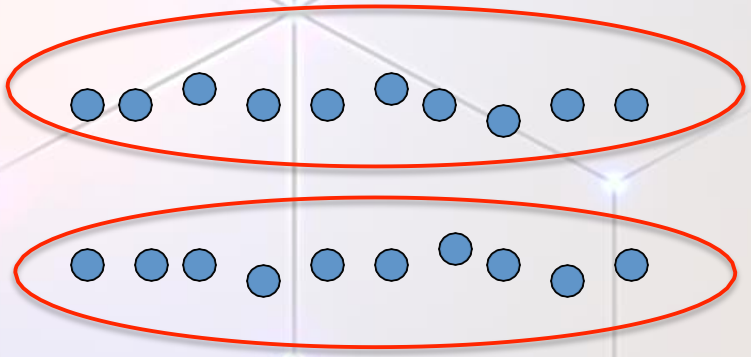


Clustering: Basic Idea

Group together similar instances

Example: 2D point patterns





K-Means Clustering

Partitioning

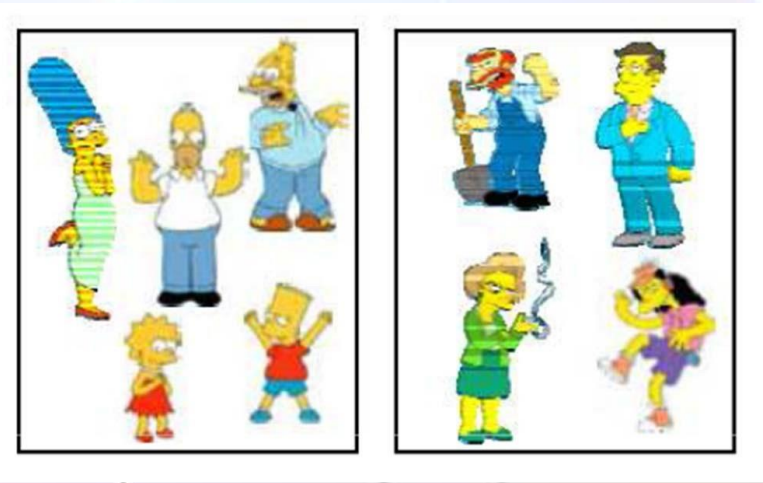
- k-Means algorithm
- PAM (k-Medoids algorithm)

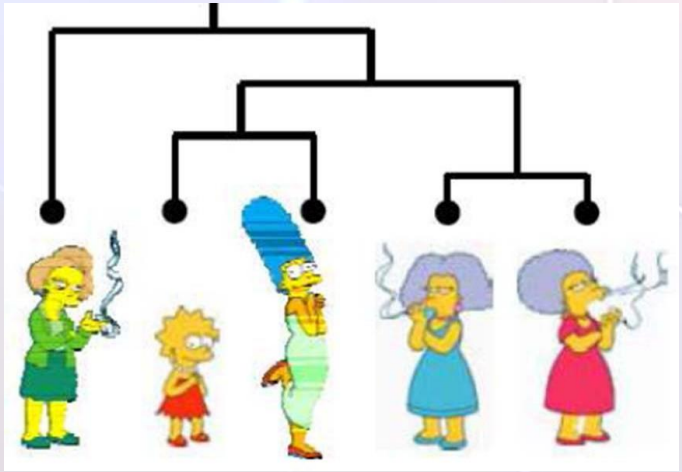
Hierarchical

- DIANA (divisive algorithm)

Density – Based

- DBSCAN

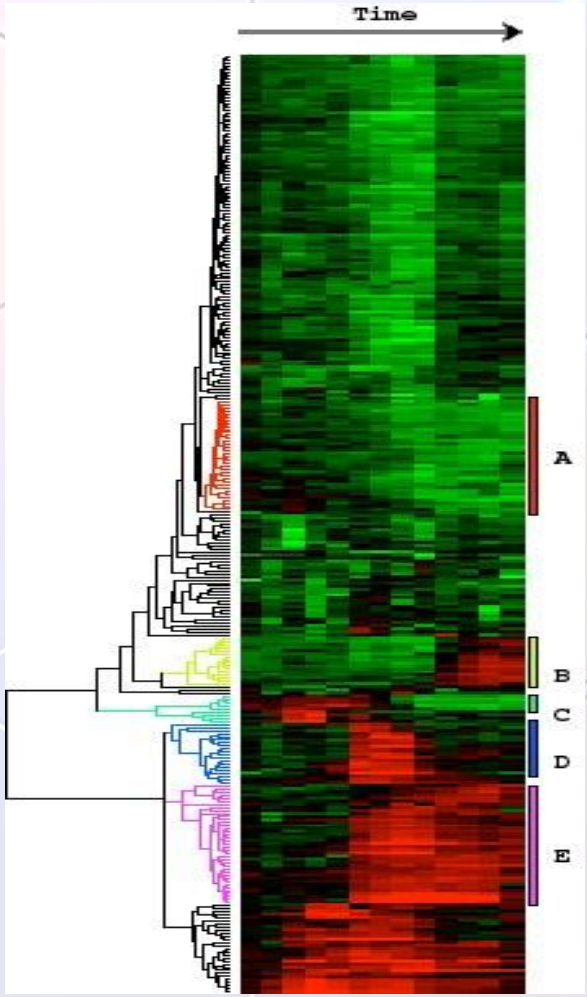




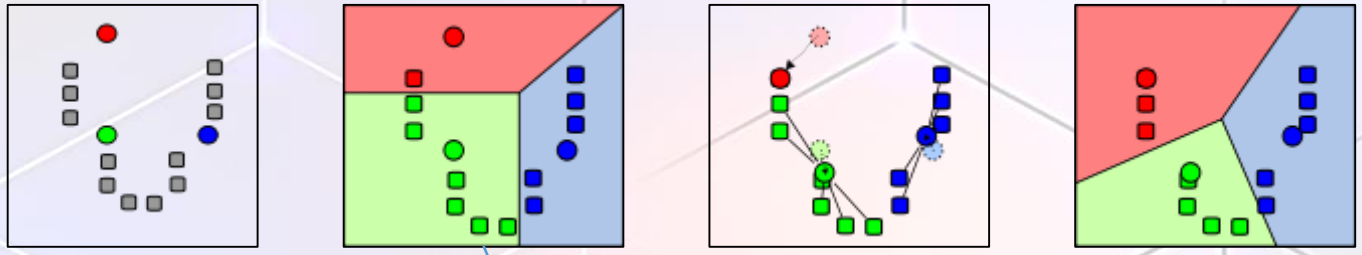
Clustering: Basic Idea

Image segmentation
Goal: Break up the image into meaningful or perceptually similar regions

Clustering gene expression data



K-Means Clustering: Basic Idea

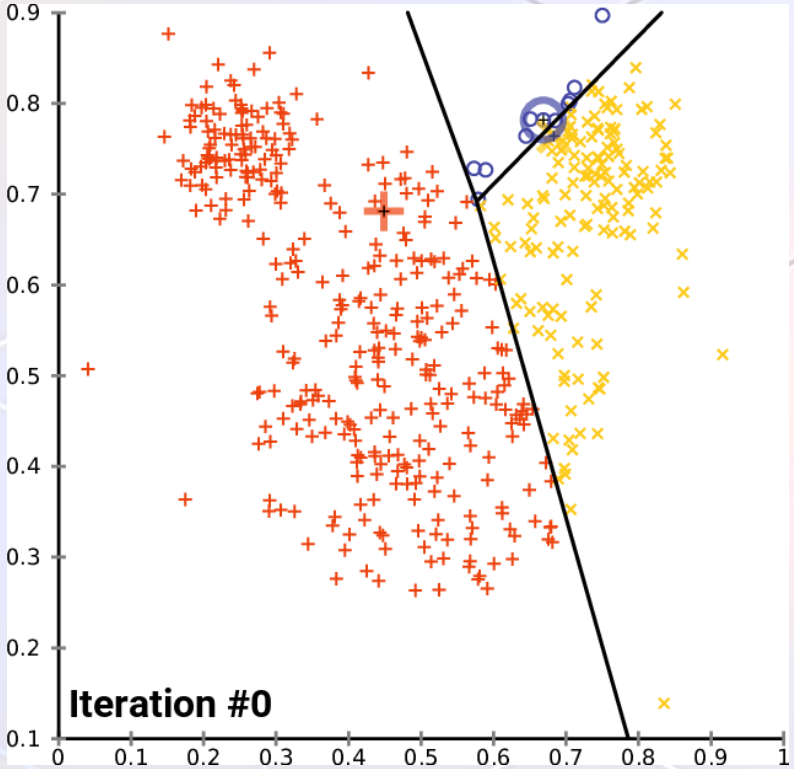


Given a set of ***n* distinct objects**, the k-Means clustering algorithm partitions the objects into ***k* number** of clusters such that **intra-cluster** similarity is high but the **inter-cluster** similarity is low.

In this algorithm, user has to specify *k*, the number of clusters and consider the objects are defined with numeric attributes and thus using any one of the distance metric to demarcate the clusters.

Voronoi Diagram

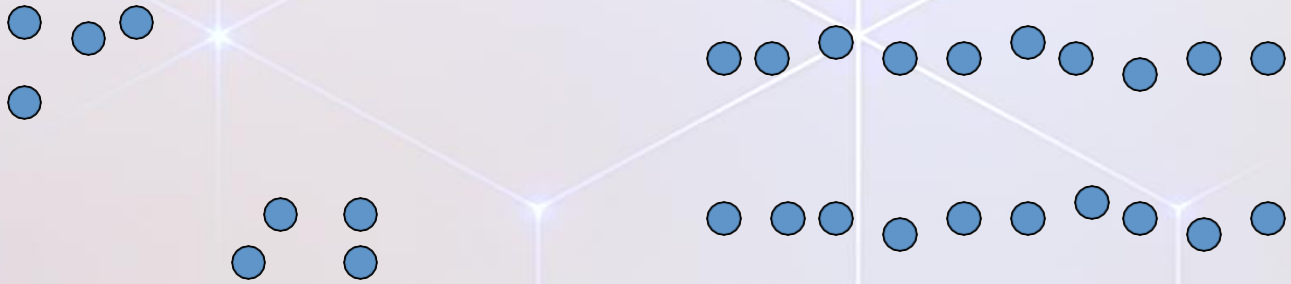
K-Means Clustering



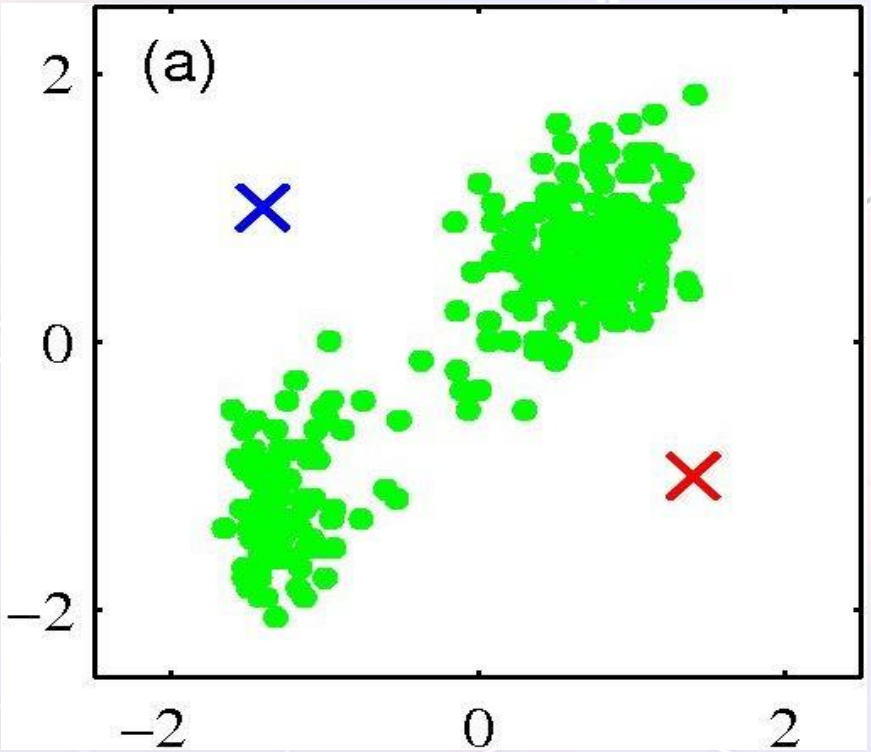
- Initialize: Pick *K* random points as cluster centers
- Alternate:
  1. Assign data points to closest cluster center
  2. Change the cluster center to the average of its assigned points
- Stop when no points' assignments change

K-Means Clustering

- Initialize: Pick *K* random points as cluster centers
- Alternate:
  1. Assign data points to closest cluster center
  2. Change the cluster center to the average of its assigned points
- Stop when no points' assignments change

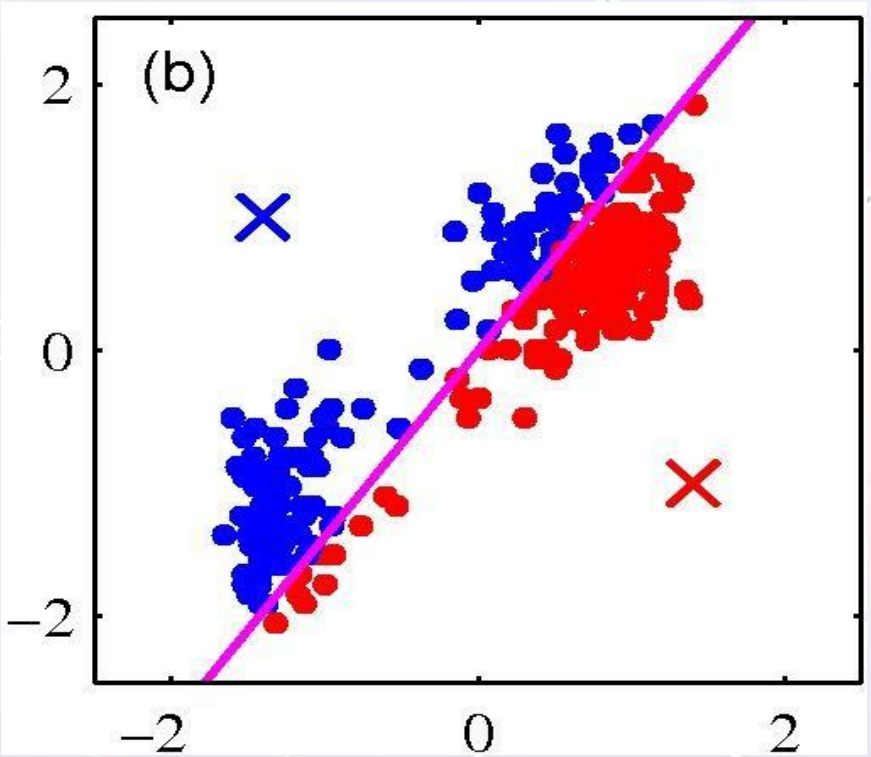


K-Means Clustering



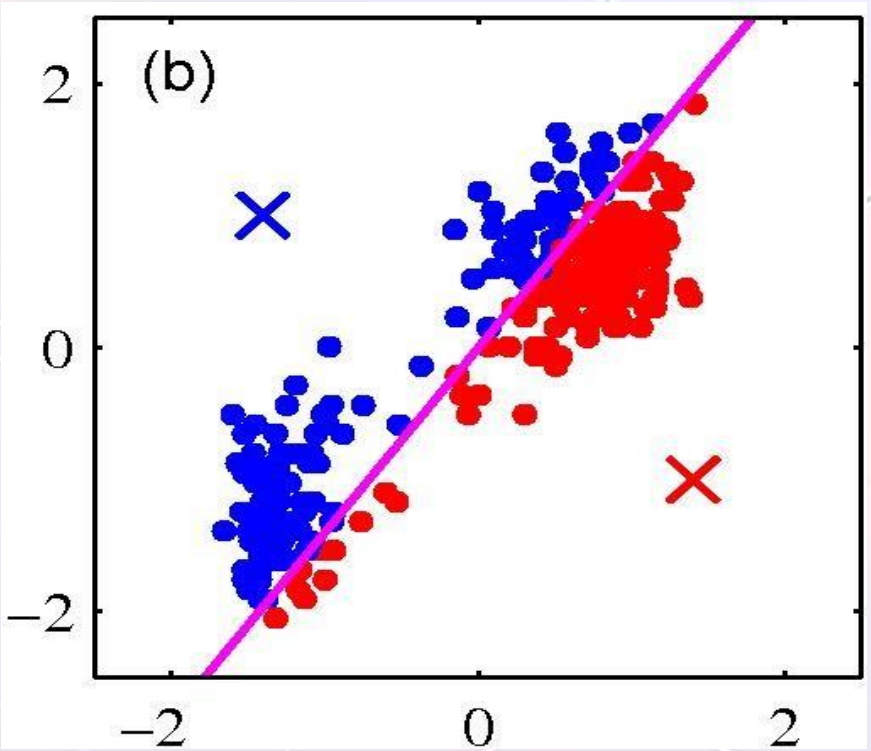
- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center

K-Means Clustering



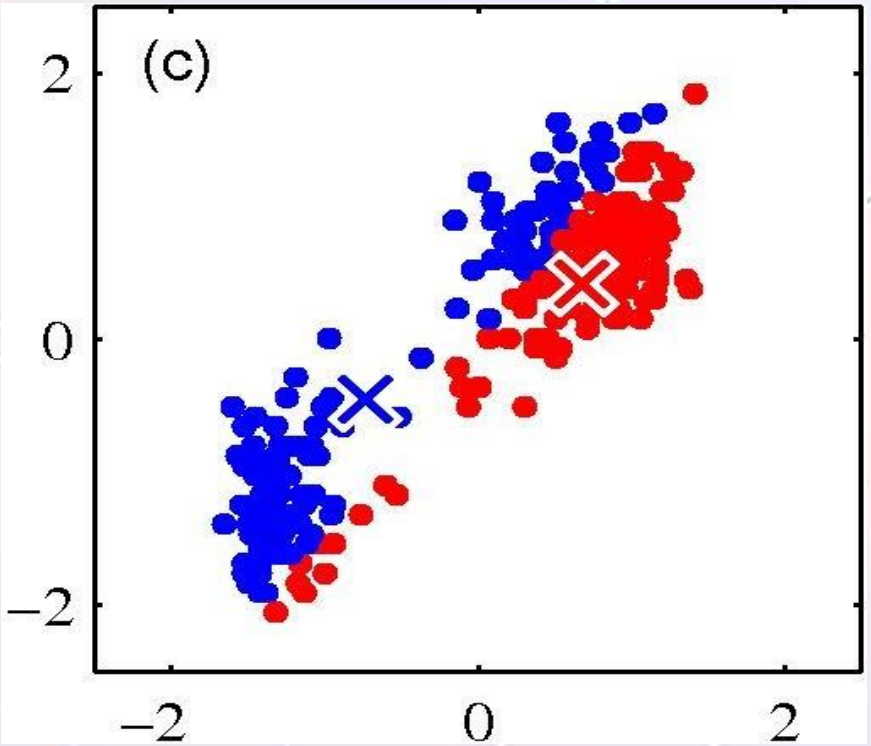
- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center

K-Means Clustering



- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center
- Iterative Step 2
- Change the cluster center to the average of the assigned points

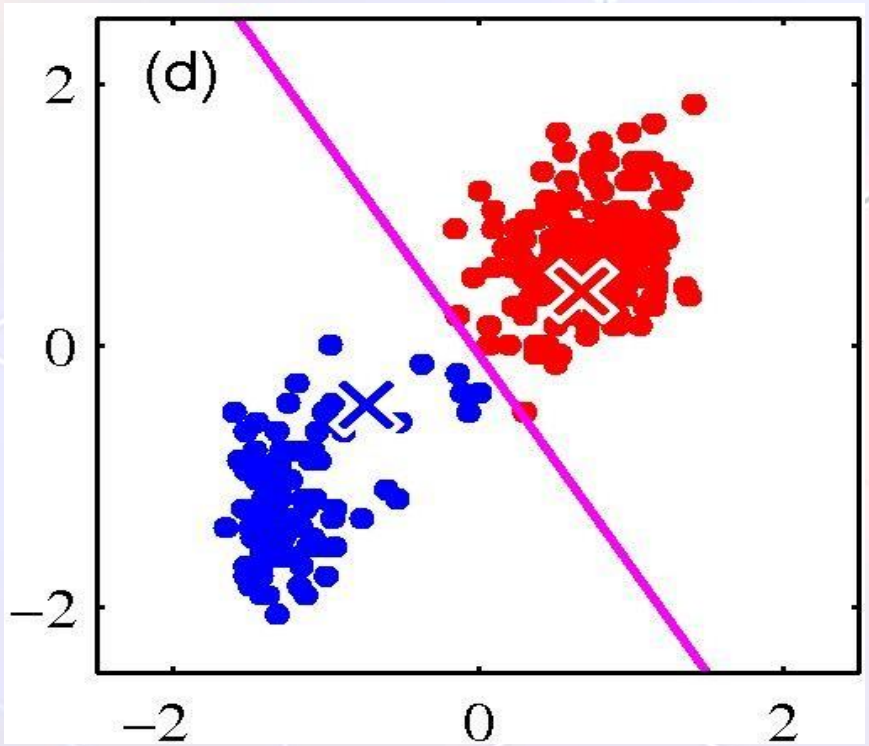
K-Means Clustering



- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center
- Iterative Step 2
- Change the cluster center to the average of the assigned points

Repeat until convergence

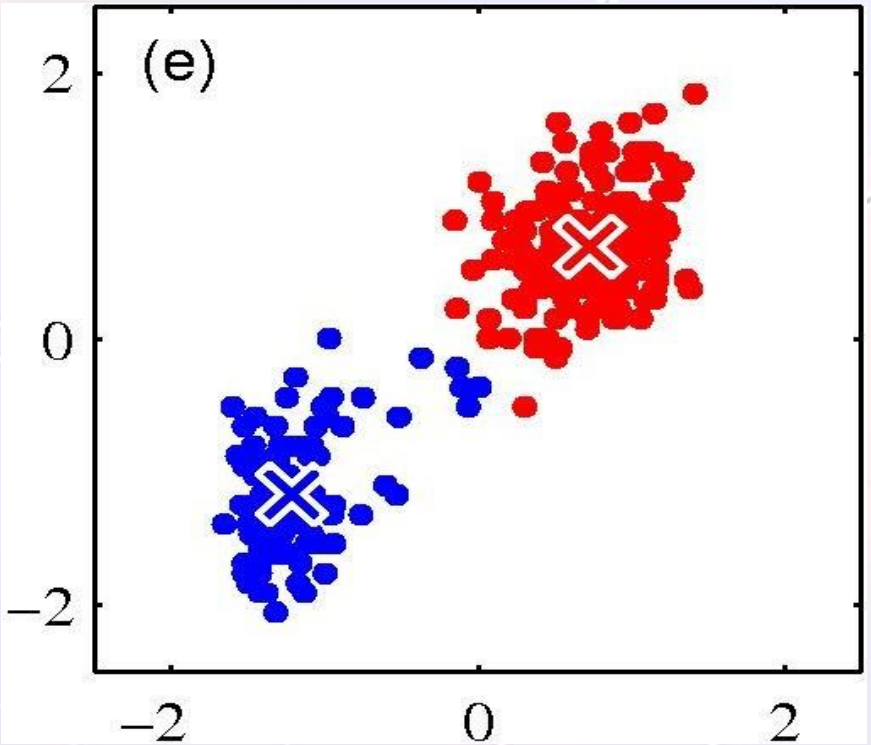
K-Means Clustering



- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center
- Iterative Step 2
- Change the cluster center to the average of the assigned points

Repeat until convergence

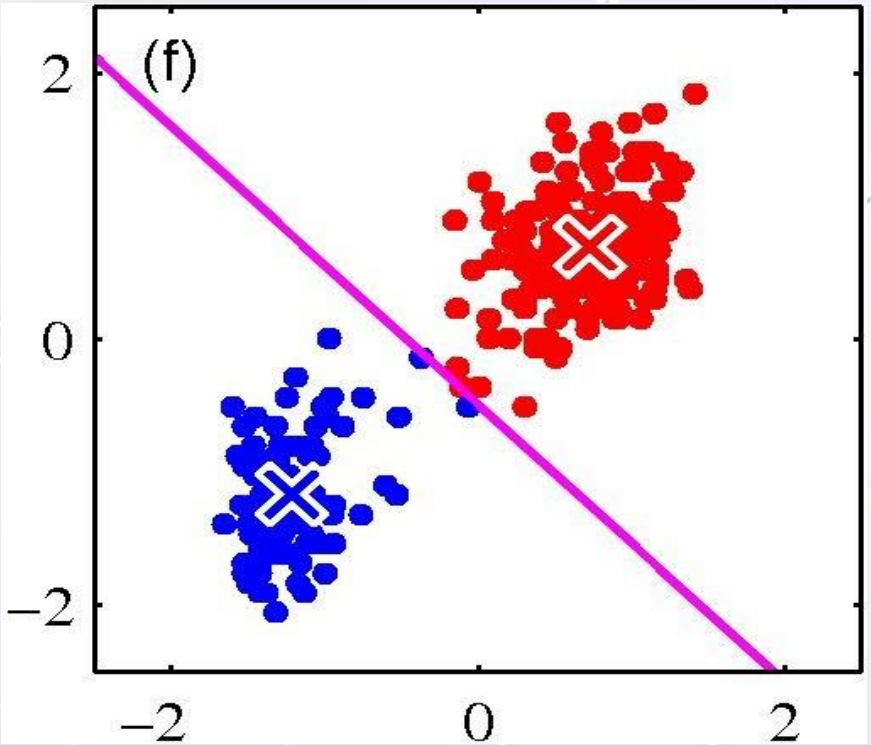
K-Means Clustering



- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center
- Iterative Step 2
- Change the cluster center to the average of the assigned points

Repeat until convergence

K-Means Clustering



- Pick K random points as cluster centers (means)
- Shown here for K=2
- Iterative Step 1
- Assign data points to closest cluster center
- Iterative Step 2
- Change the cluster center to the average of the assigned points

Repeat until convergence

$$\text{dist}(x, y) = \sqrt{\sum_{1}^{n} (x_i - y_i)^2}$$

$$\text{SSE} = \sum_{i=1}^{k} \sum_{x \in C_i} \text{dist}\,(c_i, x)^2$$

K-Means Clustering Algorithm

First it selects ***k* number** of objects at **random** from the set of n objects. These *k* objects are treated as the **centroids or center of gravities** of *k* clusters.

For each of the **remaining objects**, it is assigned to one of the **closest centroid**. Thus, it forms a collection of objects assigned to each centroid and is called a **cluster**.

Next, the **centroid** of each cluster is then **updated** (by calculating the mean values of attributes of each object).

The **assignment and update** procedure is until it reaches some **stopping criteria** (such as, number of iteration, centroids remain unchanged or no assignment, etc.)

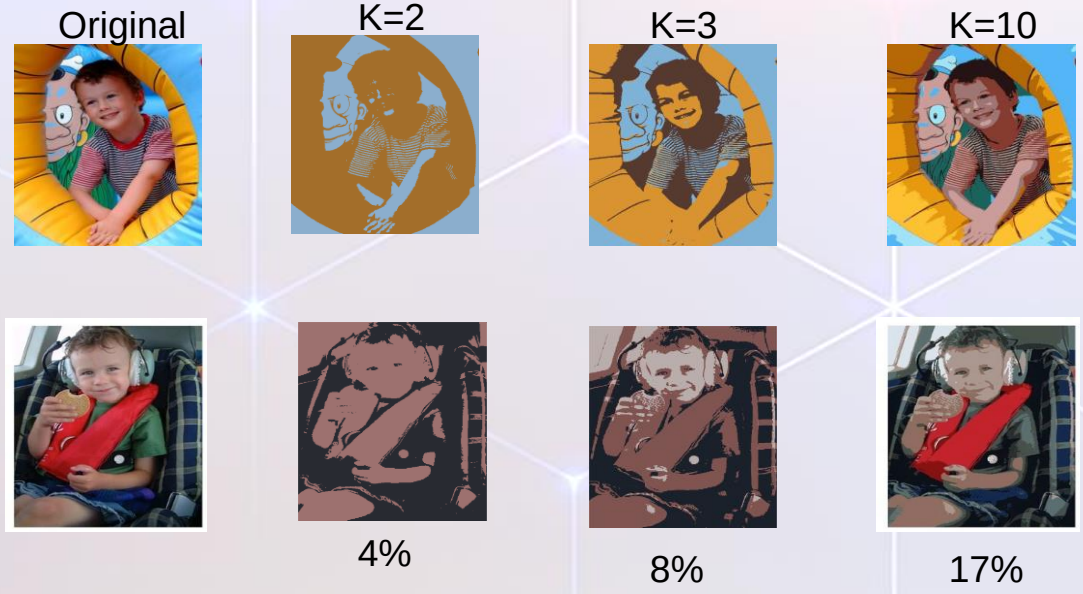
K-Means Clustering

Guaranteed to converge in a finite number of iterations

Running time per iteration:

1. Assign data points to closest cluster center O(KN) time
2. Change the cluster center to the average of its assigned points O(N)

K-Means for Segmentation



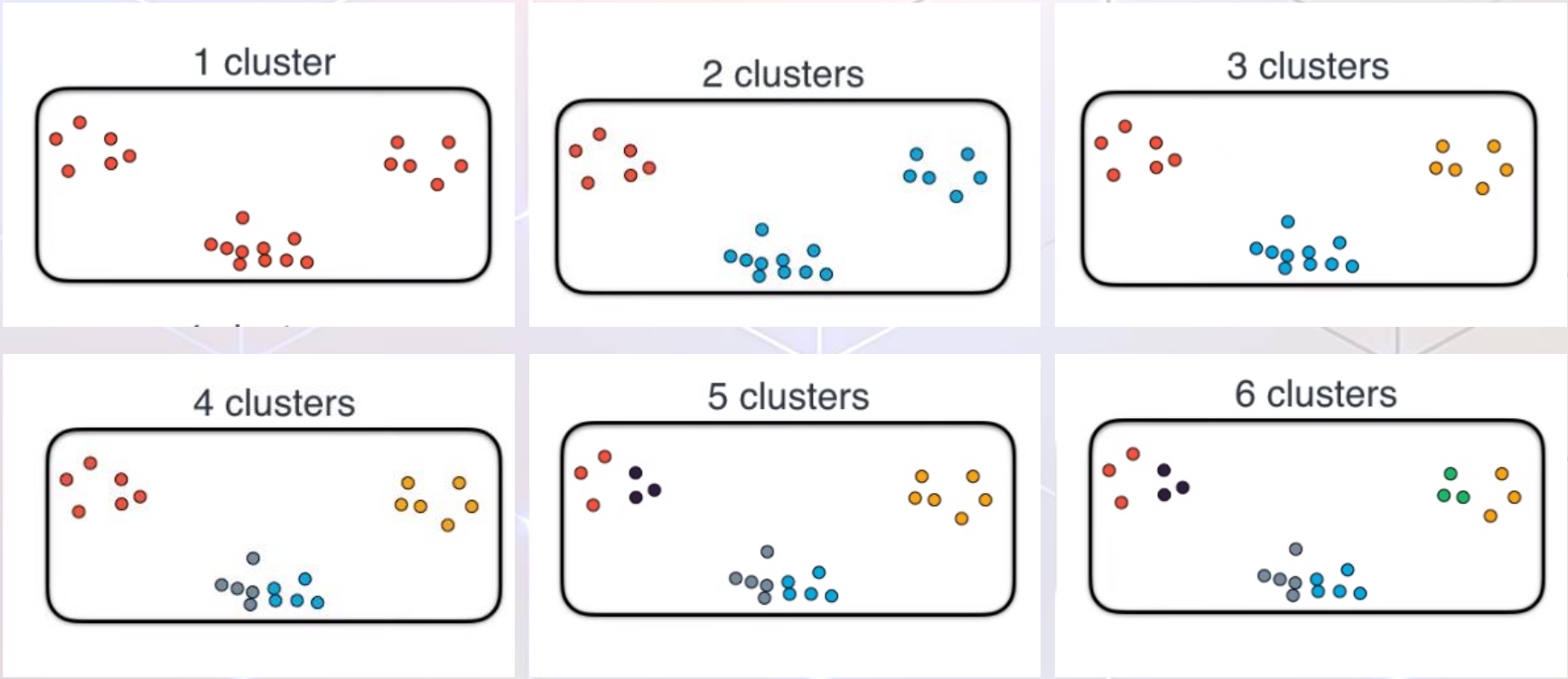
K-Means : K value

For small data set, **rule of thumb** is followed

$$k = \sqrt{\frac{n}{2}}$$

For large data set, **Elbow method**

Measures homogeneity and heterogeneity with respect to ‘k’ value



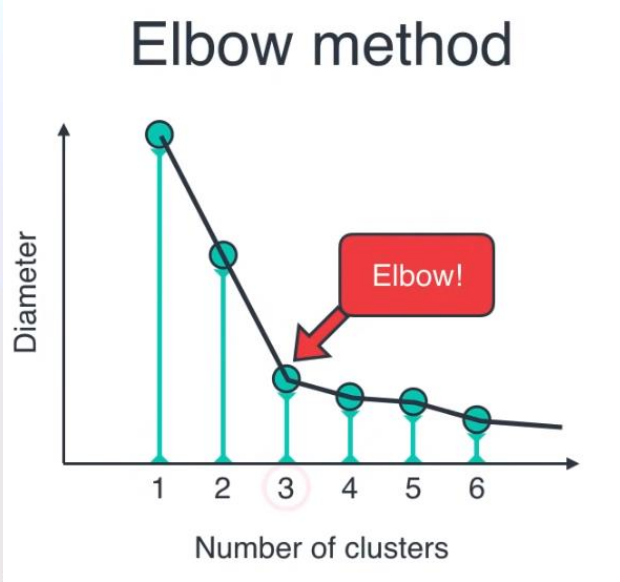
K-Means : K value

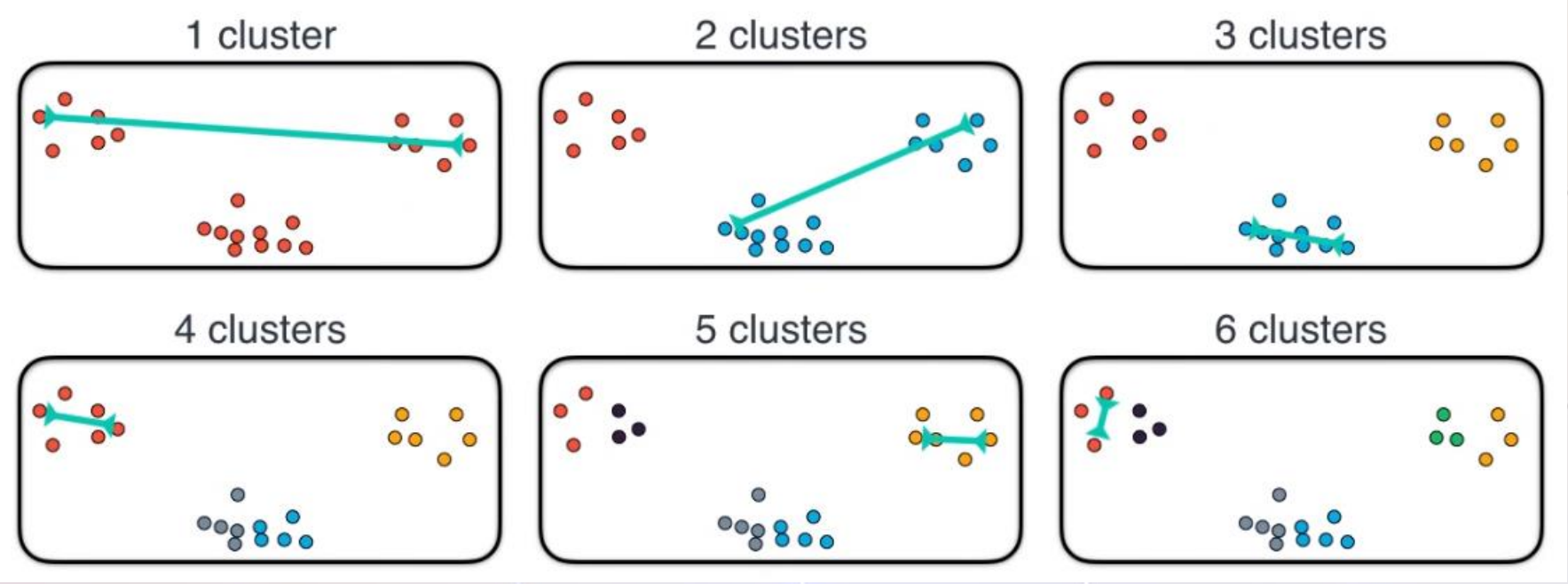
For small data set, **rule of thumb** is followed

$$k = \sqrt{\frac{n}{2}}$$

For large data set, **Elbow method**

Measures homogeneity and heterogeneity with respect to ‘k’ value





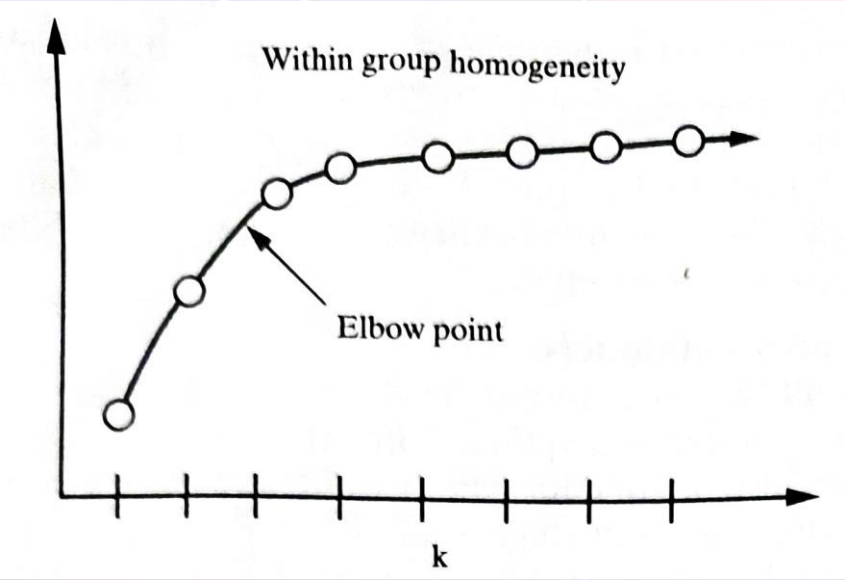
K-Means : K value

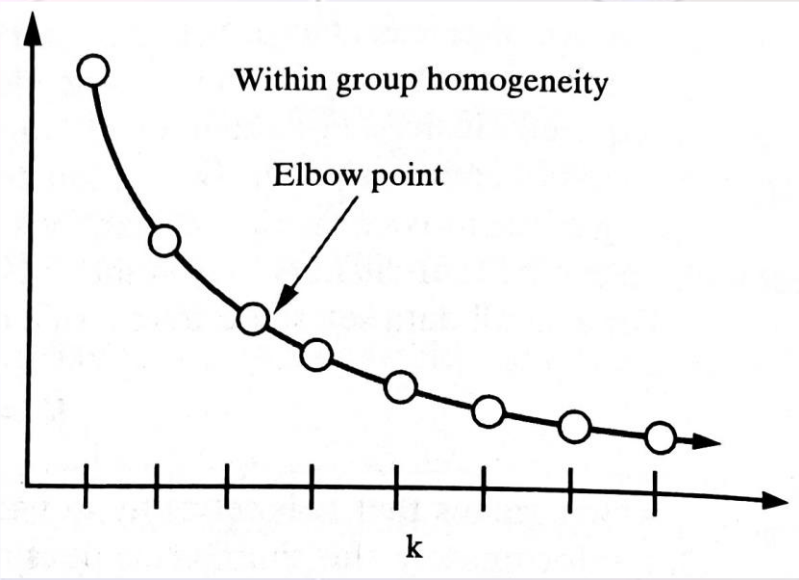
For small data set, **rule of thumb** is followed

$$k = \sqrt{\frac{n}{2}}$$

For large data set, **Elbow method**

Measures homogeneity and heterogeneity with respect to ‘k’ value

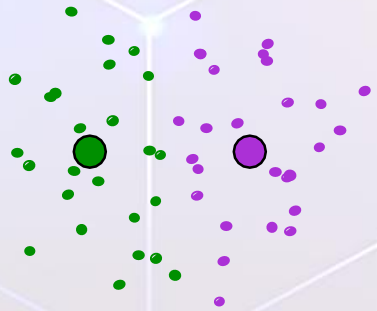




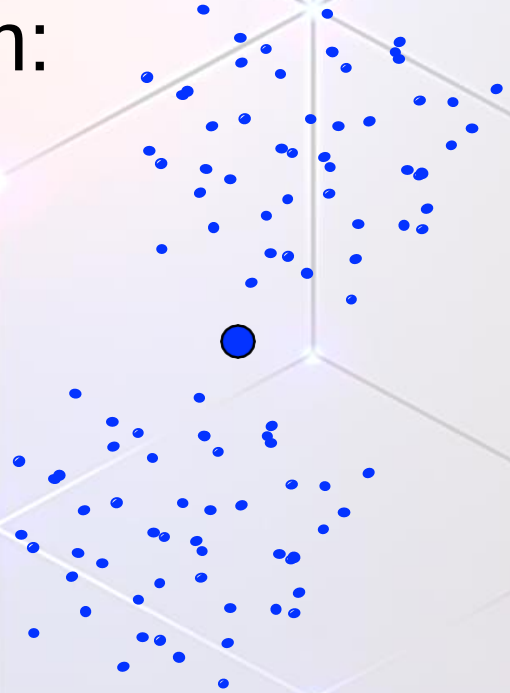
K-Means : K value

- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?

A local optimum:



Would be better to have one cluster here

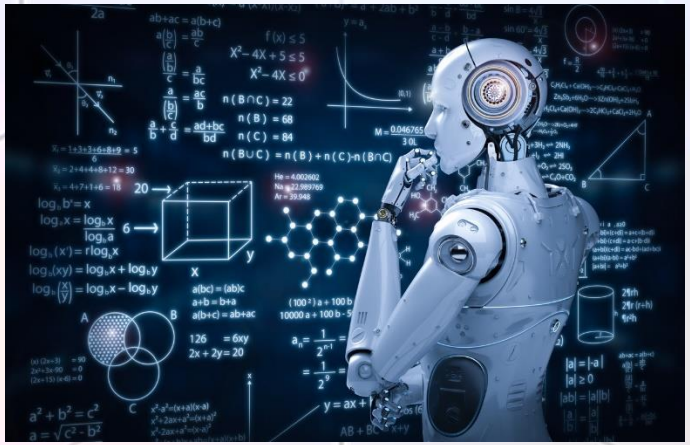


… and two clusters here

Machine Learning

GTU#3170724

B.E - Semester VII



Unit 8: Unsupervised Learning

Clustering k-medoid

Lecture#3

Instructor:

Munira Topia

Computer Engineering Department

L.J. Institutes of Engineering and Technology

K-Medoids Clustering

Mean and Outlier

Example of data points: 1, 2, 3, 6, 9, 10, 12 and 25

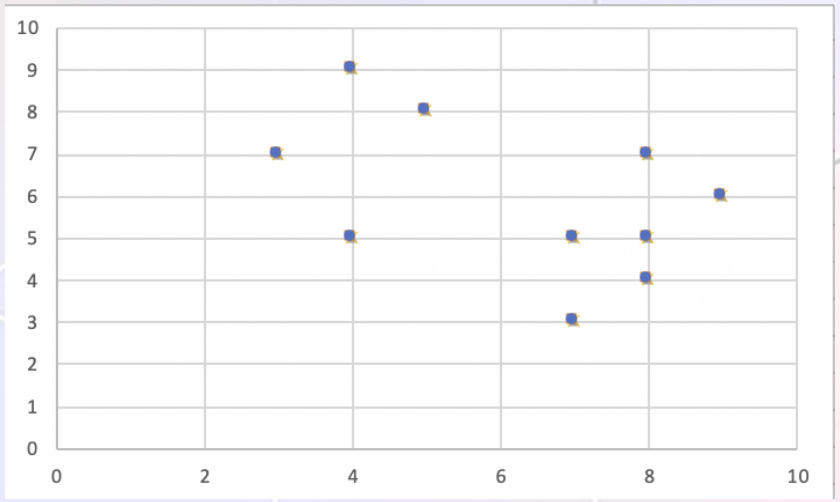
With K=2, initial clusters are {1,2,3,6} and {9,10,12,25}

Mean of cluster-1 {1,2,3,6} = 12/4 = 3
Mean of cluster-2 {9,10,12,25} = 56/4 = 14
SSE within cluster is
$(1-3)^2+(2-3)^2+(3-3)^2+(6-3)^2+(9-14)^2+(10-14)^2+(12-14)^2+(25-14)^2= 179$

If clusters are redefined, {1,2,3,6,9} and {10,12,25}
Mean of cluster-1 {1,2,3,6,9} = 21/5 = 4.2
Mean of cluster-2 {10,12,25} = 47/3 = 15.67
SSE within cluster is
$(1-4.2)^2+(2-4.2)^2+(3-4.2)^2+(6-4.2)^2+(9-4.2)^2+(10-15.67)^2+(12-15.67)^2+(25-15.67)^2= 113.84$

So instead of considering mean of data points in cluster,
using representative data points as center.

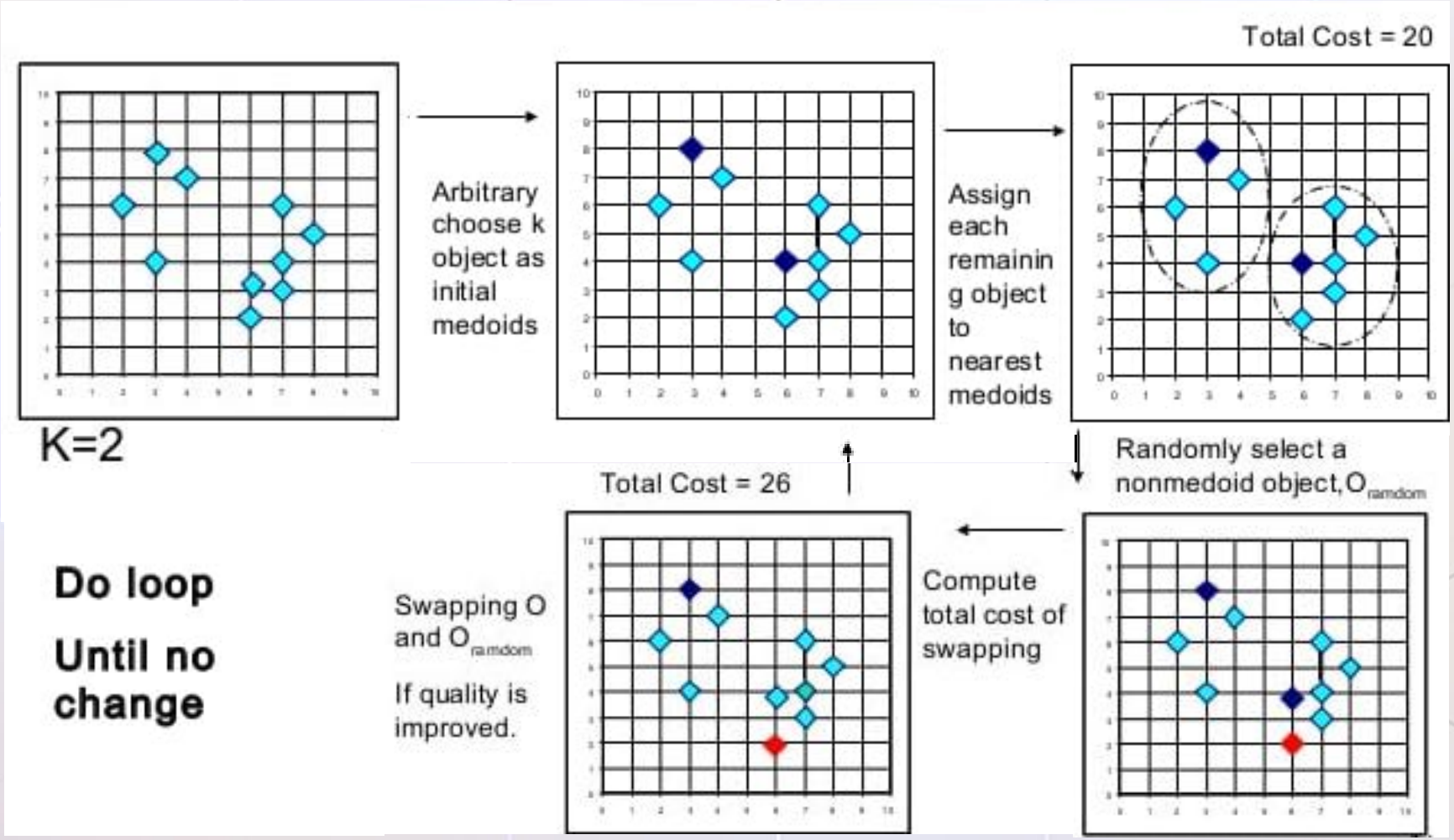
Medoid v/s Centroid



A medoid can be defined as the point in the cluster, whose dissimilarities with all the other points in the cluster is minimum.

K-Medoid

Working



K-Medoid Algorithm (PAM)

Partitioning Around Medoids (PAM)

1. Randomly choose k points in the data set as initial representative points (O_j)
2. Assign each of remaining points to the cluster which has nearest representative point O_j and compute cost.
3. Randomly select a non-representative point O_r in a cluster
4. Swap O_j with O_r and compute the new Cost after swapping.
5. If $Cost_{new} < Cost_{Old}$ then swap O_j with O_r to form new set of k representative objects.
6. Refine the k clusters on the basis of nearest representative point. Continue unitl there is no change

Complexity of algorithm is $O(k(n-k)^2)$.

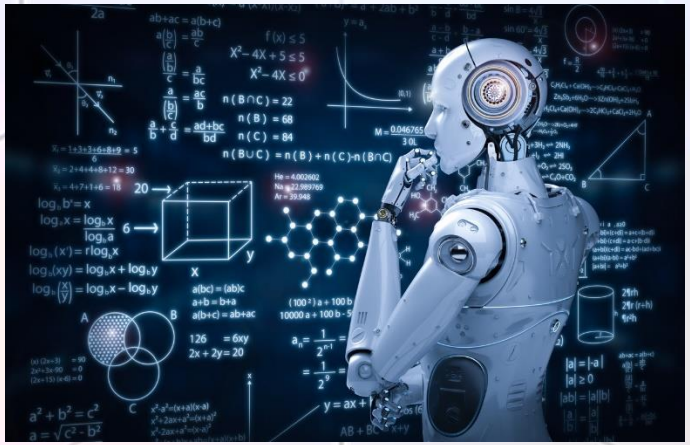
Thank You!

K-means Clustering

Machine Learning

GTU#3170724

B.E - Semester VII



Unit 8: Unsupervised Learning

Hierarchical Clustering

Density based Clustering

Lecture#32

Instructor:

Munira Topia

Computer Engineering Department

L.J. Institutes of Engineering and Technology

Outline

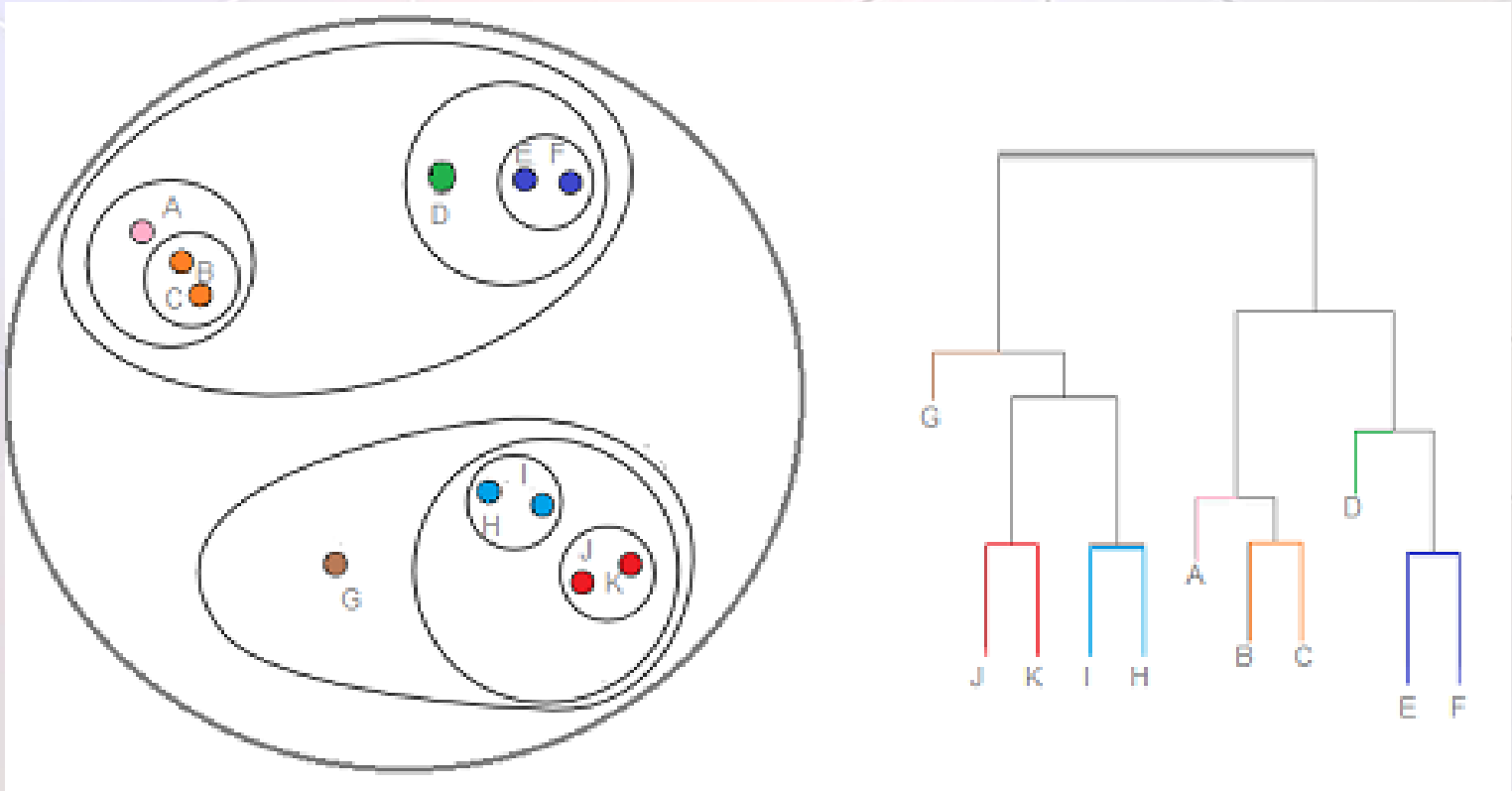
Hierarchical Clustering

Agglomerative and Divisive hierarchical clustering

DBSCAN- Density based Clustering

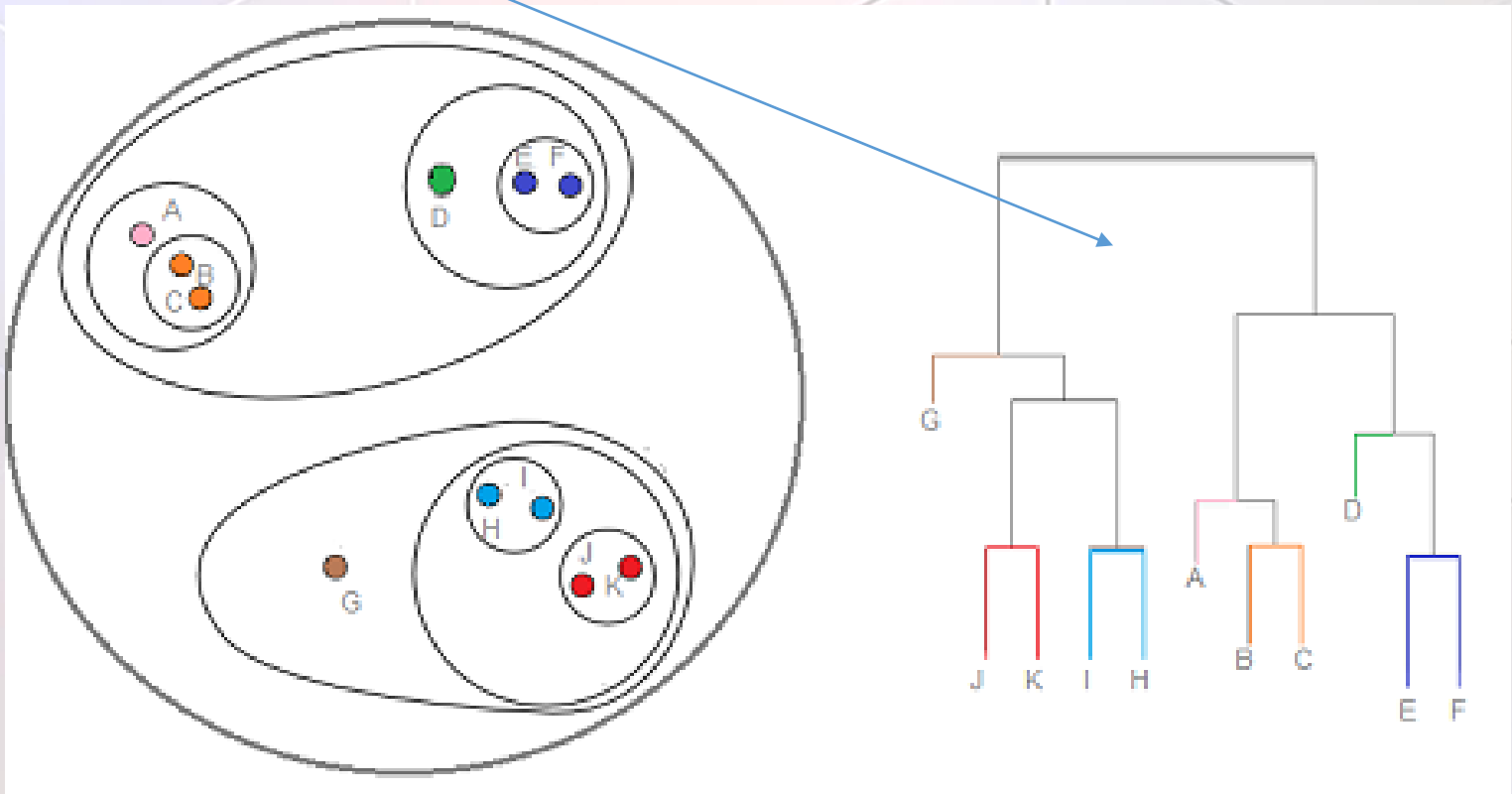
Hierarchical Clustering

- **Hierarchical clustering** is characterized by the development of a hierarchy or tree-like structure.
- The goal is to produce a hierarchical series of nested clusters, ranging from clusters of individual points at the bottom to an all-inclusive cluster at the top.



Hierarchical Clustering

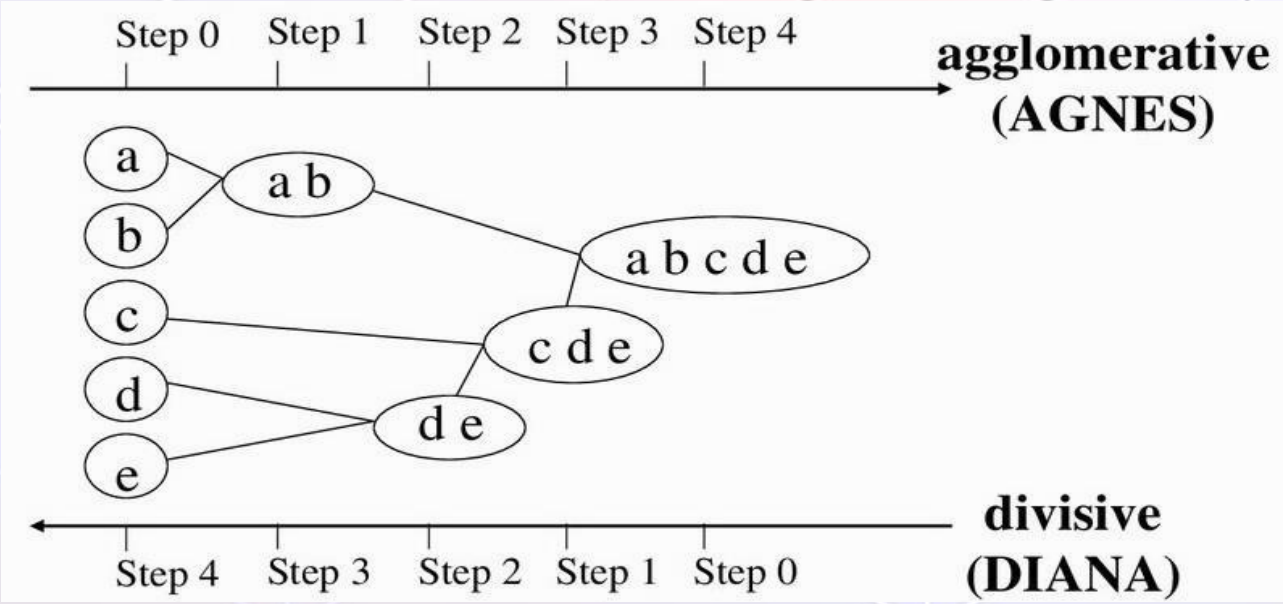
- **Hierarchical clustering** is characterized by the development of a hierarchy or tree-like structure.
- The goal is to produce a hierarchical series of nested clusters, ranging from clusters of individual points at the bottom to an all-inclusive cluster at the top.
- A **dendrogram** is a diagram representing a tree. In hierarchical clustering, it illustrates the arrangement of the cluster produced by the corresponding analysis.



Agglomerative v/s Divisive

Agglomerative clustering starts with each object in a separate cluster.

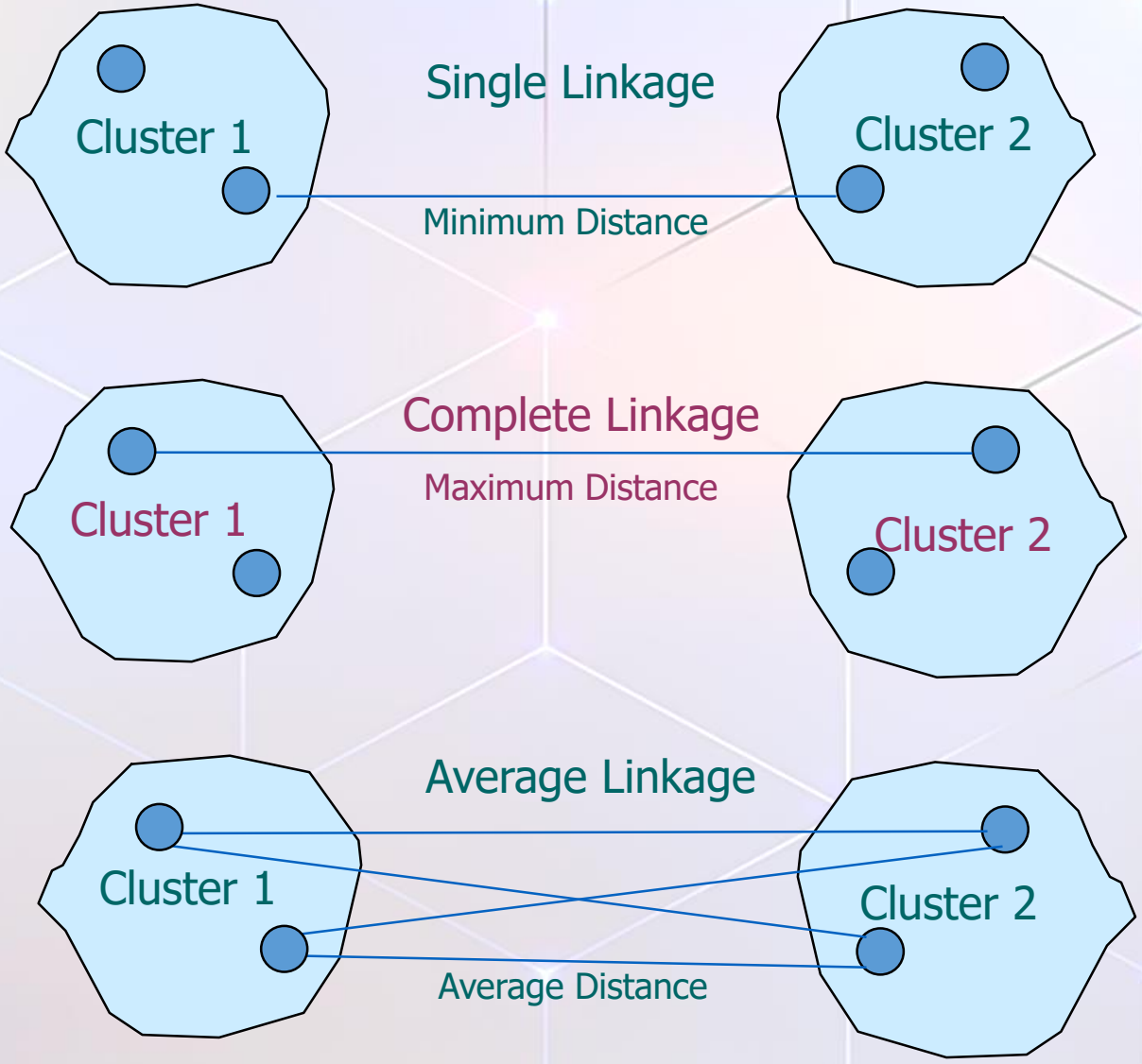
Clusters are formed by grouping objects into bigger and bigger clusters.



Divisive clustering starts with all the objects grouped in a single cluster. Clusters are divided or split until each object is in a separate cluster.

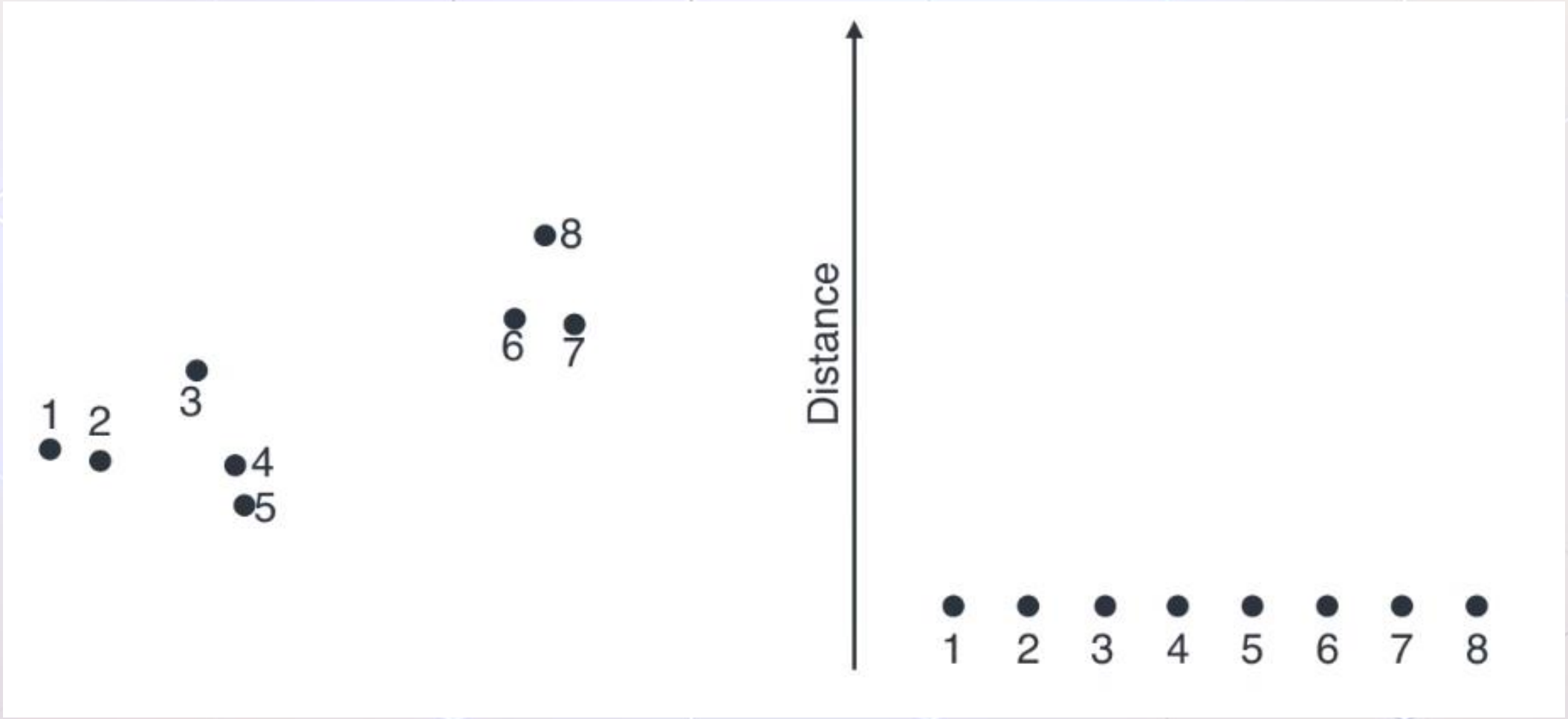
Agglomerative methods are commonly used in marketing research. They consist of linkage methods, variance methods, and centroid methods.

Linkage Methods of Clustering



Hierarchical Clustering: Example

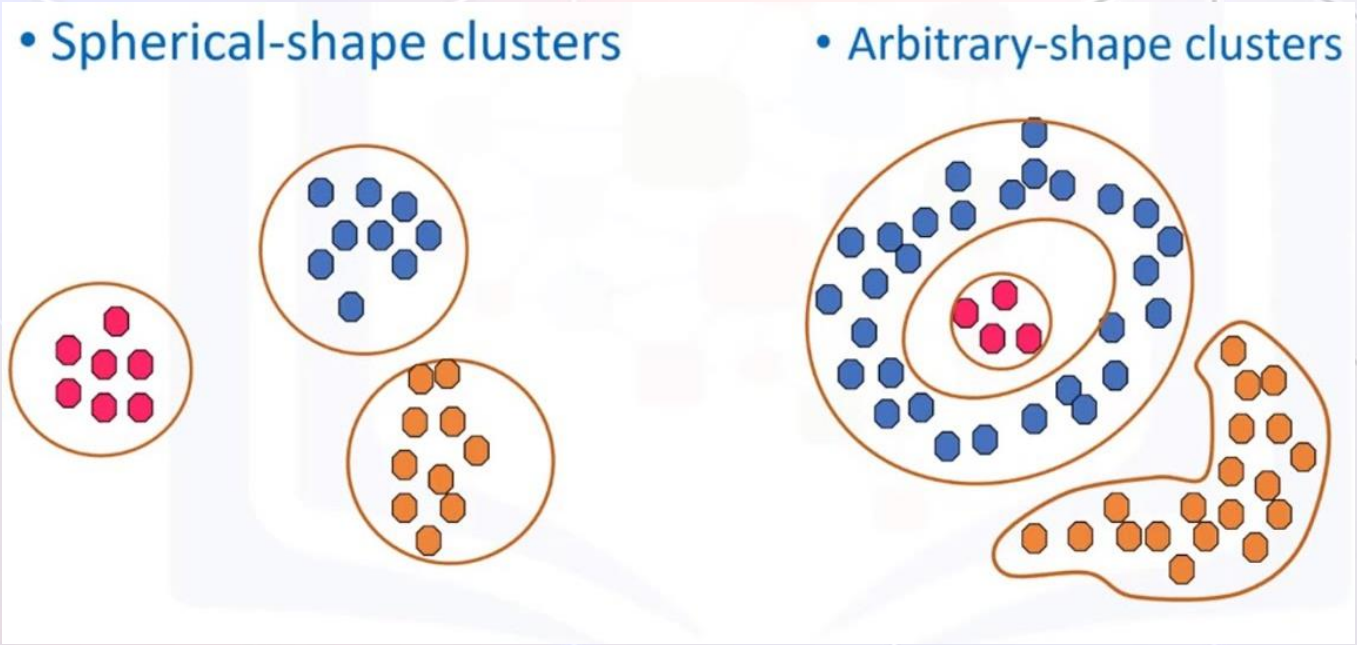
Single Link Example



DBSCAN

Density Based Spatial Clustering of Applications with Noise (DBSCAN)

- Discovers clusters of **arbitrary shape** in spatial databases with noise
- A cluster is defined as a maximal set of **density-connected** points.
- Clusters are **dense regions** in the data space, separated by regions of lower object density



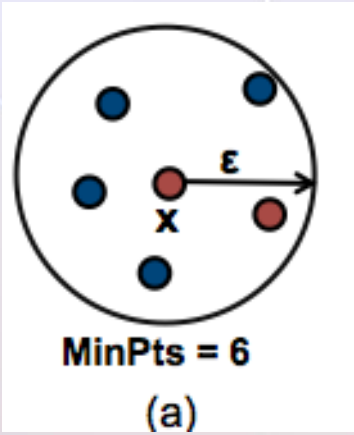
DBSCAN: Basics

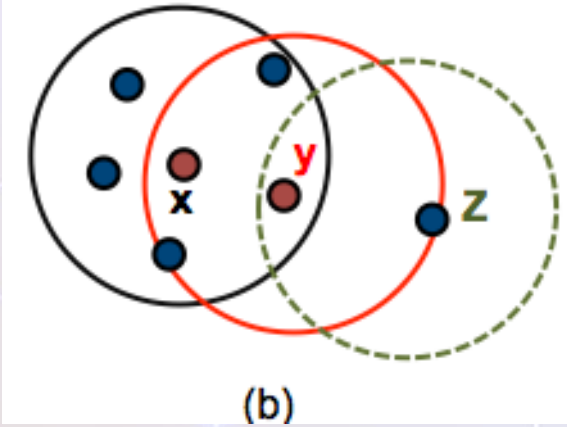
- For any point in a cluster, the local point density around that point has to exceed some threshold

  ε-Neighborhood – Objects within a radius of ε from an object.

- MinPts – minimum number of points in the given neighborhood

  "High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.





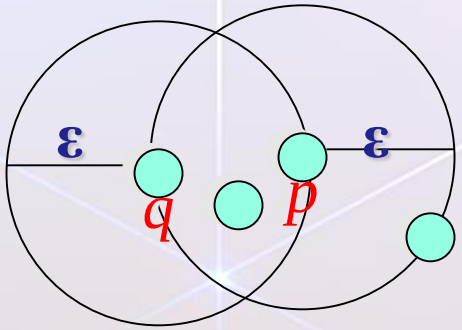
DBSCAN: Basics

- For any point in a cluster, the local point density around that point has to exceed some threshold

  ε-Neighborhood – Objects within a radius of ε from an object.

- MinPts – minimum number of points in the given neighborhood

  "High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.



ε-Neighborhood of *p*

ε-Neighborhood of *q*

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

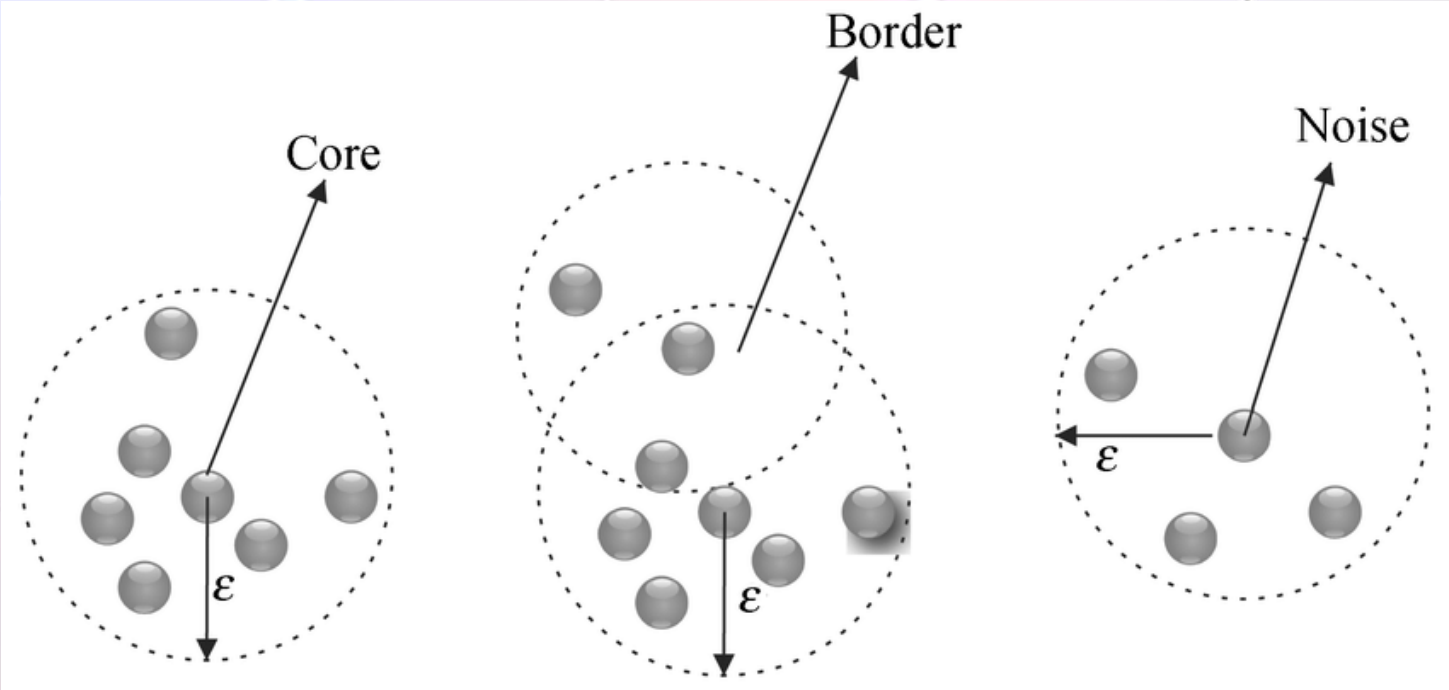
DBSCAN: Basics

- For any point in a cluster, the local point density around that point has to exceed some threshold

  ε-Neighborhood – Objects within a radius of ε from an object.

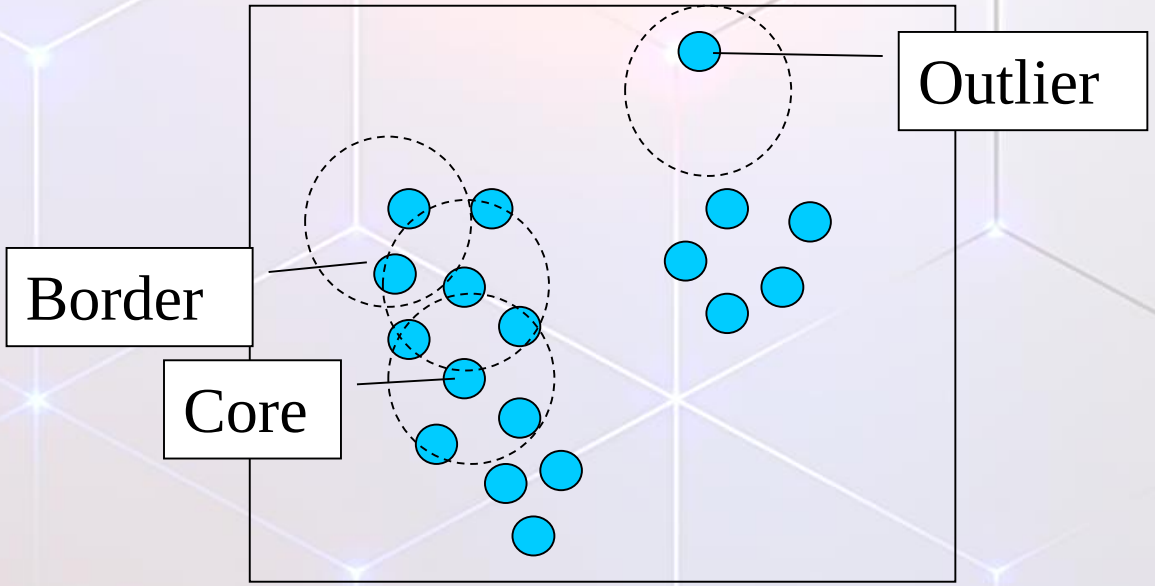
- MinPts – minimum number of points in the given neighborhood

  "High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.



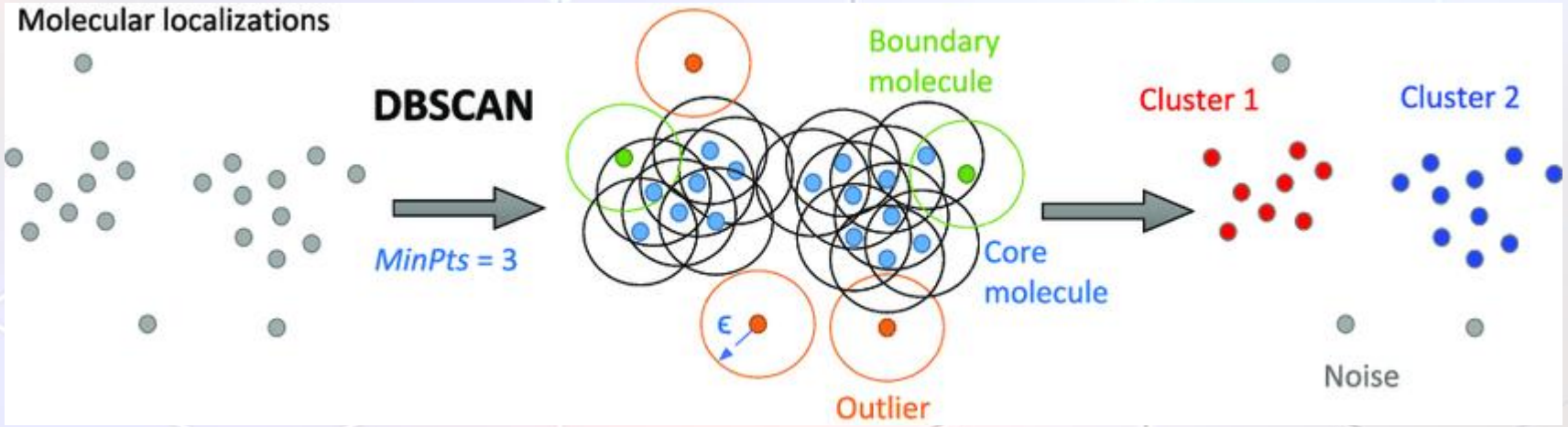
DBSCAN: Core, Border & Outlier

- A point is a **core point** if it has more than a specified number of points (MinPts) within Eps These are points that are at the interior of a cluster.
- A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point.
- A **noise point** is any point that is not a core point nor a border point.



ε = 1unit, MinPts = 5

DBSCAN: Clustering



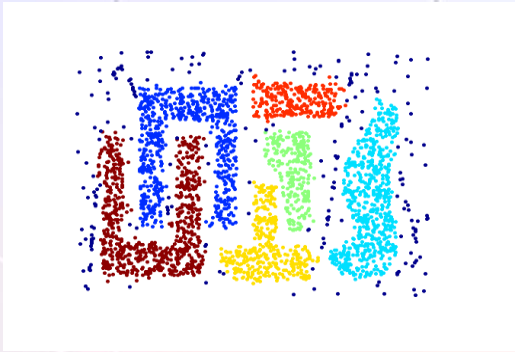
Given ε and *MinPts*, categorize the objects into three exclusive groups.

DBSCAN: Algorithm

```
Input: The data set D

Parameter: ε, MinPts

For each object p in D
      if p is a core object and not processed then
            C = retrieve all objects density-reachable from p
            mark all objects in C as processed
          report C as a cluster
      else mark p as outlier
      end if

End For
```

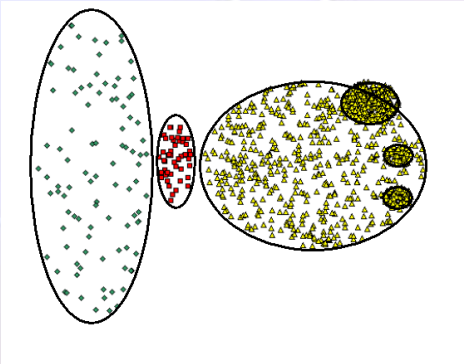
DBSCAN: Well- Not Well

Original Points

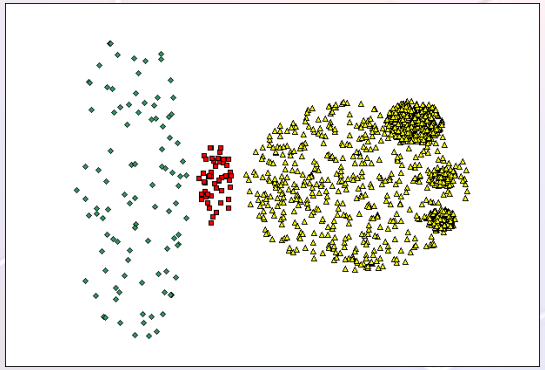


Clusters

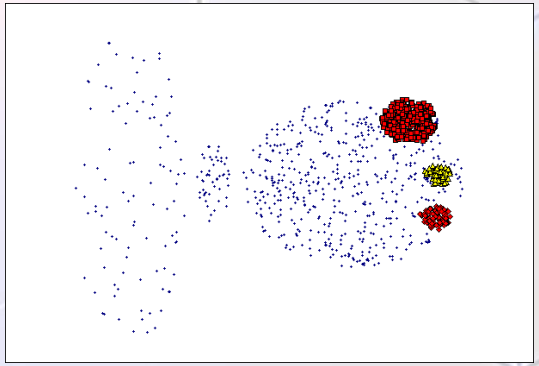
- Resistant to Noise
- Can handle clusters of different shapes and sizes



Original Points



(MinPts=4, Eps=9.92).



(MinPts=4, Eps=9.75)

- Cannot handle Varying densities
- sensitive to parameters

Thank You!

Hierarchical Clustering

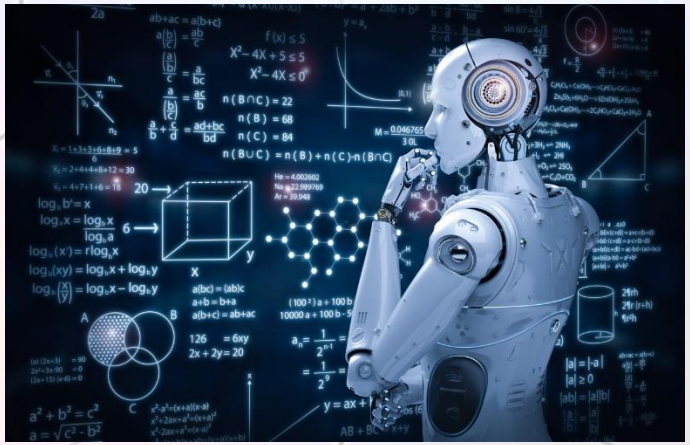
Agglomerative and Divisive hierarchical clustering

DBSCAN- Density based Clustering

Machine Learning

GTU#3170724

B.E - Semester VII



Unit 8: Unsupervised Learning

ANN- Artificial Neural Network

Lecture#33

Instructor:

Munira Topia

Computer Engineering Department

L.J. Institutes of Engineering and Technology

Outline

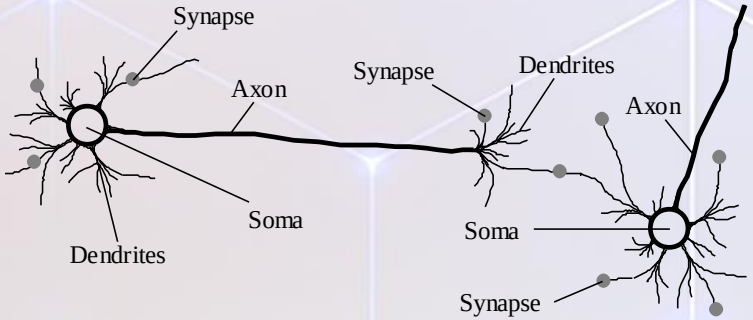
Artificial Neural Network

- Basic structure of neuron
- Activation function
- McCulloch-Pitts model of neuron

Artificial Neural Networks

Computational models inspired by the human brain:

- Algorithms that try to mimic the brain.
- Massively parallel, distributed system, made up of simple processing units (neurons).
- Synaptic connection strengths among neurons are used to store the acquired knowledge.
- Knowledge is acquired by the network from its environment through a learning process.



ANNs have been widely used in various domains for:

- Pattern recognition
- Function approximation
- Associative memory

When to Consider Neural Networks

- Input is high-dimensional discrete or raw-valued
- Output is discrete or real-valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of the result is not important

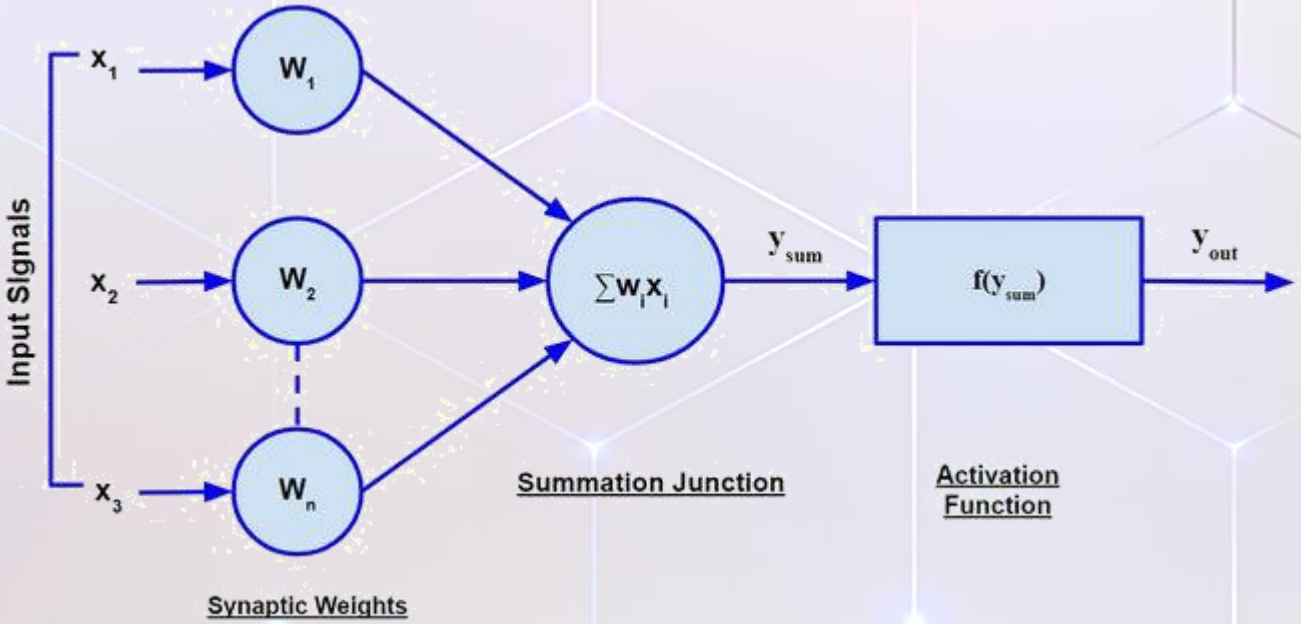
Examples:

- Speech phoneme recognition
- Image classification
- Financial prediction

Basic Structure of Neuron

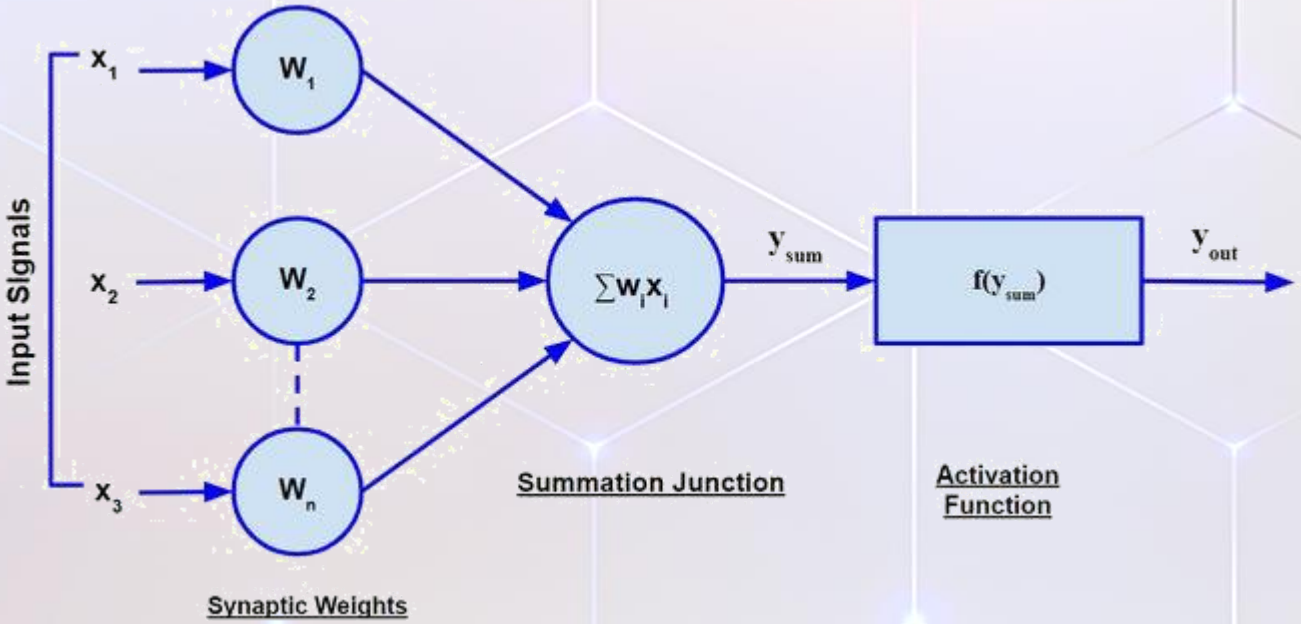
Each neuron consists of three major components:

- synaptic weights
- Summation
- Activation



Basic Structure of Neuron

- A set of **'i' synapses having weight w_i.**
- A signal x_i forms the input to the i-th synapse having weight w_i.
- The value of any weight may be positive or negative.
- A positive weight has an extraordinary effect, while a negative weight has an inhibitory effect on the output of the summation junction.



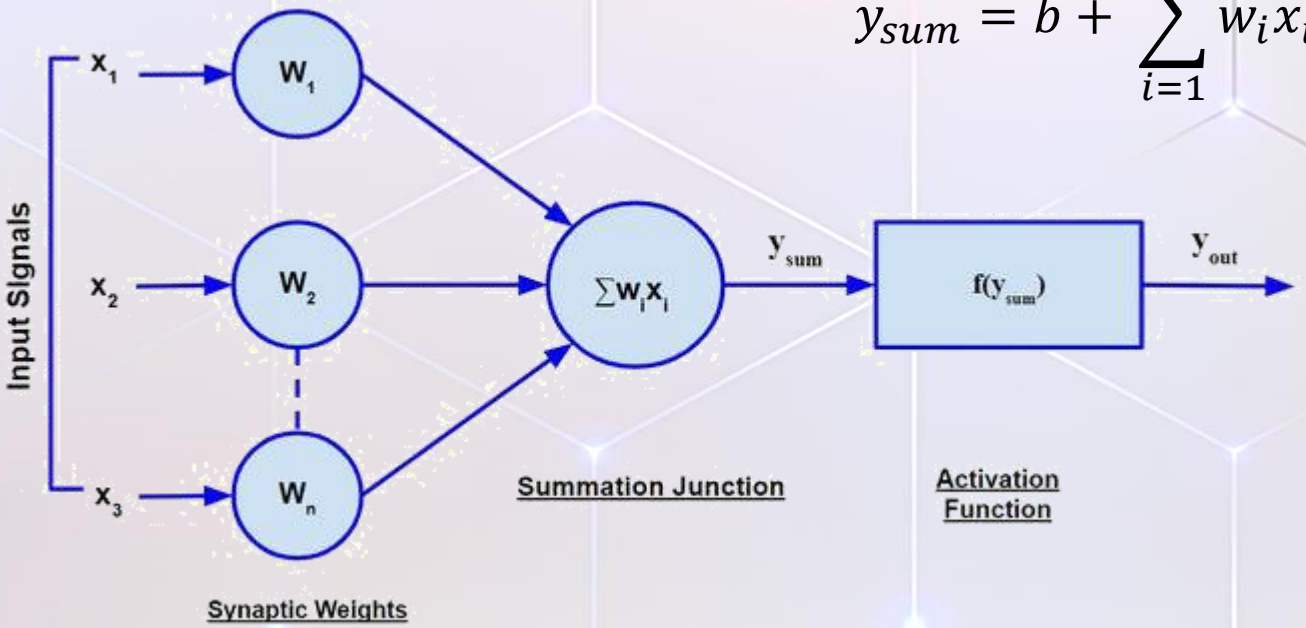
Basic Structure of Neuron

A **summation junction** for the input signals is weighted by the respective synaptic weight. Because it is a linear combiner or adder of the weighted input signals, the output of the summation junction can be expressed as follows:

$$y_{sum} = \sum_{i=1}^{n} w_i x_i$$

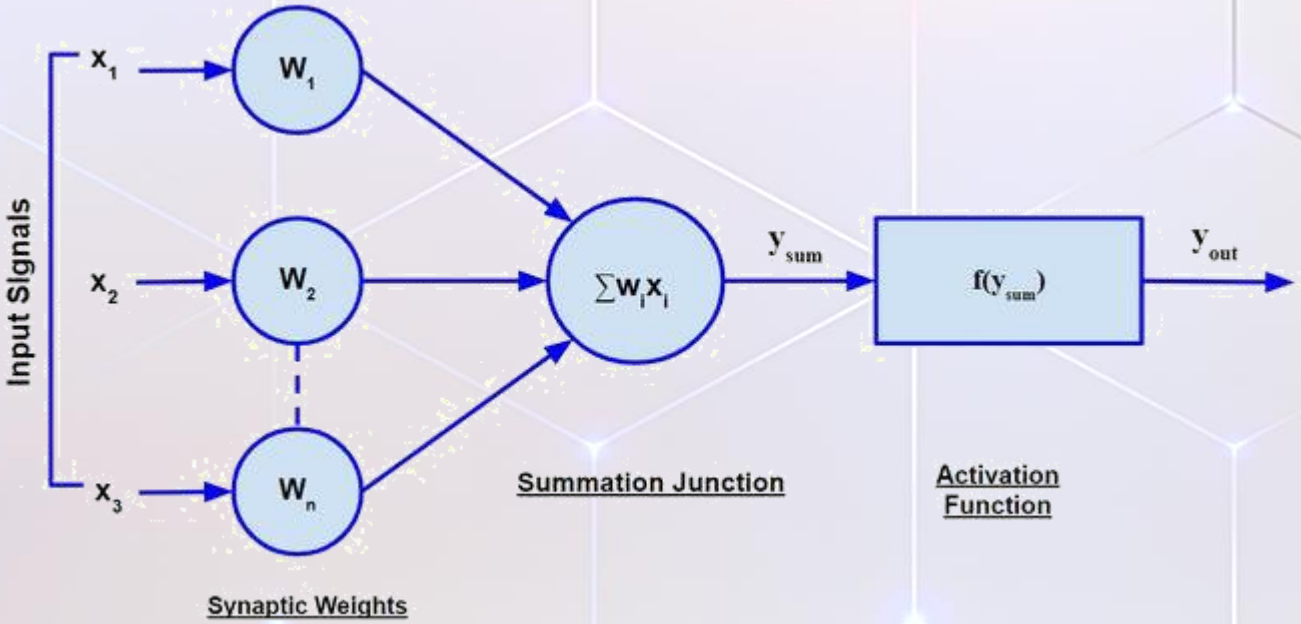
Typically a neural network also include a bias which adjust the input of the activation function. In this case of a ***bias b***, the value of y_{sum} would be:

$$y_{sum} = b + \sum_{i=1}^{n} w_i x_i$$



Basic Structure of Neuron

A ***threshold activation function*** (or simply the activation function, also known as squashing function) results in an output signal only when an input signal exceeding a specific threshold value comes as an input.



Types Of Activation Function in ANN

Identity Function: Identity function is used as an activation function for the input layer. It is a linear function having the form

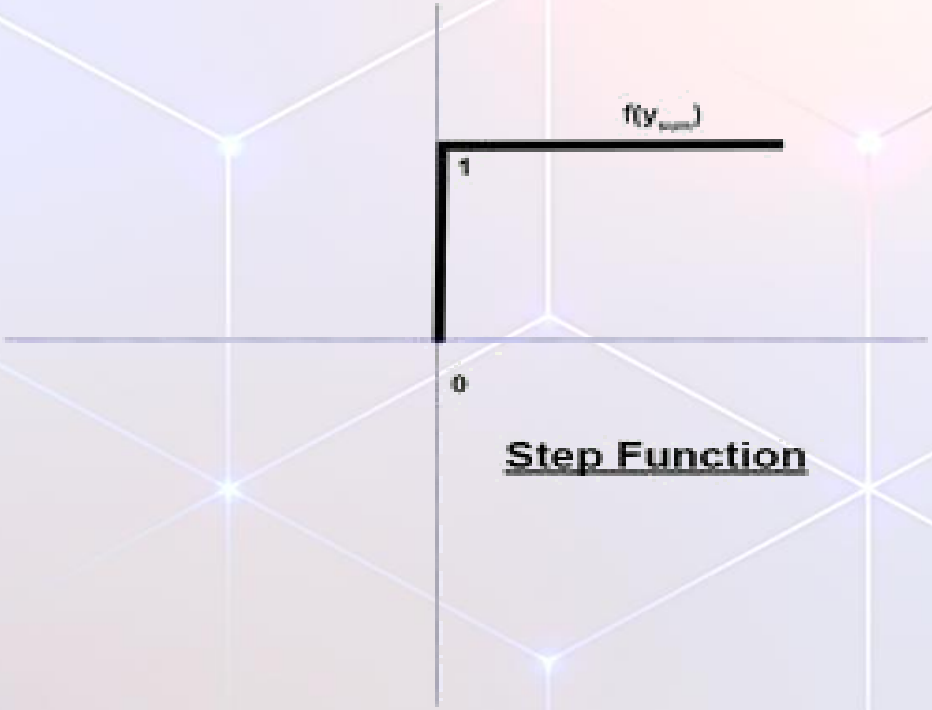
$$y_{out} = f(x) = x, \forall x$$

As obvious, the output remains the same as the input.

Types Of Activation Function in ANN

Step Function: It is a commonly used activation function. As depicted in the diagram, it gives 1 as output of the input is either 0 or positive. If the input is negative, it gives 0 as output. Expressing it mathematically,

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases}$$

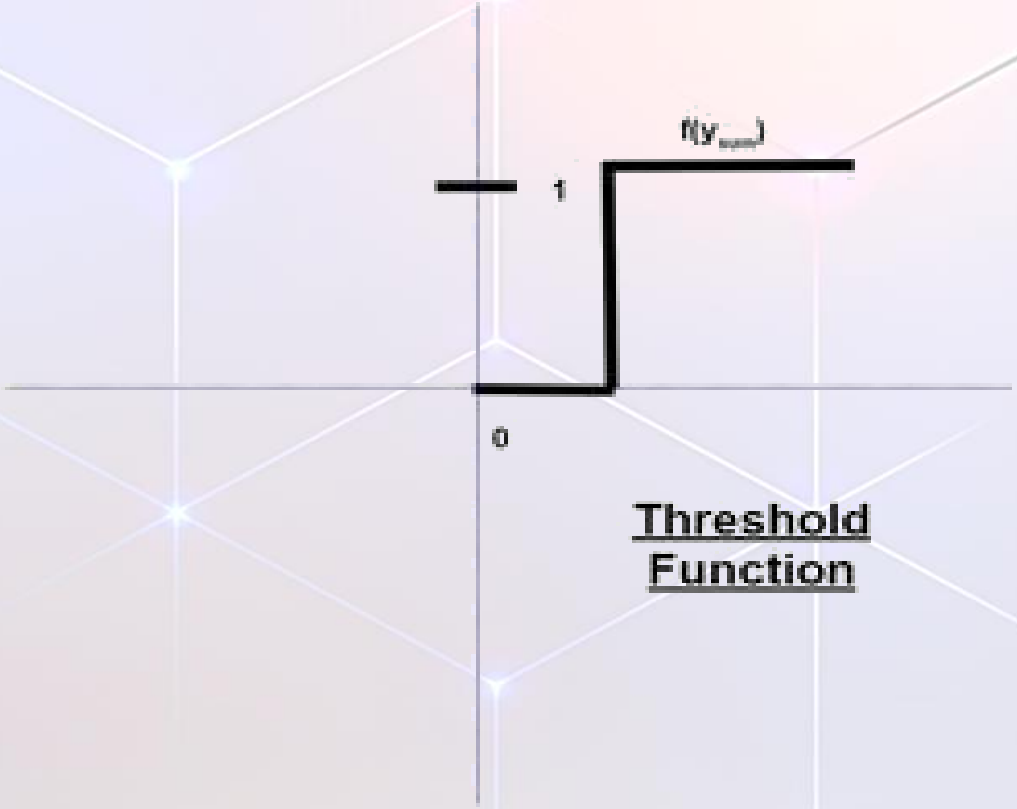


Step Function

Types Of Activation Function in ANN

Threshold function: is almost like the step function, with the only difference being a fact that θ is used as a threshold value instead of 0. Expressing mathematically,

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq \theta \\ 0, x < \theta \end{cases}$$



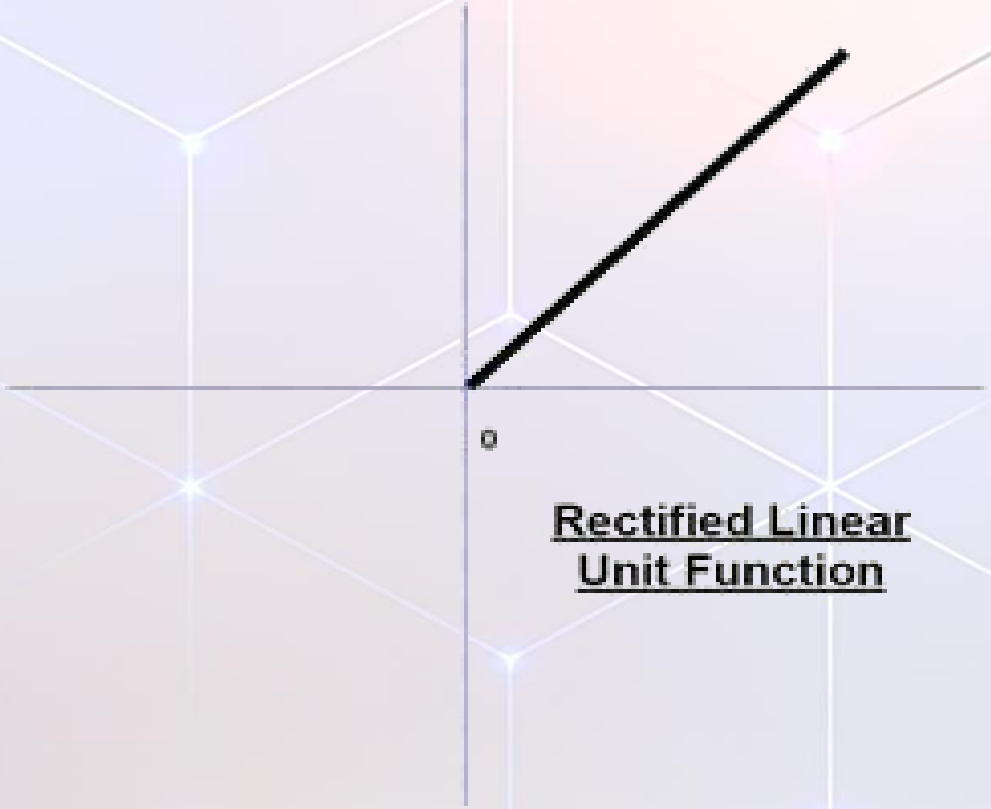
Threshold Function

Types Of Activation Function in ANN

ReLU (Rectified Linear Unit) Function: It is the most popularly used activation function in the areas of convolutional neural networks and deep learning. It is of the form:

$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$$

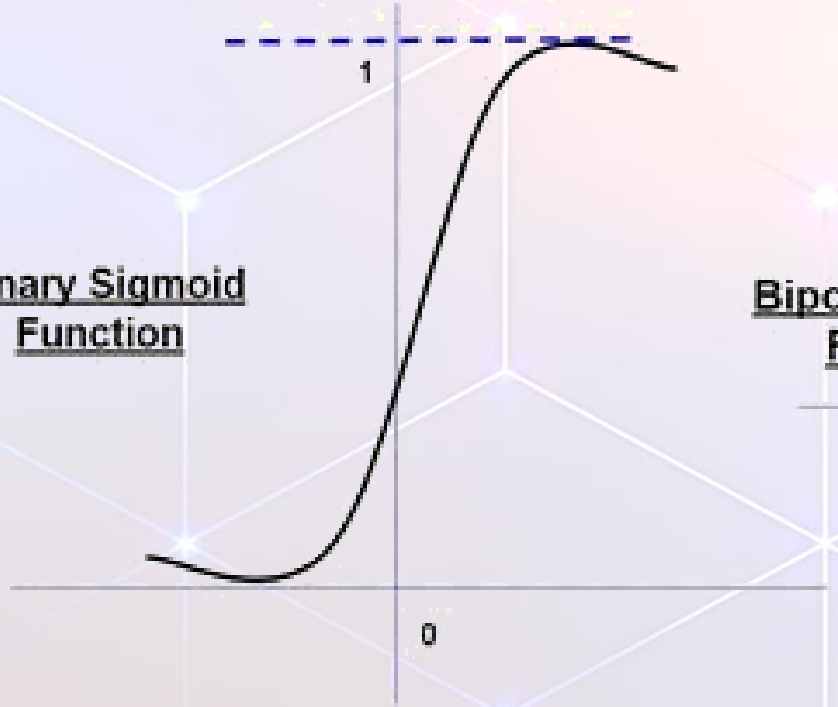
This means that f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero. This function is differentiable, except at a single point x = 0.



Rectified Linear Unit Function

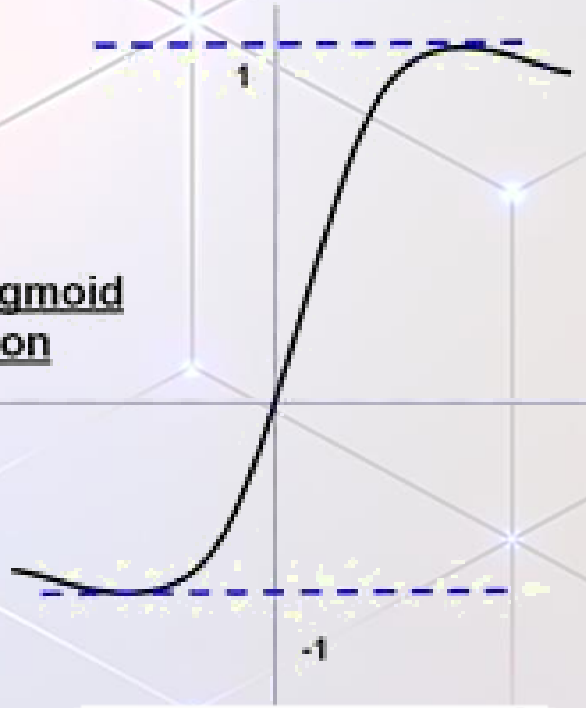
Types Of Activation Function in ANN

Sigmoid Function: It is by far the most commonly used activation function in neural networks. The need for sigmoid function stems from the fact that many learning algorithms require the activation function to be differentiable and hence continuous. There are two types of sigmoid function:



Binary Sigmoid Function

$$y_{out} = f(x) = \frac{1}{1+e^{-kx}}$$



Bipolar Sigmoid Function

$$y_{out} = f(x) = \frac{1-e^{-kx}}{1+e^{-kx}}$$

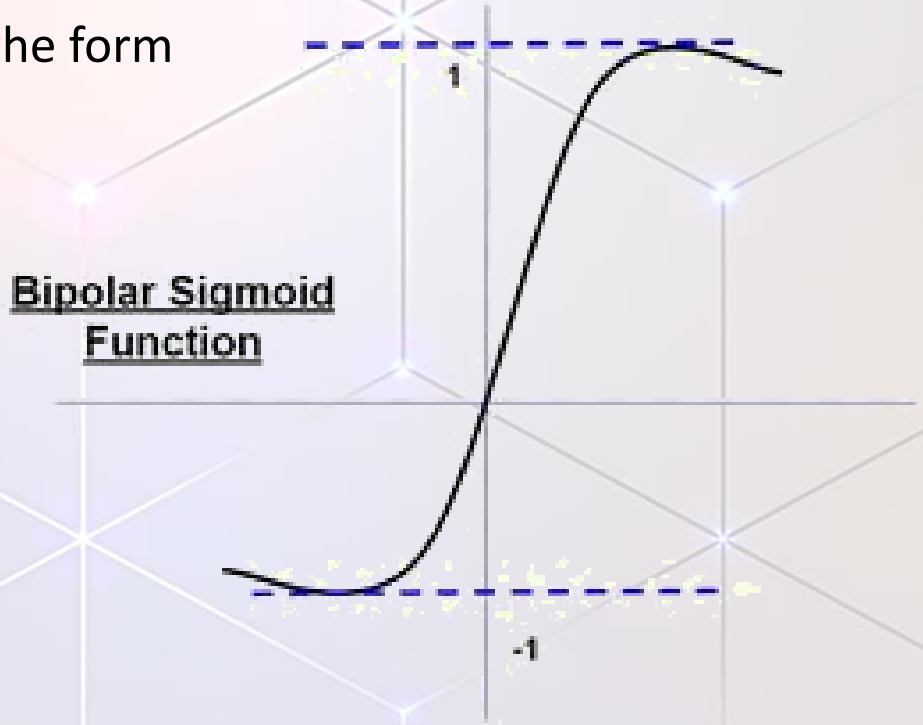
Types Of Activation Function in ANN

Hyperbolic Tangent Function: It is bipolar in nature. It is a widely adopted activation function for a special type of neural network known as Backpropogation Network.

This function is similar to the bipolar sigmoid function.

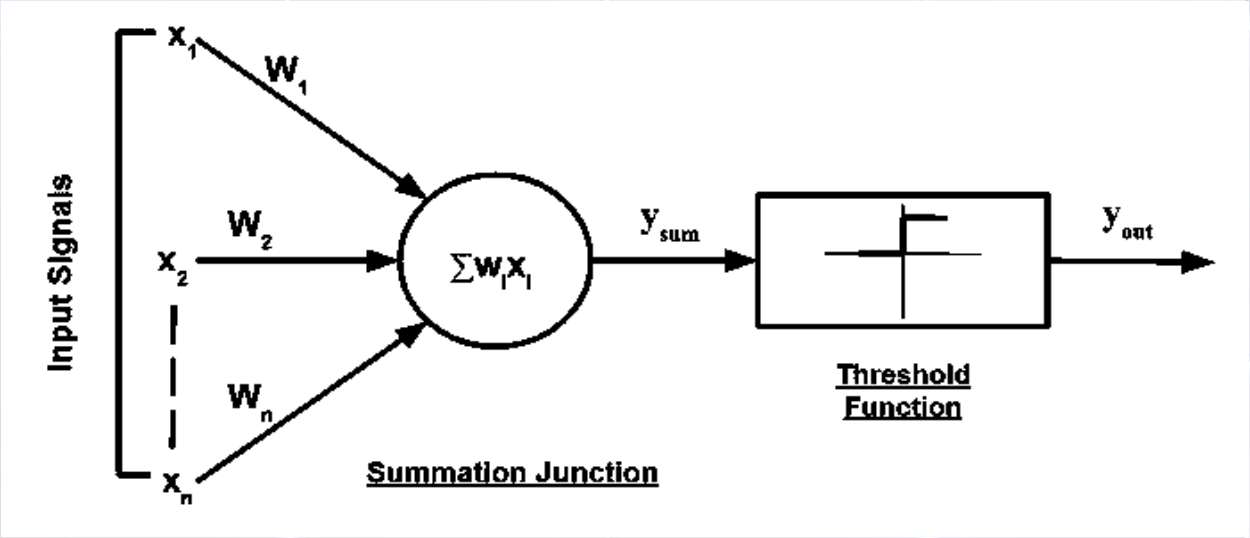
The hyperbolic tangent function is of the form

$$y_{out} = f(x)\frac{e^x - e^- x}{e^x + e^- x}$$



McCulloch-Pitts Model of Neuron

- was the earliest ANN model, has only two types of inputs — Excitatory and Inhibitory.
- The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude.
- The inputs of the McCulloch-Pitts neuron could be either 0 or 1.
- It has a threshold function as an activation function.
- So, the output signal y_{out} is 1 if the input y_{sum} is greater than or equal to a given threshold value, else 0.



McCulloch-Pitts Model of Neuron

- can be used to design logical operations.
- John carries an umbrella if it is sunny or if it is raining. There are four given situations:
  - First scenario: It is not raining, nor it is sunny
  - Second scenario: It is not raining, but it is sunny
  - Third scenario: It is raining, and it is not sunny
  - Fourth scenario: It is raining as well as it is sunny
- To analyse the situations using the McCulloch-Pitts neural model
  - X1: Is it raining?
  - X2 : Is it sunny?

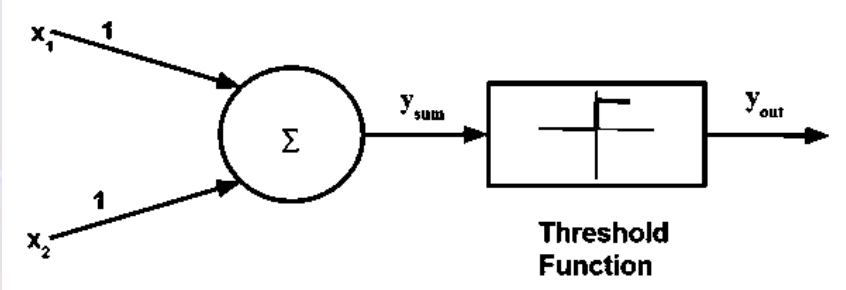
Situation	x_1	x_2	y_{sum}	y_{out}
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

McCulloch-Pitts Model of Neuron

- can be used to design logical operations.
- John carries an umbrella if it is sunny or if it is raining. There are four given situations:
  - First scenario: It is not raining, nor it is sunny
  - Second scenario: It is not raining, but it is sunny
  - Third scenario: It is raining, and it is not sunny
  - Fourth scenario: It is raining as well as it is sunny
- To analyse the situations using the McCulloch-Pitts neural model
  - X1: Is it raining?
  - X2 : Is it sunny?

Situation	x_1	x_2	y_{sum}	y_{out}
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

We can use the value of both weights X1 and X2 as 1 and a threshold function as 1.
So, the neural network model will look like



$$y_{sum} = \sum_{i=1}^{2} w_i x_i$$

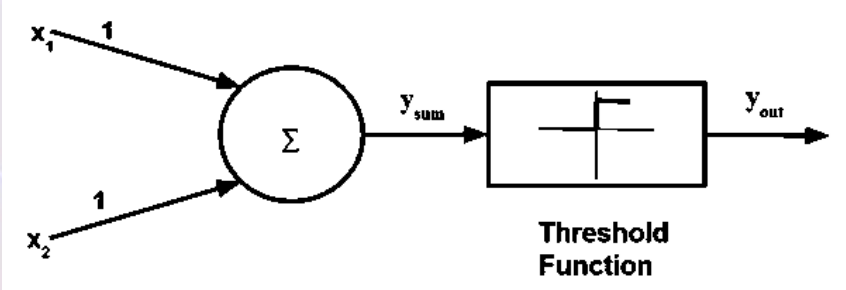
$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 1 \\ 0, x < 1 \end{cases}$$

McCulloch-Pitts Model of Neuron

- can be used to design logical operations.
- John carries an umbrella if it is sunny or if it is raining. There are four given situations:
  - First scenario: It is not raining, nor it is sunny
  - Second scenario: It is not raining, but it is sunny
  - Third scenario: It is raining, and it is not sunny
  - Fourth scenario: It is raining as well as it is sunny
- To analyse the situations using the McCulloch-Pitts neural model
  - X1: Is it raining?
  - X2 : Is it sunny?

Situation	x_1	x_2	y_{sum}	y_{out}
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

It is concluded that in the situations where the value of y_{out} is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in scenarios 2, 3 and 4.



$$y_{sum} = \sum_{i=1}^{2} w_i x_i$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, x \geq 1 \\ 0, x < 1 \end{cases}$$

Thank You!

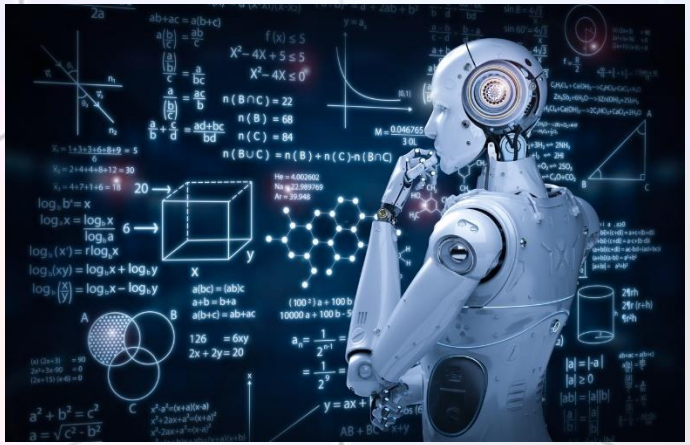
Artificial Neural Network

- Basic structure of neuron
- Activation function
- McCulloch-Pitts model of neuron

Machine Learning

GTU#3170724

B.E - Semester VII



Unit 8: Unsupervised Learning

ANN2- Artificial Neural Network

Lecture#34

Instructor:

Munira Topia

Computer Engineering Department

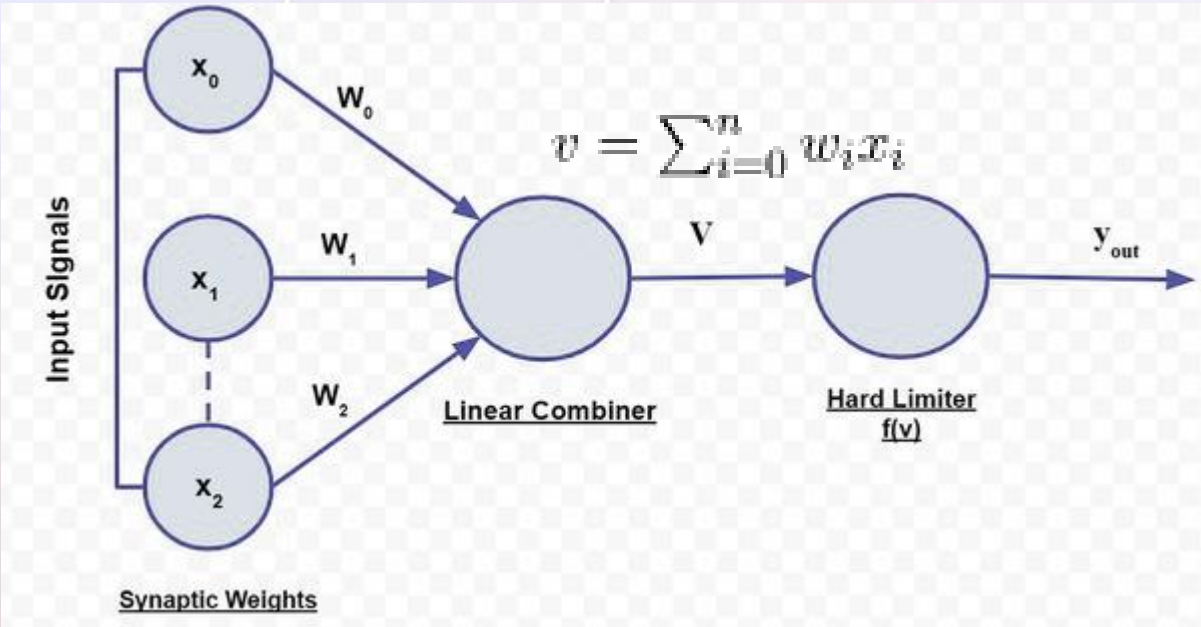
L.J. Institutes of Engineering and Technology

Outline

- Rosenblatt's Perceptron
- Multi-layer Perceptron
- Architectures of Neural Network
- Learning Process of Neural Network

Rosenblatt's Perceptron

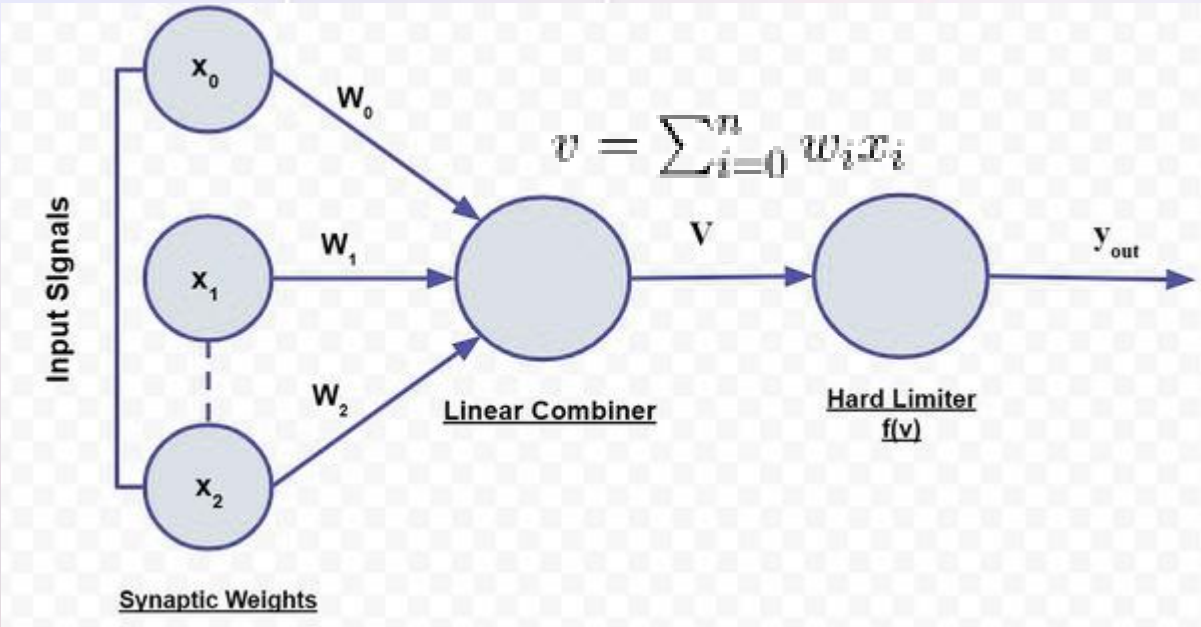
- Rosenblatt's perceptron is built around the McCulloch-Pitts neural model.
- The perceptron receives a set of input x_1, x_2,….., x_n.
- The linear combiner or the adder mode computes the linear combination of the inputs applied to the synapses with synaptic weights being w_1, w_2,……,w_n.
- Then, the hard limiter checks whether the resulting sum is positive or negative.
- If the input of the hard limiter node is positive, the output is +1, and if the input is negative, the output is -1.



$$y_{out} = f(v) = \begin{cases} +1, v > 0 \\ -1, v < 0 \end{cases}$$

Rosenblatt's Perceptron

- The objective of the perceptron is o classify a set of inputs into two classes c_1 and c_2.
- This can be done using a very simple decision rule – assign the inputs to c_1 if the output of the perceptron i.e. y_{out} is +1 and c_2 if y_{out} is -1.
- So for an n-dimensional signal space i.e. a space for 'n' input signals, the simplest form of perceptron will have two decision regions, resembling two classes, separated by a hyperplane defined by: $\sum_{i=0}^{n} w_i x_i = 0$



$$v = \sum_{i=0}^{n} w_i x_i$$

$$y_{out} = f(v) = \begin{cases} +1, v > 0 \\ -1, v < 0 \end{cases}$$

Rosenblatt's Perceptron

- This model implements the functioning of a single neuron that can solve linear classification problems through very simple learning algorithms. **Rosenblatt Perceptrons** are considered as the first generation of **neural networks** (the network is only compound of one neuron).

- Two inputs x1 and x2 forms an equation of a line:

$$w_0x_0 + w_1x_1 + w_2x_2 = 0$$
$$\text{or}, w_0 + w_1x_1 + w_2x_2 = 0[\because x_0 = 1]$$

Rosenblatt's Perceptron

- This model implements the functioning of a single neuron that can solve linear classification problems through very simple learning algorithms. **Rosenblatt Perceptrons** are considered as the first generation of **neural networks** (the network is only compound of one neuron).

- Two inputs x1 and x2 forms an equation of a line:

$$w_0x_0 + w_1x_1 + w_2x_2 = 0$$
$$\text{or}, w_0 + w_1x_1 + w_2x_2 = 0 [\because x_0 = 1]$$

Let weights be -2, ½ and ¼ for x0, x1 and x2 respectively, forms an equation:

$$-2 + ½x_1 + ¼\, x_2 = 0$$
$$\text{or}, 2x_1 + x_2 = 8$$

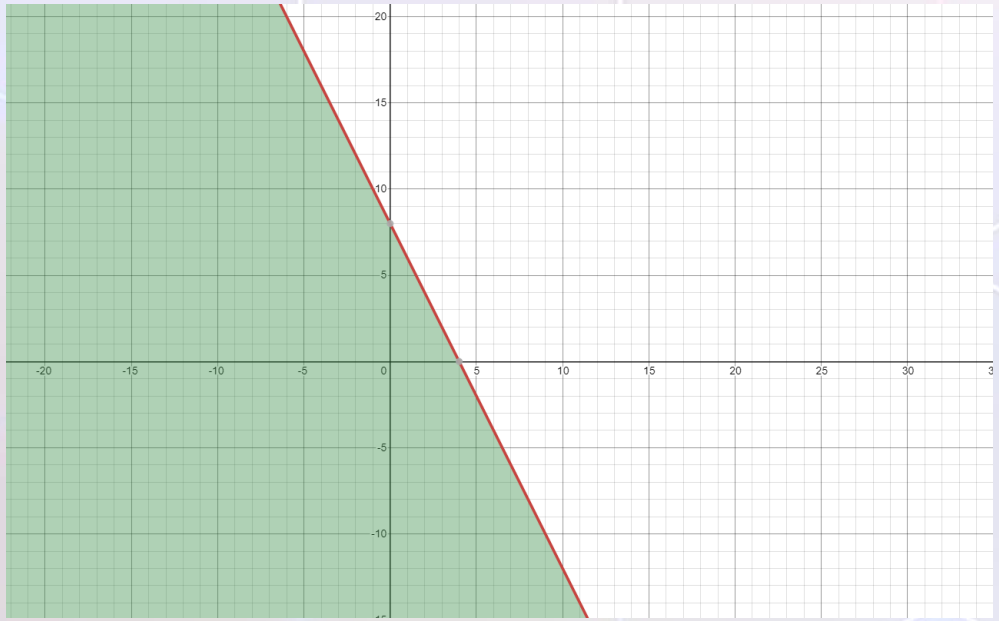
Rosenblatt's Perceptron

- Two inputs x1 and x2 forms an equation of a line:

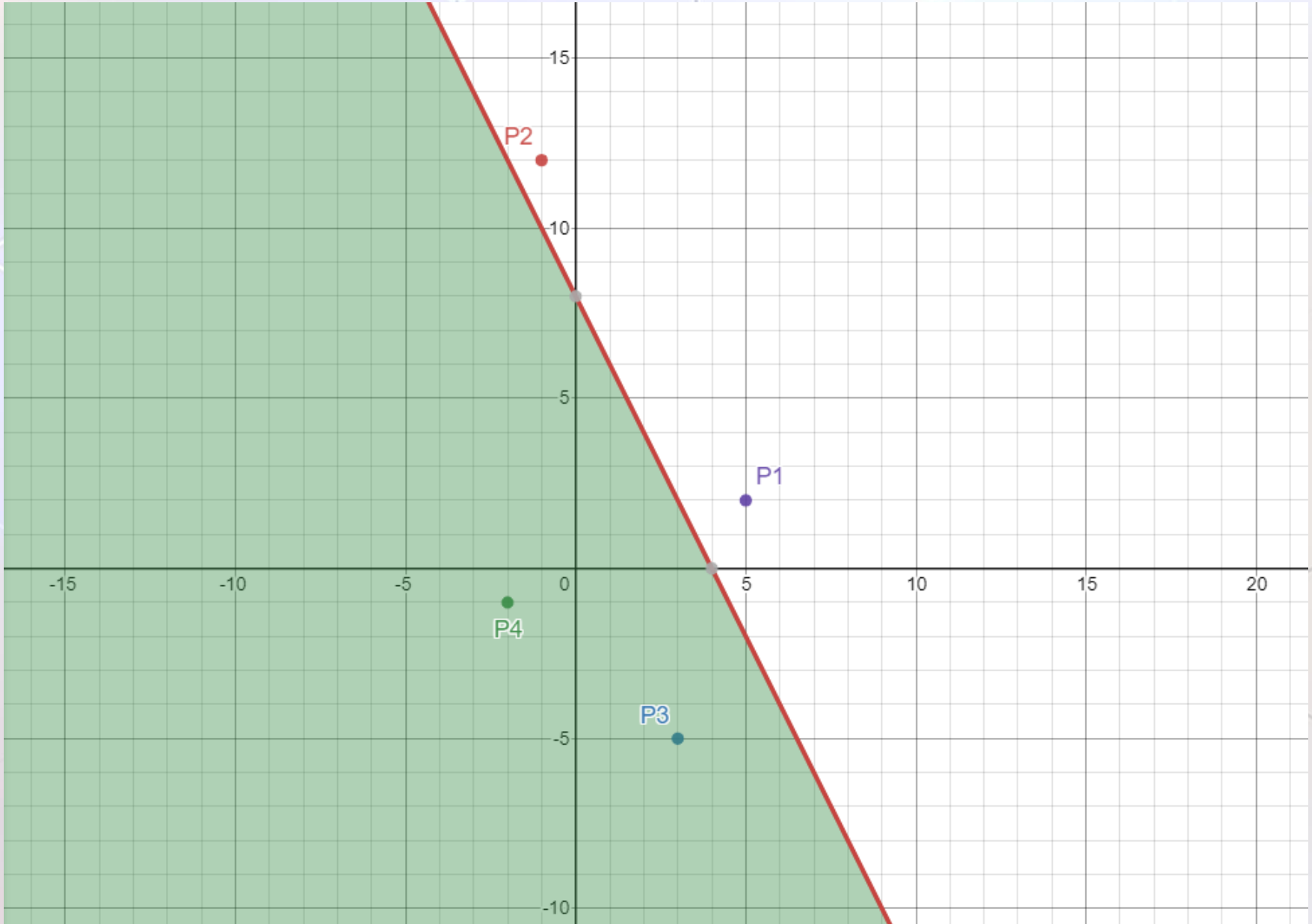
$$w_0x_0 + w_1x_1 + w_2x_2 = 0$$
$$\text{or}, w_0 + w_1x_1 + w_2x_2 = 0[\because x_0 = 1]$$

Let weights be -2, ½ and ¼ respectively, forms an equation:

$$-2 + \tfrac{1}{2}x_1 + \tfrac{1}{4}x_2 = 0$$
$$\text{or}, 2x_1 + x_2 = 8$$



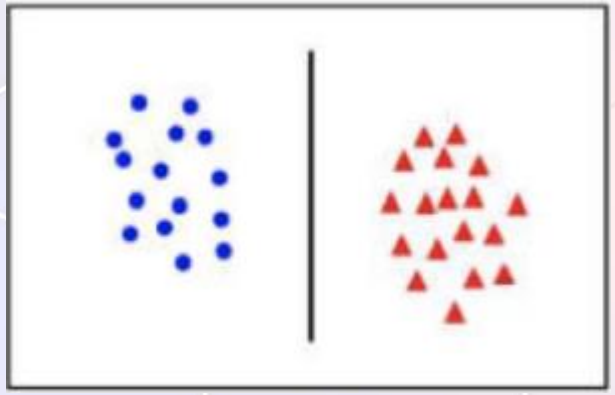
Rosenblatt's Perceptron

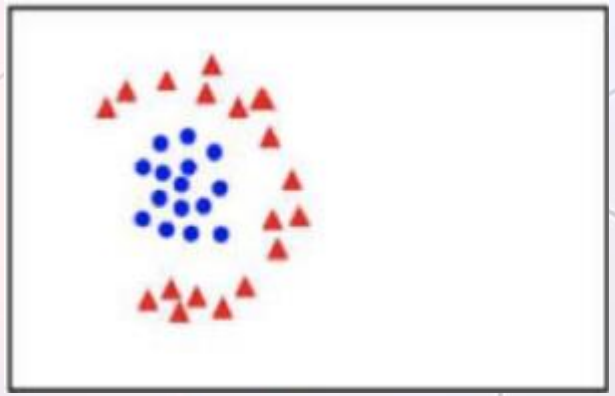


P1=(5,3)
P2=(-1,12)
P3=(3,-5)
P4=(-2,1)

Rosenblatt's Perceptron

- This simple single neuron model has the main limitation of not being able to solve non-linear separable problems.

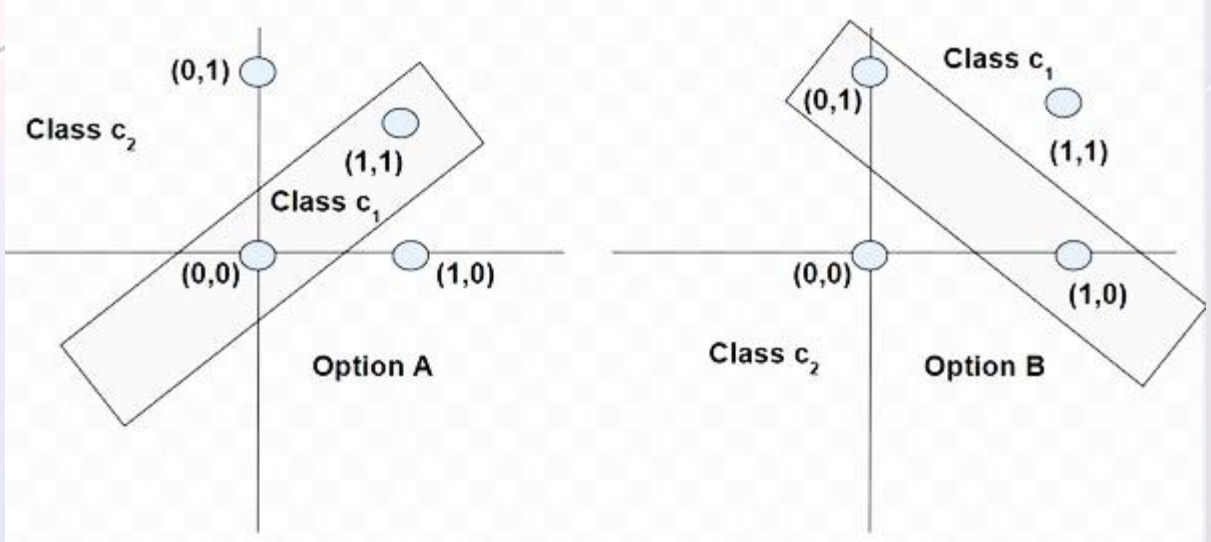




Multi-layer Perceptron

- Works with data set which is not linearly separable.
- Proposed by Minsky and Papert (in 1969).

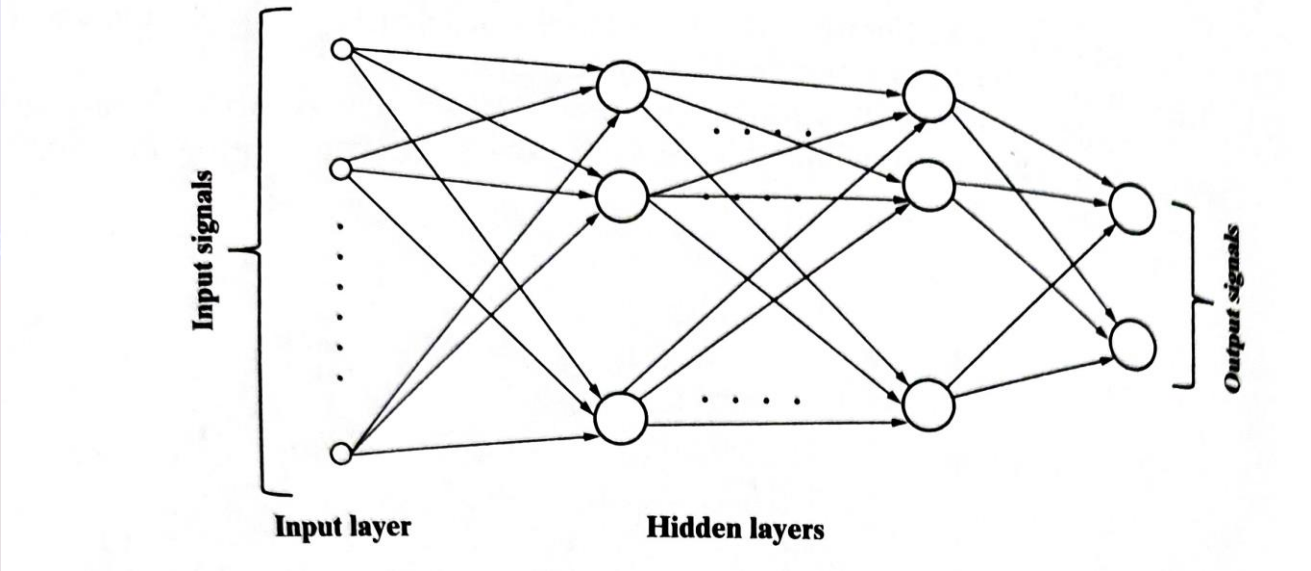
x_1	x_2	x_1 XOR x_2	Class
1	1	0	c_2
1	0	1	c_1
0	1	1	c_1
0	0	0	c_2



Multi-layer Perceptron

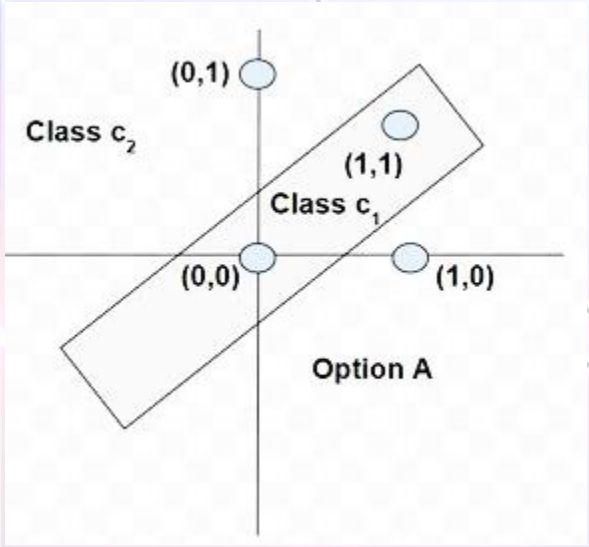
This is the philosophy used to design the multi-layer perceptron model.

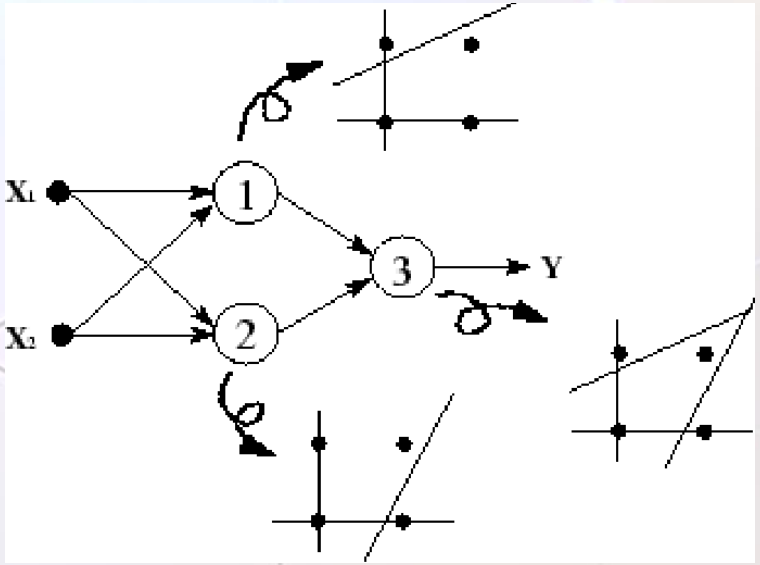
- The neural network contains one or more intermediate layers between the input and output nodes, which are hidden from both input and output nodes.
- Each neuron in the network includes a non-linear activation function that is differentiable.
- The neurons in each layer are connected with some or all the neurons in the previous layer.

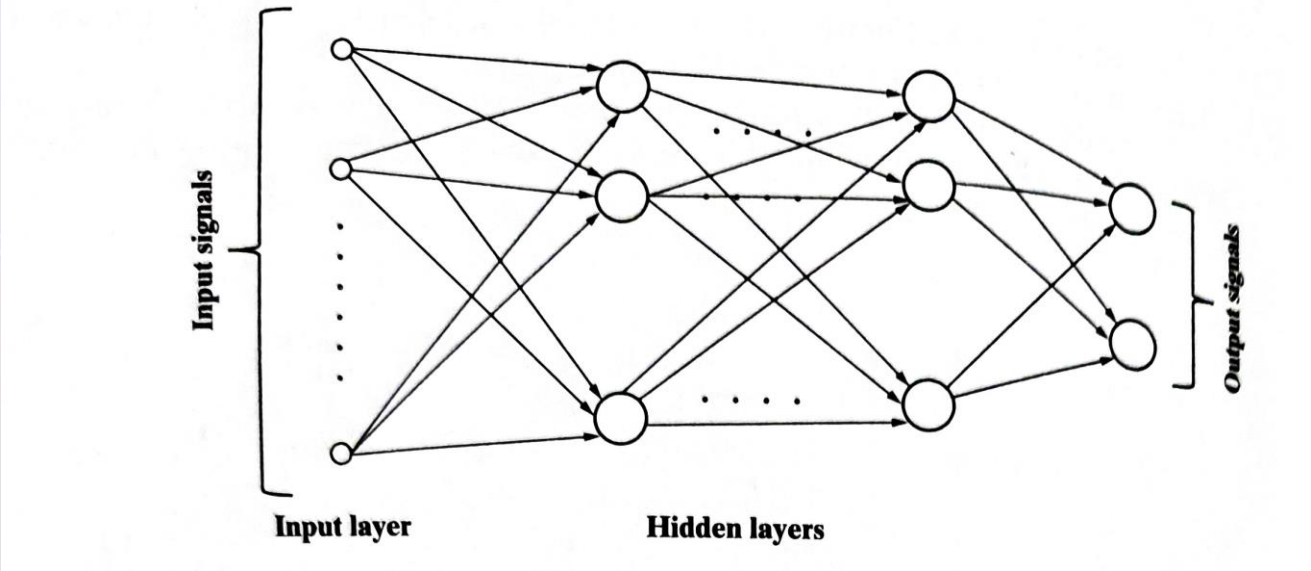


Multi-layer Perceptron – XOR

x_1	x_2	x_1 XOR x_2	Class
1	1	0	c_2
1	0	1	c_1
0	1	1	c_1
0	0	0	c_2







Architecture of Neural Network

ANN is a computational system consisting of many interconnected units called **artificial neurons**. The connection between artificial neurons can transmit a signal from one neuron to another.

*So, there are multiple possibilities for connecting the neurons based on which the **architecture** we are going to adopt for a specific solution. Some permutations and combinations are as follows:*

Architecture of Neural Network

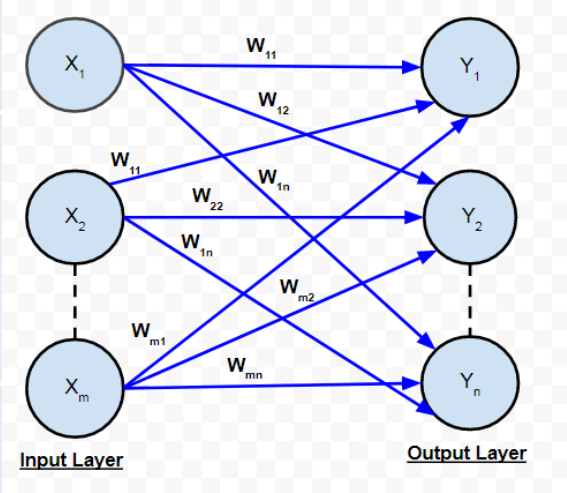
ANN is a computational system consisting of many interconnected units called **artificial neurons**. The connection between artificial neurons can transmit a signal from one neuron to another.

*So, there are multiple possibilities for connecting the neurons based on which the **architecture** we are going to adopt for a specific solution. Some permutations and combinations are as follows:*

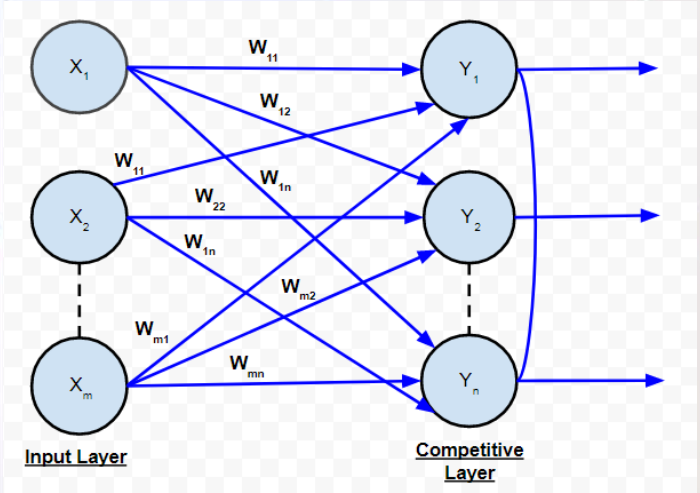
- There may be just two layers of neuron in the network – the input and output layer.
- There can be one or more intermediate **'hidden'** layers of a neuron.
- The neurons may be connected with all neurons in the next layer and so on …..

Architecture of Neural Network

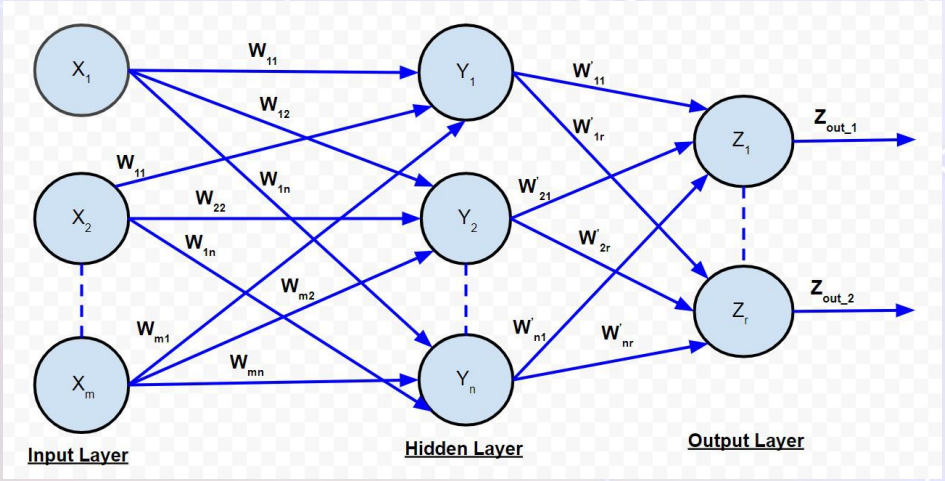
A. Single-layer Feed Forward Network:



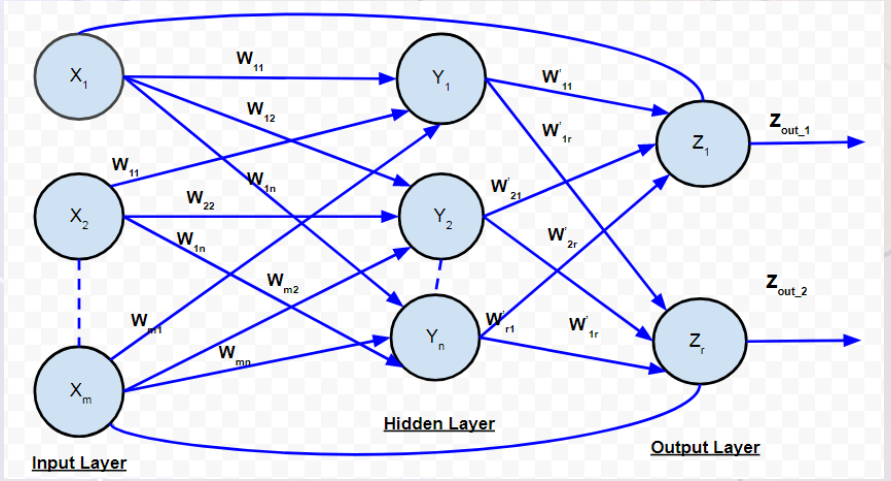
C. Competitive Network:



B. Multi-layer Feed Forward Network:



D. Recurrent Network:



Learning Process In ANN

Learning process in ANN mainly depends on four factors, they are:

1. **The number of layers in the network** (Single-layered or multi-layered)

2. **Direction of signal flow** (Feed forward or recurrent)

3. **Number of nodes in layers:**
   The number of node in the input layer is equal to the number of features of the input data set. The number of output nodes will depend on possible outcomes i.e. the number of classes in case of supervised learning. But the number of layers in the hidden layer is to be chosen by the user.

   A larger number of nodes in the hidden layer, higher the performance but too many nodes may result in over fitting as well as increased computational expense.

4. **Weight of Interconnected Nodes:**
   Deciding the value of weights attached with each interconnection between each neuron so that a specific learning problem can be solved correctly is quite a difficult problem by itself.

Backpropagation

In this method, the **difference in output values of the output layer and the expected values, are** propagated **back from the output layer to the** preceding **layers.**

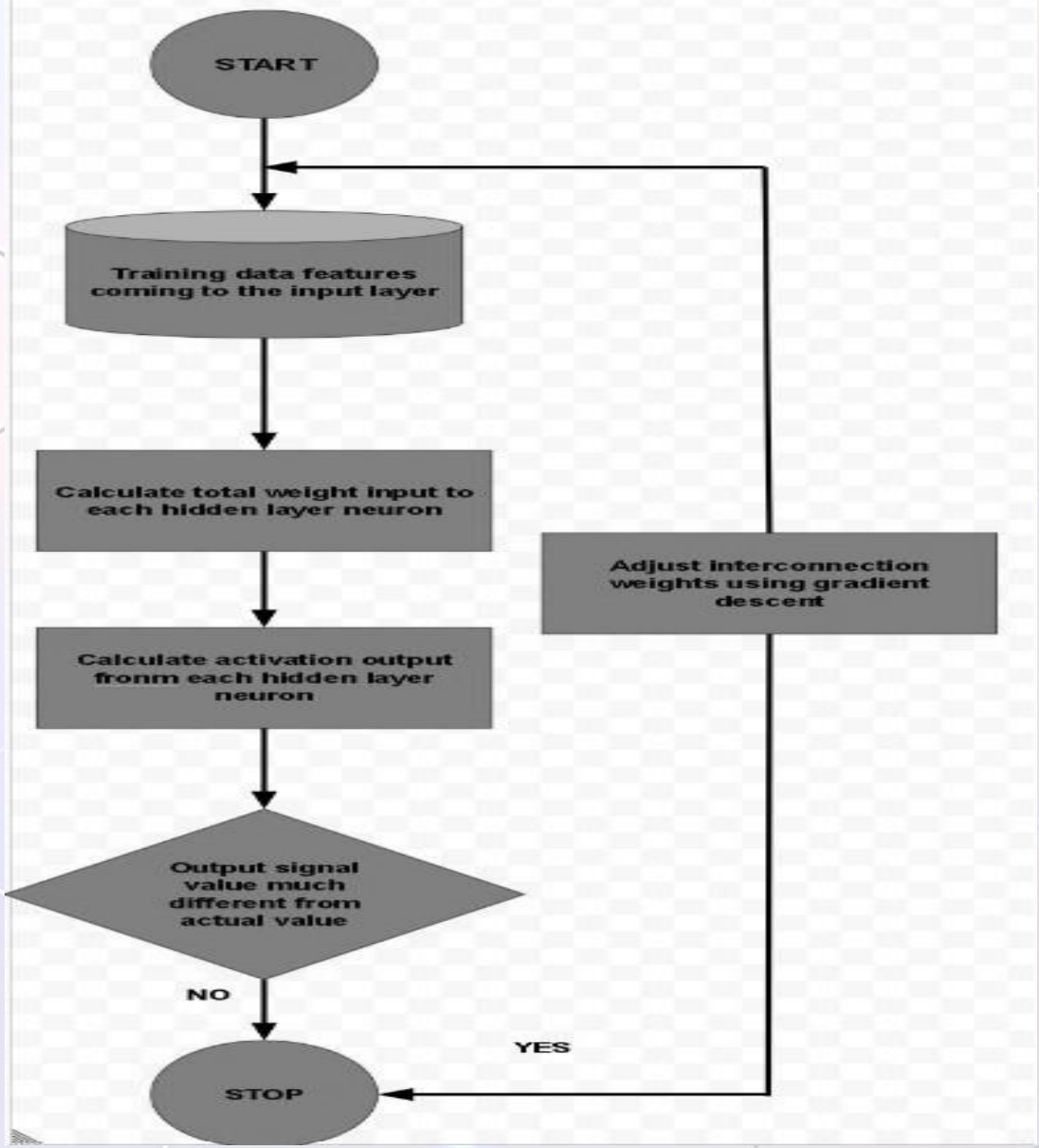
Hence, the algorithm implementing this method is known as BACK PROPAGATION **i.e. propagating the errors back to the** preceding **layers.**

- The backpropagation algorithm is applicable for multi-layer feed-forward network.
- It is a supervised learning algorithm which continues adjusting the weights of the connected neurons with an objective to reduce the deviation of the output signal from the target output.
- This algorithm consists of multiple iterations, **known as epochs.**

Backpropagation

Each epoch consists of two phases:

- **Forward Phase:** Signal flow from neurons in the input layer to the neurons in the output layer through the hidden layers. The weights of the interconnections and activation functions are used during the flow. In the output layer, the output signals are generated.
- **Backward Phase:** Signal is compared with the expected value. The computed errors are propagated backwards from the output to the preceding layer. The error propagated back are used to adjust the interconnection weights between the layers.



Thank You!

- Rosenblatt’s Perceptron
- Multi-layer Perceptron
- Architectures of Neural Network
- Learning Process of Neural Network