UNIT 2

How do neural networks work?

The human brain is the inspiration behind neural network architecture. Human brain cells, called neurons, form a complex, highly interconnected network and send electrical signals to each other to help humans process information. Similarly, an artificial neural network is made of artificial neurons that work together to solve a problem. Artificial neurons are software modules, called nodes, and artificial neural networks are software programs or algorithms that, at their core, use computing systems to solve mathematical calculations.

Simple neural network architecture

A basic neural network has interconnected artificial neurons in three layers:

Input Layer

Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.

Hidden Layer

Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.

Output Layer

The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0. However, if we have a multi-class classification problem, the output layer might consist of more than one output node.

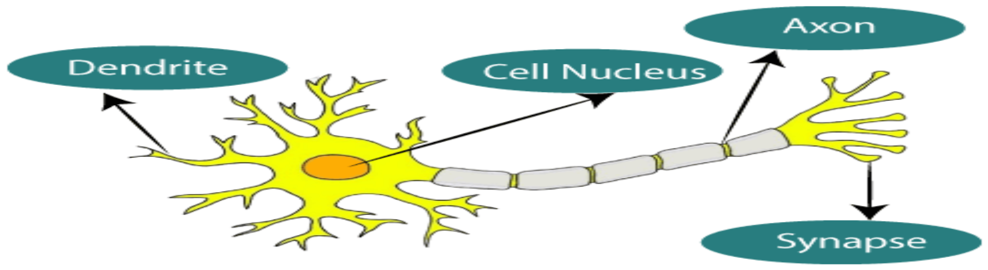
Deep neural network architecture

Deep neural networks, or deep learning networks, have several hidden layers with millions of artificial neurons linked together. A number, called weight, represents the connections between one node and another. The weight is a positive number if one node excites another, or negative if one node suppresses the other. Nodes with higher weight values have more influence on the other nodes. Theoretically, deep neural networks can map any input type to any output type. However, they also need much more training as compared to other machine learning methods. They need millions of examples of training data rather than perhaps the hundreds or thousands that a simpler network might need.

===

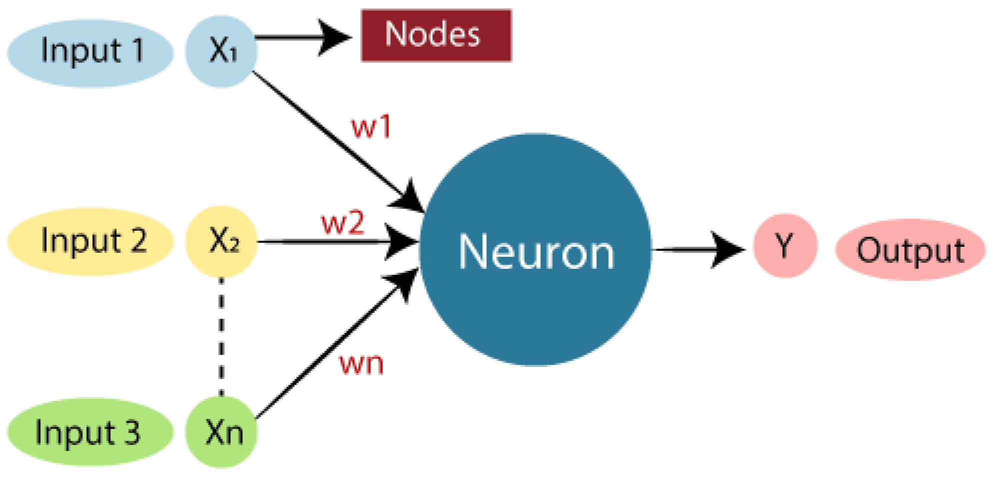
What is Artificial Neural Network?

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.

The typical Artificial Neural Network looks something like the given figure.



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

Relationship between Biological neural network and artificial neural network:

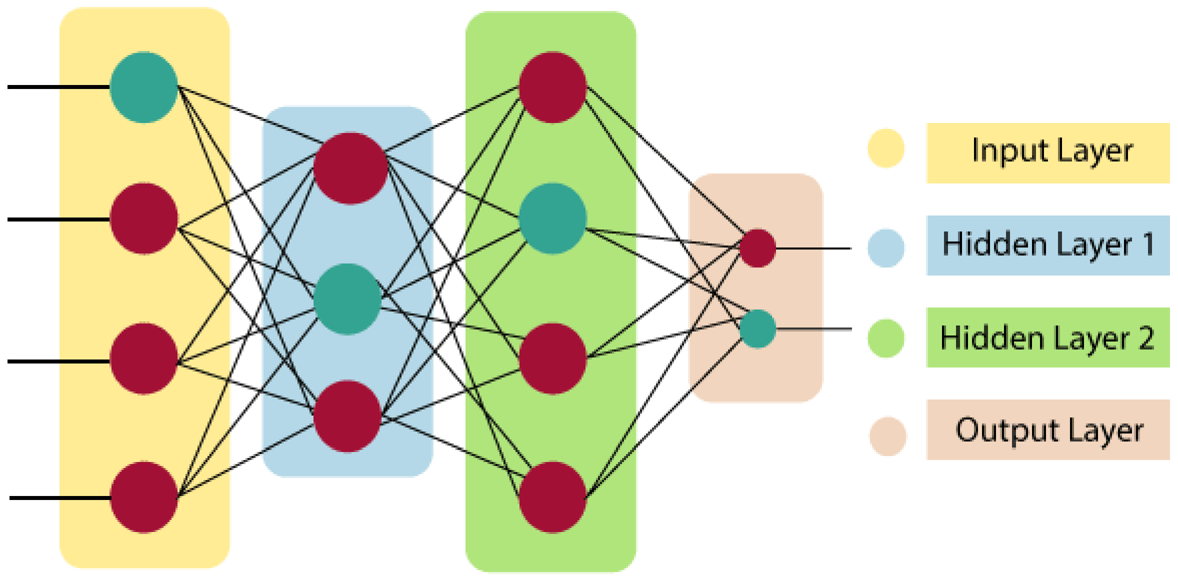
Biological Neural Network	**Artificial Neural Network**
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

The architecture of an artificial neural network:

Lets us look at various types of layers available in an artificial neural network.

Artificial Neural Network primarily consists of three layers:



Input Layer:

As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer:

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer:

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

Advantages of Artificial Neural Network (ANN)

Parallel processing capability:

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

Storing data on the entire network:

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

Capability to work with incomplete knowledge:

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

Having a memory distribution:

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

Having fault tolerance:

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network:

Assurance of proper network structure:

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

Unrecognized behavior of the network:

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

Hardware dependence:

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

Difficulty of showing the issue to the network:

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

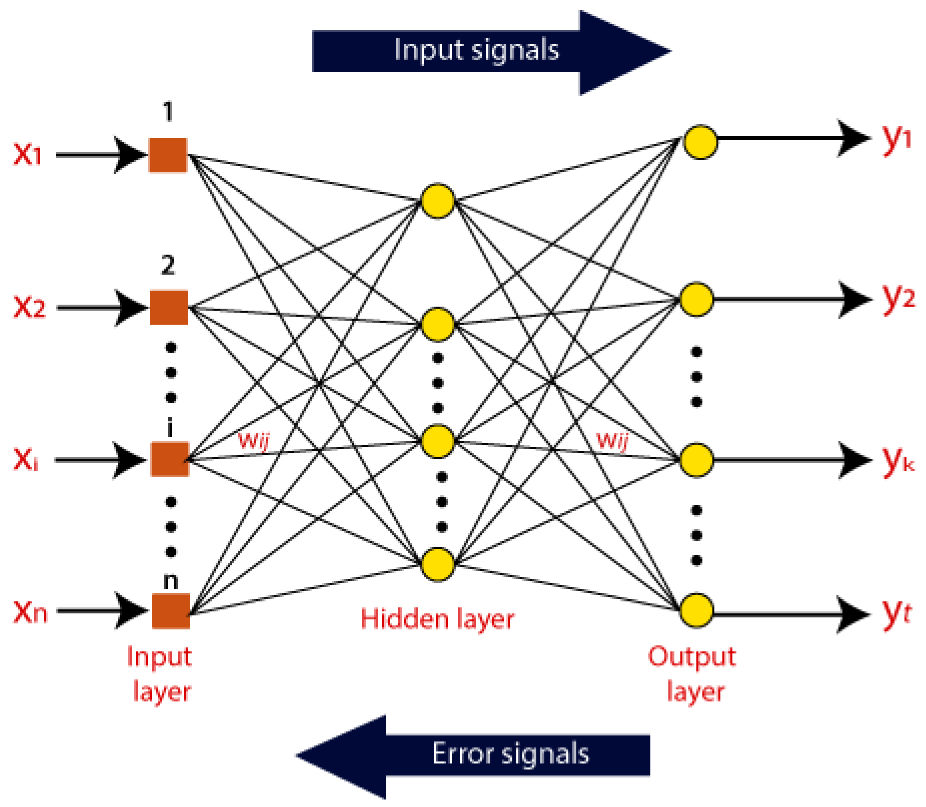
The duration of the network is unknown:

The network is reduced to a specific value of the error, and this value does not give us optimum results.

==
==================

How do artificial neural networks work?

Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:

Binary:

In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up. If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.

Sigmoidal Hyperbolic:

The Sigmoidal Hyperbola function is generally seen as an "**S**" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input. The function is defined as:

F(x) = (1/1 + exp(-????x))

Where ???? is considered the Steepness parameter.

==

Backpropagation

Introduction

The Backpropagation neural network is a multilayered, feedforward neural network and is by far the most extensively used[6]. It is also considered one of the simplest and most general methods used for supervised training of multilayered neural networks[6]. Backpropagation works by approximating the non-linear relationship between the input and the output by adjusting the weight values internally. It can further be generalized for the input that is not included in the training patterns (predictive abilities).

Generally, the Backpropagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of the person.

The following figure shows the topology of the Backpropagation neural network that includes and input layer, one hidden layer and an output layer. It should be noted that Backpropagation neural networks can have more than one hidden layer.

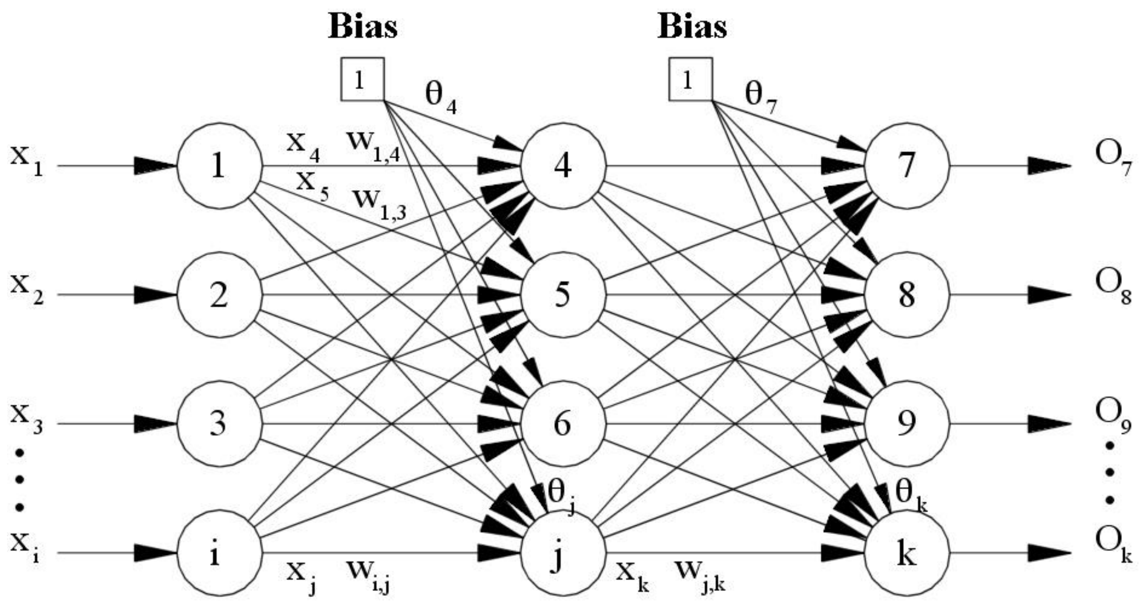


Figure 5 Backpropagation Neural Network with one hidden layer[6]

Theory

The operations of the Backpropagation neural networks can be divided into two steps: feedforward and Backpropagation.

In the feedforward step, an input pattern is applied to the input layer and its effect propagates, layer by layer, through the network until an output is produced. The network's actual output value is then compared to the expected output, and an error signal is computed for each of the output nodes. Since all the hidden nodes have, to some degree, contributed to the errors evident in the output layer, the output error signals are transmitted backwards from the output layer to each node in the hidden layer that immediately contributed to the output layer. This process is then repeated, layer by layer, until each node in the network has received an error signal that describes its relative contribution to the overall error.

Once the error signal for each node has been determined, the errors are then used by the nodes to update the values for each connection weights until the network converges to a state that allows all the training patterns to be encoded. The Backpropagation algorithm looks for the minimum value of the **error function** in weight space using a technique called the delta rule or **gradient descent**[2]. The weights that minimize the error function is then considered to be a solution to the learning problem.

There are also issues regarding generalizing a neural network. Issues to consider are problems associated with under-training and over-training data. Under-training can occur when the neural network is not complex enough to detect a pattern in a complicated data set. This is usually the result of networks with so few hidden nodes that it cannot accurately represent the solution, therefore under-fitting the data (Figure 6)[1].

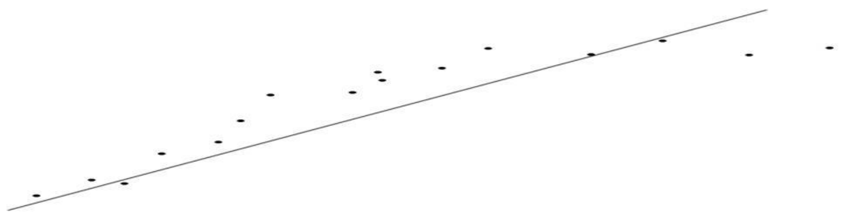


Figure 6 Under-fitting data

On the other hand, over-training can result in a network that is too complex, resulting in predictions that are far beyond the range of the training data. Networks with too many hidden nodes will tend to over-fit the solution (Figure 7)[1].

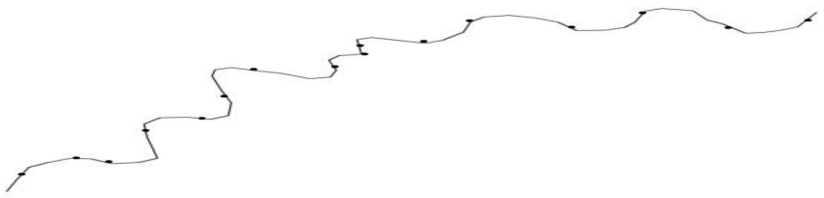


Figure 7 Over-fitting data

The aim is to create a neural network with the "right" number of hidden nodes that will lead to a good solution to the problem (Figure 8)[1].

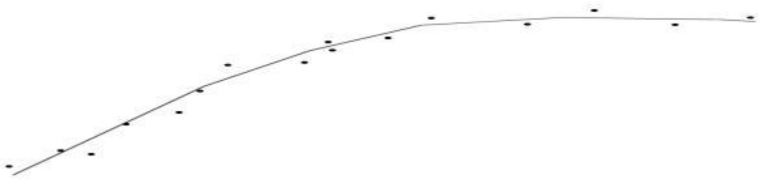


Figure 8 Good fit of the data

Algorithm

Using **Figure 3**, the following describes the learning algorithm and the equations used to train a neural network. For an extensive description on the derivation of the equations used.

Feedforward

When a specified training pattern is fed to the input layer, the weighted sum of the input to the j^{th} node in the hidden layer is given by

$$Net_j = \sum w_{i,j} x_j + \theta_j \qquad (1)$$

Equation (1) is used to calculate the aggregate input to the neuron. The θ_j term is the weighted value from a **bias** node that always has an output value of 1. The bias node is considered a "pseudo input" to each neuron in the hidden layer and the output layer, and is used to overcome the problems associated with situations where the values of an input pattern are zero. If any input pattern has zero values, the neural network could not be trained without a bias node.

To decide whether a neuron should fire, the "Net" term, also known as the action potential, is passed onto an appropriate activation function. The resulting value from the activation function determines the neuron's output, and becomes the input value for the neurons in the next layer connected to it..

Since one of the requirements for the Backpropagation algorithm is that the activation function is differentiable, a typical activation function used is the Sigmoid equation (refer to Figure 4):

$$O_j = x_k = \frac{1}{1 + e^{-Net_j}} \qquad (2)$$

It should be noted that many other types of functions can, and are, used:- hyperbolic tan being another popular choice.

Similarly, equations (1) and (2) are used to determine the output value for node k in the output layer.

Error Calculations and Weight Adjustments - Backpropagation [10]
Output Layer
If the actual activation value of the output node, k, is O_k, and the expected target output for node k is t_k, the difference between the actual output and the expected output is given by:

$$\Delta_k = t_k - O_k \qquad (3)$$

The error signal for node k in the output layer can be calculated as

$$\delta_k = \Delta_k O_k (1 - O_k)$$

or

$$\delta_k = (t_k - O_k) O_k (1 - O_k) \qquad (4)$$

where the $O_k(1-O_k)$ term is the derivative of the Sigmoid function.

With the delta rule, the change in the weight connecting input node j and output node k is proportional to the error at node k multiplied by the activation of node j.

The formulas used to modify the weight, $w_{j,k}$, between the output node, k, and the node, j is:

$$\Delta w_{j,k} = l_r \delta_k x_k \qquad (5)$$

$$w_{j,k} = w_{j,k} + \Delta w_{j,k} \qquad (6)$$

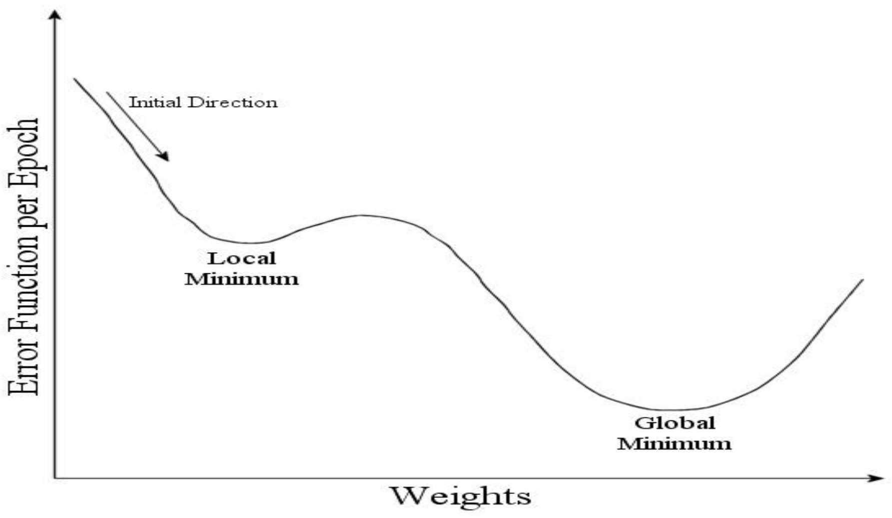
where $\Delta w_{j,k}$ is the change in the weight between nodes j and k, l_r is the **learning rate**. The learning rate is a relatively small constant that indicates the relative change in weights. If the learning rate is too low, the network will learn very slowly, and if the learning rate is too high, the network may oscillate around minimum point (refer to Figure 6), overshooting the lowest point with each weight adjustment, but never actually reaching it. Usually the learning rate is very small, with 0.01 not an uncommon number. Some modifications to the Backpropagation algorithm allows the learning rate to decrease from a large value during the learning process. This has many advantages. Since it is assumed that the network initiates at a state that is distant from the optimal set of weights, training will initially be rapid. As learning progresses, the learning rate decreases as it approaches the optimal point in the minima. Slowing the learning process near the optimal point encourages the network to converge to a solution while reducing the possibility of overshooting. If, however, the learning process initiates close to the optimal point, the system may initially oscillate, but this effect is reduced with time as the learning rate decreases.

It should also be noted that, in equation (5), the x_k variable is the input value to the node k, and is the same value as the output from node j.

To improve the process of updating the weights, a modification to equation (5) is made:

$$\Delta w_{j,k}^{n} = l_r \delta_k x_k + \Delta w_{j,k}^{(n-1)} \mu \qquad (7)$$

Here the weight update during the nth iteration is determined by including a **momentum** term (μ), which is multiplied to the (n-1)th iteration of the $\Delta w_{j,k}$. The introduction of the momentum term is used to accelerate the learning process by "encouraging" the weight changes to continue in the same direction with larger steps. Furthermore, the momentum term prevents the learning process from settling in a local minimum. by "over stepping" the small "hill". Typically, the momentum term has a value between 0 and 1.



It should be noted that no matter what modifications are made to the Backpropagation algorithm, such as including the momentum term, there are no guarantees that the network will not settle in a local minimum.

Figure 9 Global and Local Minima of Error Function[3]

Hidden Layer
The error signal for node j in the hidden layer can be calculated as

$$\delta_k = (t_k - O_k)\, O_k \sum (w_{j,k}\, \delta_k) \tag{8}$$

where the Sum term adds the weighted error signal for all nodes, k, in the output layer.

As before, the formula to adjust the weight, $w_{i,j}$, between the input node, i, and the node, j is:

$$\Delta w_{i,j}^{n} = l_r\, \delta_j\, x_j + \Delta w_{i,j}^{(n-1)}\, \mu \tag{9}$$

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} \tag{10}$$

Global Error
Finally, Backpropagation is derived by assuming that it is desirable to minimize the error on the output nodes over all the patterns presented to the neural network. The following equation is used to calculate the **error function**, E, for all patterns

$$E = \tfrac{1}{2} \sum \left(\sum (t_k - O_k)^2 \right) \tag{11}$$

Ideally, the error function should have a value of zero when the neural network has been correctly trained. This, however, is numerically unrealistic.

================================ ==

Types of Perceptron Models

Based on the layers, Perceptron models are divided into two types. These are as follows:

1. Single-layer Perceptron Model
2. Multi-layer Perceptron model

Single Layer Perceptron Model:

This is one of the easiest Artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

In a single layer perceptron model, its algorithms do not contain recorded data, so it begins with inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a pre-determined value, the model gets activated and shows the output value as +1.

If the outcome is same as pre-determined or threshold value, then the performance of this model is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight inputs values are fed into the model. Hence, to find desired output and minimize errors, some changes should be necessary for the weights input.

"Single-layer perceptron can learn only linearly separable patterns."

https://media.geeksforgeeks.org/wp-content/uploads/20221219111343/Single-Layer-Perceptron.png

Multi-Layered Perceptron Model:

Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.

The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- **Forward Stage:** Activation functions start from the input layer in the forward stage and terminate on the output layer.

- **Backward Stage:** In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model has considered as multiple artificial neural networks having various layers in which activation function does not remain linear, similar to a single layer perceptron model. Instead of linear, activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, NOR.

Advantages of Multi-Layer Perceptron:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training

==

Genetic Algorithm in Machine Learning

A genetic algorithm is an adaptive heuristic search algorithm inspired by "Darwin's theory of evolution in Nature." It is used to solve optimization problems in machine learning. It is one of the important algorithms as it helps solve complex problems that would take a long time to solve.

Genetic Algorithms are being widely used in different real-world applications, for example, **Designing electronic circuits, code-breaking, image processing, and artificial creativity.**

https://cdn.analyticsvidhya.com/wp-content/uploads/2020/02/ANN-Graph.gif

What is a Genetic Algorithm?

Before understanding the Genetic algorithm, let's first understand basic terminologies to better understand this algorithm:

- **Population:** Population is the subset of all possible or probable solutions, which can solve the given problem.
- **Chromosomes:** A chromosome is one of the solutions in the population for the given problem, and the collection of gene generate a chromosome.
- **Gene:** A chromosome is divided into a different gene, or it is an element of the chromosome.
- **Allele:** Allele is the value provided to the gene within a particular chromosome.
- **Fitness Function:** The fitness function is used to determine the individual's fitness level in the population. It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function.
- **Genetic Operators:** In a genetic algorithm, the best individual mate to regenerate offspring better than parents. Here genetic operators play a role in changing the genetic composition of the next generation.
- **Selection:** After calculating the fitness of every existent in the population, a selection process is used to determine which of the individualities in the population will get to reproduce and produce the seed that will form the coming generation.

Types of selection styles available

- **Roulette wheel selection**
- **Event selection**
- **Rank- grounded selection**

So, now we can define a genetic algorithm as a heuristic search algorithm to solve optimization problems. It is a subset of evolutionary algorithms, which is used in computing. A genetic algorithm uses genetic and natural selection concepts to solve optimization problems.

How Genetic Algorithm Work?

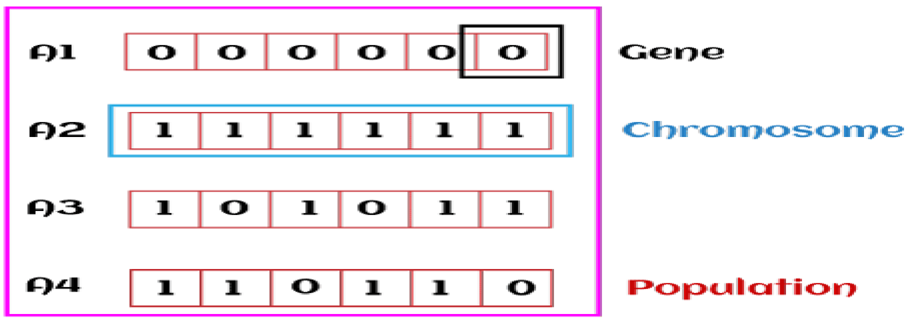
The genetic algorithm works on the evolutionary generational cycle to generate high-quality solutions. These algorithms use different operations that either enhance or replace the population to give an improved fit solution.

It basically involves five phases to solve the complex optimization problems, which are given as below:

- **Initialization**
- **Fitness Assignment**
- **Selection**
- **Reproduction**
- **Termination**

1. Initialization

The process of a genetic algorithm starts by generating the set of individuals, which is called population. Here each individual is the solution for the given problem. An individual contains or is characterized by a set of parameters called Genes. Genes are combined into a string and generate chromosomes, which is the solution to the problem. One of the most popular techniques for initialization is the use of random binary strings.



2. Fitness Assignment

Fitness function is used to determine how fit an individual is? It means the ability of an individual to compete with other individuals. In every iteration, individuals are evaluated based on their fitness function. The fitness function provides a fitness score to each individual. This score further determines the probability of being selected for reproduction. The high the fitness score, the more chances of getting selected for reproduction.

3. Selection

The selection phase involves the selection of individuals for the reproduction of offspring. All the selected individuals are then arranged in a pair of two to increase reproduction. Then these individuals transfer their genes to the next generation.

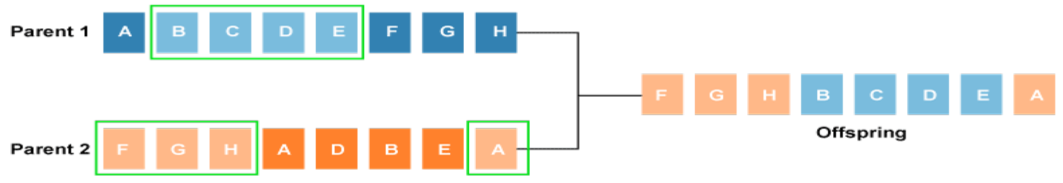
There are three types of Selection methods available, which are:

- Roulette wheel selection
- Tournament selection
- Rank-based selection

4. Reproduction

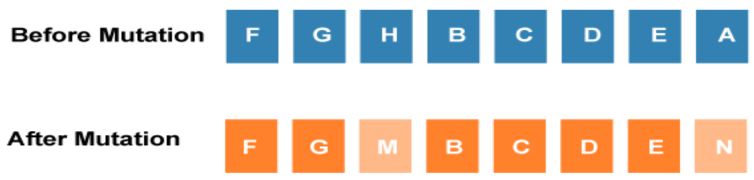
After the selection process, the creation of a child occurs in the reproduction step. In this step, the genetic algorithm uses two variation operators that are applied to the parent population. The two operators involved in the reproduction phase are given below:

- **Crossover:** The crossover plays a most significant role in the reproduction phase of the genetic algorithm. In this process, a crossover point is selected at random within the genes. Then the crossover operator swaps genetic information of two parents from the current generation to produce a new individual representing the offspring.

  The genes of parents are exchanged among themselves until the crossover point is met. These newly generated offspring are added to the population. This process is also called or crossover. Types of crossover styles available:
  - One point crossover
  - Two-point crossover
  - Livery crossover
  - Inheritable Algorithms crossover
- **Mutation**
  The mutation operator inserts random genes in the offspring (new child) to maintain the diversity in the population. It can be done by flipping some bits in the chromosomes.
  Mutation helps in solving the issue of premature convergence and enhances diversification. The below image shows the mutation process:
  Types of mutation styles available,
  - **Flip bit mutation**

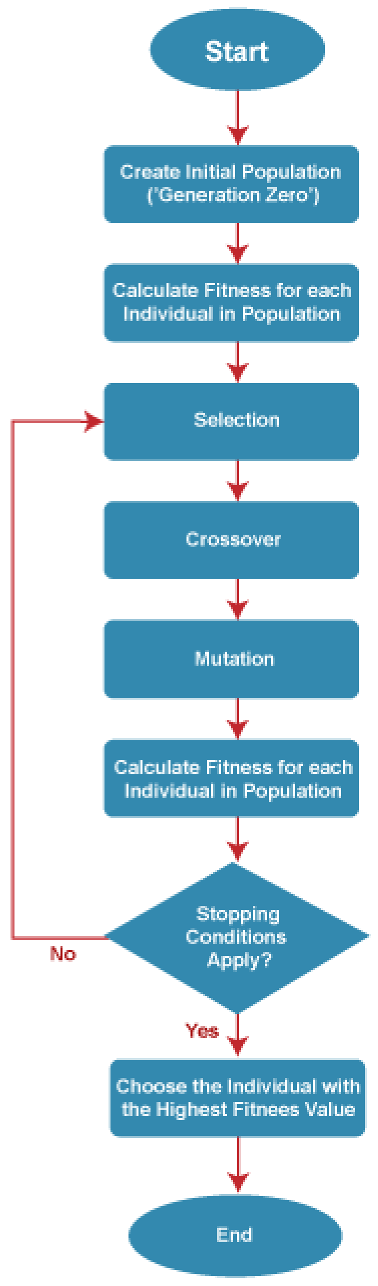
- **Gaussian mutation**
- **Exchange/Swap mutation**



5. Termination

After the reproduction phase, a stopping criterion is applied as a base for termination. The algorithm terminates after the threshold fitness solution is reached. It will identify the final solution as the best solution in the population.

General Workflow of a Simple Genetic Algorithm


Start
Create Initial Population ('Generation Zero')
Calculate Fitness for each Individual in Population
Selection
Crossover
Mutation
Calculate Fitness for each Individual in Population
Stopping Conditions Apply?
No
Yes
Choose the Individual with the Highest Fitnees Value
End

Advantages of Genetic Algorithm

- The parallel capabilities of genetic algorithms are best.
- It helps in optimizing various problems such as discrete functions, multi-objective problems, and continuous functions.
- It provides a solution for a problem that improves over time.
- A genetic algorithm does not need derivative information.

Limitations of Genetic Algorithms

- Genetic algorithms are not efficient algorithms for solving simple problems.
- It does not guarantee the quality of the final solution to a problem.
- Repetitive calculation of fitness values may generate some computational challenges.

What Are the Disadvantages Of Genetic Algorithm

1. ***Genetic algorithms are often criticized for being too slow.***
There are several disadvantages of using genetic algorithms. One is that they can be quite slow, particularly when compared to other optimization methods. Another disadvantage is that they can be difficult to understand and interpret, making it hard to know why a particular solution was found. Finally, they can be sensitive to the initial conditions and may not find the global optimum solution.

2. ***They can be expensive to implement.***
There are a few disadvantages to using a genetic algorithm. They can be expensive to implement and can be time consuming to run. Additionally, they can be difficult to interpret the results of.

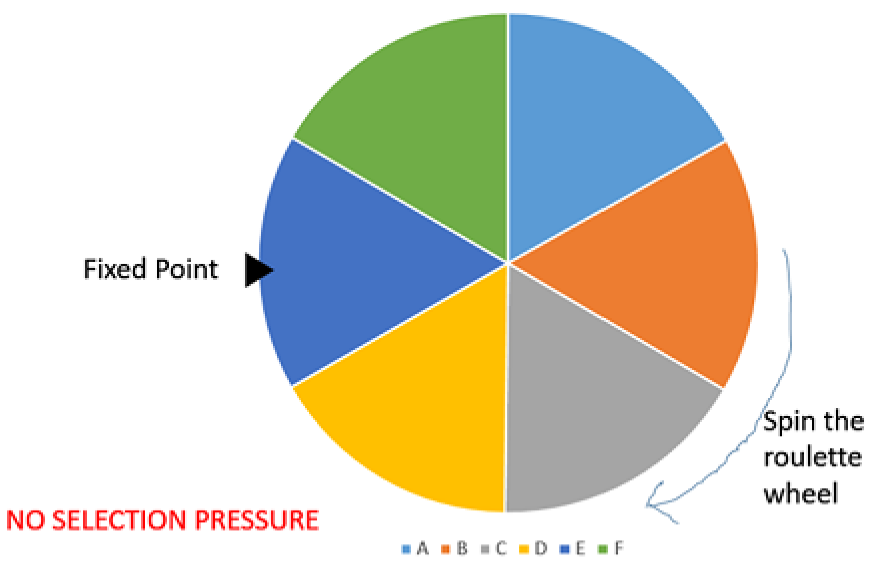
3. ***They can be difficult to understand.***
There are a few disadvantages to using genetic algorithms. They can be difficult to understand and can be time-consuming to run. Additionally, they may not always find the optimal solution to a problem.

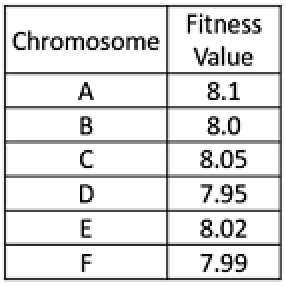
4. ***They can be difficult to debug.***
There are a few disadvantages to using genetic algorithms. They can be difficult to debug and can be computationally expensive. Additionally, they can be sensitive to the initial conditions and can sometimes converge to local optima.

5. ***They can be difficult to optimize.***
There are a few disadvantages to using genetic algorithms. They can be difficult to optimize and can be time-consuming to run. Additionally, they may not find the global optimum and can be stuck in local optima.

Rank Selection

Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run). This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following image and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent. This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.





Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

In this, we remove the concept of a fitness value while selecting a parent. However, every individual in the population is ranked according to their fitness. The selection of the parents depends on the rank of each individual and not the fitness. The higher ranked individuals are preferred more than the lower ranked ones.

Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

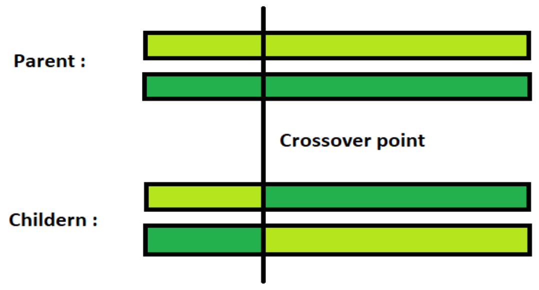
1. Rank selection first ranks the population and then every chromosome receives fitness from this ranking.
2. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).
3. After this all the chromosomes have a chance to be selected.
4. Rank-based selection schemes can avoid premature convergence.
5. But can be computationally expensive because it sorts the populations based on fitness value.
6. But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones.

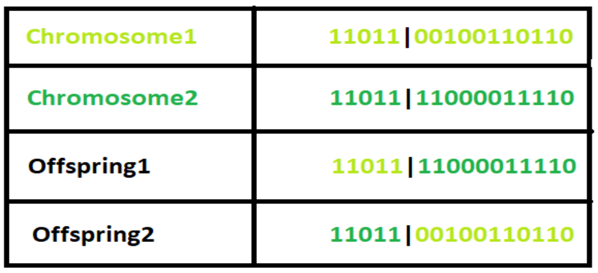
Crossover in Genetic Algorithm

Crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. Crossover is sexual reproduction. Two strings are picked from the mating pool at random to crossover in order to produce superior offspring. The method chosen depends on the Encoding Method.

Different types of crossover :

Single Point Crossover: A crossover point on the parent organism string is selected. All data beyond that point in the organism string is swapped between the two parent organisms. Strings are characterized by Positional Bias.

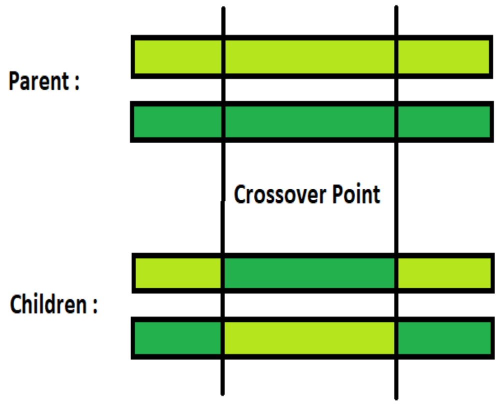




Chromosome1	11011|00100110110
Chromosome2	11011|11000011110
Offspring1	11011|11000011110
Offspring2	11011|00100110110

Single Point Crossover

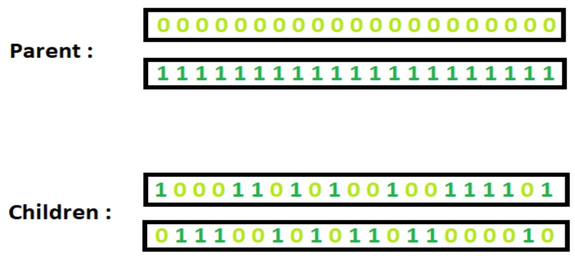
Two-Point Crossover : This is a specific case of a N-point Crossover technique. Two random points are chosen on the individual chromosomes (strings) and the genetic material is exchanged at these points.



Chromosome1	11011|00100|110110
Chromosome2	10101|11000|011110
Offspring1	11011|11000|110110
Offspring2	10101|00100|011110

Two Point Crossover

Uniform Crossover: Each gene (bit) is selected randomly from one of the corresponding genes of the parent chromosomes. Use tossing of a coin as an example technique.



Uniform Crossover

The crossover between two good solutions may not always yield a better or as good a solution. Since parents are good, the probability of the child being good is high. If offspring is not good (poor solution), it will be removed in the next iteration during “Selection”.

Problems with Crossover:

- Depending on coding, simple crossovers can have a high chance to produce illegal offspring.
  E.g. in TSP with simple binary or path coding, most offspring will be illegal because not all cities will be in the offspring and some cities will be there more than once.
- Uniform crossover can often be modified to avoid this problem
  E.g. in TSP with simple path coding:
  Where the mask is 1, copy cities from one parent
  Where the mask is 0, choose the remaining cities in the order of the other parent.

Important questions

1. What is ANN?
2. What do you mean by Perceptron?
3. What do you mean by Cost Function?
4. What do you mean by Backpropagation?
5. What are the types of Perceptron?
6. What is the use of Loss Function?

7. What is the role of the Activation functions in Neural Networks?
8. What do you mean by Hyperparameters?
9. Define the term ‘Data Normalization’.
10. Explain the different types of Gradient Descent in detail.
11. Differentiate between Forward & Backward Propagation.
12. Define ‘Genetic Algorithm’.
13. Explain about Mutation in Genetic Algorithm.
14. Explain about Variation operators in Genetic Algorithm.
15. List down the names of some popular Activation Functions used in Neural Networks.

16. How do Neural Networks Work?
17. What are the disadvantages in Genetic Algorithm?
18. Write short notes on Perceptron.
19. Give a brief note on Single Layer Perceptron.
20. Give a brief note on Multi-Layer Perceptron.
21. Discuss about Back Propagation Algorithm with example.
22. Explain the process of data representation
23. Write short notes on different layers of Neural Network.
24. How to perform Rank based selection in Genetic Algorithm.
25. Discuss about optimizer.

26. Compare Single-point &Two-point Cross over.
27. How to initialize weights & biases in neural network.
28. Explain temporal data representation.
29. Explain in detail about Single layer & multi-layer perceptron.
30. Describe in detail about ANN.