First Order Predicate Logic

- In PL, seen that how to represent statements using PL. But only represent facts, which are either true or false.
- PL is not sufficient to represent the complex sentences or natural language statements.
- PL has very limited expressive power. Some sentence cannot represent using PL logic.

“All books available in liberary”

"Some Dogs like Biscuit"

- PL logic is not sufficient, so more powerful logic required, such as first-order logic.

First Order Predicate Logic

- It is an extension to propositional logic.
- FOL is sufficiently expressive to represent the natural language statements in a concise way.
- FOL is a powerful language that develops information about the objects in a more easy way & can also express the relationship b/w those objects.
- FOL does not only assume that the world contains facts like PL but also assumes **objects, functions & predicate or relations** in the world.
- As a natural language, first-order logic also has two main parts:
  - **Syntax**
  - **Semantics**

FOL Statement

- **First-order logic statements can be divided into two parts**
- **Subject:** Subject is the main part of the statement
- **Predicate:** A predicate can be defined as a relation, which binds two atoms together in a statement.
- **Ajay is Student**
- **Tommy is a dog**
- **Ravi and Ajay are brothers**

Quantifiers in FOL

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
  - **Universal Quantifier, (for all, everyone, everything)**
  - **Existential quantifier, (for some, at least one).**

Universal Quantifier

- Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.
- The Universal quantifier is represented by a symbol ∀, which resembles an inverted A.
- Note: In universal quantifier we use implication "→".
- If x is a variable, then ∀x is read as:
  - **For all x**
  - **For each x**
  - **For every x.**

Universal Quantifier

- **All man drink coffee. {X1, X2, X3,……..,Xn} drink coffee**

  $\forall x \; man(x) \rightarrow drink(x, coffee)$.

- **All birds fly.**

  $\forall x \; bird(x) \rightarrow fly(x)$.

- **Every man respects his parent.**

  $\forall x \; man(x) \rightarrow respects(x, parent)$.

- **Not all students like both Math and Science.**

  $\neg \forall (x) \; [\, student(x) \rightarrow like(x, Math) \wedge like(x, Science)]$.

Existential Quantifier

- It is the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.
- denoted by the logical operator ∃, which resembles as inverted E. When it is used with a predicate variable, called as an existential quantifier.
- In Existential quantifier, always use AND or Conjunction symbol (∧).
- If x is a variable, then existential quantifier:- ∃x / ∃(x). And it will be read as:
  - **There exists a 'x.'**
  - **For some 'x.'**
  - **For at least one 'x.'**

Existential Quantifier

- **Some boys are intelligent.**

  $\exists x$: boys(x) $\wedge$ intelligent(x)

- **Some boys play cricket.**

  $\exists x$ boys(x) $\wedge$ play(x, cricket).

- **Only one student failed in Math.**

  $\exists(x)$ [student(x) $\rightarrow$ failed (x, Math) $\wedge \forall$ (y) [$\neg$(x==y) $\wedge$ student(y) $\rightarrow \neg$failed (x, Math)].

Unification

- Unification is a kind of binding logic between two or more variables.
- In propositional logic it is easy to determine that two literals can not both be true at the same time.

  Eg. Man(Sunil) and ¬Man(Sunil) is a contradiction

  while Man(Sunil) and ¬Man(Arun) is not .
- In predicate logic, this matching process is more complicated, since bindings of variables must be considered.
- In Predicate logic in order to determine contradiction we need a matching procedure that compares two literals and discovers whether there exist a set of substitutions that make them identical. **Unification algorithms**

Unification...

- The goal of unification is to make two expression look like identical by using substitution
- It means the meaning of the sentence should not be changed, but it should be expressed in multiple ways.
- The **UNIFY** algorithm in unification takes two sentences as input and then returns a unifier if one exists:
  - Substitution means replacing one variable with another term.
  - It takes two literals as input and makes them identical using substitution.
  - It returns fail if the expressions do not match with each other.

UNIFY(p, q)= θ where SUBST(θ , p) = SUBST(θ, q).

Example #1

- Let's say there are two different expressions,
  P(x, y), and P(a, f(z)).
- we need to make both above statements identical to each other.
- Perform the substitution.

  P(x, y)..... (i)
  P(a, f(z))..... (ii)
- Substitute x with a, and y with f(z) in the first expression, and it will be represented as **a/x** and **f(z)/y**.
- With both the substitutions, the first expression will be identical to the second expression and the substitution set will be: **[a/x, f(z)/y]**.

Example #2

- **Given:** Knows(Ram,x). Is a predicate
- **Whom does Ram knows?**
- The UNIFY algorithm will search all the related sentences in the knowledge base, which could unify with Knows(Ram, x).

  UNIFY (Knows(Ram, x), Knows(Ram, Shyam))≡{x/Shyam}

  UNIFY (Knows{Ram,x}, Knows{y, Aakash})≡{x/Aakash, y/Ram}

  UNIFY (Knows{Ram,x}, Knows{x, Raman})≡fails. **Unifier is empty**
- The last one failed because we have used the same variable for two persons or the two sentences happen to use the same variable name, x
- unifications are attempted only with sentences that have some chance of unifying.
- For example, there is no point in trying to

  unify Knows(Ram, x) with Brother(Laxman, Ram).

Conditions for Unification

- Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.

  tryassassinate (Marcus, Caesar)

  hate (Marcus, Caesar)

- Number of Arguments in both expressions must be identical.

  hate(Marcus)

  hate (Marcus, Caesar)

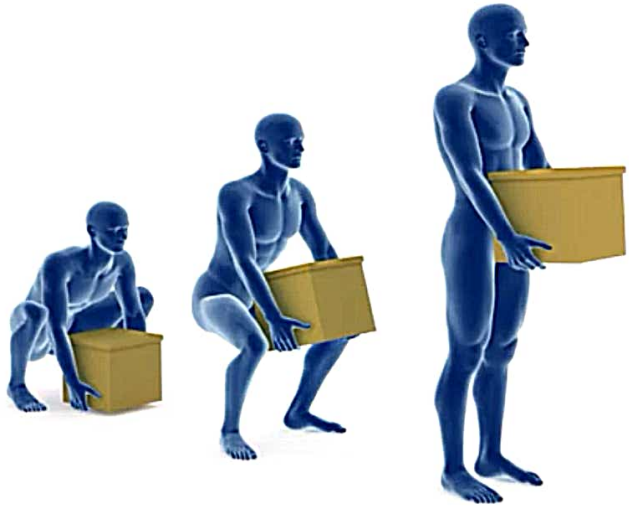
- Unification will fail if there are two similar variables present in the same expression.

Lifting

Lifting is a technique used in AI to generalize knowledge from specific examples.

It involves creating higher-level rules or concepts from lower-level ones.

Lifting allows AI systems to reason and make decisions based on abstract knowledge.



Forward chaining

- Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference rules (Modus Ponens) in the forward direction to extract more data until a goal is reached
- Here facts are held in a working memory and condition action rules represent actions to take when specified facts occur in working memory it may add and delete facts from working memory.

Properties

- It is a bottom-up approach, as it moves from bottom to top.
- It is a process of making a conclusion based on known facts or data, by starting from the initial state and reaches the goal state.
- Forward-chaining approach is also called as data-driven as the data determines which rules are selected and used.

- Example:

A	He exercises regularly.
A -> B	If he is exercising , he is fit.
B	He is fit.

Backward Chaining

- Backward chaining is a goal driven method of deriving a particular goal from a given knowledge base and set of inference rules
- The inference system knows the final decision or goal, this system starts from the goal and works backwards to determine what facts must be asserted so that the goal can be achieved.
- Example

  B He is fit.

  A -> B If he is exercising , he is fit

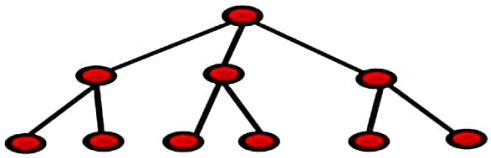
  A He exercises regularly.

Properties of backward chaining

- It is known as a top-down approach.
- Backward-chaining is based on modus ponens inference rule.
- The goal is broken into sub-goal or sub-goals to prove the facts true.
- It is called a goal-driven approach, as a list of goals decides which rules are selected and used.

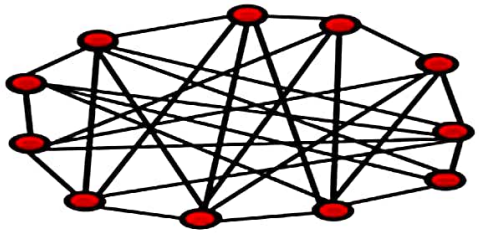
What is Forward and Backward chaining?

- A backward chaining algorithm is a form of reasoning, which starts with the goal and works backward, chaining through rules to find known facts that support the goal.



"Top-down"

- Forward chaining is a form of reasoning which start with atomic sentences in the knowledge base and applies inference in the forward direction to extract more data until a goal is reached.



"Bottom-up"

Forward Chaining V/S Backward Chaining

Forward chaining	Backward chaining
• It follows bottom up approach. • process of making a conclusion based on known data, by starting from initial state and reaches the goal state. • Forward-chaining approach is also called as data-driven as we reach to the goal using available data. • Forward -chaining approach is commonly used in the expert system, such as CLIPS,business, and production rule systems.	• It follows top down approach. • In backward chaining, the goal is broken into sub-goal or sub-goals to prove the facts true. • It is called a goal-driven approach, as a list of goals decides which rules are selected and used. • Backward -chaining algorithm is used in game theory, automated theorem proving tools, inference engines, proof assistants, and various AI applications.

Forward Chaining

Example: **Solved in class**

1. John likes all kind of food
2. Apple is food
3. Chicken is food
4. Anyone eats anything and alive is food
5. Bill eats peanut and alive
6. Peter eats everything Bill eats

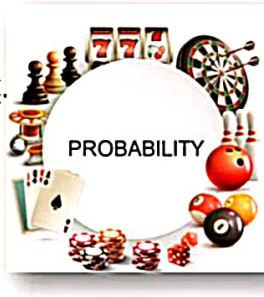
Question : John likes peanuts

Uncertainty

- When an agent knows enough facts about its environment, the logical plans and actions produces a guaranteed work.
- Unfortunately, *agents never have access to the whole truth about their environment.* Agents act under **uncertainty.**

About Probabiliy

- Probability means possibility.
- It is a branch of mathematics that deals with the occurrence of a random event.
- Probability is a measure of the likelihood of an event to occur.



Example

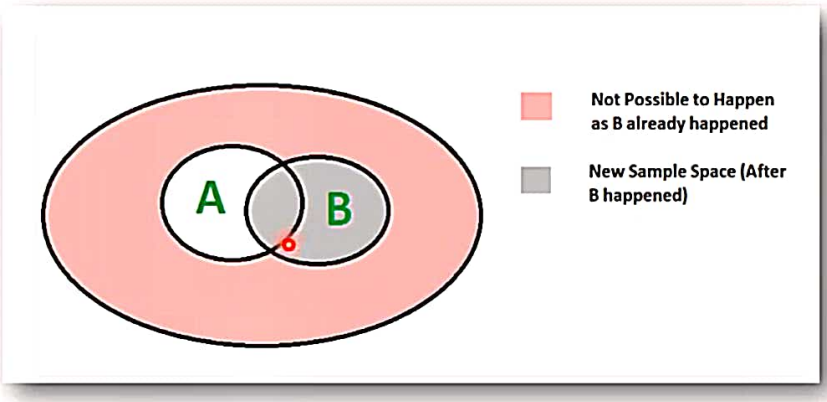
- When we toss a coin, either we get Head OR Tail, only two possible outcomes are possible (H, T).
- But if we toss two coins in the air, there could be three possibilities of events to occur, such as both the coins show heads or both show tails or one shows heads and one tail.
- i.e.(H, H), (H, T),(T, T).

Probability Formula:

$$P(A) = \frac{\text{Number of favorable outcomes to A}}{\text{Total number of outcomes}}$$

Conditional Probability

- The probability of an event occurring given that another event has already occurred is called a **conditional probability**.



Conditional Probability Formula

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Probability of A given B

Probability of A and B

Probability of B

Conditional Probability Example

- Neha took two tests.
- The probability of her passing both tests is 0.6.
- The probability of her passing the first test is 0.8.
- What is the probability of her passing the second test given that she has passed the first test?

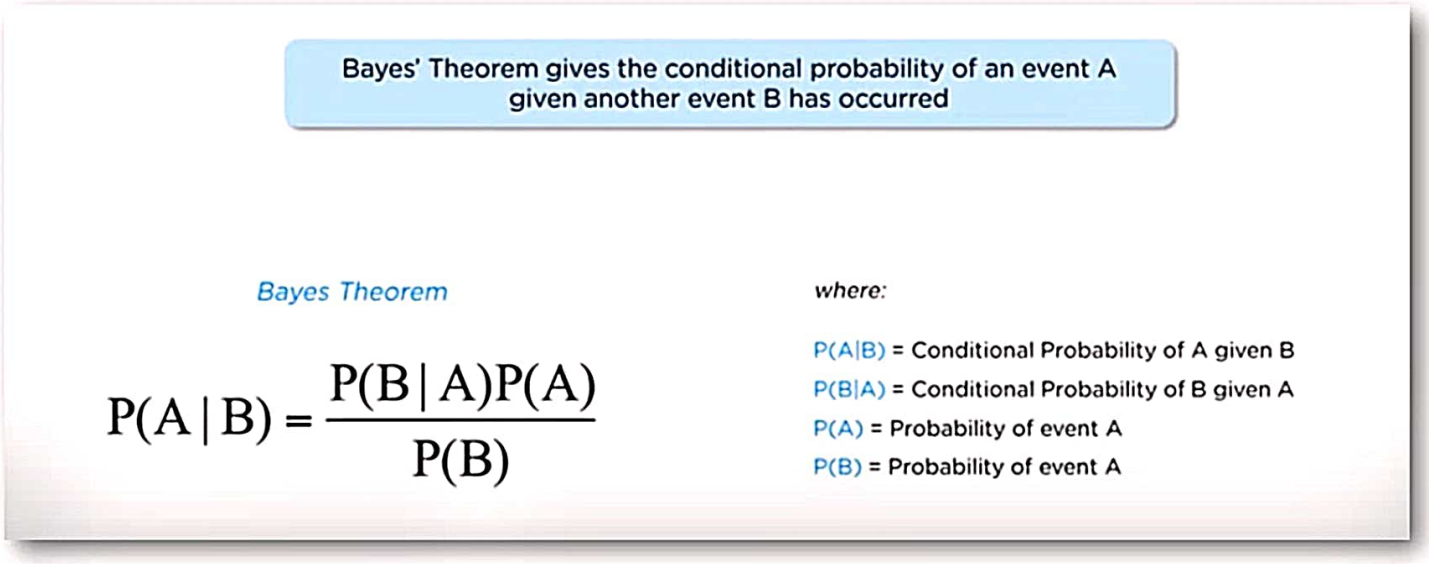
Solution:

$$P(\text{second} \mid \text{first}) = \frac{P(\text{first and second})}{P(\text{first})} = \frac{0.6}{0.8} = 0.75$$

About Bayes Theorem

- Bayes' theorem is also known as **Bayes' rule, Bayes' law**, or **Bayesian reasoning**, which determines the probability of an event with uncertain knowledge.
- Bayes' theorem was named after the British mathematician **Thomas Bayes**.
- It is a way to **calculate the value of P(B|A) with the knowledge of P(A|B).**
- Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

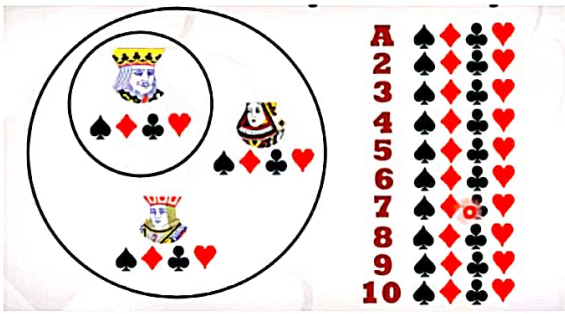
Bayes Theorem Formula



Example of Bayes Theorem

- From the deck of the cards, find the probability of the card being picked is king given that it is the face card.
- This can be represented as : **P(King|Face)**

- There are total 52 cards in a deck (n=52), from which 12 cards are face cards; king, queen and jack with club, diamond, heart and spade
- There is a set of face cards with 12 members.
- There is a subset of king cards with 4 members.
- So 4 face cards out of 12 face cards (4/12) are king cards.

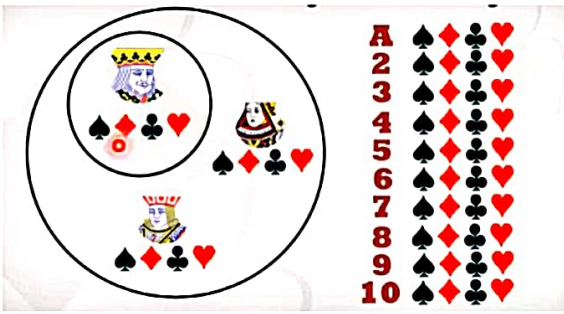


Example of Bayes Theorem

According to Bayes' theorem: $$P(King|Face) = \frac{P(Face|King)P(King)}{P(Face)}$$

- 4 members are king cards out of 52 cards: $P(King) = \frac{4}{52} = \frac{1}{13}$
- 12 members are face cards out of 52 cards: $P(Face) = \frac{12}{52} = \frac{3}{13}$
- Probability of being face given king: $P(Face|King) = 1$

Here we have prior knowledge that if the card is king, it is face card only. It is 100% sure so it is always 1.



Example of Bayes Theorem

- **Putting all in Bayes' formula:**

$$P(King|Face) = \frac{P(Face|King)P(King)}{P(Face)}$$

$$= \frac{1 * \frac{1}{13}}{\frac{3}{13}}$$

$$= \frac{1}{3}$$

➢Bayes' Theorem proof:

$$P(King|Face) = P(King \cap Face)/P(Face)$$

So from the image above

$$P(King \cap Face) = 4$$

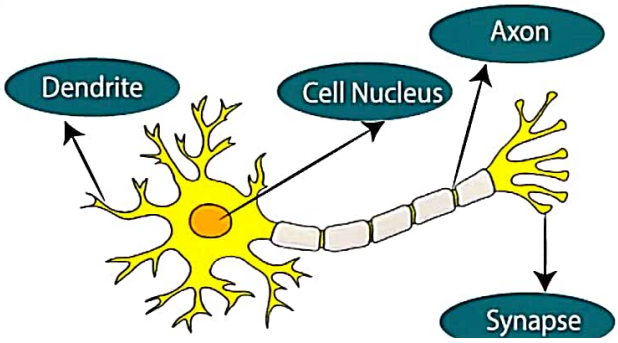
$$P(Face) = 12$$

$$P(King|Face) = 4/12 = 1/3$$

$$\therefore 4/12 = 1/3$$

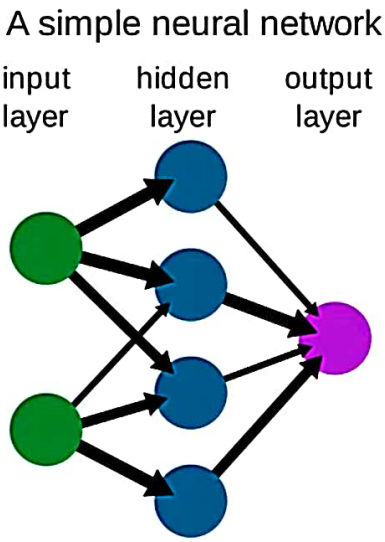
Artificial Neural Network (ANN)

- Artificial Neural Network (ANN) is inspired by the working of a human brain.
- The human brain that has neurons interconnected to one another, ANN also have neurons, that are interconnected to one another in various layers of the networks. These neurons are known as nodes.
- Neural Networks are a set of algorithms that tries to recognize the patterns, relationships, and information from the data through the process which is inspired by and works like the human brain.



Components of ANN

- A Simple Neural network consists of three components
- Input layer
- Hidden layer (middle layer)
- Output layer

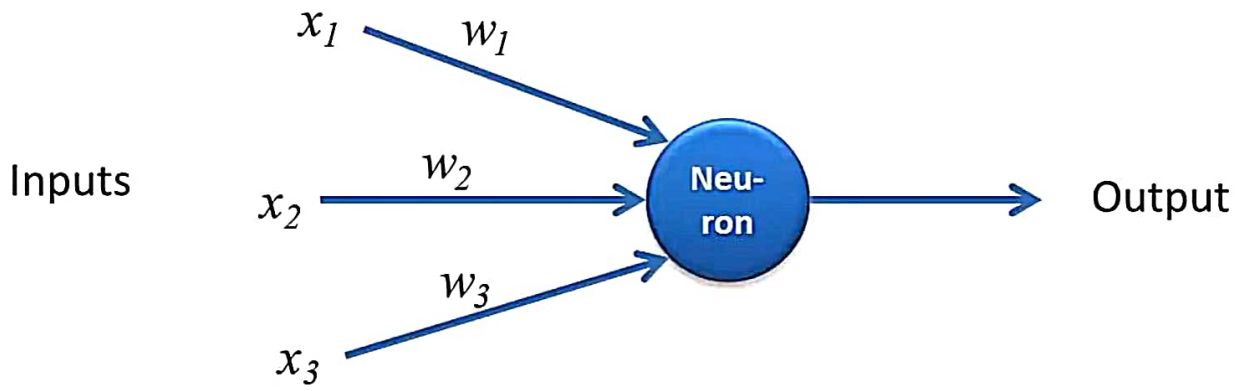


Components of ANN…

- Input Layer: Input nodes receive inputs/information from the outside world.
- Hidden Layer: Hidden layer is the set of neurons where all the computations are performed on the input data.
- There can be any number of hidden layers in a neural network. The simplest network consists of a single hidden layer.
- Output layer: The output layer is the output/conclusions derived from all the computations performed.
- There can be single or multiple nodes in the output layer.
- If we have a binary classification problem the output node is 1 but in the case of multi-class classification, the output nodes can be more than 1.

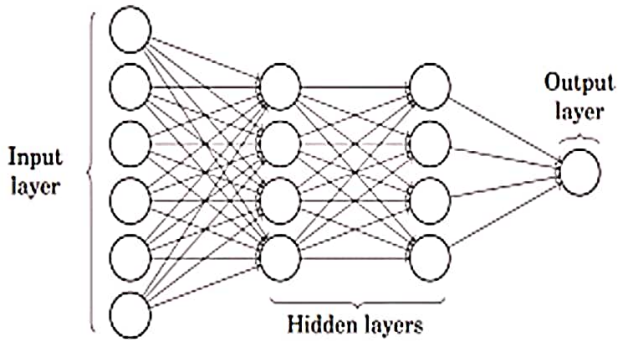
Perceptron

- Perceptron is a simple form of Neural Network and consists of a single layer where all the mathematical computations are performed.

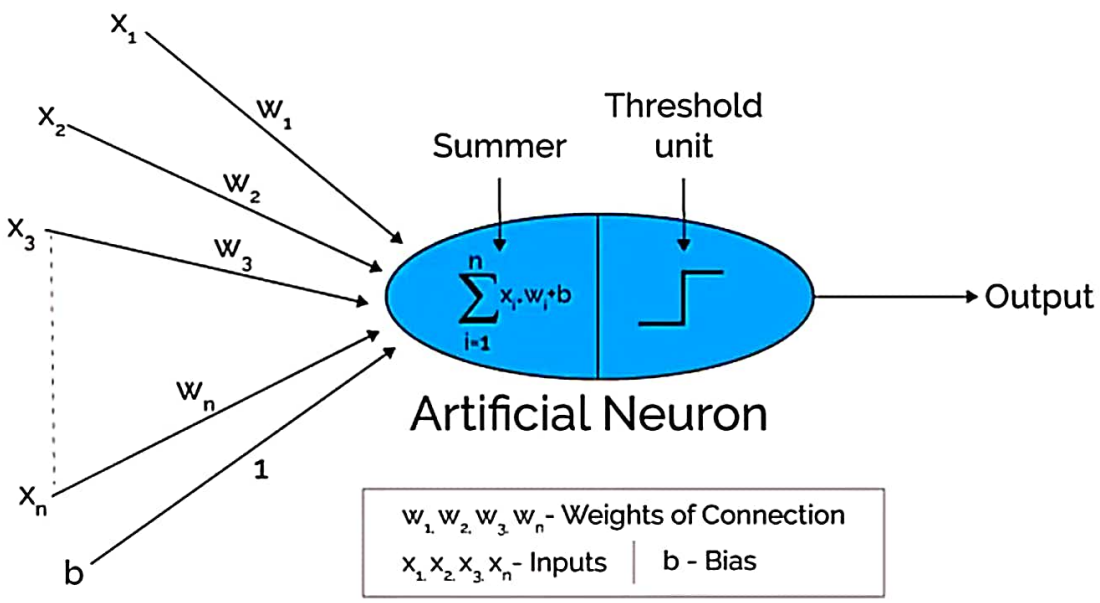


Multilayer Perceptron

- **Multilayer Perceptron** also known as **Artificial Neural Networks** consists of more than one perception which is grouped together to form a multiple layer neural network.
- An input layer, with 6 input nodes
- Hidden Layer 1, with 4 hidden nodes/4 perceptrons
- Hidden layer 2, with 4 hidden nodes
- Output layer with 1 output node

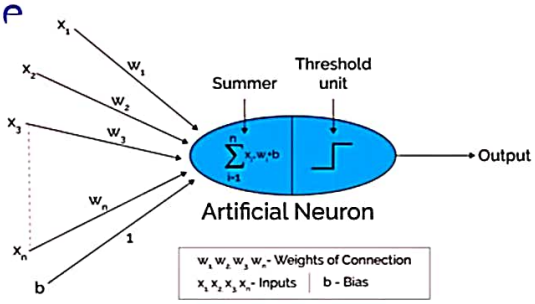


Working of the Artificial Neural Network



Computation performed in hidden layers are done in two steps which are as follows :

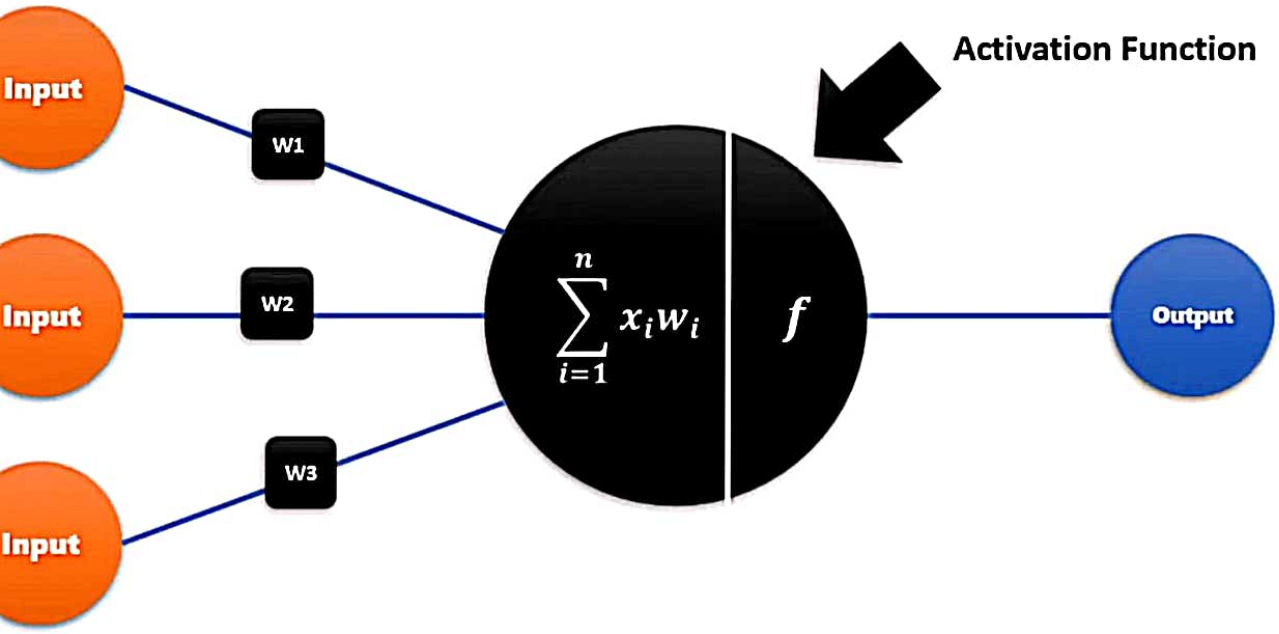
- **Step 1. All the inputs are multiplied by their weights**.
- Weight is the gradient or coefficient of each variable It shows the strength of the particular input. After assigning the weights, a bias variable is added.
- Bias is a constant that helps the model to fit in the best way possible.
- $Z_1 = W_1*x_1 + W_2*x_2 + \ldots + W_n*x_n + b$
- Where W1, W2, W3, W4, W5 are the weights assigned to the inputs x1, x2, x3, … xn, and b is the bias.



Computation performed in hidden layers are done in two steps which are as follows …

- **Step 2. Activation function is applied to the linear equation Z1.**($Z_1 = W_1*x_1 + W_2*x_2 + \ldots + W_n*x_n + b$)
- The activation function is a nonlinear transformation that is applied to the input before sending it to the next layer of neurons.

How actually Neuron Works?



Formula

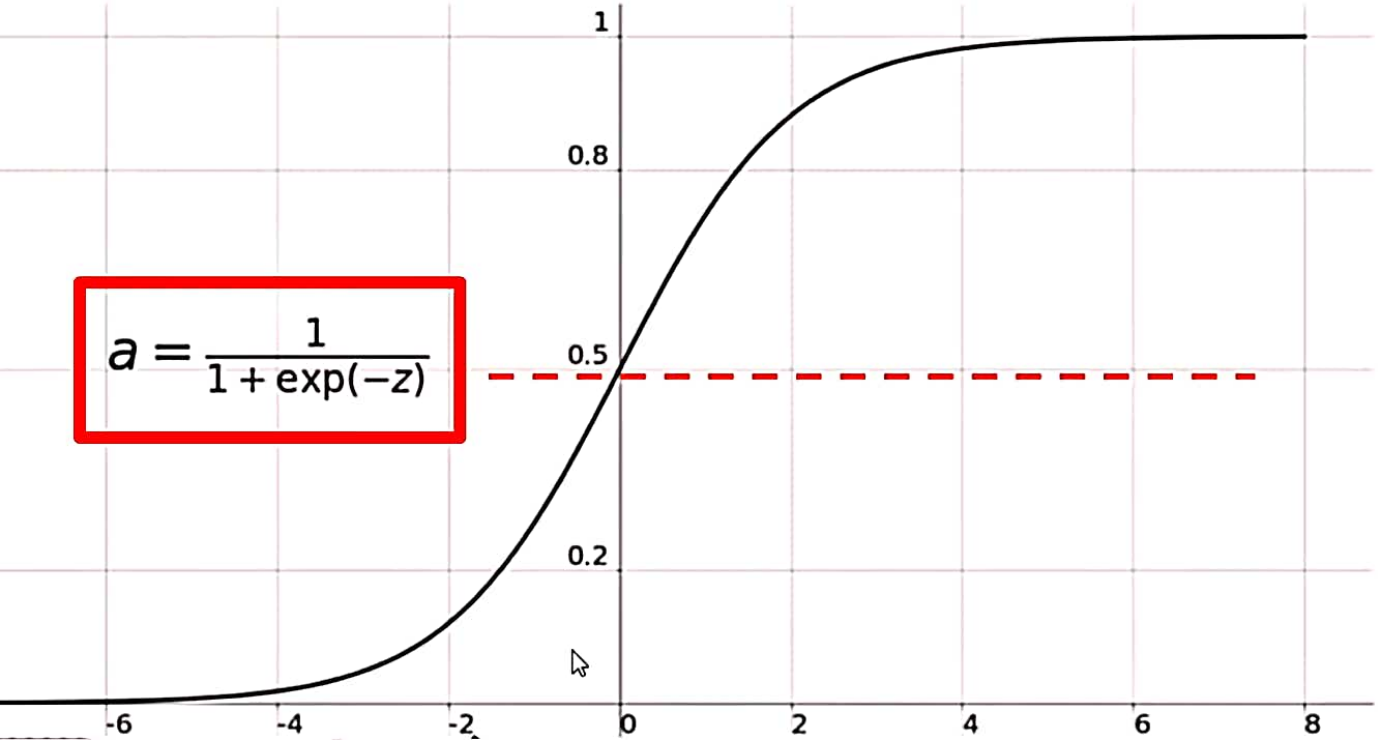
$$Activation\ Function\left(\sum_{i=1}^{n} x_i w_i + b\right)$$

Types

Linear and Non - Linear

Sigmoid Function

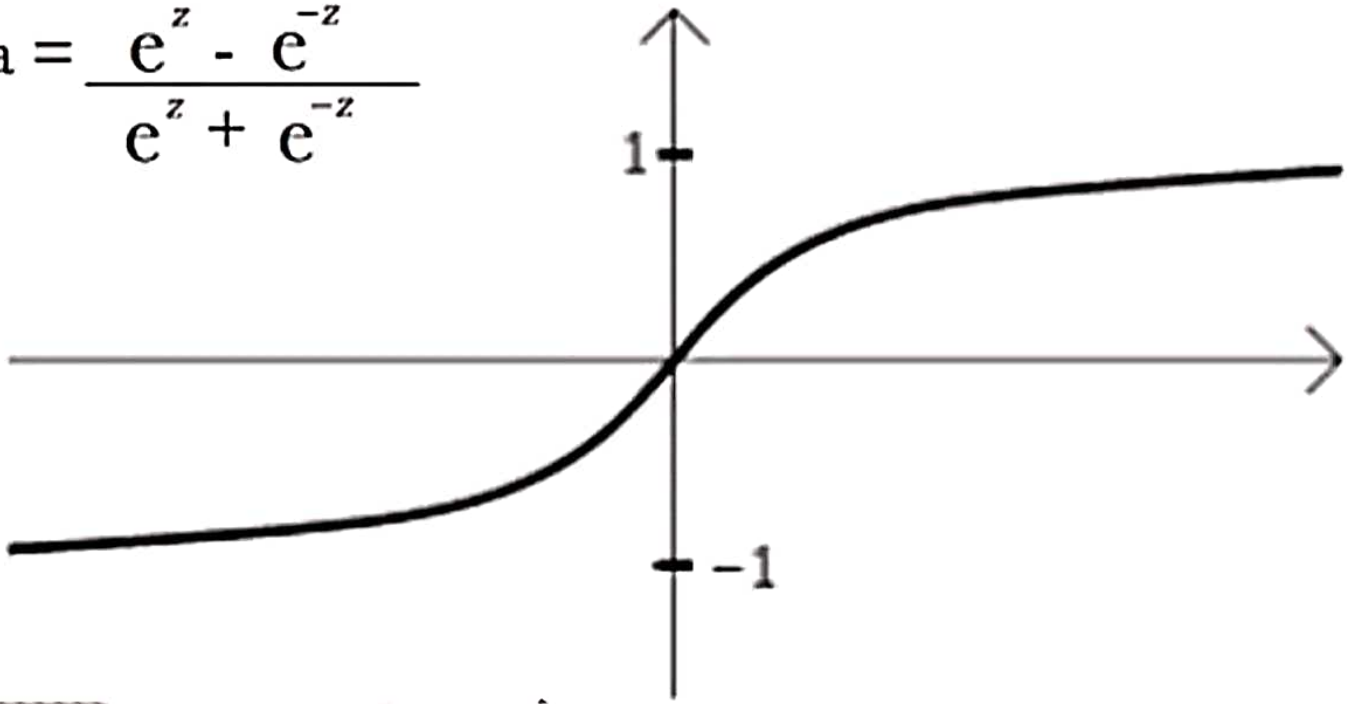
$$a = \frac{1}{1 + \exp(-z)}$$



Hyperbolic Tangent(tanh) Function

Tanh Function

$$a = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



Tan h

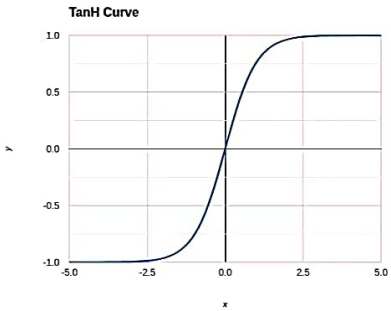
$$Hyperbolic\ Tangent\ (Tanh)\ Function\ Calculation\ for\ x = 1$$

$$Given\ the\ tanh\ function:$$

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

$$For\ x = 1:$$

$$\tanh(1) = \frac{e^1 - e^{-1}}{e^1 + e^{-1}} \approx 0.762$$



ReLU

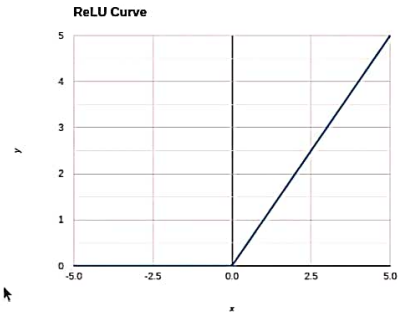
Rectified Linear Unit (ReLU) Function Calculation for $x = -3$

Given the ReLU function :

$$\text{ReLU}(x) = \max(0, x)$$

For $x = -3$ **:**

$$\text{ReLU}(-3) = \max(0, -3) = 0$$



Advantages OF ANN

- They are massively parallel .i.e. they can perform multiple tasks parallelly.
- Fault-tolerant – like biological ANNs can also survive and function in case some of its functional units stop functioning.
- Capable to learn and generalize means they keep on updating their knowledge with time, exposure and experience.
- They support Black box functioning (i.e. a system that produces results without the user being able to see or understand how it works).

Disadvantages OF ANN

- ANNs need massive amount of data to be trained.

- ANNs black box nature also turns out to be its disadvantage (the developer can't figure out how or why ANN came up to certain output).

- ANNs are computationally expensive than traditional machine learning algorithms.

- ANNs are not suitable for every type of problems as these are more complicated and the development takes longer.