

## MODULE-4

**Bayes' Theorem** is a fundamental concept in probability theory and forms the foundation of **Bayesian learning** in machine learning. It allows you to update the probability of a hypothesis (or event) based on new evidence.

### Bayes' Theorem Explained

At its core, Bayes' Theorem relates current knowledge or belief about an event (the **prior** probability) to new data or evidence (the **likelihood**) to produce an updated belief (the **posterior** probability).

Mathematically, Bayes' Theorem is:

Mathematically, Bayes' Theorem is:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

Where:

- $P(H | D)$  is the **posterior probability**: the probability of the hypothesis  $H$  being true given the data  $D$ .
- $P(D | H)$  is the **likelihood**: the probability of observing the data  $D$  given that hypothesis  $H$  is true.
- $P(H)$  is the **prior probability**: the initial belief about the hypothesis  $H$  before any data is observed.
- $P(D)$  is the **marginal likelihood or evidence**: the total probability of the data under all possible hypotheses. This acts as a normalizing constant to ensure that the posterior is a valid probability distribution.

### Breaking Down the Components of Bayes' Theorem

1. **Prior Probability ( $P(H)$ ):**
  - This represents what we know or believe about a hypothesis before seeing any new data.
  - Example: In a medical test scenario, it could be the prior probability of a person having a disease before considering the test results (e.g., based on the general population statistics).
2. **Likelihood ( $P(D | H)$ ):**
  - This is the probability of observing the data, assuming the hypothesis is true. It expresses how likely it is to see the given data under the assumption of the hypothesis.
  - Example: The likelihood would be the probability of getting a positive test result assuming the person has the disease.
3. **Evidence ( $P(D)$ ):**
  - This is the total probability of the data across all hypotheses. It serves to normalize the posterior probability so that it sums to 1.
  - Example: The probability of getting a positive test result across all people, whether they have the disease or not.
4. **Posterior Probability ( $P(H | D)$ ):**
  - This is the updated belief about the hypothesis after considering the new data (the evidence).
  - Example: The posterior would give the probability of a person having the disease after considering both the prior knowledge and the test results.

### Intuition Behind Bayes' Theorem

Bayes' Theorem can be understood in terms of **updating beliefs**. When you receive new evidence, you modify your prior belief to form a new belief that incorporates both your prior knowledge and the new data.

- **Before** you collect any data, you have a prior belief about a hypothesis (e.g., the probability of a patient having a disease).
- **After** seeing new data (e.g., the result of a medical test), you update your belief about the hypothesis to reflect this new evidence.

Bayes' Theorem lets you do this systematically, ensuring that your updated belief (posterior) is proportional to the prior belief and the likelihood of observing the new data.

## Example: Disease Diagnosis

Consider a simple example of diagnosing a disease using a medical test.

1. **Prior Probability ( $P(H)$ ):**
  - The prior belief is the probability that a person has the disease. For example, in a population, 1% of people might have the disease, so  $P(H)=0.01$ .
2. **Likelihood ( $P(D | H)$ ):**
  - This is the probability of getting a positive test result if the person has the disease. Suppose the test correctly identifies the disease 95% of the time, so  $P(D|H)=0.95$ .
3. **Evidence ( $P(D)$ ):**
  - This is the total probability of a positive test result in the population. It includes both people who have the disease and those who do not.
4. **Posterior Probability ( $P(H | D)$ ):**
  - After receiving a positive test result, we want to calculate the probability that the person actually has the disease.

Let's compute the **posterior probability** using Bayes' Theorem:

$$P(H|D) = \frac{P(D|H)P(H)}{P(D)}$$

Where:

- $P(D|H) = 0.95$  (the probability of a positive test result given the person has the disease)
- $P(H) = 0.01$  (the prior probability that the person has the disease)
- $P(D)$  is the total probability of getting a positive test result (both true positives and false positives).

To calculate  $P(D)$ , we use the law of total probability:

$$P(D) = P(D|H)P(H) + P(D|\neg H)P(\neg H)$$

To calculate  $P(D)$ , we use the law of total probability:

$$P(D) = P(D|H)P(H) + P(D|\neg H)P(\neg H)$$

Where:

- $P(D|\neg H)$  is the probability of getting a positive test result when the person does **not** have the disease (e.g., a false positive rate). Let's assume this is 5%, so  $P(D|\neg H) = 0.05$ .
- $P(\neg H) = 0.99$  is the prior probability that the person does not have the disease.

Now, we can calculate  $P(D)$ :

$$P(D) = (0.95)(0.01) + (0.05)(0.99) = 0.0095 + 0.0495 = 0.059$$

Finally, we can compute the **posterior probability**:

$$P(H|D) = \frac{(0.95)(0.01)}{0.059} = \frac{0.0095}{0.059} \approx 0.161$$

So, after a positive test result, the probability that the person actually has the disease is approximately **16.1%**, even though the test is 95% accurate. This is due to the low prior probability of the disease in the population.

### Algorithm 8.1: Naïve Bayes

1. Compute the prior probability for the target class.
2. Compute Frequency matrix and likelihood Probability for each of the feature.
3. Use Bayes theorem Eq. (8.1) to calculate the probability of all hypotheses.
4. Use Maximum A Posteriori (MAP) Hypothesis,  $h_{MAP}$  Eq. (8.2) to classify the test object to the hypothesis with the highest probability.

**Example 8.2:** Assess a student's performance using Naïve Bayes algorithm with the dataset provided in Table 8.1. Predict whether a student gets a job offer or not in his final year of the course.

**Table 8.1: Training Dataset**

S.No.	CGPA	Interactiveness	Practical Knowledge	Communication Skills	Job Offer
1.	≥9	Yes	Very good	Good	Yes
2.	≥8	No	Good	Moderate	Yes
3.	≥9	No	Average	Poor	No
4.	<8	No	Average	Good	No
5.	≥8	Yes	Good	Moderate	Yes
6.	≥9	Yes	Good	Moderate	Yes
7.	<8	Yes	Good	Poor	No
8.	≥9	No	Very good	Good	Yes
9.	≥8	Yes	Good	Good	Yes
10.	≥8	Yes	Average	Good	Yes

**Solution:** The training dataset  $T$  consists of 10 data instances with attributes such as 'CGPA', 'Interactiveness', 'Practical Knowledge' and 'Communication Skills' as shown in Table 8.1. The target variable is Job Offer which is classified as Yes or No for a candidate student.

**Step 1:** Compute the prior probability for the target feature 'Job Offer'. The target feature 'Job Offer' has two classes, 'Yes' and 'No'. It is a binary classification problem. Given a student instance, we need to classify whether 'Job Offer = Yes' or 'Job Offer = No'.

From the training dataset, we observe that the frequency or the number of instances with 'Job Offer = Yes' is 7 and 'Job Offer = No' is 3.

The prior probability for the target feature is calculated by dividing the number of instances belonging to a particular target class by the total number of instances.

Hence, the prior probability for 'Job Offer = Yes' is 7/10 and 'Job Offer = No' is 3/10 as shown in Table 8.2.

**Table 8.2: Frequency Matrix and Prior Probability of Job Offer**

Job Offer Classes	No. of Instances	Probability Value
Yes	7	$P(\text{Job Offer} = \text{Yes}) = 7/10$
No	3	$P(\text{Job Offer} = \text{No}) = 3/10$

**Step 2:** Compute Frequency matrix and Likelihood Probability for each of the feature.

**Step 2(a): Feature – CGPA**

Table 8.3 shows the frequency matrix for the feature CGPA.

**Table 8.3: Frequency Matrix of CGPA**

CGPA	Job Offer = Yes	Job Offer = No
$\geq 9$	3	1
$\geq 8$	4	0
$< 8$	0	2
Total	7	3

Table 8.4 shows how the likelihood probability is calculated for CGPA using conditional probability.

**Table 8.4: Likelihood Probability of CGPA**

CGPA	$P(\text{Job Offer} = \text{Yes})$	$P(\text{Job Offer} = \text{No})$
$\geq 9$	$P(\text{CGPA} \geq 9   \text{Job Offer} = \text{Yes}) = 3/7$	$P(\text{CGPA} \geq 9   \text{Job Offer} = \text{No}) = 1/3$
$\geq 8$	$P(\text{CGPA} \geq 8   \text{Job Offer} = \text{Yes}) = 4/7$	$P(\text{CGPA} \geq 8   \text{Job Offer} = \text{No}) = 0/3$
$< 8$	$P(\text{CGPA} < 8   \text{Job Offer} = \text{Yes}) = 0/7$	$P(\text{CGPA} < 8   \text{Job Offer} = \text{No}) = 2/3$

As explained earlier the Likelihood probability is stated as the sampling density for the evidence given the hypothesis. It is denoted as  $P(\text{Evidence} | \text{Hypothesis})$ , which says how likely is the occurrence of the evidence given the parameters.

It is calculated as the number of instances of each attribute value and for a given class value divided by the number of instances with that class value.

For example  $P(\text{CGPA} \geq 9 | \text{Job Offer} = \text{Yes})$  denotes the number of instances with 'CGPA  $\geq 9$ ' and 'Job Offer = Yes' divided by the total number of instances with 'Job Offer = Yes'.

From the Table 8.3 Frequency Matrix of CGPA, number of instances with 'CGPA  $\geq 9$ ' and 'Job Offer = Yes' is 3. The total number of instances with 'Job Offer = Yes' is 7. Hence,  $P(\text{CGPA} \geq 9 | \text{Job Offer} = \text{Yes}) = 3/7$ .

Similarly, the Likelihood probability is calculated for all attribute values of feature CGPA.

**Step 2(b): Feature – Interactiveness**

Table 8.5 shows the frequency matrix for the feature Interactiveness.

**Table 8.5: Frequency Matrix of Interactiveness**

Interactiveness	Job Offer = Yes	Job Offer = No
YES	5	1
NO	2	2
Total	7	3

**Table 8.10: Likelihood Probability of Communication Skills**

Communication Skills	P (Job Offer = Yes)	P (Job Offer = No)
Good	$P(\text{Communication Skills} = \text{Good}   \text{Job Offer} = \text{Yes}) = 4/7$	$P(\text{Communication Skills} = \text{Good}   \text{Job Offer} = \text{No}) = 1/3$
Moderate	$P(\text{Communication Skills} = \text{Moderate}   \text{Job Offer} = \text{Yes}) = 3/7$	$P(\text{Communication Skills} = \text{Moderate}   \text{Job Offer} = \text{No}) = 0/3$
Poor	$P(\text{Communication Skills} = \text{Poor}   \text{Job Offer} = \text{Yes}) = 0/7$	$P(\text{Communication Skills} = \text{Poor}   \text{Job Offer} = \text{No}) = 2/3$

**Step 3:** Use Bayes theorem Eq. (8.1) to calculate the probability of all hypotheses.

Given the test data = (CGPA  $\geq 9$ , Interactiveness = Yes, Practical knowledge = Average, Communication Skills = Good), apply the Bayes theorem to classify whether the given student gets a Job offer or not.

$$P(\text{Job Offer} = \text{Yes} | \text{Test data}) = (P(\text{CGPA} \geq 9 | \text{Job Offer} = \text{Yes}) P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{Yes}) P(\text{Practical knowledge} = \text{Average} | \text{Job Offer} = \text{Yes}) P(\text{Communication Skills} = \text{Good} | \text{Job Offer} = \text{Yes}) P(\text{Job Offer} = \text{Yes})) / (P(\text{Test Data}))$$

We can ignore  $P(\text{Test Data})$  in the denominator since it is common for all cases to be considered.

$$\begin{aligned} \text{Hence, } P(\text{Job Offer} = \text{Yes} | \text{Test data}) &= (P(\text{CGPA} \geq 9 | \text{Job Offer} = \text{Yes}) P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{Yes}) \\ &\quad P(\text{Practical knowledge} = \text{Average} | \text{Job Offer} = \text{Yes}) P(\text{Communication Skills} = \text{Good} | \text{Job Offer} = \text{Yes}) P(\text{Job Offer} = \text{Yes})) \\ &= 3/7 \times 5/7 \times 1/7 \times 4/7 \times 7/10 \end{aligned}$$

$$= 0.0175$$

Similarly, for the other case 'Job Offer = No',

We compute the probability,

$$P(\text{Job Offer} = \text{No} | \text{Test data}) = (P(\text{CGPA} \geq 9 | \text{Job Offer} = \text{No}) P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{No}) P(\text{Practical knowledge} = \text{Average} | \text{Job Offer} = \text{No}) P(\text{Communication Skills} = \text{Good} | \text{Job Offer} = \text{No}) P(\text{Job Offer} = \text{No})) / (P(\text{Test Data}))$$

$$\begin{aligned} P(\text{CGPA} \geq 9 | \text{Job Offer} = \text{No}) P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{No}) P(\text{Practical knowledge} = \text{Average} | \text{Job Offer} = \text{No}) \\ P(\text{Communication Skills} = \text{Good} | \text{Job Offer} = \text{No}) P(\text{Job Offer} = \text{No}) \\ = 1/3 \times 1/3 \times 2/3 \times 1/3 \times 3/10 \end{aligned}$$

$$= 0.0074$$

**Step 4:** Use Maximum A Posteriori (MAP) Hypothesis,  $h_{\text{MAP}}$  Eq. (8.2) to classify the test object to the hypothesis with the highest probability.

Since  $P(\text{Job Offer} = \text{Yes} | \text{Test data})$  has the highest probability value, the test data is classified as 'Job Offer = Yes'.

### Zero Probability Error

In Example 8.1, consider the test data to be (CGPA  $\geq 8$ , Interactiveness = Yes, Practical knowledge = Average, Communication Skills = Good)

When computing the posterior probability,

$$P(\text{Job Offer} = \text{Yes} | \text{Test data}) = (P(\text{CGPA} \geq 8 | \text{Job Offer} = \text{Yes}) P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{Yes}) P(\text{Practical knowledge} = \text{Average} | \text{Job Offer} = \text{Yes}) P(\text{Communication Skills} = \text{Good} | \text{Job Offer} = \text{Yes}) P(\text{Job Offer} = \text{Yes})) / (P(\text{Test Data}))$$

$P(\text{Job Offer} = \text{Yes} \mid \text{Test data}) = (P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{Yes}) P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{Yes}) P(\text{Practical knowledge} = \text{Average} \mid \text{Job Offer} = \text{Yes}) P(\text{Communication Skills} = \text{Good} \mid \text{Job Offer} = \text{Yes}) P(\text{Job Offer} = \text{Yes})$

$$= 4/7 \times 5/7 \times 1/7 \times 4/7 \times 7/10$$

$$= 0.0233$$

Similarly, for the other case 'Job Offer = No',

When we compute the probability:

$P(\text{Job Offer} = \text{No} \mid \text{Test data}) = (P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{No}) P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{No}) P(\text{Practical knowledge} = \text{Average} \mid \text{Job Offer} = \text{No}) P(\text{Communication Skills} = \text{Good} \mid \text{Job Offer} = \text{No}) P(\text{Job Offer} = \text{No}))/(P(\text{Test Data}))$

$= P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{No}) P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{No}) P(\text{Practical knowledge} = \text{Average} \mid \text{Job Offer} = \text{No}) P(\text{Communication Skills} = \text{Good} \mid \text{Job Offer} = \text{No}) P(\text{Job Offer} = \text{No})$

$$= 0/3 \times 1/3 \times 2/3 \times 1/3 \times 3/10$$

$$= 0$$

Since the probability value is zero, the model fails to predict, and this is called as Zero-Probability error. This problem arises because there are no instances in the given Table 8.1 for the attribute value CGPA  $\geq 8$  and Job Offer = No and hence the probability value of this case is zero. This zero-probability error can be solved by applying a smoothing technique called Laplace correction which means given 1000 data instances in the training dataset, if there are zero instances for a particular value of a feature we can add 1 instance for each attribute value pair of that feature which will not make much difference for 1000 data instances and the overall probability does not become zero.

Now, let us scale the values given in Table 8.1 for 1000 data instances. The scaled values without Laplace correction are shown in Table 8.11.

**Table 8.11:** Scaled Values to 1000 without Laplace Correction

CGPA	$P(\text{Job Offer} = \text{Yes})$	$P(\text{Job Offer} = \text{No})$
$\geq 9$	$P(\text{CGPA} \geq 9 \mid \text{Job Offer} = \text{Yes}) = 300/700$	$P(\text{CGPA} \geq 9 \mid \text{Job Offer} = \text{No}) = 100/300$
$\geq 8$	$P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{Yes}) = 400/700$	$P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{No}) = 0/300$
$< 8$	$P(\text{CGPA} < 8 \mid \text{Job Offer} = \text{Yes}) = 0/700$	$P(\text{CGPA} < 8 \mid \text{Job Offer} = \text{No}) = 200/300$

Now, add 1 instance for each CGPA-value pair for 'Job Offer = No'. Then,

$$P(\text{CGPA} \geq 9 \mid \text{Job Offer} = \text{No}) = 101/303 = 0.333$$

$$P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{No}) = 1/303 = 0.0033$$

$$P(\text{CGPA} < 8 \mid \text{Job Offer} = \text{No}) = 201/303 = 0.6634$$

With scaled values to 1003 data instances, we get

$P(\text{Job Offer} = \text{Yes} \mid \text{Test data}) = (P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{Yes}) P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{Yes}) P(\text{Practical knowledge} = \text{Average} \mid \text{Job Offer} = \text{Yes}) P(\text{Communication Skills} = \text{Good} \mid \text{Job Offer} = \text{Yes}) P(\text{Job Offer} = \text{Yes})$

$$= 400/700 \times 500/700 \times 100/700 \times 400/700 \times 700/1003$$

$$= 0.02325$$

$$P(\text{Job Offer} = \text{No} \mid \text{Test data}) = P(\text{CGPA} \geq 8 \mid \text{Job Offer} = \text{No}) P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{No}) P(\text{Practical knowledge} = \text{Average} \mid \text{Job Offer} = \text{No}) P(\text{Communication Skills} = \text{Good} \mid \text{Job Offer} = \text{No})$$

$$= 1/303 \times 100/300 \times 200/300 \times 100/300 \times 303/1003$$

$$= 0.000007385$$

Thus, using Laplace Correction, Zero Probability error can be solved with Naïve Bayes classifier.

### 8.3.2 Brute Force Bayes Algorithm

Applying Bayes theorem, Brute Force Bayes algorithm relies on the idea of concept learning wherein given a hypothesis space  $H$  for the training dataset  $T$ , the algorithm computes the posterior probabilities for all the hypothesis  $h_i \in H$ . Then, Maximum A Posteriori (MAP) Hypothesis,  $h_{MAP}$  is used to output the hypothesis with maximum posterior probability. The algorithm is quite expensive since it requires computations for all the hypotheses. Although computing posterior probabilities is inefficient, this idea is applied in various other algorithms which is also quite interesting.

### 8.3.3 Bayes Optimal Classifier

Bayes optimal classifier is a probabilistic model, which in fact, uses the Bayes theorem to find the most probable classification for a new instance given the training data by combining the predictions of all posterior hypotheses. This is different from Maximum A Posteriori (MAP) Hypothesis,  $h_{MAP}$  which chooses the maximum probable hypothesis or the most probable hypothesis.

Here, a new instance can be classified to a possible classification value  $C_i$  by the following Eq. (8.4).

$$= \max_{C_i} \sum_{h_i \in H} P(C_i \mid h_i) P(h_i \mid T) \quad (8.4)$$

**Example 8.3:** Given the hypothesis space with 4 hypothesis  $h_1, h_2, h_3$  and  $h_4$ . Determine if the patient is diagnosed as COVID positive or COVID negative using Bayes Optimal classifier.

**Solution:** From the training dataset  $T$ , the posterior probabilities of the four different hypotheses for a new instance are given in Table 8.12.

Table 8.12: Posterior Probability Values

$P(h_i \mid T)$	$P(\text{COVID Positive} \mid h_i)$	$P(\text{COVID Negative} \mid h_i)$
0.3	0	1
0.1	1	0
0.2	1	0
0.1	1	0

$h_{MAP}$  chooses  $h_1$  which has the maximum probability value 0.3 as the solution and gives the result that the patient is COVID negative. But Bayes Optimal classifier combines the predictions of  $h_2, h_3$  and  $h_4$  which is 0.4 and gives the result that the patient is COVID positive.

$$\sum_{h_i \in H} P(\text{COVID Negative} \mid h_i) P(h_i \mid T) = 0.3 \times 1 = 0.3$$

$$\sum_{h_i \in H} P(\text{COVID Positive} \mid h_i) P(h_i \mid T) = 0.1 \times 1 + 0.2 \times 1 + 0.1 \times 1 = 0.4$$

Therefore,  $\max_{C_i \in \{\text{COVID Positive, COVID Negative}\}} \sum_{h_i \in H} P(C_i|h_i)P(h_i|T) = \text{COVID Positive}$ .

Thus, this algorithm, diagnoses the new instance to be COVID positive.

### 8.3.4 Gibbs Algorithm

The main drawback of Bayes optimal classifier is that it computes the posterior probability for all hypotheses in the hypothesis space and then combines the predictions to classify a new instance.

Gibbs algorithm is a sampling technique which randomly selects a hypothesis from the hypothesis space according to the posterior probability distribution and classifies a new instance. It is found that the prediction error occurs twice with the Gibbs algorithm when compared to Bayes Optimal classifier.

## 8.4 NAÏVE BAYES ALGORITHM FOR CONTINUOUS ATTRIBUTES

There are two ways to predict with Naive Bayes algorithm for continuous attributes:

1. Discretize continuous feature to discrete feature.
2. Apply Normal or Gaussian distribution for continuous feature.

### *Gaussian Naive Bayes Algorithm*

In Gaussian Naive Bayes, the values of continuous features are assumed to be sampled from a Gaussian distribution.

**Example 8.4:** Assess a student's performance using Naïve Bayes algorithm for the continuous attribute. Predict whether a student gets a job offer or not in his final year of the course. The training dataset  $T$  consists of 10 data instances with attributes such as 'CGPA' and 'Interactiveness' as shown in Table 8.13. The target variable is Job Offer which is classified as Yes or No for a candidate student.

Table 8.13: Training Dataset with Continuous Attribute

S.No.	CGPA	Interactiveness	Job Offer
1.	9.5	Yes	Yes
2.	8.2	No	Yes
3.	9.3	No	No
4.	7.6	No	No
5.	8.4	Yes	Yes
6.	9.1	Yes	Yes
7.	7.5	Yes	No
8.	9.6	No	Yes
9.	8.6	Yes	Yes
10.	8.3	Yes	Yes

**Solution:**

**Step 1:** Compute the prior probability for the target feature 'Job Offer'.

Prior probabilities of both the classes are calculated using the same formula (refer to Table 8.14).

**Table 8.14: Prior Probability of Target Class**

Job Offer Classes	No. of Instances	Probability Value
Yes	7	$P(\text{Job Offer} = \text{Yes}) = 7/10$
No	3	$P(\text{Job Offer} = \text{No}) = 3/10$

**Step 2:** Compute Frequency matrix and Likelihood Probability for each of the feature.

Likelihood probabilities for a continuous attribute is obtained from Gaussian (Normal) Distribution. In the above data set, CGPA is a continuous attribute for which we need to apply Gaussian distribution to calculate the likelihood probability.

Gaussian distribution for continuous feature is calculated using the given formula,

$$P(X_i = x_k | C_j) = g(x_k, \mu_{ij}, \sigma_{ij}) \quad (8.5)$$

where,

$X_i$  is the  $i^{\text{th}}$  continuous attribute in the given dataset and  $x_k$  is a value of the attribute.

$C_j$  denotes the  $j^{\text{th}}$  class of the target feature.

$\mu_{ij}$  denotes the mean of the values of that continuous attribute  $X_i$  with respect to the class  $j$  of the target feature.

$\sigma_{ij}$  denotes the standard deviation of the values of that continuous attribute  $X_i$  with respect to the class  $j$  of the target feature.

Hence, the normal distribution formula is given as:

$$P(X_i = x_k | C_j) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(x_k - \mu_{ij})^2}{2\sigma_{ij}^2}} \quad (8.6)$$

**Step 2(a): Consider the feature CGPA**

In this example CGPA is a continuous attribute,

To calculate the likelihood probability for this continuous attribute, first compute the mean and standard deviation for CGPA with respect to the target class 'Job Offer'.

Here,  $X_i = \text{CGPA}$

$C_j = \text{'Job Offer} = \text{Yes}'$

Mean and Standard Deviation for class 'Job Offer = Yes' are given as:

$$\mu_{ij} = \mu_{\text{CGPA - YES}} = 8.814286$$

$$\sigma_{ij} = \sigma_{\text{CGPA - YES}} = 0.58146$$

Mean and Standard Deviation for class 'Job Offer = No' are given as:

$C_j = \text{'Job Offer} = \text{No}'$

$$\mu_{ij} = \mu_{\text{CGPA - NO}} = 8.133333$$

$$\sigma_{ij} = \sigma_{\text{CGPA - NO}} = 1.011599$$

Once Mean and Standard Deviation are computed, the likelihood probability for any test value using Gaussian distribution formula can be calculated.

### Step 2(b): Consider the feature Interactiveness

Interactiveness is a discrete feature whose probability is calculated as earlier.

Table 8.15 shows the frequency matrix for the feature Interactiveness.

**Table 8.15: Frequency Matrix of Interactiveness**

Interactiveness	Job Offer = Yes	Job Offer = No
YES	5	1
NO	2	2
Total	7	3

Table 8.16 shows how the likelihood probability is calculated for Interactiveness using conditional probability.

**Table 8.16: Likelihood Probability of Interactiveness**

Interactiveness	P (Job Offer = Yes)	P (Job Offer = No)
YES	$P(\text{Interactiveness} = \text{Yes}   \text{Job Offer} = \text{Yes}) = 5/7$	$P(\text{Interactiveness} = \text{Yes}   \text{Job Offer} = \text{No}) = 1/3$
NO	$P(\text{Interactiveness} = \text{No}   \text{Job Offer} = \text{Yes}) = 2/7$	$P(\text{Interactiveness} = \text{No}   \text{Job Offer} = \text{No}) = 2/3$

### Step 3: Use Bayes theorem to calculate the probability of all hypotheses.

Consider the test data to be (CGPA = 8.5, Interactiveness = Yes).

For the hypothesis 'Job Offer = Yes':

$$P(\text{Job Offer} = \text{Yes} | \text{Test data}) = (P(\text{CGPA} = 8.5 | \text{Job Offer} = \text{Yes}) \times P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{Yes})) \times P(\text{Job Offer} = \text{Yes})$$

To compute  $P(\text{CGPA} = 8.5 | \text{Job Offer} = \text{Yes})$  use Gaussian distribution formula:

$$P(X_i = x_k | C_j) = g(x_k, \mu_{ij}, \sigma_{ij})$$

$$P(X_{\text{CGPA}} = 8.5 | C_{\text{Job Offer} = \text{Yes}}) = \frac{1}{\sigma_{ij}\sqrt{2\pi}} e^{-\frac{(x_k - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

$$P(X_{\text{CGPA}} = 8.5 | C_{\text{Job Offer} = \text{Yes}}) = \frac{1}{\sigma_{\text{CGPA-YES}}\sqrt{2\pi}} e^{-\frac{(8.5 - \mu_{\text{CGPA-YES}})^2}{2\sigma_{\text{CGPA-YES}}^2}}$$

$$P(\text{CGPA} = 8.5 | \text{Job Offer} = \text{Yes}) = g(x_k = 8.5, \mu_{ij} = 8.814, \sigma_{ij} = 0.581)$$

$$= \frac{1}{0.581\sqrt{2\pi}} e^{-\frac{(8.5 - 8.814)^2}{2(0.581)^2}} = 0.594$$

$$P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{Yes}) = 5/7$$

$$P(\text{Job Offer} = \text{Yes}) = 7/10$$

Hence:

$$P(\text{Job Offer} = \text{Yes} | \text{Test data}) = (P(\text{CGPA} = 8.5 | \text{Job Offer} = \text{Yes}) \times P(\text{Interactiveness} = \text{Yes} | \text{Job Offer} = \text{Yes})) \times P(\text{Job Offer} = \text{Yes})$$

$$= 0.594 \times 5/7 \times 7/10$$

$$= 0.297$$

Similarly, for the hypothesis 'Job Offer = No':

$$P(\text{Job Offer} = \text{No} \mid \text{Test data}) = P(\text{CGPA} = 8.5 \mid \text{Job Offer} = \text{No}) \times P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{No}) \times P(\text{Job Offer} = \text{No})$$

$$P(\text{CGPA} = 8.5 \mid \text{Job Offer} = \text{No}) = g(x_k = 8.5, \mu_g = 8.133, \sigma_g = 1.0116)$$

$$P(X_{\text{CGPA}} = 8.5 \mid C_{\text{Job Offer} = \text{Yes}}) = \frac{1}{\sigma_{\text{CGPA-NO}} \sqrt{2\pi}} e^{-\frac{(8.5 - \mu_{\text{CGPA-NO}})^2}{2\sigma_{\text{CGPA-NO}}^2}}$$

$$= \frac{1}{1.0116 \sqrt{2\pi}} e^{-\frac{(8.5 - 8.133)^2}{2 \times 1.0116^2}} = 0.369$$

$$P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{No}) = 1/3$$

$$= P(\text{Job Offer} = \text{No}) = 0.369$$

Hence,

$$P(\text{Job Offer} = \text{No} \mid \text{Test data}) = P(\text{CGPA} = 8.5 \mid \text{Job Offer} = \text{No}) P(\text{Interactiveness} = \text{Yes} \mid \text{Job Offer} = \text{No}) \times P(\text{Job Offer} = \text{No})$$

$$= 0.369 \times 1/3 \times 3/10$$

$$= 0.0369$$

**Step 4:** Use Maximum A Posteriori (MAP) Hypothesis,  $h_{\text{MAP}}$  to classify the test object to the hypothesis with the highest probability.

Since  $P(\text{Job Offer} = \text{Yes} \mid \text{Test data})$  has the highest probability value of 0.297, the test data is classified as 'Job Offer = Yes'.

## Chapter 10

### Artificial Neural Networks

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modelled after the brain.

An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain.

Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

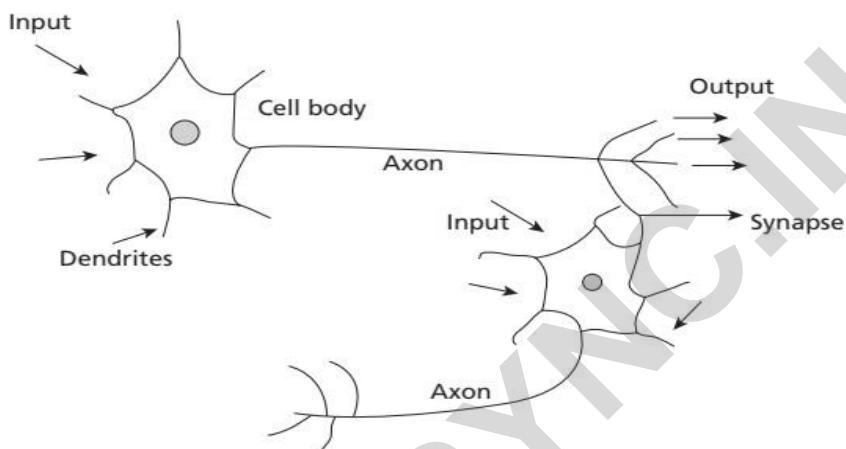
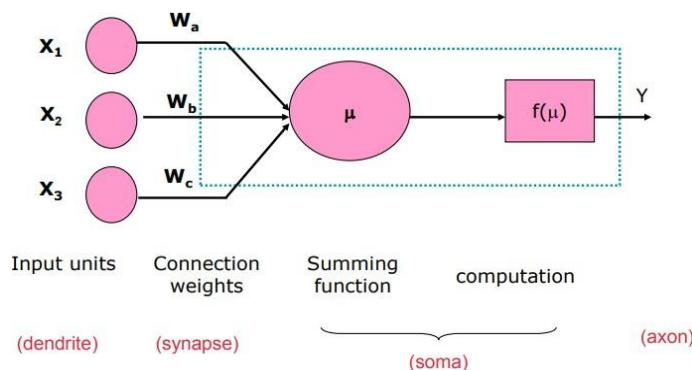


Figure 10.1: A Biological Neuron

The biological neuron consists of main **four** parts:

- **dendrites**: nerve fibres carrying electrical signals to the cell .
- **cell body**: computes a non-linear function of its inputs
- **axon**: single long fiber that carries the electrical signal from the cell body to other neurons
- **synapse**: the point of contact between the axon of one cell and the dendrite of another, regulating a chemical connection whose strength affects the input to the cell.

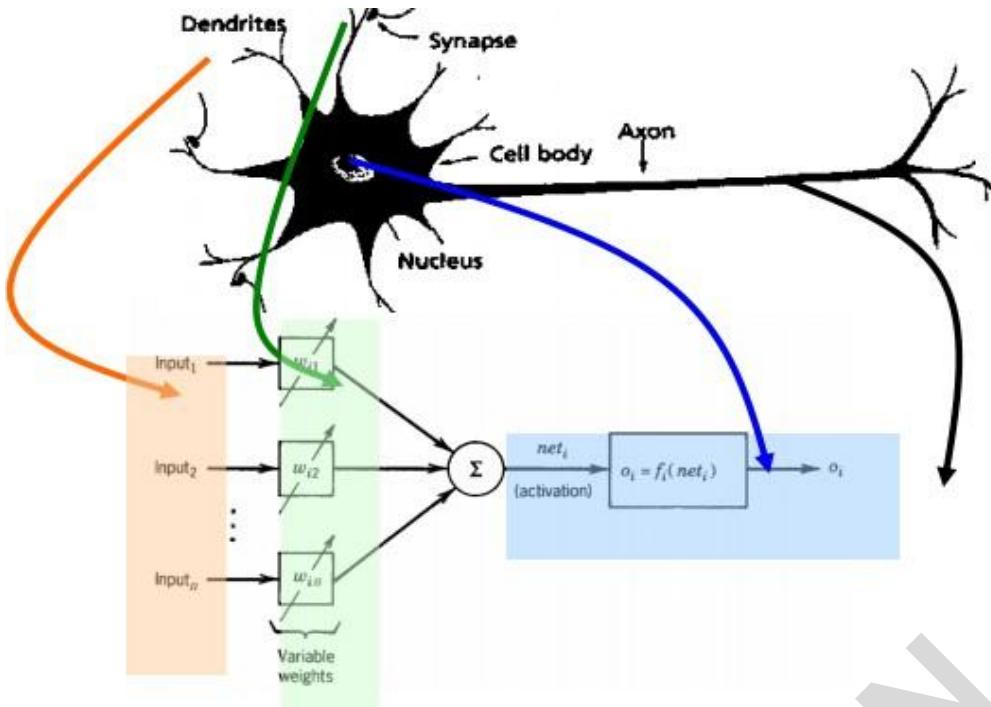


**Dendrites** are tree like networks made of nerve fiber connected to the cell body.

An **Axon** is a single, long connection extending from the cell body and carrying signals from the neuron. The end of axon splits into fine strands. It is found that each strand terminated into small bulb like organs called as synapse. It is through synapse that the neuron introduces its signals to other nearby neurons. The receiving ends of these synapses on the nearby neurons can be found both on the dendrites and on the cell body. There are approximately 104 synapses per neuron in the human body. Electric impulse is passed between synapse and dendrites. It is a chemical process which results in increase/decrease in the electric potential inside the body of the receiving cell. If the electric potential reaches a thresh hold value, receiving cell fires & pulse / action potential of fixed strength and duration is send through the axon to synaptic junction of the cell. After that, cell has to wait for a period called refractory period.

### Difference between biological and Artificial Neuron

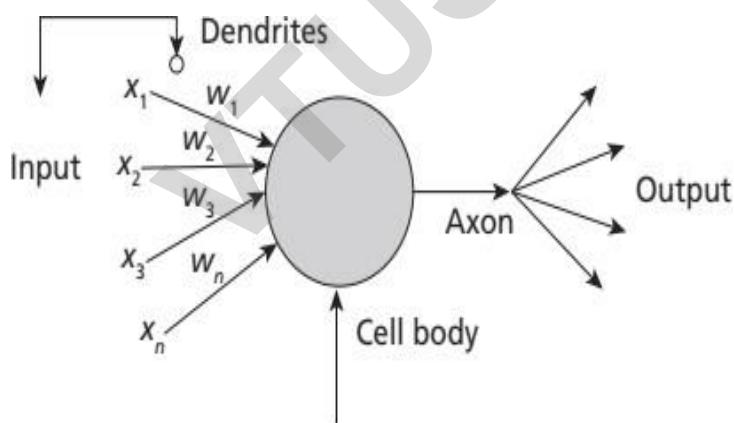
Characteristics	Artificial Neural Network	Biological(Real) Neural Network
<b>Speed</b>	Faster in processing information. Response time is in nanoseconds.	Slower in processing information. The response time is in milliseconds.
<b>Processing</b>	Serial processing.	Massively parallel processing.
<b>Size &amp; Complexity</b>	Less size & complexity. It does not perform complex pattern recognition tasks.	Highly complex and dense network of interconnected neurons containing neurons of the order of $10^{11}$ with $10^{15}$ of interconnections.
<b>Storage</b>	Information storage is replaceable means new data can be added by deleting an old one.	Highly complex and dense network of interconnected neurons containing neurons of the order of $10^{11}$ with $10^{15}$ of interconnections.
<b>Fault tolerance</b>	Fault intolerant. Information once corrupted cannot be retrieved in case of failure of the system.	Information storage is adaptable means new information is added by adjusting the interconnection strengths without destroying old information
<b>Control Mechanism</b>	There is a control unit for controlling computing activities	No specific control mechanism external to the computing task.



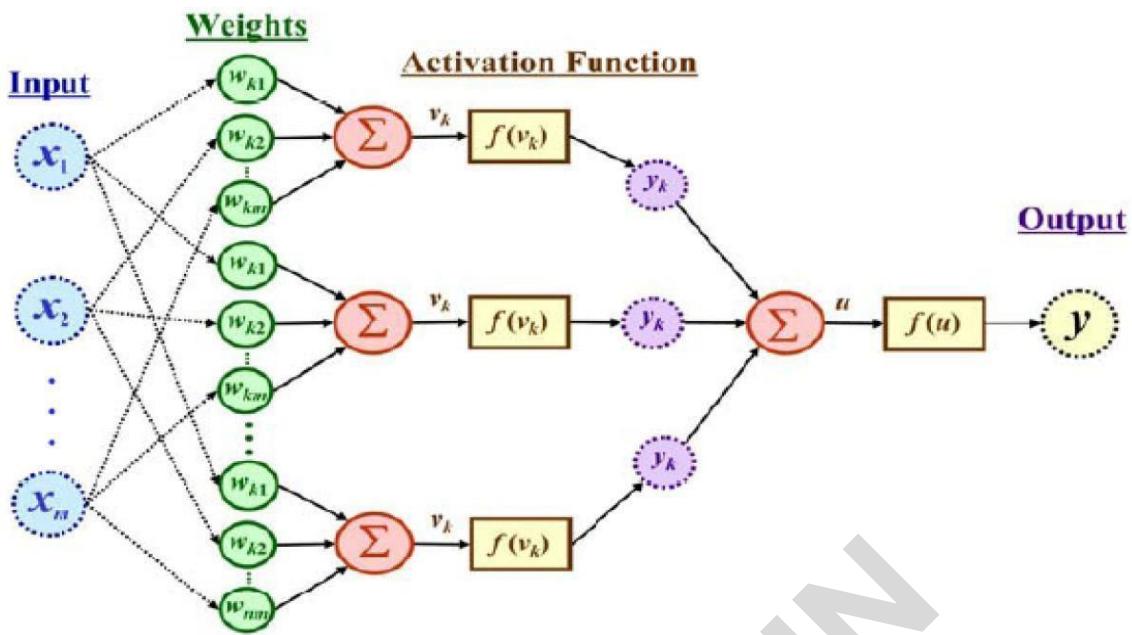
### ARTIFICIAL NEURONS:

Artificial neurons are like biological neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

A node or a neuron can receive one or more input information and process it. artificial neurons are connected by connection links to another neuron. Each connection link is associated with a synaptic weight. The structure of a single neuron is shown below:



**Figure 10.2:** An Artificial Neuron



**Fig:** McCulloch-Pitts Neuron Mathematical model.

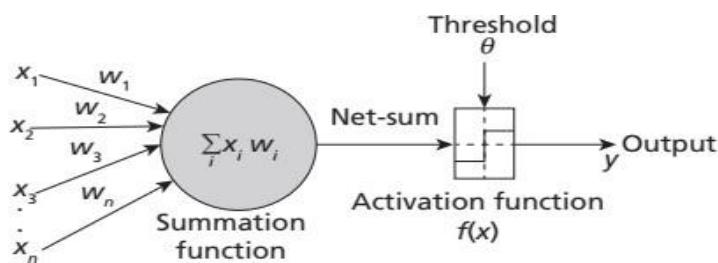
### Simple Model of an ANN

The first mathematical model of a biological neuron was designed by McCulloch-Pitts in 1943.

It includes 2 steps:

1. It receives weighted inputs from other neurons.
2. It operates with a threshold function or activation function.

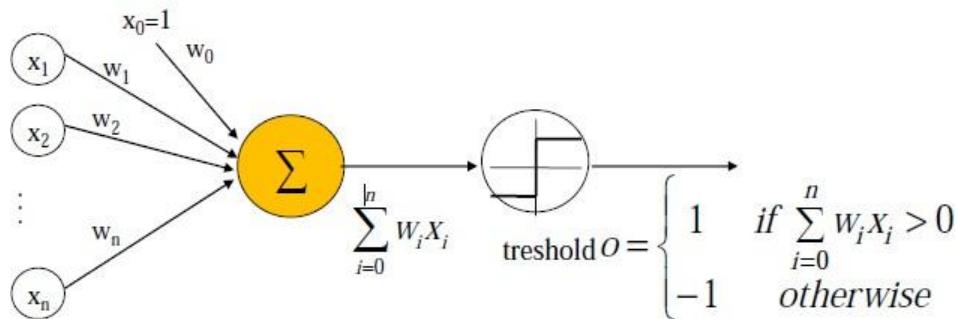
Basically, a neuron takes an input signal (dendrite), processes it like the CPU (soma), passes the output through a cable like structure to other connected neurons (axon to synapse to other neuron's dendrite).



**Figure 10.3:** McCulloch & Pitts Neuron Mathematical Model

**OR**

$$W_0 + W_1X_1 + \dots + W_nX_n = \sum_{i=0}^n W_iX_i = W \cdot X$$



### Working:

The received input are computed as a weighted sum which is given to the activation function and if the sum exceeds the threshold value the neuron gets fired. The neuron is the basic processing unit that receives a set of inputs  $x_1, x_2, x_3, \dots, x_n$  and their associated weights  $w_1, w_2, w_3, \dots, w_n$ . The summation function computes the weighted sum of the inputs received by the neuron.

$$\text{Sum} = \sum x_i w_i$$

### Activation functions:

- To make work more efficient and for exact output, some force or activation is given. Like that, activation function is applied over the net input to calculate the output of an ANN. Information processing of processing element has two major parts: input and output. An integration function ( $f$ ) is associated with input of processing element.
- Several **activation functions** are there.

- Identity function or Linear Function:** It is a linear function which is defined as  $f(x) = x$  for all  $x$

The output is same as the input ie the weighted sum. The function is useful when we do not apply any threshold. The output value ranged between  $-\infty$  and  $+\infty$

- Binary step function:** This function can be defined as

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$f(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Where,  $\theta$  represents threshold value. It is used in single layer nets to convert the net input to an output that is binary (0 or 1).

### 3. Bipolar step function:

This function can be defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ -1 & \text{if } x < \theta \end{cases}$$

Where,  $\theta$  represents threshold value. It is used in single layer nets to convert the net input to an output that is bipolar (+1 or -1).

### 4. Sigmoid function:

It is used in Back propagation nets.

Two types:

- a) **Binary sigmoid function:** It is also termed as logistic sigmoid function or unipolar sigmoid function. It is defined as

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

where,  $\lambda$  represents steepness parameter. The range of sigmoid function is 0 to 1

- b) **Bipolar sigmoid function:** This function is defined as

$$f(x) = \frac{2}{1 + e^{-\lambda x}} - 1 = \frac{1 - e^{-\lambda x}}{1 + e^{-\lambda x}}$$

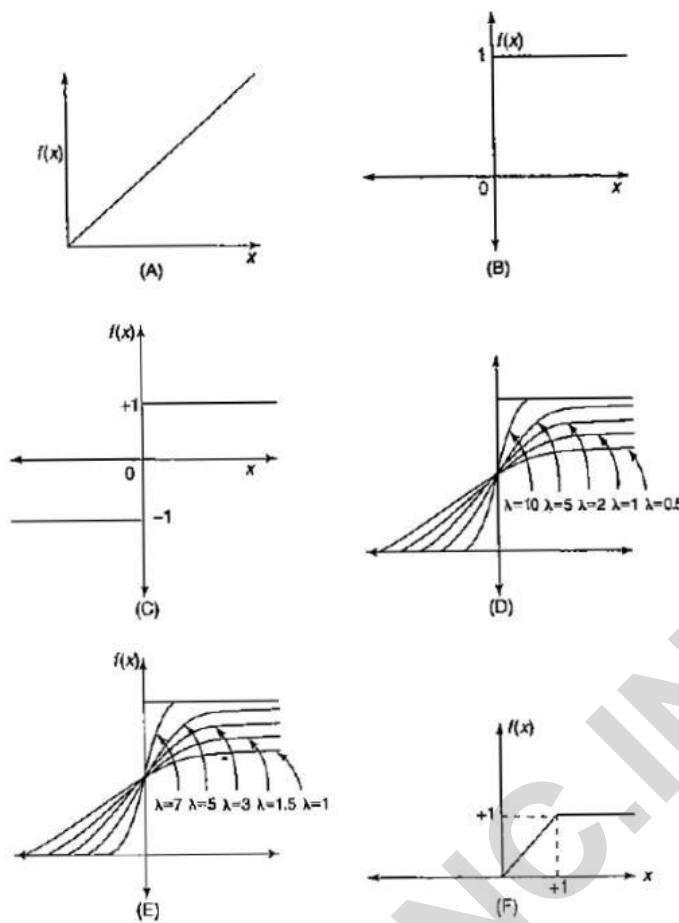
Where  $\lambda$  represents steepness parameter and the sigmoid range is between -1 and +1.

### 5. Ramp function:

The ramp function is defined as:

$$f(x) = \begin{cases} 1 & \text{if } x > 1 \\ x & \text{if } 0 \leq x \leq 1 \\ 0 & \text{if } x < 0 \end{cases}$$

It is a linear function whose upper and lower limits are fixed.



Depiction of activation functions: (A) identity function; (B) binary step function; (C) bipolar step function; (D) binary sigmoidal function; (E) bipolar sigmoidal function; (F) ramp function.

**6. Tanh-Hyperbolic tangent function :** Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

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

## 7. ReLU Function

ReLU stands for **Rectified Linear Unit**.

Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient.

The main catch here is that the ReLU function does not activate all the neurons at the same time.

The neurons will only be deactivated if the output of the linear transformation is less than 0

8. **Softmax function:** Softmax is an activation function that scales numbers/logits into probabilities. The output of a Softmax is a vector (say  $v$ ) with probabilities of each possible outcome. The probabilities in vector  $v$  sums to one for all possible outcomes or classes.

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^n \exp(y_j)}$$

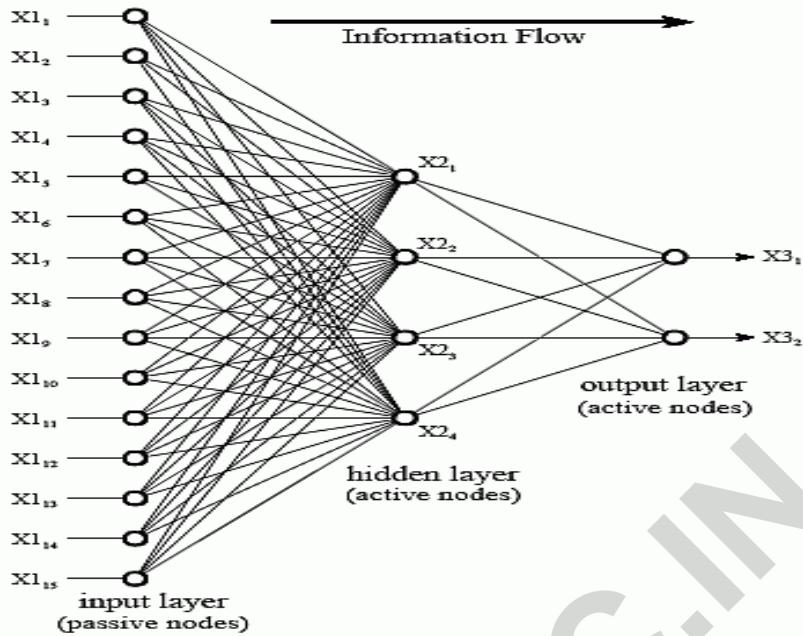
where,

$y$	is an input vector to a softmax function, $S$ . It consists of $n$ elements for $n$ classes (possible outcomes)
$y_i$	the $i$ -th element of the input vector. It can take any value between $-\infty$ and $+\infty$
$\exp(y_i)$	standard exponential function applied on $y_i$ . The result is a small value (close to 0 but never 0) if $y_i < 0$ and a large value if $y_i$ is large. eg <ul style="list-style-type: none"> <li><math>\exp(55) = 7.69e+23</math> (A very large value)</li> <li><math>\exp(-55) = 1.30e-24</math> (A very small value close to 0)</li> </ul> <b>Note:</b> $\exp(*)$ is just $e^*$ where $e = 2.718$ , the Euler's number.
$\sum_{j=1}^n \exp(y_j)$	A normalization term. It ensures that the values of output vector $S(y)_i$ sum to 1 for $i$ -th class and each of them and each of them is in the range 0 and 1 which makes up a valid probability distribution.
$n$	Number of classes (possible outcomes)

## Artificial Neural Network Structure

- Artificial Neural Networks Computational models inspired by the human 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.
  
- **The Neural Network is constructed from 3 type of layers:**
- Input layer — initial data for the neural network.

- Hidden layers — intermediate layer between input and output layer and place where all the computation is done.
- Output layer — produce the result for given inputs.



## PERCEPTRON AND LEARNING THEORY

- The perceptron is also a simplified model of a biological neuron.
- The perceptron is an algorithm for supervised learning of binary classifiers. It is a type of linear classifier, i.e. a classification algorithm that makes all of its predictions based on a linear predictor function combining a set of weights with the feature vector.
- One type of ANN system is based on a unit called a perceptron.

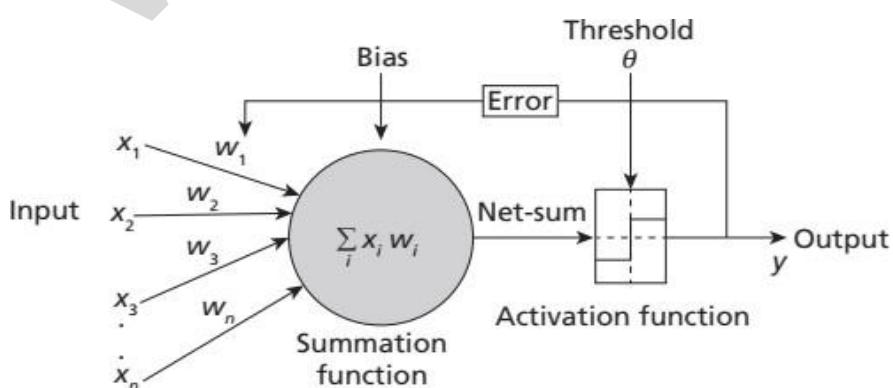
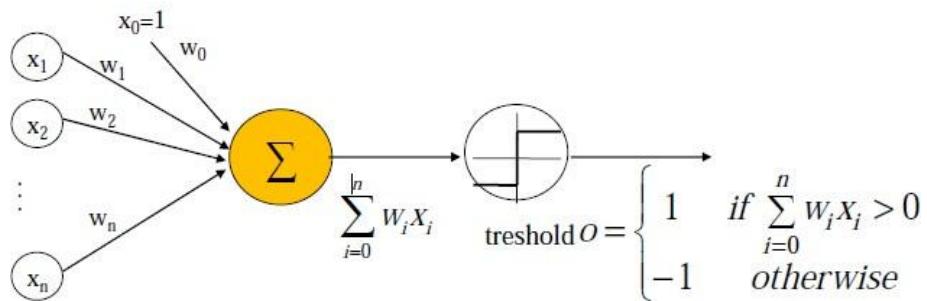


Figure 10.5: Perceptron Model

OR

$$W_0 + W_1X_1 + \dots + W_nX_n = \sum_{i=0}^n W_iX_i = \vec{W} \cdot \vec{X}$$



- The perceptron can represent all boolean primitive functions AND, OR, NAND , NOR.
- Some boolean functions can not be represented .
  - E.g. the XOR function.

### Major components of a perceptron

- Input
- Weight
- Bias
- Weighted summation
- Step/activation function
- output

### WORKING:

- Feed the features of the model that is required to be trained as input in the first layer. All weights and inputs will be multiplied – the multiplied result of each weight and input will be added up. The Bias value will be added to shift the output function .This value will be presented to the activation function (the type of activation function will depend on the need) The value received after the last step is the output value.

The activation function is a binary step function which outputs a value 1, if  $f(x)$  is above the threshold value  $\Theta$  and a 0 if  $f(x)$  is below the threshold value  $\Theta$ . Then the output of a neuron is:

$$f(x) = \begin{cases} 1 & \text{if } f(x) \geq \theta \\ 0 & \text{if } f(x) < \theta \end{cases}$$

**Algorithm 10.1: Perceptron Algorithm**

Set initial weights  $w_1, w_2, \dots, w_n$  and bias  $\theta$  to a random value in the range  $[-0.5, 0.5]$ .

For each Epoch,

1. Compute the weighted sum by multiplying the inputs with the weights and add the products.

2. Apply the activation function on the weighted sum:

$$Y = \text{Step}((x_1 w_1 + x_2 w_2) - \theta)$$

3. If the sum is above the threshold value, output the value as positive else output the value as negative.

4. Calculate the error by subtracting the estimated output  $Y_{\text{estimated}}$  from the desired output  $Y_{\text{desired}}$ :

$$\text{error } e(t) = Y_{\text{desired}} - Y_{\text{estimated}}$$

[If error  $e(t)$  is positive, increase the perceptron output  $Y$  and if it is negative, decrease the perceptron output  $Y$ .]

5. Update the weights if there is an error:

$$\Delta w_i = \alpha \times e(t) \times x_i$$

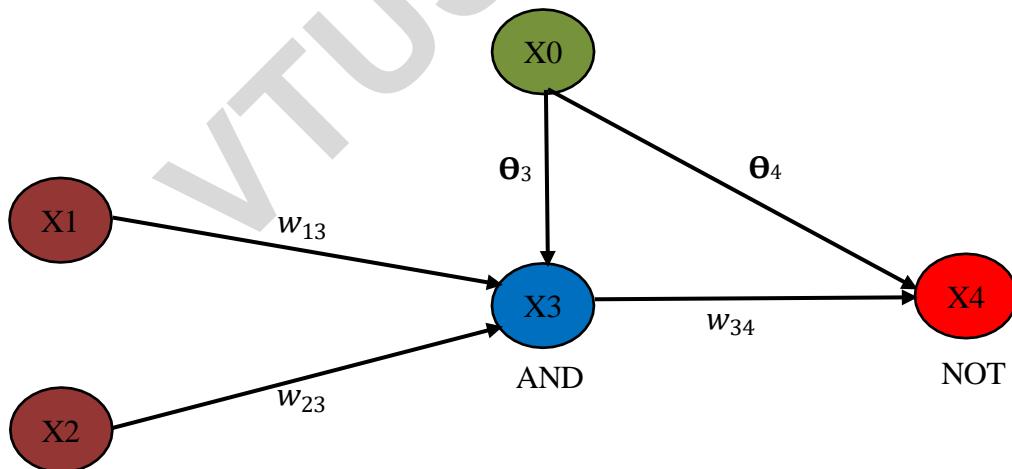
$$w_i = w_i + \Delta w_i$$

where,  $x_i$  is the input value,  $e(t)$  is the error at step  $t$ ,  $\alpha$  is the learning rate and  $\Delta w_i$  is the difference in weight that has to be added to  $w_i$ .

**PROBLEM:**

Design a 2 layer network of perceptron to implement NAND gate. Assume your own weights and biases in the range of  $[-0.5, 0.5]$ . Use learning rate as 0.4.

**Solution:**



**Figure 1 Two Layer Network for NAND gate**

**Table 1: Weights and Biases**

X <sub>1</sub>	X <sub>2</sub>	O <sub>desired</sub>	w <sub>13</sub>	w <sub>23</sub>	w <sub>34</sub>	θ <sub>3</sub>	θ <sub>4</sub>	X <sub>0</sub>
0	1	1	0.1	-0.4	0.3	0.2	-0.3	1

**Table 2: Truth Table of NAND Gate**

$X_1$	$X_2$	$X_1 \text{ AND } X_2$	$NAND = NOT(X_1 \text{ AND } X_2)$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

### **ITERATION 1:**

#### ***Step 1: FORWARD PROPAGATION***

- Calculate net inputs and outputs in input layer as shown in Table 3.

**Table 3: Net Input and Output Calculation**

<i>Input Layer</i>	$I_j$	$O_j$
$X_1$	0	0
$X_2$	1	1

- Calculate net inputs and outputs in hidden and output layer as shown in Table 4.

**Table 4: Net Input and Output Calculation in Hidden and Output layer**

<i>Unit<sub>j</sub></i>	<i>Net Input I<sub>j</sub></i>	<i>Net output O<sub>j</sub></i>
$X_3$	$I_3 = X_1W_{13} + X_2W_{23} + X_0\Theta_3$ $= 0(0.1) + 1(-0.4) + 1(0.2)$ $= -0.2$	$\Theta_3 = \frac{1}{1 + e^{-I_3}}$ $= \frac{1}{1 + e^{-(0.2)}}$ $= 0.450$
$X_4$	$I_4 = O_3W_{34} + X_0\Theta_4$ $= (0.450 * 0.3) + 1(-0.3)$ $= -0.165$	$\Theta_4 = \frac{1}{1 + e^{-I_4}}$ $= \frac{1}{1 + e^{(-0.165)}}$ $= 0.458$

### **3. Calculate Error**

$$\begin{aligned}
 \text{Error} &= O_{desired} - O_{estimated} \\
 &= 1 - 0.458
 \end{aligned}$$

$$Error = 0.542$$

### Step 2: BACKWARD PROPAGATION

- For each  $unit_k$  in the output layer

$$Error_k = O_k * (1 - O_k) * (O_{desired} - O_k)$$

For each  $unit_j$  in the hidden layer

$$Error_j = O_j * (1 - O_j) * (\sum_k Error_k * W_{jk})$$

**Table 5: Error Calculation**

<i>For each output layer unit<sub>k</sub></i>	<i>Error<sub>k</sub></i>
$X_4$	$\begin{aligned} Error_k &= O_k * (1 - O_k) * (O_{desired} - O_k) \\ &= 0.458(1 - 0.458)(1 - 0.458) \\ &= 0.134 \end{aligned}$
<i>For each hidden layer unit<sub>j</sub></i>	<i>Error<sub>j</sub></i>
$X_3$	$\begin{aligned} Error_j &= O_j * (1 - O_j) * (\sum_k Error_k * W_{jk}) \\ &= 0.450 * (1 - 0.450) * 0.134 * 0.3 \\ &= 0.0099 \end{aligned}$

- Update Weights and biases

**Table 6: Weight and Bias Calculation**

$w_{ij}$	$w_{ij} = w_{ij} + (\alpha * Error_j * O_i)$	<i>Net Weight</i>
$w_{13}$	$\begin{aligned} w_{13} &= w_{13} + (0.4 * Error_3 * O_1) \\ &= 0.1 * (0.4 * 0.0099 * 0) \end{aligned}$	0.1
$w_{23}$	$\begin{aligned} w_{23} &= w_{23} + (0.4 * Error_3 * O_2) \\ &= -0.4 * (0.4 * 0.0099 * 1) \end{aligned}$	-0.396
$w_{24}$	$\begin{aligned} w_{24} &= w_{24} + (0.4 * Error_4 * O_2) \\ &= 0.3 * (0.4 * 0.134 * 0.450) \end{aligned}$	0.324

$\Theta_j$	$\Theta_j = \Theta_j + (\alpha * Error_j)$	Net Bias
$\Theta_3$	$\Theta_3 = \Theta_3 + (0.4 * Error_3)$ $= 0.2 + (0.4 * 0.0099)$	0.203
$\Theta_4$	$\Theta_4 = \Theta_4 + (0.4 * Error_4)$ $= -0.3 + (0.4 * 0.134)$	-0.246

## ITERATION 2:

### Step 1: FORWARD PROPAGATION

- Calculate net inputs and outputs in hidden and output layer

Table 7: Inputs and Outputs in Hidden and Output layer

<i>Unit<sub>j</sub></i>	<i>Net Input I<sub>j</sub></i>	<i>Net output O<sub>j</sub></i>
$X_3$	$I_3 = X_1W_{13} + X_2W_{23} + X_0\Theta_3$ $= 0(0.1) + 1(-0.396) + 1(0.203)$ $= -0.193$	$O_3 = \frac{1}{1 + e^{-I_3}}$ $= \frac{1}{1 + e^{-(0.193)}}$ $= 0.451$
$X_4$	$I_4 = O_3W_{34} + X_0\Theta_4$ $= (0.451 * 0.324) + 1(-0.246)$ $= -0.099$	$O_4 = \frac{1}{1 + e^{-I_4}}$ $= \frac{1}{1 + e^{(-0.099)}}$ $= 0.475$

- Calculate Error

$$\begin{aligned} Error &= O_{desired} - O_{estimated} \\ &= 1 - 0.475 \\ Error &= 0.525 \end{aligned}$$

ITERATION	ERROR
1	0.542
2	0.525

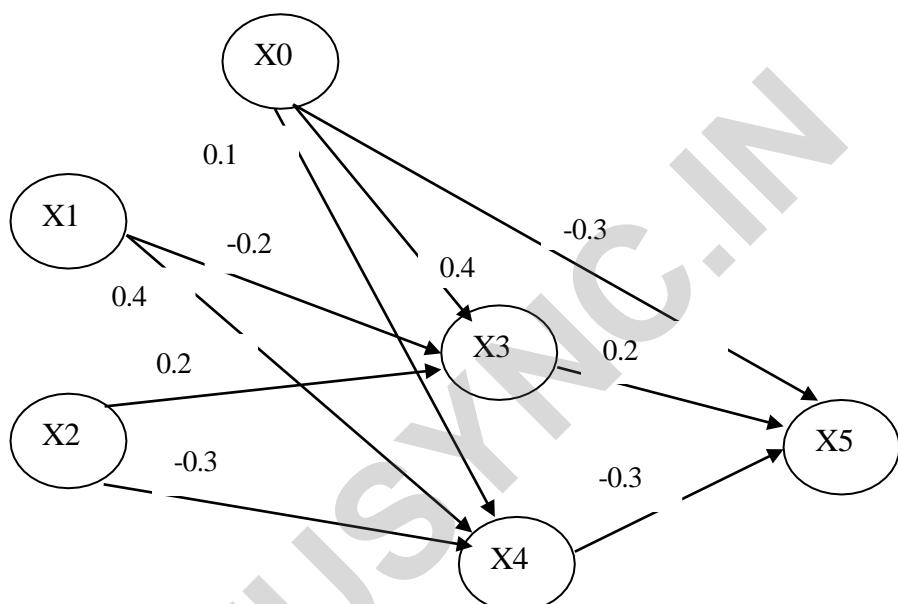
$$\begin{aligned} &= 0.542 - 0.525 \\ &= 0.017 \end{aligned}$$

In iteration 2 the error gets reduced to 0.525. This process will continue until desired output is achieved.

How a Multi-Layer Perceptron does solves the XOR problem. Design an MLP with back propagation to implement the XOR Boolean function.

**Solution:**

X <sub>1</sub>	X <sub>2</sub>	Y
0	0	1
0	1	0
1	0	0
1	1	1



**Figure 2: Multi Layer Perceptron for XOR**

Learning rate: =0.8

**Table 8: Weights and Biases**

X <sub>1</sub>	X <sub>2</sub>	W <sub>13</sub>	W <sub>14</sub>	W <sub>23</sub>	W <sub>24</sub>	W <sub>35</sub>	W <sub>45</sub>	θ <sub>3</sub>	θ <sub>4</sub>	θ <sub>5</sub>
1	0	-0.2	0.4	0.2	-0.3	0.2	-0.3	0.4	0.1	-0.3

### Step 1: Forward Propagation

1. Calculate Input and Output in the Input Layer shown in Table 9.

**Table 9: Net Input and Output Calculation**

Input Layer	I <sub>j</sub>	O <sub>j</sub>
<b>X<sub>1</sub></b>	1	1
<b>X<sub>2</sub></b>	0	0

2. Calculate Net Input and Output in the Hidden Layer and Output Layer shown in Table 10.

**Table 10: Unit j at Hidden Layer and Output Layer – Net Input and Output Calculation**

Unit j	Net Input I <sub>j</sub>	Output O <sub>j</sub>
X <sub>3</sub>	$I_3 = X_1 * W_{13} + X_2 * W_{23} + X_0 * \theta_3$ $I_3 = 1 * -0.2 + 0 * 0.2 + 1 * 0.4 = 0.2$	$O_3 = \frac{1}{1+e^{-I_3}} = \frac{1}{1+e^{-0.2}} = 0.549$
X <sub>4</sub>	$I_4 = X_1 * W_{14} + X_2 * W_{24} + X_0 * \theta_4$ $I_4 = 1 * 0.4 + 0 * -0.3 + 1 * 0.1 = 0.5$	$O_4 = \frac{1}{1+e^{-I_4}} = \frac{1}{1+e^{-0.5}} = 0.622$
X <sub>5</sub>	$I_5 = O_3 * W_{35} + O_4 * W_{45} + X_0 * \theta_5$ $I_5 = 0.549 * 0.2 + 0.622 * -0.3 + 1 * -0.3 = -0.376$	$O_5 = \frac{1}{1+e^{-I_5}} = \frac{1}{1+e^{0.376}} = 0.407$

3. Calculate Error = O<sub>desired</sub> – O<sub>Estimated</sub>

So error for this network is,

$$\text{Error} = O_{\text{desired}} - O_7 = 1 - 0.407 = 0.593$$

## Step 2: Backward Propagation

1. Calculate Error at each node as shown in Table 11.

For each unit k in the output layer, calculate

$$\text{Error}_k = O_k (1-O_k) (Y_N - O_k)$$

For each unit j in the hidden layer, calculate

$$\text{Error}_j = O_j (1-O_j) \sum_k \text{Error}_k W_{jk}$$

**Table 11: Error Calculation for each unit in the Output layer and Hidden layer**

For Output Layer Unit k	Error <sub>k</sub>
X <sub>5</sub>	$\text{Error}_5 = O_5 (1-O_5) (1 - O_5)$ $= 0.407 * (1-0.407) * (1- 0.407)$ $= 0.143$
For Hidden layer Unit j	Error <sub>j</sub>
X <sub>4</sub>	$\text{Error}_4 = O_4 (1-O_4) \sum_k \text{Error}_k W_{jk} = O_4 (1-O_4) \text{Error}_5 W_{45}$ $= 0.622 (1-0.622) * -0.3 * 0.143$ $= -0.010$
X <sub>3</sub>	$\text{Error}_3 = O_3 (1-O_3) \sum_k \text{Error}_k W_{jk} = O_3 (1-O_3) \text{Error}_5 W_{35}$ $= 0.549 (1- 0.549) * 0.143 * 0.2$

	= -0.007
--	----------

2. Update weight using the below formula,

Learning rate  $\alpha = 0.8$

$$\Delta W_{ij} = \alpha * \text{Error}_j * O_i$$

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

The updated weight and bias is shown in Table 12 and Table 13.

**Table 12: Weight Updation**

$W_{ij}$	$W_{ij} = W_{ij} + \alpha * \text{Error}_j * O_i$	New Weight
$W_{13}$	$W_{13} = W_{13} + 0.8 * \text{Error}_3 * O_1$ = $-0.2 + 0.8 * 0.007 * 1$	-0.194
$W_{14}$	$W_{14} = W_{14} + 0.8 * \text{Error}_4 * O_1$ = $0.4 + 0.8 * -0.01 * 1$	0.392
$W_{23}$	$W_{23} = W_{23} + 0.8 * \text{Error}_3 * O_2$ = $0.2 + 0.8 * 0.007 * 0$	0.2
$W_{24}$	$W_{24} = W_{24} + 0.8 * \text{Error}_4 * O_2$ = $-0.3 + 0.8 * -0.001 * 0$	-0.3
$W_{35}$	$W_{35} = W_{35} + 0.8 * \text{Error}_5 * O_3$ = $0.2 + 0.8 * 0.143 * 0.4$	0.154
$W_{45}$	$W_{45} = W_{45} + 0.8 * \text{Error}_5 * O_4$ = $0.3 + 0.8 * 0.143 * 0.1$	-0.288

Update bias using the below formula,

$$\Delta \theta_j = \alpha * \text{Error}_j$$

$$\theta_j = \theta_j + \Delta \theta_j$$

**Table 13: Bias Updation**

$\theta_j$	$\theta_j = \theta_j + \alpha * \text{Error}_j$	New Bias
$\theta_3$	$\theta_3 = \theta_3 + \alpha * \text{Error}_3$ = $0.4 + 0.8 * 0.007$	0.405
$\theta_4$	$\theta_4 = \theta_4 + \alpha * \text{Error}_4$ = $0.1 + 0.8 * -0.01$	0.092
$\theta_5$	$\theta_5 = \theta_5 + \alpha * \text{Error}_5$ = $-0.3 + 0.8 * 0.143$	-0.185

## Iteration 2

Now with the updated weights and biases,

1. Calculate Input and Output in the Input Layer shown in Table 14.

**Table 14: Net Input and Output Calculation**

Input Layer	I <sub>j</sub>	O <sub>j</sub>
X <sub>1</sub>	1	1
X <sub>2</sub>	0	0

2. Calculate Net Input and Output in the Hidden Layer and Output Layer shown in Table 15.

**Table 15: Net Input and Output Calculation in the Hidden Layer and Output Layer**

Unit j	Net Input I <sub>j</sub>	Output O <sub>j</sub>
X <sub>3</sub>	$I_3 = X_1 * W_{13} + X_2 * W_{23} + X_0 * \theta_3$ $I_3 = 1 * -0.194 + 0 * 0.2 + 1 * 0.405 = 0.211$	$O_3 = \frac{1}{1+e^{-I_3}} = \frac{1}{1+e^{-0.211}} = 0.552$
X <sub>4</sub>	$I_4 = X_1 * W_{14} + X_2 * W_{24} + X_0 * \theta_4$ $I_4 = 1 * 0.392 + 0 * -0.3 + 1 * 0.092 = 0.484$	$O_4 = \frac{1}{1+e^{-I_4}} = \frac{1}{1+e^{-0.484}} = 0.618$
X <sub>5</sub>	$I_5 = O_3 * W_{35} + O_4 * W_{45} + X_0 * \theta_5$ $I_5 = 0.552 * 0.154 + 0.618 * -0.288 + 1 * -0.185 = -0.282$	$O_5 = \frac{1}{1+e^{-I_5}} = \frac{1}{1+e^{0.282}} = 0.429$

The output we receive in the network at node 5 is 0.407.

$$\text{Error} = 1 - 0.429 = 0.571$$

Now when we compare the error, we get in the previous iteration and in the current iteration, the network has learnt which reduces the error by 0.022.

**Error is reduced by 0.055:** 0.593 – 0.571.

Consider the Network architecture with 4 input units and 2 output units. Consider four training samples each vector of length 4.

Training samples

$$i1: (1, 1, 1, 0)$$

$$i2: (0, 0, 1, 1)$$

$$i3: (1, 0, 0, 1)$$

$$i4: (0, 0, 1, 0)$$

Output Units: Unit 1, Unit 2

Learning rate  $\eta(t) = 0.6$

Initial Weight matrix

$$[\text{Unit 1}]: [ \begin{array}{cccc} 0.2 & 0.8 & 0.5 & 0.1 \end{array}]$$
$$[\text{Unit 2}]: [ \begin{array}{cccc} 0.3 & 0.5 & 0.4 & 0.6 \end{array}]$$

Identify an algorithm to learn without supervision? How do you cluster them as we expected?

**Solution:**

Use Self Organizing Feature Map (SOFM)

**Iteration 1:**

Training Sample  $X_1: (1, 1, 1, 0)$

Weight matrix

$$[\text{Unit 1}]: [ \begin{array}{cccc} 0.2 & 0.8 & 0.5 & 0.1 \end{array}]$$
$$[\text{Unit 2}]: [ \begin{array}{cccc} 0.3 & 0.5 & 0.4 & 0.6 \end{array}]$$

Compute Euclidean distance between  $X_1: (1, 1, 1, 0)$  and Unit 1 weights.

$$d^2 = (0.2 - 1)^2 + (0.8 - 1)^2 + (0.5 - 1)^2 + (0.1 - 0)^2$$
$$= 0.94$$

Compute Euclidean distance between  $X_1: (1, 1, 1, 0)$  and Unit 2 weights.

$$d^2 = (0.3 - 1)^2 + (0.5 - 1)^2 + (0.4 - 1)^2 + (0.6 - 0)^2$$
$$= 1.46$$

**Unit 1 wins**

Update the weights of the winning unit

$$\text{New Unit 1 weights} = [0.2 \ 0.8 \ 0.5 \ 0.2] + 0.6 ([1 \ 1 \ 1 \ 0] - [0.2 \ 0.8 \ 0.5 \ 0.2])$$
$$= [0.2 \ 0.8 \ 0.5 \ 0.2] + 0.6 [0.8 \ 0.2 \ 0.5 \ -0.2]$$
$$= [0.2 \ 0.8 \ 0.5 \ 0.2] + [0.48 \ 0.12 \ 0.30 \ -0.12]$$
$$= [0.68 \ 0.92 \ 0.80 \ 0.08]$$

$$[\text{Unit 1}]: [ \begin{array}{cccc} 0.68 & 0.92 & 0.80 & 0.08 \end{array}]$$
$$[\text{Unit 2}]: [ \begin{array}{cccc} 0.3 & 0.5 & 0.4 & 0.6 \end{array}]$$

**Iteration 2:**

Training Sample  $X_2: (0, 0, 1, 1)$

Weight matrix

$$[\text{Unit 1}]: [ \begin{array}{cccc} 0.68 & 0.92 & 0.80 & 0.08 \end{array}]$$
$$[\text{Unit 2}]: [ \begin{array}{cccc} 0.3 & 0.5 & 0.4 & 0.6 \end{array}]$$

Compute Euclidean distance between  $X_2$ : (0, 0, 1, 1) and Unit 1 weights.

$$\begin{aligned}d^2 &= (0.68 - 0)^2 + (0.92 - 0)^2 + (0.80 - 1)^2 + (0.08 - 1)^2 \\&= 2.1952\end{aligned}$$

Compute Euclidean distance between  $X_2$ : (0, 0, 1, 1) and Unit 2 weights.

$$\begin{aligned}d^2 &= (0.3 - 0)^2 + (0.5 - 0)^2 + (0.4 - 1)^2 + (0.6 - 1)^2 \\&= 0.86\end{aligned}$$

### **Unit 2 wins**

Update the weights of the winning unit

$$\begin{aligned}\text{New Unit 2 weights} &= [0.3 \ 0.5 \ 0.4 \ 0.6] + 0.6 ([0 \ 0 \ 1 \ 1] - [0.3 \ 0.5 \ 0.4 \ 0.6]) \\&= [0.3 \ 0.5 \ 0.4 \ 0.6] + 0.6 [-0.3 \ -0.5 \ 0.6 \ 0.4] \\&= [0.3 \ 0.5 \ 0.4 \ 0.6] + [-0.18 \ -0.30 \ 0.36 \ 0.24] \\&= [0.12 \ 0.2 \ 0.76 \ 0.84]\end{aligned}$$

$$[\text{Unit 1}]: [0.68 \ 0.92 \ 0.80 \ 0.08]  
[\text{Unit 2}]: [0.12 \ 0.2 \ 0.76 \ 0.84]$$

### **Iteration 3:**

Training Sample  $X_3$ : (1, 0, 0, 1)

Weight matrix

$$[\text{Unit 1}]: [0.68 \ 0.92 \ 0.80 \ 0.08]  
[\text{Unit 2}]: [0.12 \ 0.2 \ 0.76 \ 0.84]$$

Compute Euclidean distance between  $X_3$ : (1, 0, 0, 1) and Unit 1 weights.

$$\begin{aligned}d^2 &= (0.68 - 1)^2 + (0.92 - 0)^2 + (0.80 - 0)^2 + (0.08 - 1)^2 \\&= 2.44\end{aligned}$$

Compute Euclidean distance between  $X_3$ : (1, 0, 0, 1) and Unit 2 weights.

$$\begin{aligned}d^2 &= (0.12 - 1)^2 + (0.2 - 0)^2 + (0.76 - 0)^2 + (0.84 - 1)^2 \\&= 1.42\end{aligned}$$

### **Unit 2 wins**

Update the weights of the winning unit

$$\begin{aligned}\text{New Unit 2 weights} &= [0.12 \ 0.2 \ 0.76 \ 0.84] + 0.6 ([1 \ 0 \ 0 \ 1] - [0.12 \ 0.2 \ 0.76 \ 0.84]) \\&= [0.12 \ 0.2 \ 0.76 \ 0.84] + 0.6 [0.88 \ -0.2 \ -0.76 \ 0.16]\end{aligned}$$

$$= [0.12 \ 0.2 \ 0.76 \ 0.84] + [0.53 \ -0.12 \ -0.46 \ 0.096]$$

$$= [0.65 \ 0.08 \ 0.3 \ 0.94]$$

$$[\text{Unit 1}]: [\begin{matrix} 0.68 & 0.92 & 0.80 & 0.08 \end{matrix}]$$

$$[\text{Unit 2}]: [\begin{matrix} 0.65 & 0.08 & 0.3 & 0.94 \end{matrix}]$$

#### **Iteration 4:**

Training Sample  $X_4$ : (0, 0, 1, 0)

Weight matrix

$$[\text{Unit 1}]: [\begin{matrix} 0.68 & 0.92 & 0.80 & 0.08 \end{matrix}]$$

$$\text{Unit 2} \quad [\begin{matrix} 0.65 & 0.08 & 0.3 & 0.94 \end{matrix}]$$

Compute Euclidean distance between  $X_4$ : (0, 0, 1, 0) and Unit 1 weights.

$$d^2 = (0.68 - 0)^2 + (0.92 - 0)^2 + (0.80 - 1)^2 + (0.08 - 0)^2$$

$$= 1.36$$

Compute Euclidean distance between  $X_1$ : (0, 0, 1, 0) and Unit 2 weights.

$$d^2 = (0.65 - 0)^2 + (0.08 - 0)^2 + (0.3 - 1)^2 + (0.94 - 0)^2$$

$$= 1.8025$$

#### **Unit 1 wins**

Update the weights of the winning unit

$$\text{New Unit 1 weights} = [0.68 \ 0.92 \ 0.80 \ 0.08] + 0.6 ([0 \ 0 \ 1 \ 0] - [0.68 \ 0.92 \ 0.80 \ 0.08])$$

$$= [0.68 \ 0.92 \ 0.80 \ 0.08] + 0.6 [-0.68 \ -0.92 \ 0.2 \ -0.08]$$

$$= [0.68 \ 0.92 \ 0.80 \ 0.08] + [-0.408 \ -0.552 \ 0.12 \ -0.258]$$

$$= [0.27 \ 0.37 \ 0.92 \ -0.178]$$

$$[\text{Unit 1}]: [\begin{matrix} 0.27 & 0.37 & 0.92 & -0.178 \end{matrix}]$$

$$[\text{Unit 2}]: [\begin{matrix} 0.65 & 0.08 & 0.3 & 0.94 \end{matrix}]$$

Best mapping unit for each of the sample taken are,

$X_1$ : (1, 1, 1, 0)  $\rightarrow$  Unit 1

$X_2$ : (0, 0, 1, 1)  $\rightarrow$  Unit 2

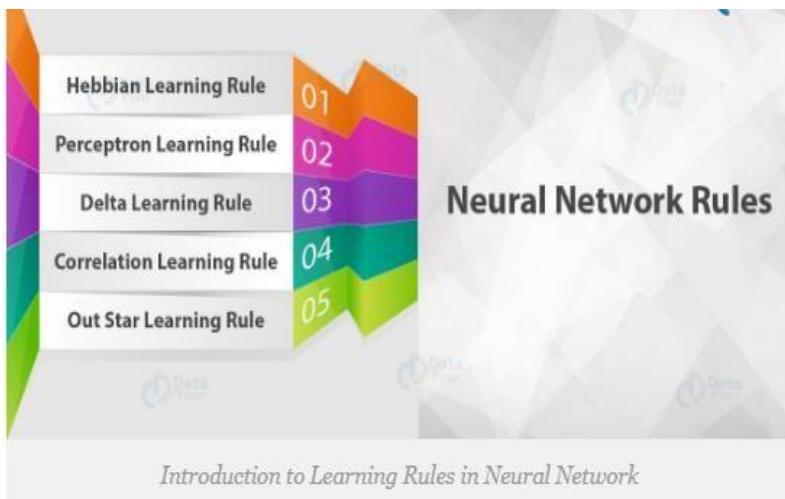
$X_3$ : (1, 0, 0, 1)  $\rightarrow$  Unit 2

$X_4$ : (0, 0, 1, 0)  $\rightarrow$  Unit 1

This process is continued for many epochs until the feature map doesn't change.

## Learning Rules

Learning in NN is performed by adjusting the network weights in order to minimize the difference between the desired and estimated output.



## Delta Learning Rule and Gradient Descent

- Developed by **Widrow and Hoff**, the delta rule, is one of the most common learning rules.
- It is **supervised** learning.
- Delta rule is derived from gradient descent method(Back-propagation).
- It is **Non-linearly separable**. Also called as continuous perceptron Learning rule.
- It updates the connection weights with the difference between the target and the output value. It is the least mean square learning algorithm.
- The Delta difference is measured as an **error function** or also called as **cost** function.

$$\text{Training Error} = \frac{1}{2} \sum_{d \in T} (O_{\text{Desired}} - O_{\text{Estimated}})^2$$

where,  $T$  is the training dataset,  $O_{\text{Desired}}$  and  $O_{\text{Estimated}}$  are the desired target output and estimated actual output, respectively, for a training instance  $d$ .

The principle of gradient descent is an optimization approach which is used to minimize the cost function by converging to a local minimal point moving in the negative direction of the gradient and each step size during movement is determined by the learning rate and the slope of the gradient.

## **TYPES OF ANN**

1. Feed Forward Neural Network
2. Fully connected Neural Network
3. Multilayer Perceptron
4. Feedback Neural Network

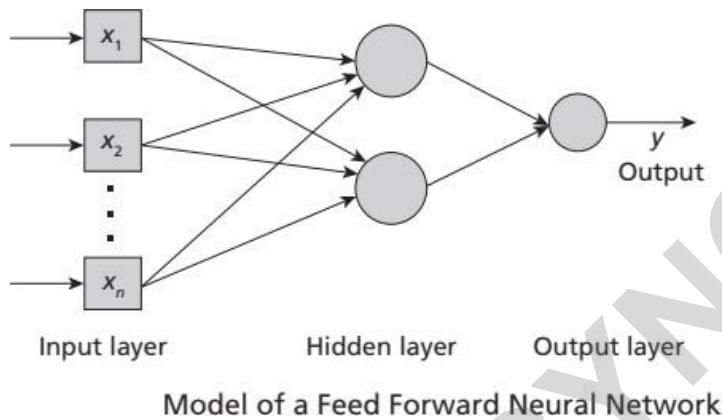
## **Feed Forward Neural Network:**

Feed-Forward Neural Network is a single layer perceptron. A sequence of inputs enters the layer and are multiplied by the weights in this model. The weighted input values are then summed together to form a total. If the sum of the values is more than a predetermined threshold, which is normally set at zero, the output value is usually 1, and if the sum is less than the threshold, the output value is usually -1.

The single-layer perceptron is a popular feed-forward neural network model that is frequently used for classification.

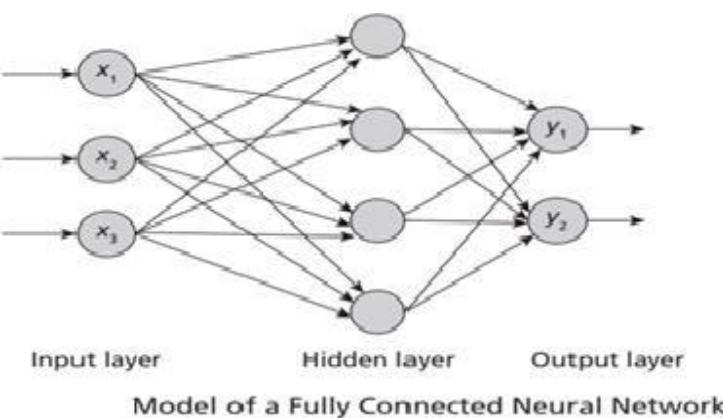
The model may or may not contain hidden layer and there is no backpropagation.

Based on the number of hidden layers they are further classified into single-layered and multilayered feed forward network.



## **Fully connected Neural Network:**

- A fully connected neural network consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layer.
- The major advantage of fully connected networks is that they are “structure agnostic” i.e. there are no special assumptions needed to be made about the input.



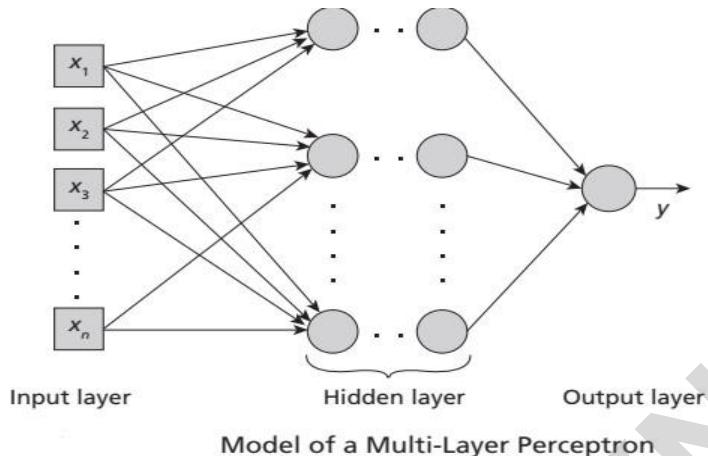
## Multilayer Perceptron:

A multi-layer perceptron has one input layer and for each input, there is one neuron (or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes.

The information flows in both directions.

The weight adjustment training is done via **backpropagation**.

Every node in the multi-layer perception uses a sigmoid activation function. The sigmoid activation function takes real values as input and converts them to numbers between 0 and 1 using the sigmoid formula.



## Feedback Neural Network:

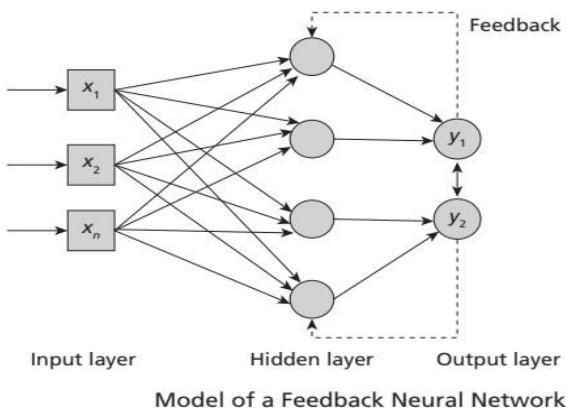
Feedback networks also known as **recurrent neural network** or interactive neural network are the deep learning models in which information flows in **backward** direction.

It allows feedback loops in the network. Feedback networks are dynamic in nature, powerful and can get much complicated at some stage of execution

Neuronal connections can be made in any way.

RNNs may process input sequences of different lengths by using their internal state, which can represent a form of memory.

They can therefore be used for applications like speech recognition or handwriting recognition.



## **Advantages and Disadvantages of ANN**

1. ANN can solve complex problems involving non-linear processes.
2. ANNs can learn and recognize complex patterns and solve problems as humans solve a problem.
3. ANNs have a parallel processing capability and can predict in less time.
4. They have an ability to work with inadequate knowledge. It can even handle incomplete and noisy data.
5. They can scale well to larger data sets and outperforms other learning mechanisms.

## ***Limitations of ANN***

1. An ANN requires processors with parallel processing capability to train the network running for many epochs. The function of each node requires a CPU capability which is difficult for very large networks with a large amount of data.
2. They work like a 'black box' and it is exceedingly difficult to understand their working in inner layers. Moreover, it is hard to understand the relationship between the representations learned at each layer.

3. The modelling with ANN is also extremely complicated and the development takes a much longer time.
4. Generally, neural networks require more data than traditional machine learning algorithms, and they do not perform well on small datasets.
5. They are also more computationally expensive than traditional learning techniques.

## **Challenges of Artificial Neural Networks**

The major challenges while modelling a real-time application with ANNs are:

1. Training a neural network is the most challenging part of using this technique. Overfitting or underfitting issues may arise if datasets used for training are not correct. It is also hard to generalize to the real-world data when trained with some simulated data. Moreover, neural network models normally need a lot of training data to be robust and are usable for a real-time application.
2. Finding the weight and bias parameters for neural networks is also hard and it is difficult to calculate an optimal model.