UNSUPERVISED LEARNING

Deep learning

INTRODUCTION TO CLUSTERING

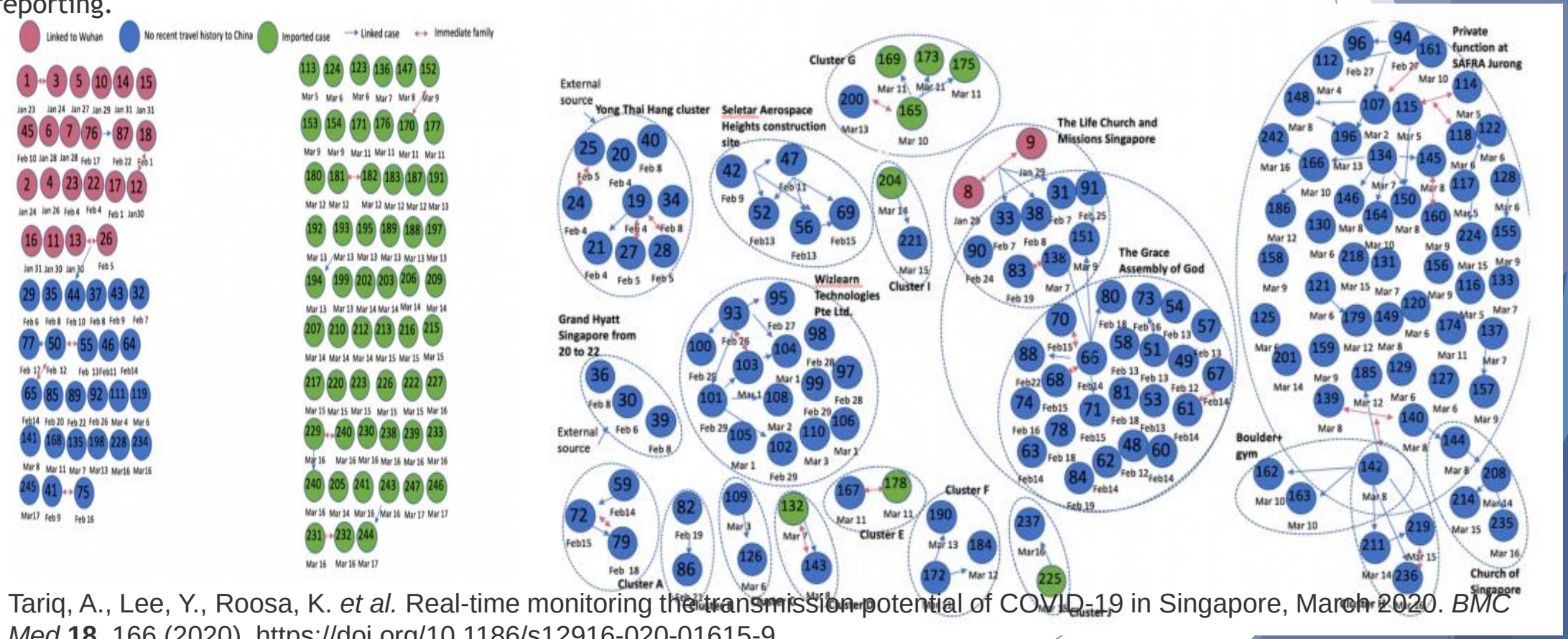
- ☑ Clustering is a popular technique where data is grouped based on the similarity of the data-points
- ☑ It is an unsupervised learning problem
- ☑ It tries to predict which class or category some entity belongs to, based on its features
- ☑ Clustering determines the intrinsic grouping among the unlabeled data present; so it is helpful for pattern discovery or knowledge discovery

Real life application

- Segmentation for customer data for various data analytics applications
- Fraud detection for cyber crime frauds
- Clustering in search engines - is the backbone behind the search engines
- Clustering Algorithm can be used effectively in Wireless Sensor Network's based application

UNSUPERVISED LEARNING - CLUSTERING

Cluster network of the cases in Singapore for the COVID-19 global epidemic as of February 19, 2020 . The green circles represent the imported cases, pink circles represent direct-linked to Wuhan, where as the blue circles represent cases with no travel history to China. The larger dotted circles represent the COVID-19 disease clusters. Each blue arrow represents the direction in which the disease was transmitted. Pink arrows represent immediate family. Dates below the circles are the dates of case reporting.



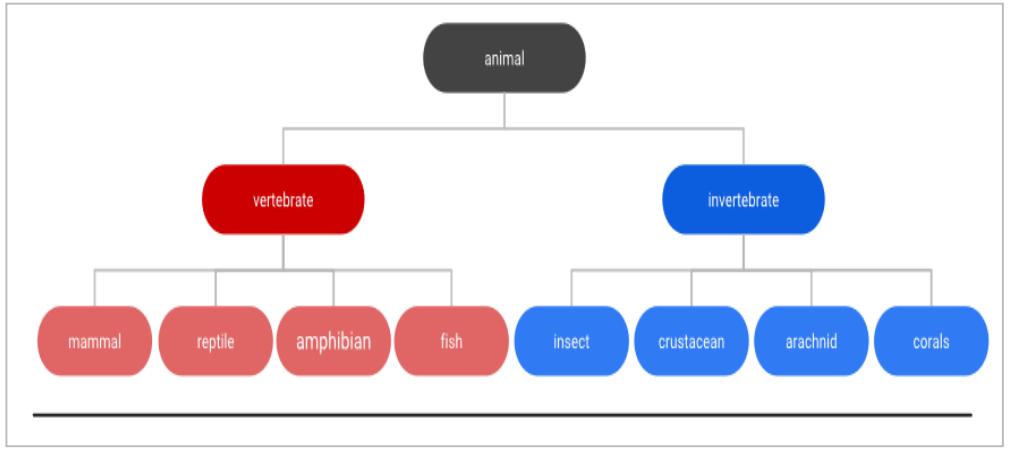
Tariq, A., Lee, Y., Roosa, K. *et al.* Real-time monitoring the transmission potential of COVID-19 in Singapore, March 2020. *BMC Med* **18,** 166 (2020). https://doi.org/10.1186/s12916-020-01615-9

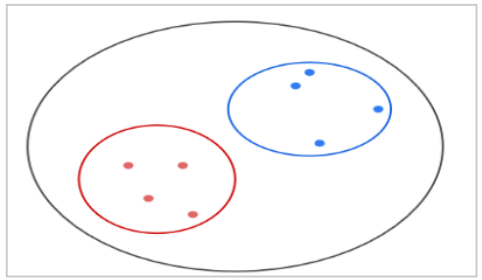
TYPES OF CLUSTERING

Hierarchical Clustering

- It is a tree of clusters
- It is well suited for hierarchical data, such as taxonomies

Example of a hierarchical tree clustering animal:

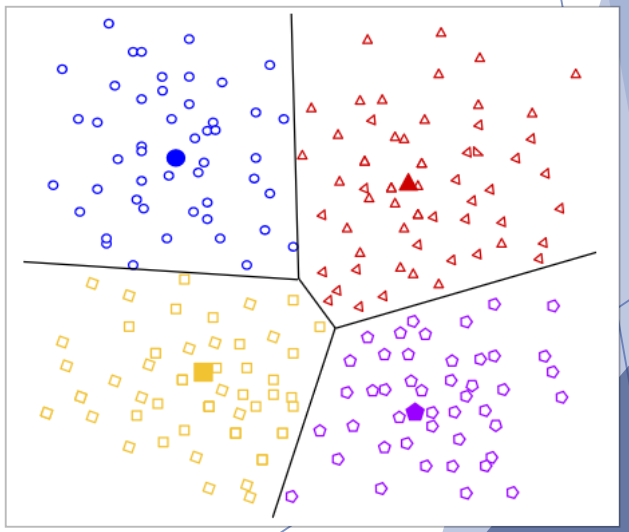




Centroid-based Clustering

- It organizes the data into non-hierarchical clusters
- It is an efficient, effective, and simple clustering algorithm
- **k-means** is the most widely-used centroid-based clustering algorithm

Example of a centroid-based clustering:

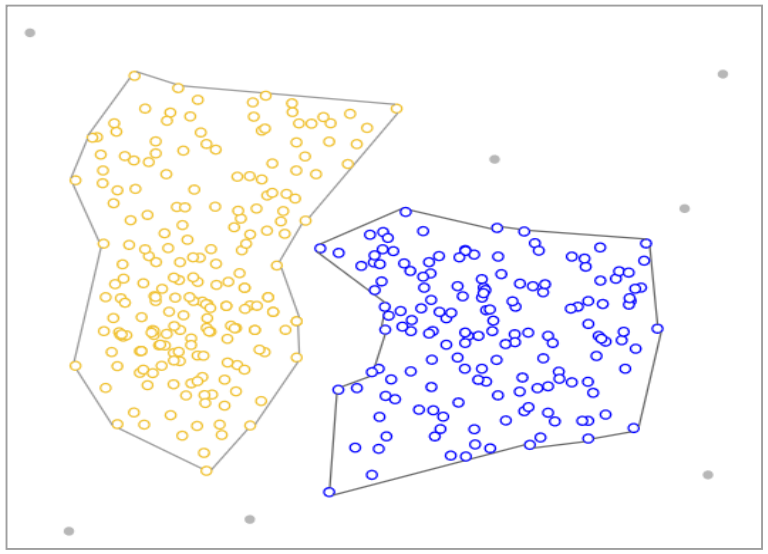


TYPES OF CLUSTERING

Density-based Clustering

- It connects areas of high example density into clusters
- It has difficulty with data of varying densities and high dimensions
- This algorithms do not assign outliers to clusters

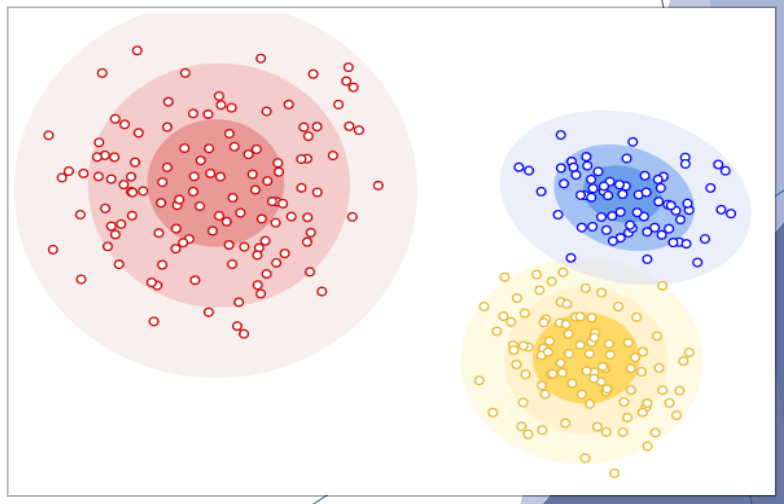
Example of a density-based clustering:



Distribution-based Clustering

- In this approach, data is composed of distributions, such as Gaussian distributions or normal distributions, that is , this algorithm clusters data into Gaussian distributions
- As distance from the distribution's center increases, the probability that a point belongs to the distribution decreases

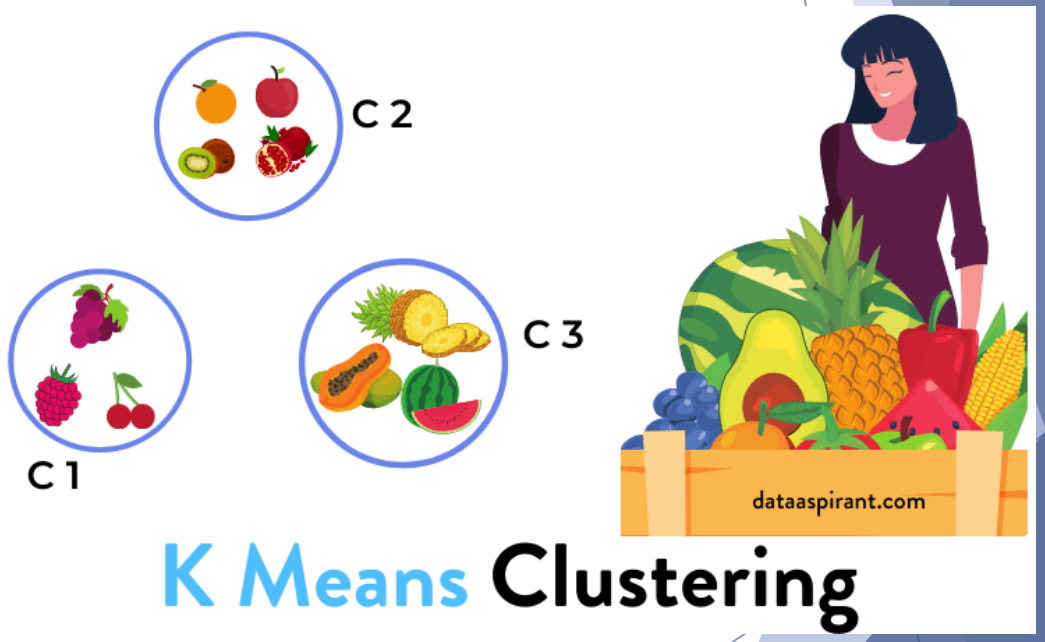
Example of a distribution-based clustering



INTRODUCTION TO K-MEANS CLUSTERING

- K-Means clustering algorithm aims to partition “n” observation into “k” clusters in which each observation belongs to the nearest cluster based on Euclidean distance

```
In [2]: import time
        %matplotlib inline
        import numpy as np
        from IPython import display
        import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
```



GENERATE DATA

Simulation of two set of clusters cluster_1 and cluster_2 with different centers:

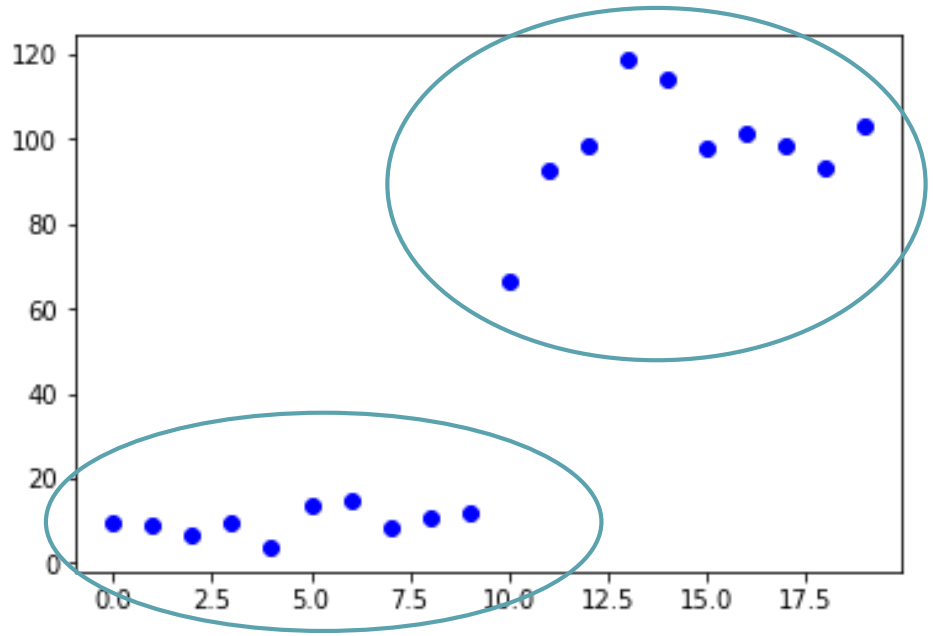
- ***np.random.normal()*** function helps to specify the number of samples along with the required mean and variance
- ***np.hstack()*** can be used to join two list into a single list
- In the example given below, cluster_1 will hold ten values (third parameter), with mean 10 (first parameter) and variance 5 (second parameter)
- Second cluster will also hold ten values but the mean and variance are 100 and 10 respectively

```
In [3]: cluster_1 = np.random.normal(10, 5, 10)
        cluster_2 = np.random.normal(100, 10, 10)
        data = np.hstack([cluster_1, cluster_2])
        timestamp = list(range(len(data)))
        fig = plt.figure()

        <matplotlib.figure.Figure at 0x77d06fe940>
```

PLOTTING DATA POINTS

```
In [ ]: plt.plot(timestamp, data, 'bo')
```



There are two set of clusters:
Cluster-1
and
Cluster-2

FITTING THE MODEL

In [7]:
```
data_kmeans = np.column_stack([timestamp, data])
init = [(10,20), (10,20)]
kmeans = KMeans(n_clusters=2, init=np.array(init), n_init=1)
fit = kmeans.fit(data_kmeans)
fit
```

Out[7]:
```
KMeans(algorithm='auto', copy_x=True,
    init=array([[10, 20],
       [10, 20]]), max_iter=300, n_clusters=2,
    n_init=1, n_jobs=1, precompute_distances='auto', random_state=None,
    tol=0.0001, verbose=0)
```

CENTROIDS USING SKLEARN

- **Kmean.cluster_centers_** is used to identify the centroids of the clusters
- 'Number of clusters' is one of the important parameter that a Kmeans() function takes
- Deciding upon optimal number of clusters is very crucial because the results will be primarily driven by this parameter
- The optional parameter call ***init***, is the initial guess for the cluster centroids

GETTING CENTROIDS

- The K-Means algorithm will compute the centroids for each cluster
- The coordinates of the centroids can be retrieved using ***kmeans.cluster_centers_***

```
In [8]: centroids_sklearn = kmeans.cluster_centers_
        centroids_sklearn

Out[8]: array([[  4.5       ,   9.78573431],
               [ 14.5       ,  98.46308902]])
```

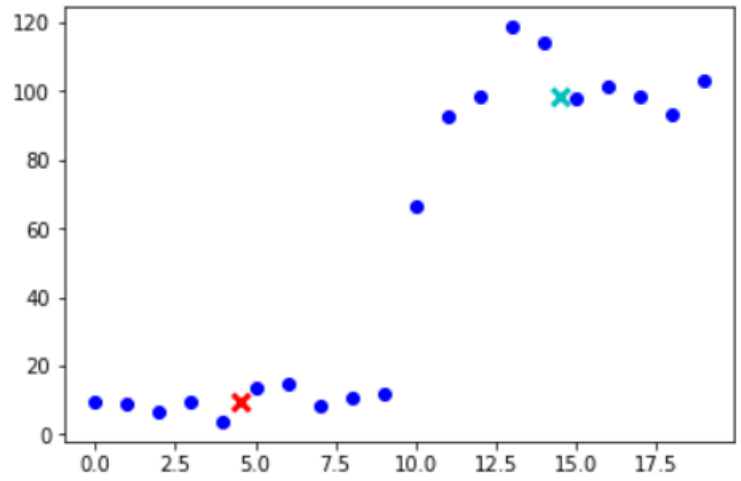
- The coordinates for the first & second clusters are (4.5,9.7) and (14.5, 98.46) respectively

PLOTTING CENTROIDS

Add the centroids to the plot for better understanding the results of k-means

In [9]:
```
plt.plot(timestamp, data, 'bo')
plt.plot(centroids_sklearn[0][0], centroids_sklearn[0][1], 'rx', mew=12, ms=2)
plt.plot(centroids_sklearn[1][0], centroids_sklearn[1][1], 'cx', mew=12, ms=2)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x77d1fc2710>]
```



STEPS TO CREATE K-MEANS ALGORITHMS

- Initialize the centroids with random coordinates for each cluster
- Loop in and follow the given steps for each data point
  - Calculate the Euclidean distance between the data point and each centroid separately
  - Identify the nearest centroid for which the distance is small
  - Categorize or add the current data point under the identified cluster
  - Re-calculate the centroids by computing the average of the data points under each cluster
- Repeat the above process multiple times, say 7 to 10 times
- For every repeat, update the initial centroids based on the previous loop
- After repeating the process multiple times, it returns the final centroids for each cluster

COMPARING RESULTS

Centroid returned by sklearn.cluster Vs. Centroid return by manual method

The centroids returned via sklearn.cluster method is (4.5, 9.7) & (14.5, 98.4)

```
In [16]: print (centroids_sklearn)

[[  4.5         9.78573431]
 [ 14.5        98.46308902]]
```

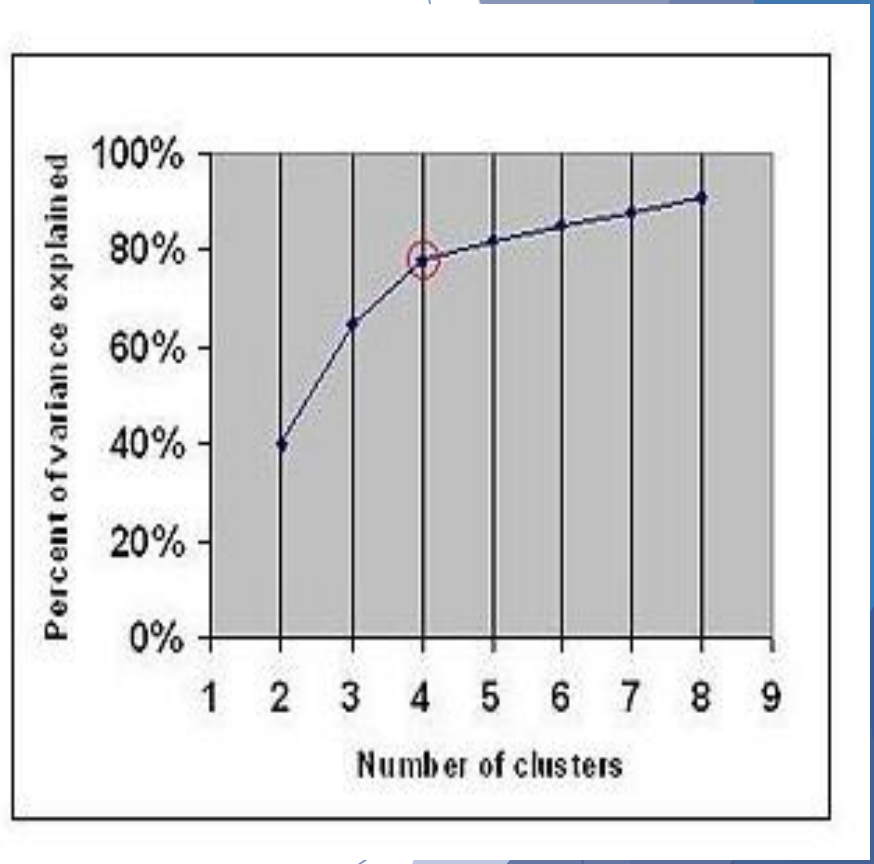
The centroids returned via manual method is also (4.5, 9.7) & (14.5, 98.4)

```
In [17]: print (centroids_manual)

[[14.5, 98.463089017970702], [4.5, 9.7857343138850545]]
```

OPTIMAL NUMBER OF CLUSTERS

- The Elbow method is used to determine the optimal number of clusters
- Explained Variance: The percentage of variance explained as a function of number of clusters
- One should choose a number of cluster, so that adding another cluster doesn't give much better modelling of the data
- If one plots the percentage of variance explained by the clusters against the number of clusters, the first few clusters will add much information (explains lot of variance), but at some point the marginal gain in explained variance will drop, giving an angle in the graph.
- The number of clusters is chosen at this point, hence the name "elbow criterion".



HANDS-ON PRACTICE

Lets Try

DEEP LEARNING

Introduction to Neural Networks

DEEP LEARNING

Deep learning is a subset of machine learning in Artificial Intelligence (AI) that has **networks capable of learning unsupervised from data that is unstructured or unlabeled.**

Deep Learning is the **most exciting and powerful branch** of Machine Learning.

Deep Learning models can be used for a variety of complex tasks:

- ❑ **Artificial Neural Networks for Regression and Classification**
- ❑ **Convolutional Neural Networks for Computer Vision**
- ❑ **Recurrent Neural Networks for Time Series Analysis**
- ❑ **Self Organizing Maps for Feature Extraction**
- ❑ **Deep Boltzmann Machines for Recommendation Systems**
- ❑ **Auto Encoders for Recommendation Systems**

EXPONENTIAL GROWTH IN STORAGE

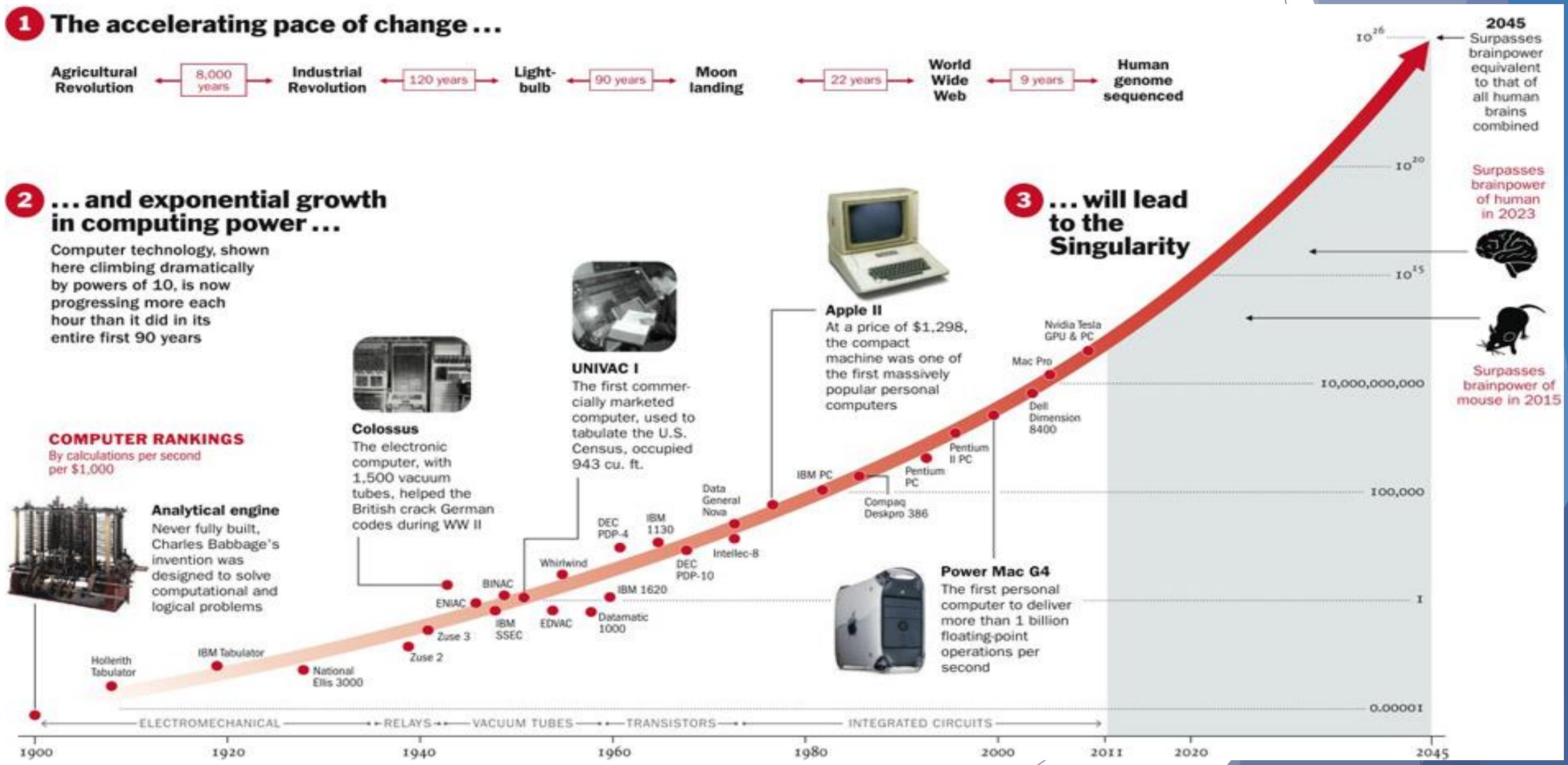
STORAGE LIMITS

Estimates based on bacterial genetics suggest that digital DNA could one day rival or exceed today's storage technology.

	Hard disk	Flash memory	Bacterial DNA
Read–write speed (μs per bit)	~3,000–5,000	~100	<100
Data retention (years)	>10	>10	>100
Power usage (watts per gigabyte)	~0.04	~0.01–0.04	$<10^{-10}$
Data density (bits per cm^3)	$\sim10^{13}$	$\sim10^{16}$	$\sim10^{19}$

Source: nature.com

EXPONENTIAL GROWTH IN COMPUTING POWER

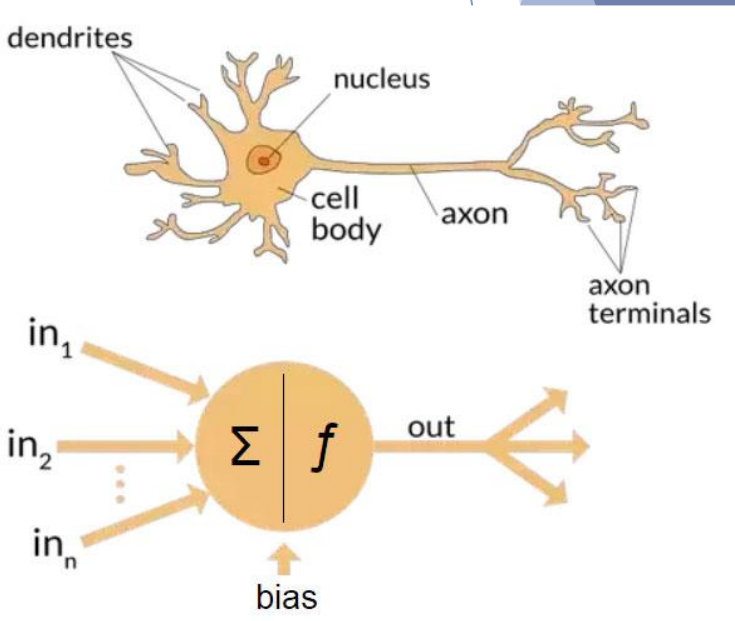


NEURAL NETWORKS - INTRODUCTION

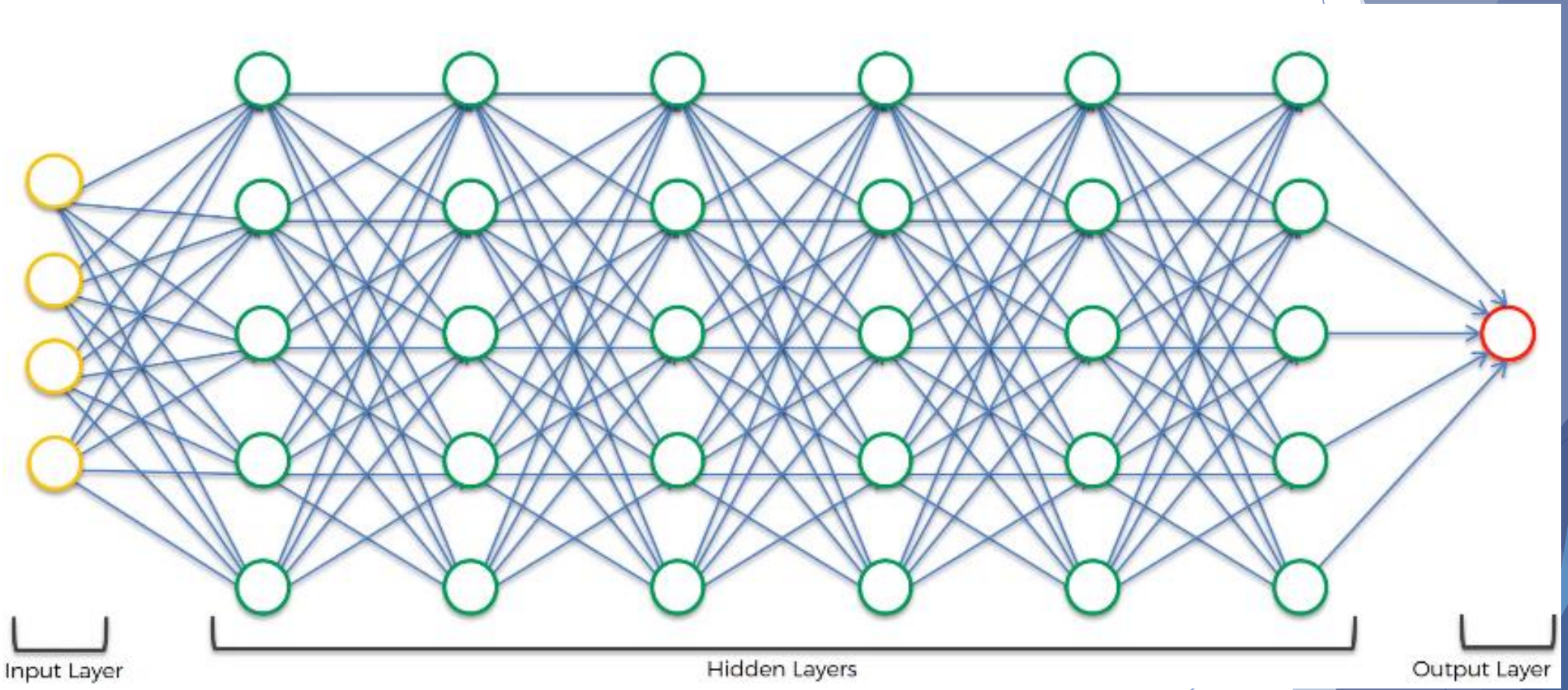
- Neural network learning methods provide a robust approach to approximating real-valued, discrete-valued, and vector-valued target functions.
- For certain types of problems, such as learning to interpret complex real-world sensor data, artificial neural networks are among the most effective learning methods currently known.
- The study of artificial neural networks (ANNs) has been inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons.

Artificial Neural Network

- It is a computational model inspired by the way biological neural networks in the human brain process information
- It has a lot of excitement and breakthrough in
  - Machine Learning research and industry
  - Speech recognition
  - Computer vision
  - Text processing
- Used for regression and classification problems
- It combines hundreds of algorithms and their variants

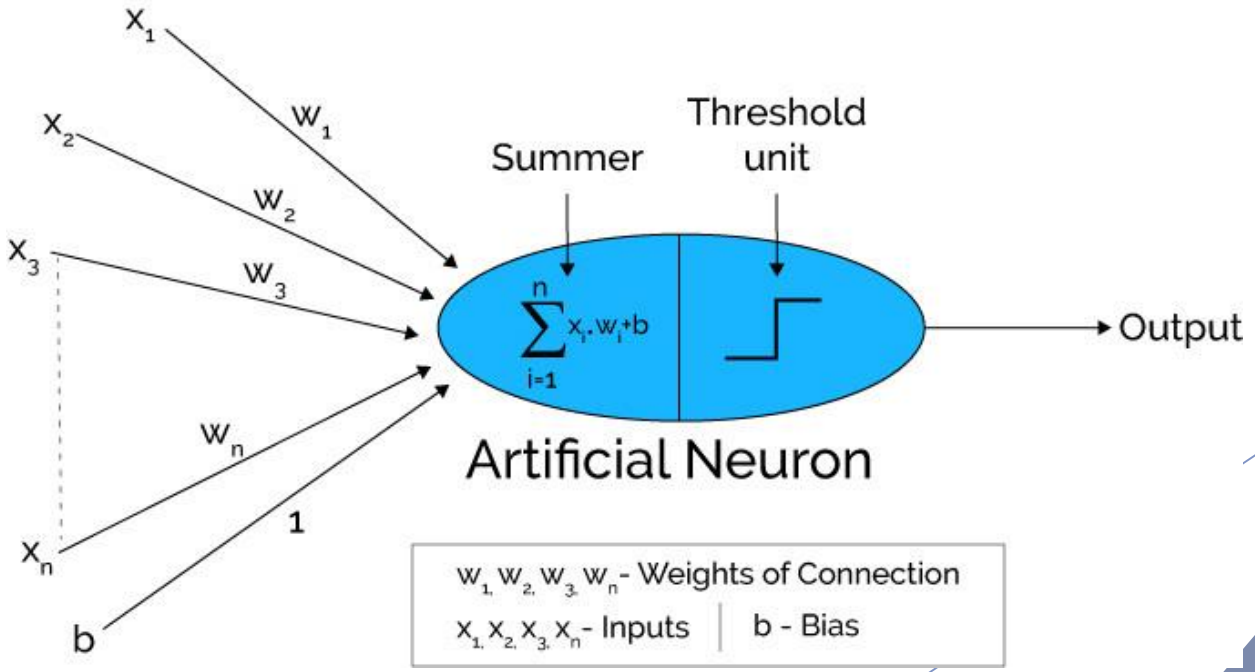


ANN



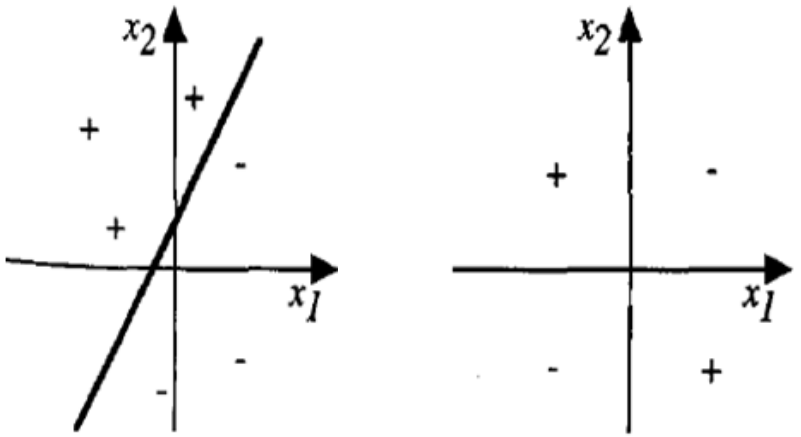
APPROPRIATE PROBLEMS FOR NEURAL NETWORK LEARNING

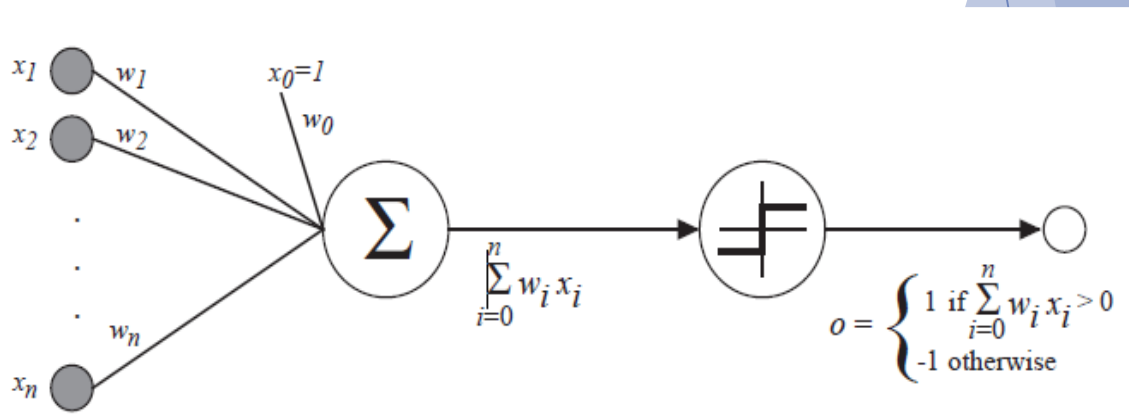
- ANN learning is well-suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras and microphones.
- It is also applicable to problems for which more symbolic representations are often used, such as the decision tree learning tasks.
- In these cases, ANN and decision tree learning often produce results of comparable accuracy.
- The back-propagation algorithm is the most commonly used ANN learning technique.



BASIC UNDERSTANDING - NEURAL NETWORK

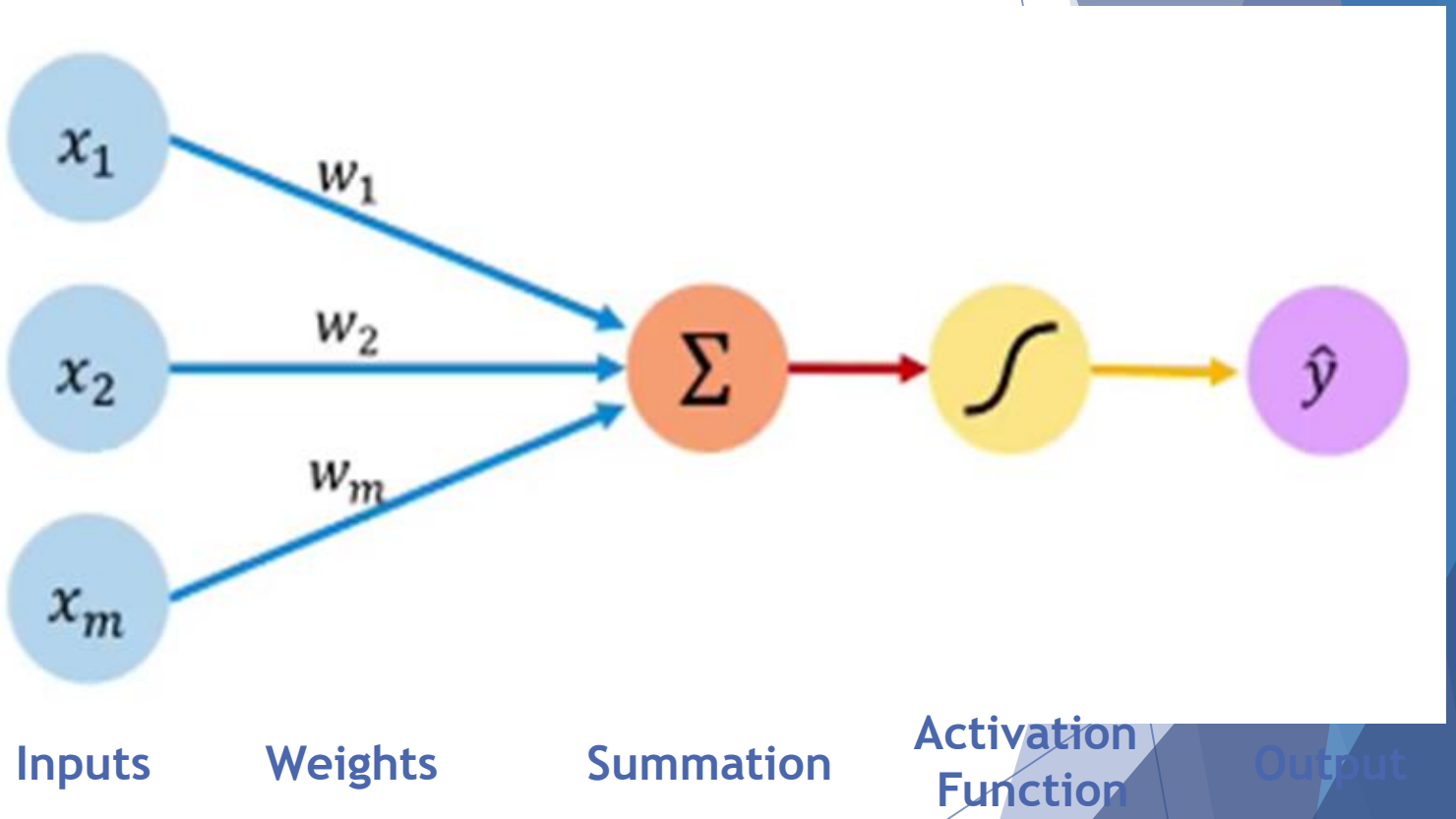
- Perceptron is the basic operational unit of artificial neural networks. It employs supervised learning rule and is able to classify the data into two classes.
- We can view the perceptron as representing a hyperplane decision surface in the n dimensional space of instances (i.e., points). The perceptron outputs a 1 for instances lying on one side of the hyperplane and outputs a -1 for instances lying on the other side, as illustrated in Figure given below.





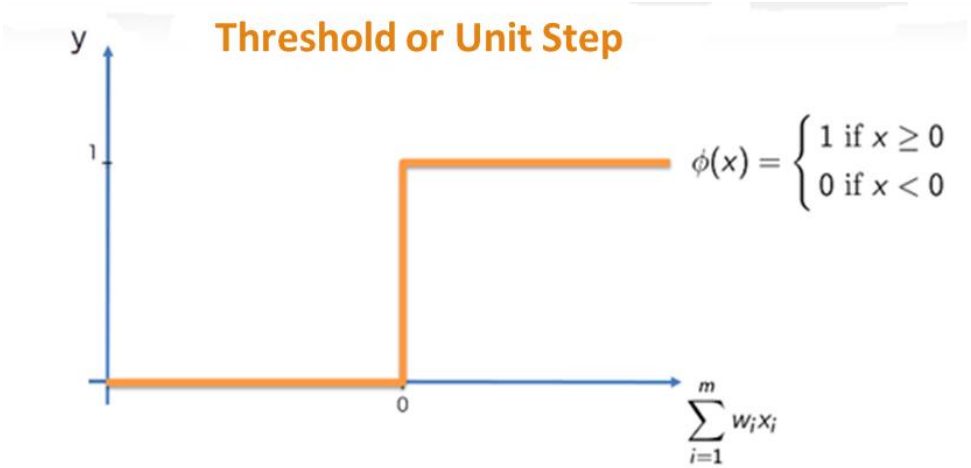
SINGLE NEURON

- The basic unit of computation in a neural network is the neuron, often called a **node or unit**
- Receives input from some other nodes, or from an external source and computes an output
- Each input has an associated **weight (w)** which is assigned on the basis of its relative importance to other inputs
- The node applies a **function f** to the **weighted sum** of its inputs as shown in the fig

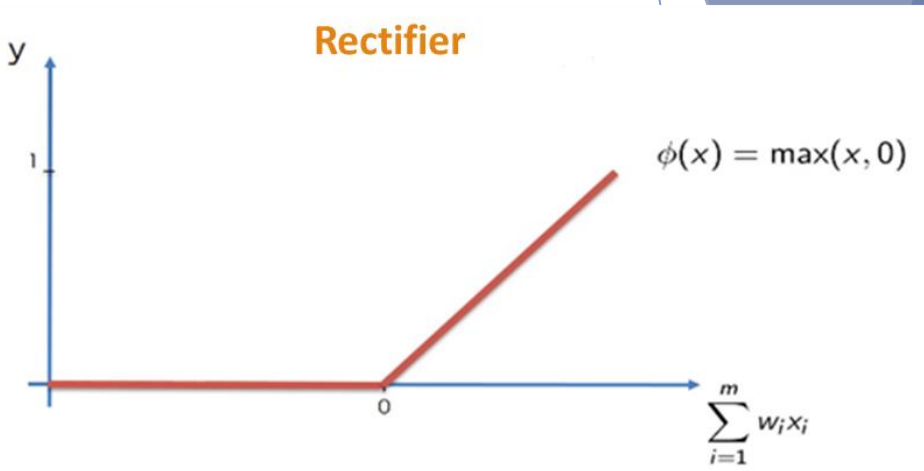


ACTIVATION FUNCTIONS

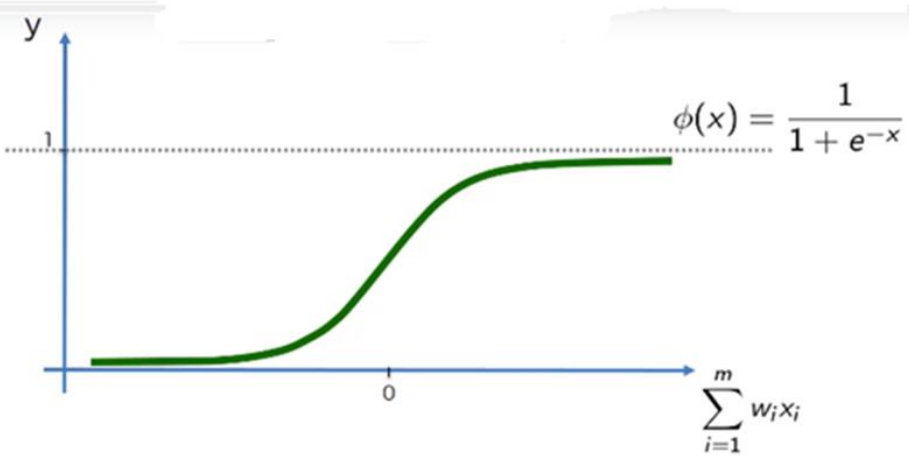
Threshold or Unit Step



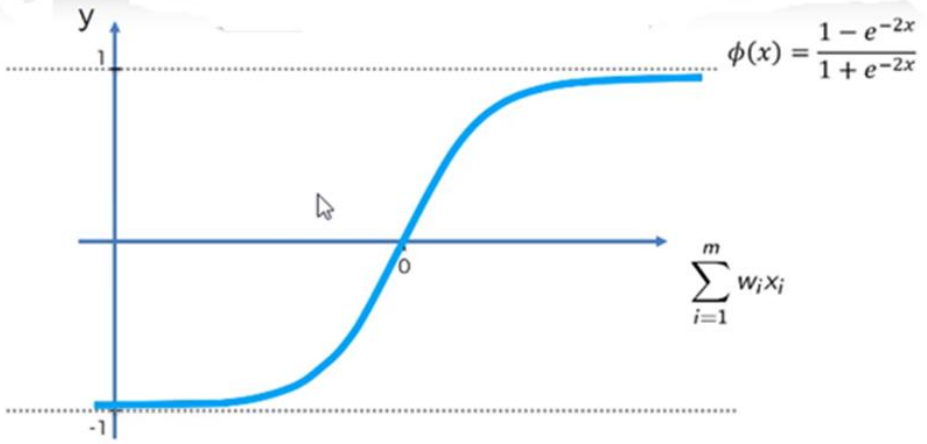
Rectifier



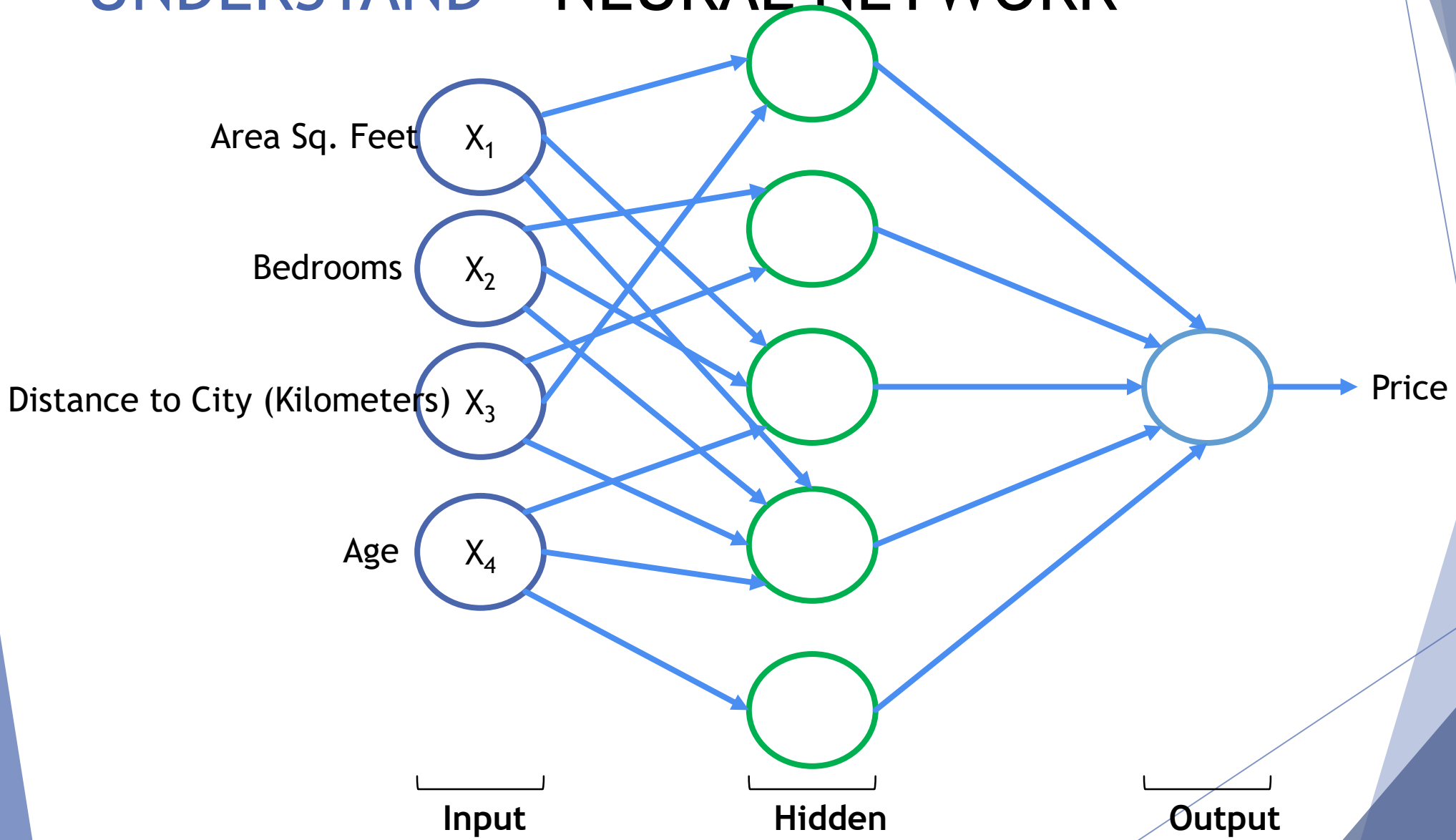
Sigmoid



Hyperbolic Tangent

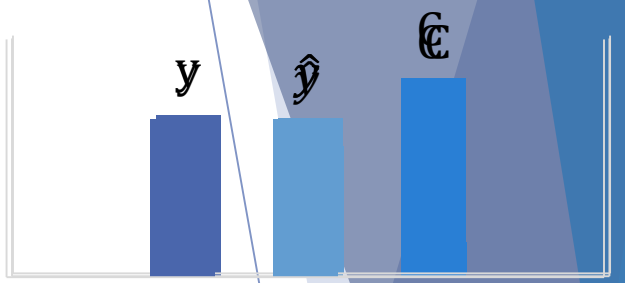


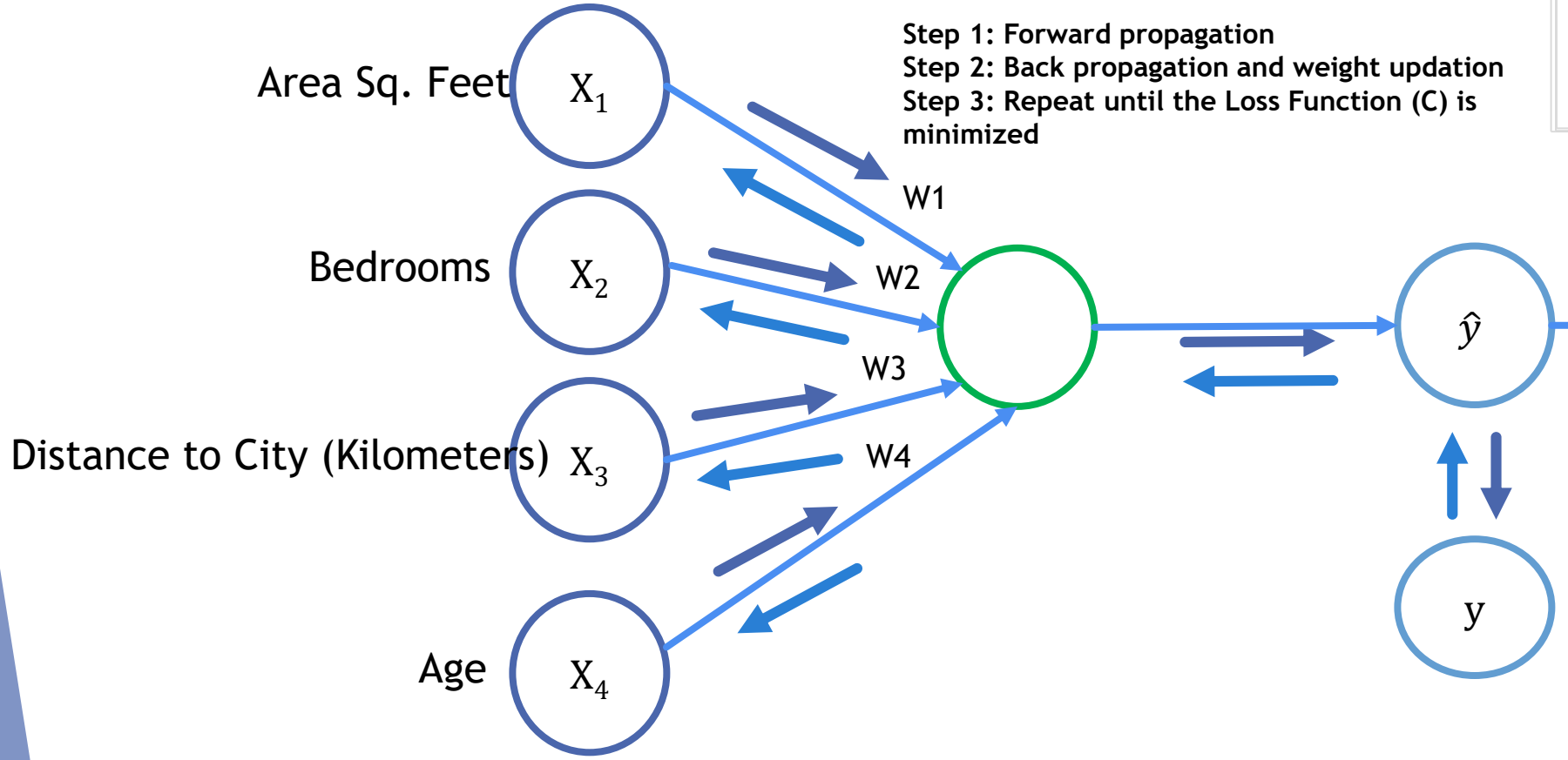
UNDERSTAND – NEURAL NETWORK



HOW NEURAL NETWORKS LEARN ?

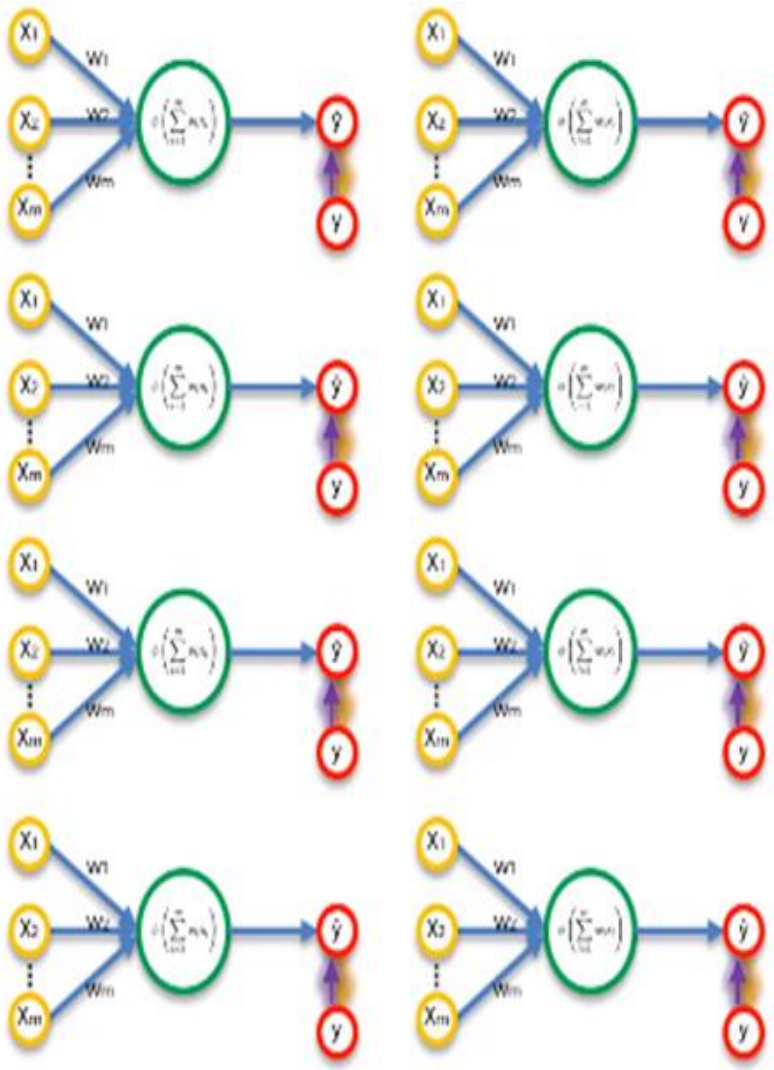
Row ID	Area Sq. Feet	Bedrooms	Distance to City (KMs)	Age	Price
01	250	3	15	36	2500000



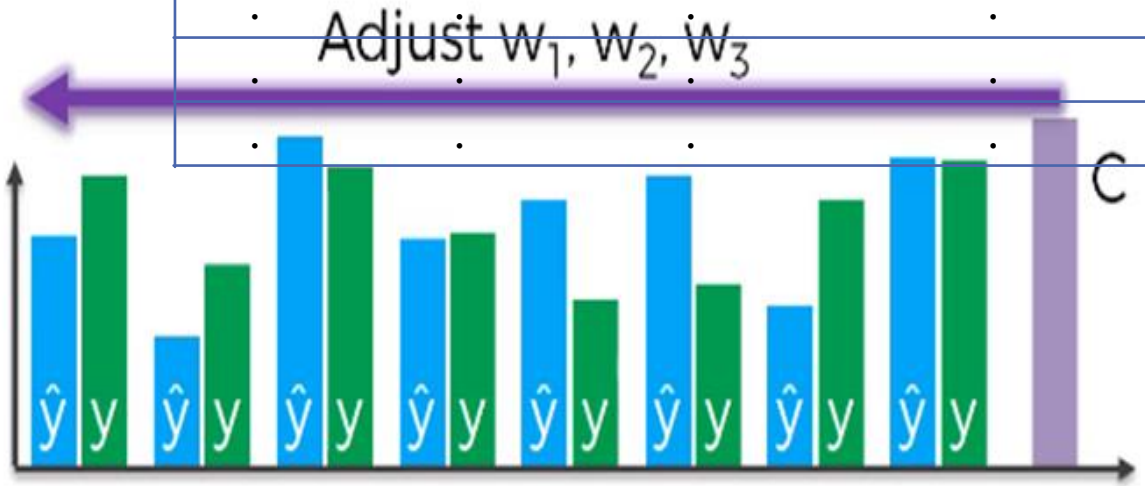


Step 1: Forward propagation
Step 2: Back propagation and weight updation
Step 3: Repeat until the Loss Function (C) is minimized

BACK PROPOGATION – BackProp is like “learning from mistakes“



Row ID	Area Sq. Feet	Bedrooms	Distance to City (KMs)	Age	Price
01	250	3	15	36	2500000
02	300	2	5	40	5000000
03	175	1	10	26	1500000
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

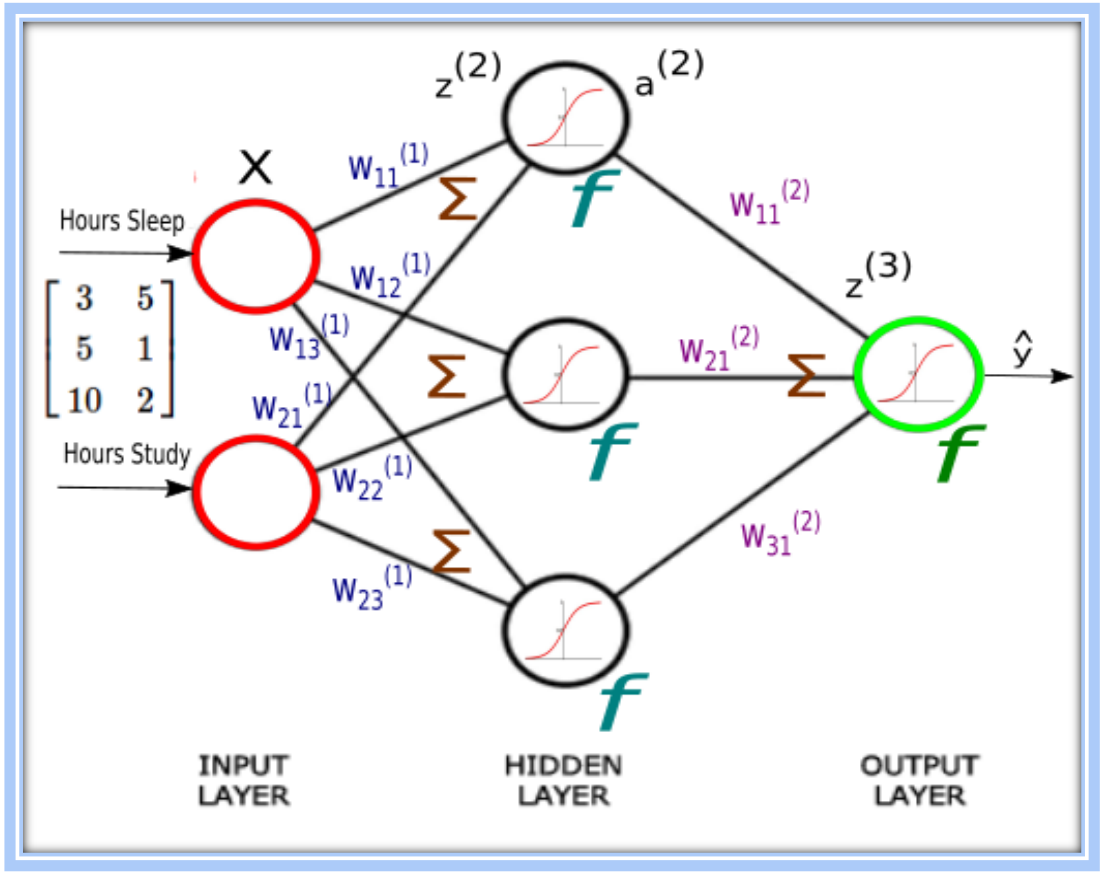


$$C = \frac{1}{2} * (\hat{y} - y)$$

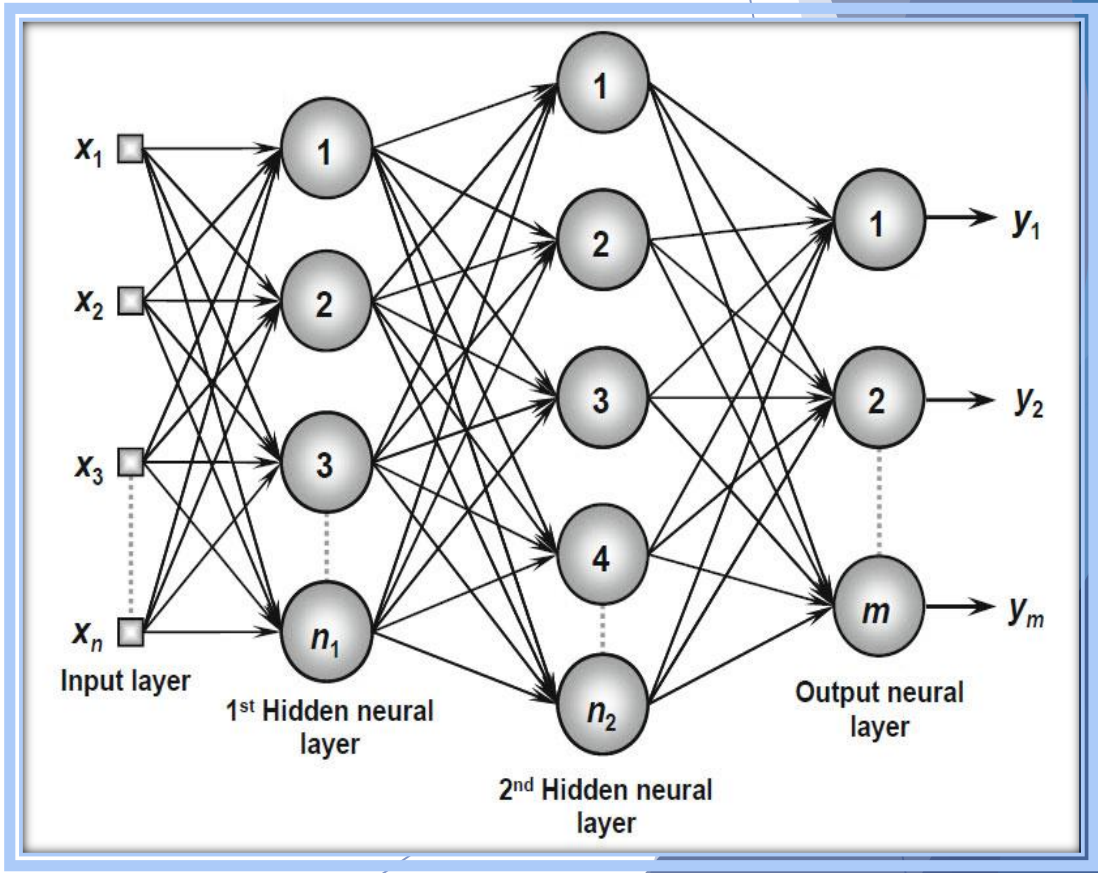
ARCHITECTURES OF ARTIFICIAL NEURAL NETWORKS

The main architectures of artificial neural networks, considering the neuron disposition, as well as how they are interconnected and how its layers are composed, classified as follows:

01 single-layer feed forward network



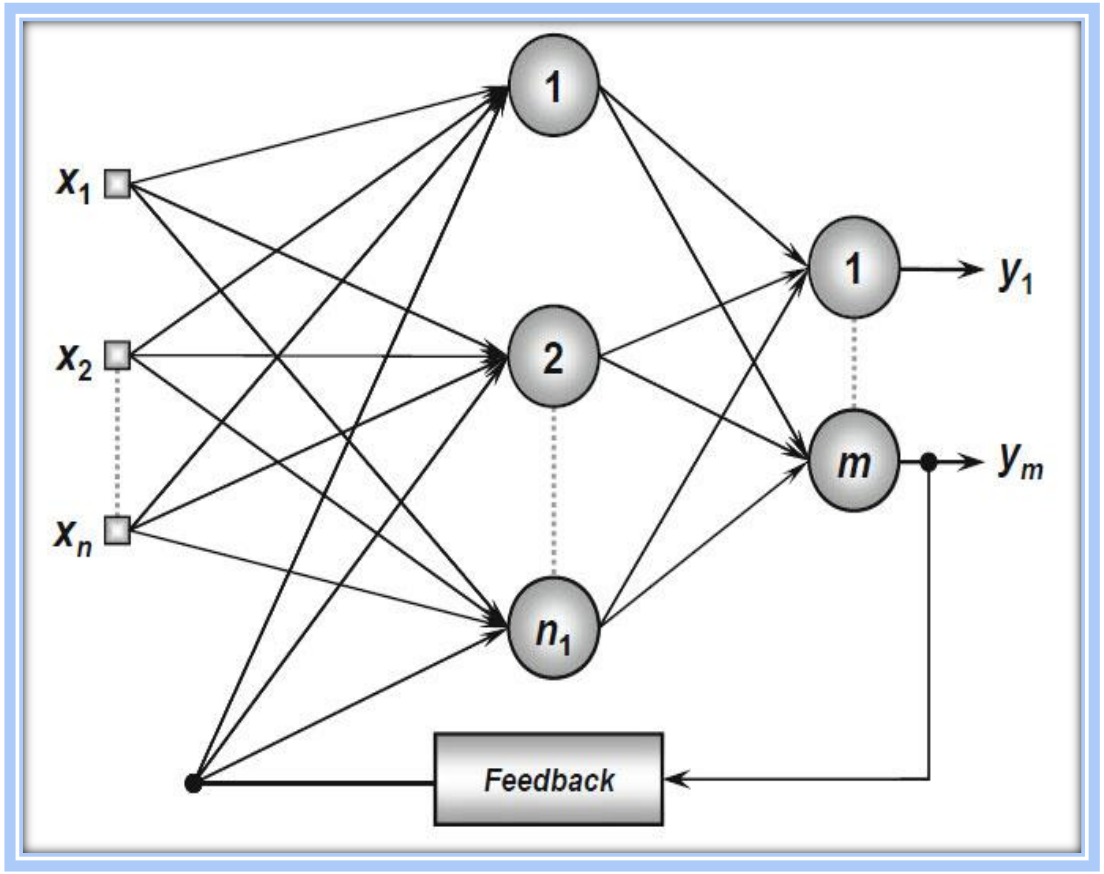
02 multilayer feed forward networks



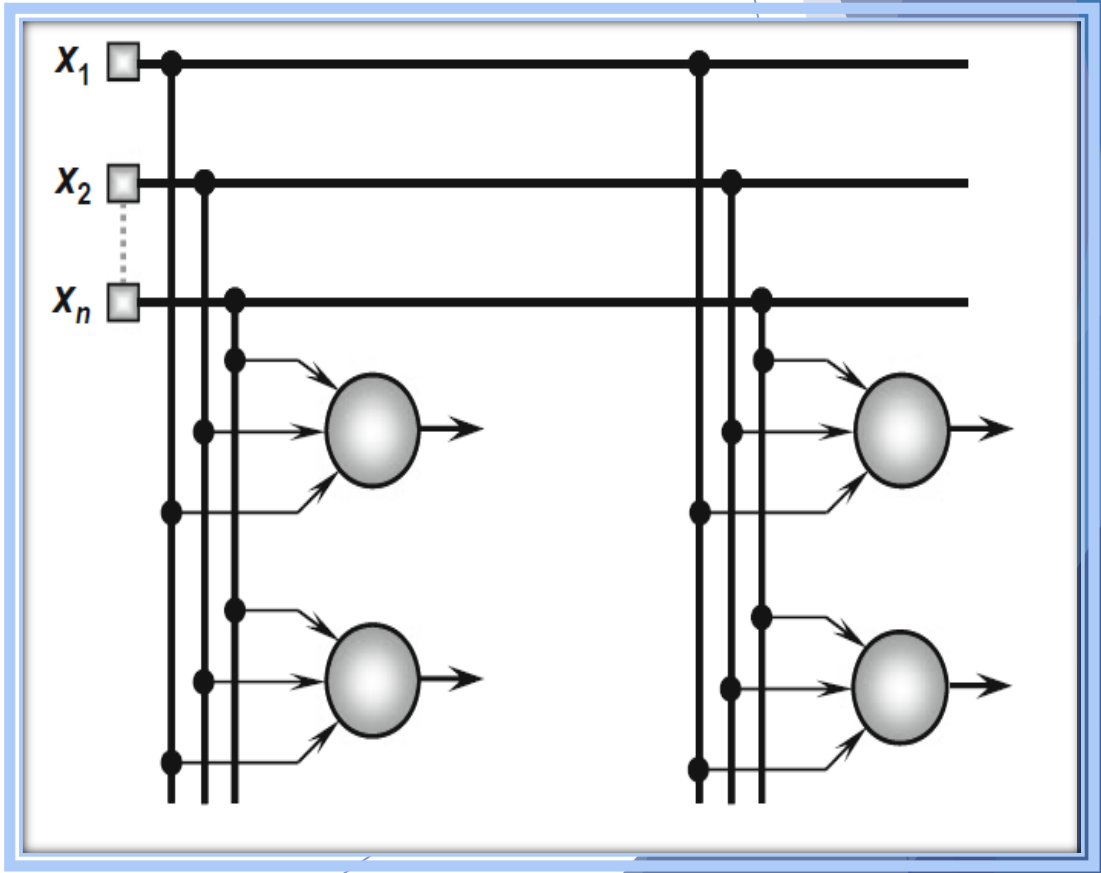
ARCHITECTURES OF ARTIFICIAL NEURAL NETWORKS

The main architectures of artificial neural networks, considering the neuron disposition, as well as how they are interconnected and how its layers are composed, classified as follows:

03 recurrent networks



04 mesh networks



GRADIENT DESCENT

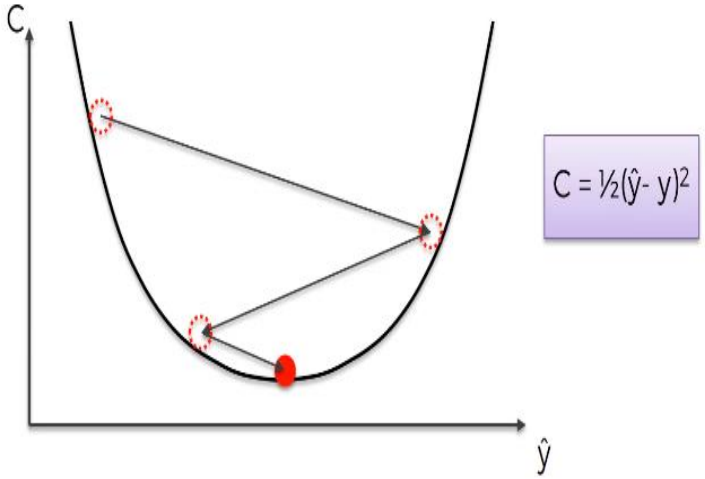
Method

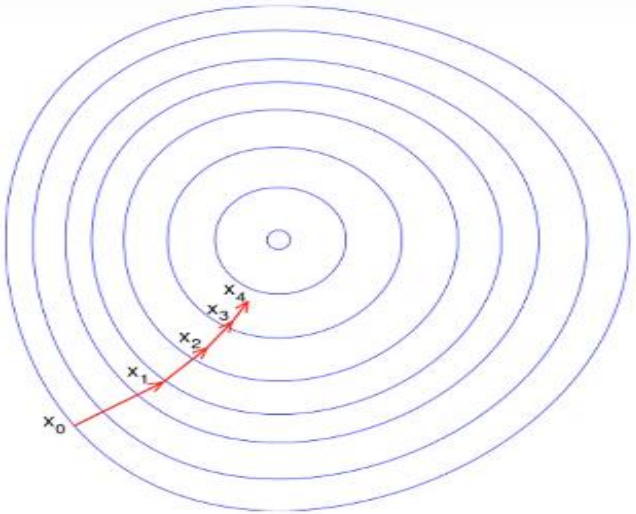
1. **Pick a random point in the function**
2. While gradient hasn't converged, on each iteration you will
   - **2a. Compute the negative gradient at *w*.**
   - **2b. Update point *w* with it the result of 2a, and go back to step 1.**

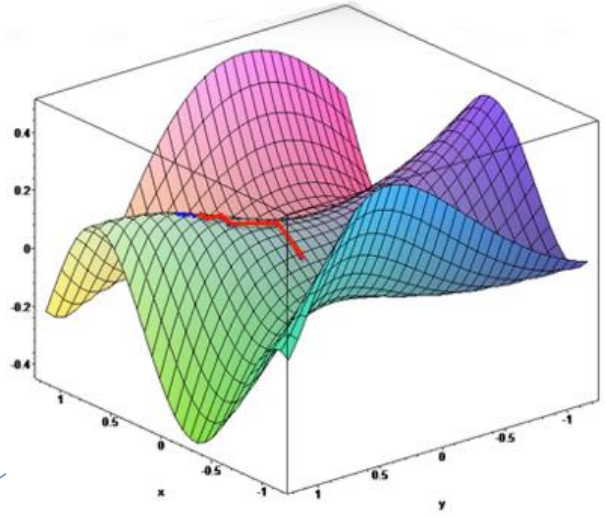
$$\underbrace{\mathrm{w}^{(t+1)}}_{\text{position of next iteration}} = \mathrm{w}^{(t)} \underbrace{- \alpha \nabla f(\mathrm{w}^{(t)})}_{\text{step}}$$

learning rate

position of previous step

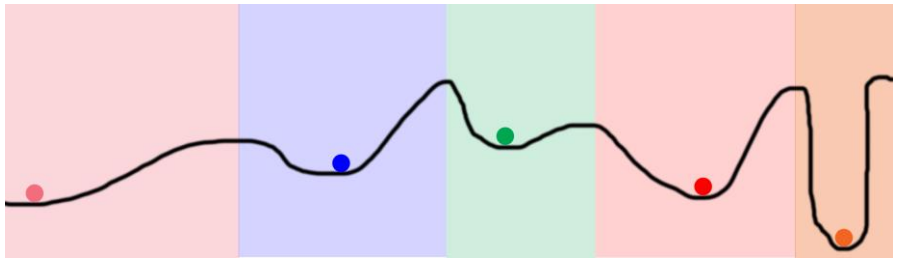






STOCHASTIC GRADIENT DESCENT

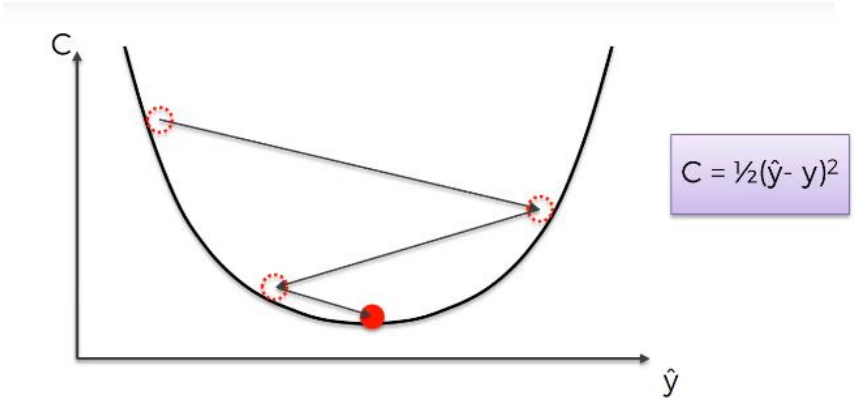
Stochastic Gradient Descent is a probabilistic approximation of Gradient Descent.
It is an approximation because, at each step, the algorithm calculates the gradient for one observation picked at random, instead of calculating the gradient for the entire dataset.



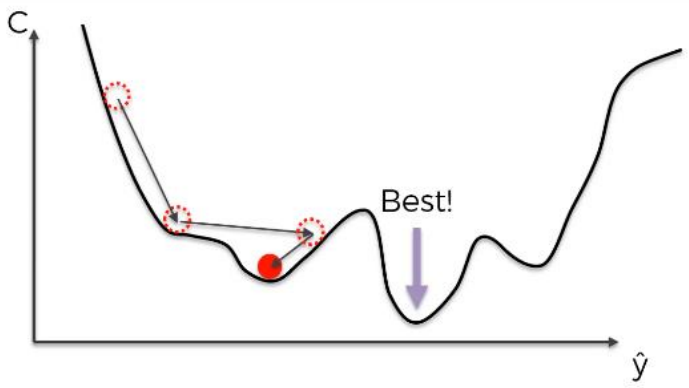
$$\underbrace{w^{(t+1)}}_{\text{position of next iteration}} = \underbrace{w}_{\text{position of previous step}} - \underbrace{\alpha \nabla f_i(w^{(t)})}_{\text{step}}$$

learning rate

observation *i*



Convex function



Non-Convex function

Additional Reading:

A Neural Network in 13 lines of Python (Part 2 - Gradient Descent)
Improving our neural network by optimizing Gradient Descent

Andrew Trask (2015)
by
Link:
https://iamtrask.github.io/2015/07/27/python-network-part2/