Unit-5

Association Rules Learning

Association Rule Learning

- Association rule mining finds interesting associations and relationships among large sets of data items.
- **Market Based Analysis is one of the key techniques used by large relations to show associations between items**. It allows retailers to identify relationships between the items that people buy together frequently.

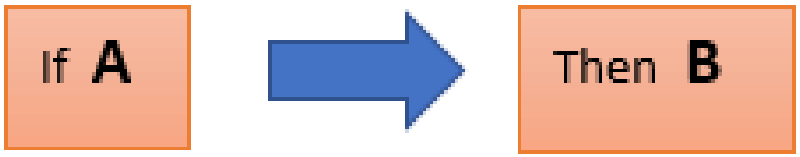
https://www.geeksforgeeks.org/association-rule/

Applications of Association Rule Learning

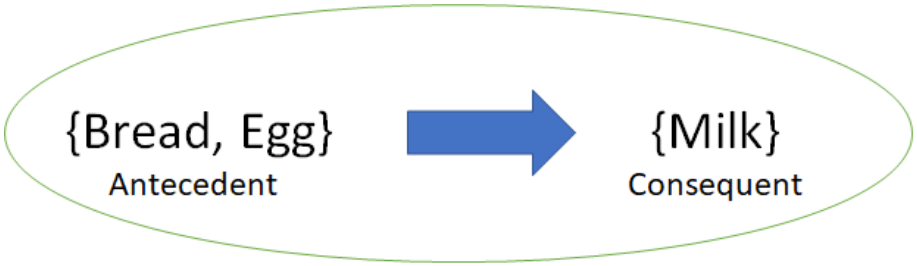
- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

https://www.javatpoint.com/association-rule-learning

How does Association Rule Learning work?



Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly.



Itemset = {Bread, Egg, Milk}

https://www.javatpoint.com/association-rule-learning

Support & Confidence

$$Support(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

Associate Rule Mining Algorithms

- Naïve Algorithm
- Apriori Algorithm
- Eclat Algorithm
- F-P Growth Algorithm

Naïve Algorithm

Table 2.2 Transactions for Example 2.1

Transaction ID	*Items*
100	Bread, Cheese
200	Bread, Cheese, Juice
300	Bread, Milk
400	Cheese, Juice, Milk

Table 2.3 The list of all itemsets and their frequencies

Itemsets	*Frequency*
Bread	3
Cheese	3
Juice	2
Milk	2
(Bread, Cheese)	2
(Bread, Juice)	1
(Bread, Milk)	1
(Cheese, Juice)	2
(Cheese, Milk)	1
(Juice, Milk)	1
(Bread, Cheese, Juice)	1
(Bread, Cheese, Milk)	0
(Bread, Juice, Milk)	0
(Cheese, Juice, Milk)	1
(Bread, Cheese, Juice, Milk)	0

Given the required minimum support of 50%, we find the itemsets that occur in at least two transactions. Such itemsets are called *frequent*. The list of frequencies shows that all four items (Bread, Cheese, Juice and Milk) are frequent. The frequency goes down as we look at 2-itemsets (only two out of a possible six are frequent), 3-itemsets (none are frequent) and 4-itemsets (again, none frequent).

The frequent itemsets are given in Table 2.4.

Table 2.4 The set of all frequent itemsets

Itemsets	*Frequency*
Bread	3
Cheese	3
Juice	2
Milk	2
Bread, Cheese	2
Cheese, Juice	2

We can now proceed to determine if the two 2-itemsets (Bread, Cheese) and (Cheese, Juice) lead to association rules with required confidence of 75%. Every 2-itemset (A, B) can lead to two rules $A \rightarrow B$ and $B \rightarrow A$ if both satisfy the required confidence. As defined earlier, confidence of $A \rightarrow B$ is given by the support for A and B together divided by the support for A.

We therefore have four possible rules and their confidence as follows:

Bread $\rightarrow$ Cheese with confidence of 2/3 = 67%
Cheese $\rightarrow$ Bread with confidence of 2/3 = 67%
Cheese $\rightarrow$ Juice with confidence of 2/3 = 67%
Juice $\rightarrow$ Cheese with confidence of 100%

Exercise for Naïve Algorithm

Transaction ID	*Items*
100	Bread, Cornflakes
101	Bread, Cornflakes, Jam
102	Bread, Milk
103	Cornflakes, Jam, Milk

Itemsets	*Frequency*
Bread	**3**
Cornflakes	**3**
Jam	**2**
Milk	**2**
(Bread, Cornflakes)	**2**
(Bread, Jam)	1
(Bread, Milk)	1
(Cornflakes, Jam)	**2**
(Cornflakes, Milk)	1
(Jam, Milk)	1
(Bread, Cornflakes, Jam)	1
(Bread, Cornflakes, Milk)	0
(Bread, Jam, Milk)	0
(Cornflakes, Jam, Milk)	1
(Bread, Cornflakes, Jam, Milk)	0

Itemsets	*Frequency*
Bread	3
Cornflakes	3
Jam	2
Milk	2
(Bread, Cornflakes)	2
(Cornflakes, Jam)	2

Confidence = Support of (Cornflakes, Bread) / Support of (Cornflakes) = 2/3 = 67%

Cornflakes→Jam

Confidence = Support of (Cornflakes, Jam) / Support of (Cornflakes) = 2/3 = 67%

Jam→Cornflakes

Confidence = Support of (Jam, Cornflakes) / Support of (Jam) = 2/2 = 100%

Therefore, only the last rule Jam→Cornflakes has more than the minimum required confidence, *i.e.*, 75% and it qualifies. The rules having more than the user-specified minimum confidence are known as confident.

Apriori Algorithm

What is Apriori Algorithm

- **Apriori algorithm** is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule.
- Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent item sets are used to find k+1 item sets.

Apriori Property

- *All subsets of a frequent item set must be frequent. If an item set is infrequent, all its supersets will be infrequent.*

Table 2.14 Transaction data for Example 2.3

TID	*Items*
1	Biscuits, Bread, Cheese, Coffee, Yogurt
2	Bread, Cereal, Cheese, Coffee
3	Cheese, Chocolate, Donuts, Juice, Milk
4	Bread, Cheese, Coffee, Cereal, Juice
5	Bread, Cereal, Chocolate, Donuts, Juice
6	Milk, Tea
7	Biscuits, Bread, Cheese, Coffee, Milk
8	Eggs, Milk, Tea
9	Bread, Cereal, Cheese, Chocolate, Coffee
10	Bread, Cereal, Chocolate, Donuts, Juice
11	Bread, Cheese, Juice
12	Bread, Cheese, Coffee, Donuts, Juice
13	Biscuits, Bread, Cereal
14	Cereal, Cheese, Chocolate, Donuts, Juice
15	Chocolate, Coffee
16	Donuts
17	Donuts, Eggs, Juice
18	Biscuits, Bread, Cheese, Coffee
19	Bread, Cereal, Chocolate, Donuts, Juice
20	Cheese, Chocolate, Donuts, Juice
21	Milk, Tea, Yogurt
22	Bread, Cereal, Cheese, Coffee
23	Chocolate, Donuts, Juice, Milk, Newspaper
24	Newspaper, Pastry, Rolls
25	Rolls, Sugar, Tea

Computing L_1

To find the frequent set L_1 we count the number of times each of the 16 items has been sold. Since we wish to scan the transaction database only once, we first set up the list of items and we look at one transaction at a time and update the count of every item that appears in that transaction. We obtain the list given in Table 2.15.

Table 2.15 Frequency count for all items

Item no.	*Item name*	*Frequency*
1	Biscuits	4
2	Bread	13
3	Cereal	10
4	Cheese	11
5	Chocolate	9
6	Coffee	9
7	Donuts	10
8	Eggs	2
9	Juice	11
10	Milk	6
11	Newspaper	2
12	Pastry	1
13	Rolls	2
14	Sugar	1
15	Tea	4
16	Yogurt	2

The items that have the necessary support (25% support in 25 transactions) must occur in at least seven transactions. The frequent 1-itemset or L_1 is now given in Table 2.16.

Table 2.16 The frequent 1-itemset or L_1

Item	*Frequency*
Bread	13
Cereal	10
Cheese	11
Chocolate	9
Coffee	9
Donuts	10
Juice	11

Candidate 2-itemset	*Frequency*
{Bread, Cereal}	9
{Bread, Cheese}	8
{Bread, Chocolate}	4
{Bread, Coffee}	8
{Bread, Donuts}	4
{Bread, Juice}	6
{Cereal, Cheese}	5
{Cereal, Chocolate}	4
{Cereal, Coffee}	5
{Cereal, Donuts}	4
{Cereal, Juice}	6
{Cheese, Chocolate}	4
{Cheese, Coffee}	9
{Cheese, Donuts}	3
{Cheese, Juice}	4
{Chocolate, Coffee}	1
{Chocolate, Donuts}	7
{Chocolate, Juice}	7
{Coffee, Donuts}	1
{Coffee, Juice}	2
{Donuts, Juice}	9

Table 2.19 The frequent 2-itemsets or L_2

Frequent 2-itemset	*Frequency*
{Bread, Cereal}	9
{Bread, Cheese}	8
{Bread, Coffee}	8
{Cheese, Coffee}	9
{Chocolate, Donuts}	7
{Chocolate, Juice}	7
{Donuts, Juice}	9

Computing C_3

To find the candidate 3-itemsets or C_3, we combine the appropriate frequent 2-itemsets from L_2 (these must have the same first item) and obtain four such itemsets. Now we again scan the transaction database to find frequencies. These are given in Table 2.20.

Table 2.20 Candidate 3-itemsets or C_3 and their frequencies

Candidate 3-itemset	*Frequency*
{Bread, Cereal, Cheese}	4
{Bread, Cereal, Coffee}	4
{Bread, Cheese, Coffee}	8
{Chocolate, Donuts, Juice}	7

Computing L_3

Table 2.21 now presents the frequent 3-itemsets or L_3. Note that only one member of L_2 is not a subset of any itemset in L_3. That 2-itemset is {Bread, Cereal}.

Table 2.21 The frequent 3-itemsets or L_3

Frequent 3-itemset	*Frequency*
{Bread, Cheese, Coffee}	8
{Chocolate, Donuts, Juice}	7

Table 2.21 The frequent 3-itemsets or L_3

Frequent 3-itemset	*Frequency*
{Bread, Cheese, Coffee}	8
{Chocolate, Donuts, Juice}	7

Table 2.22 Confidence of association rules from {Bread, Cheese, Coffee}

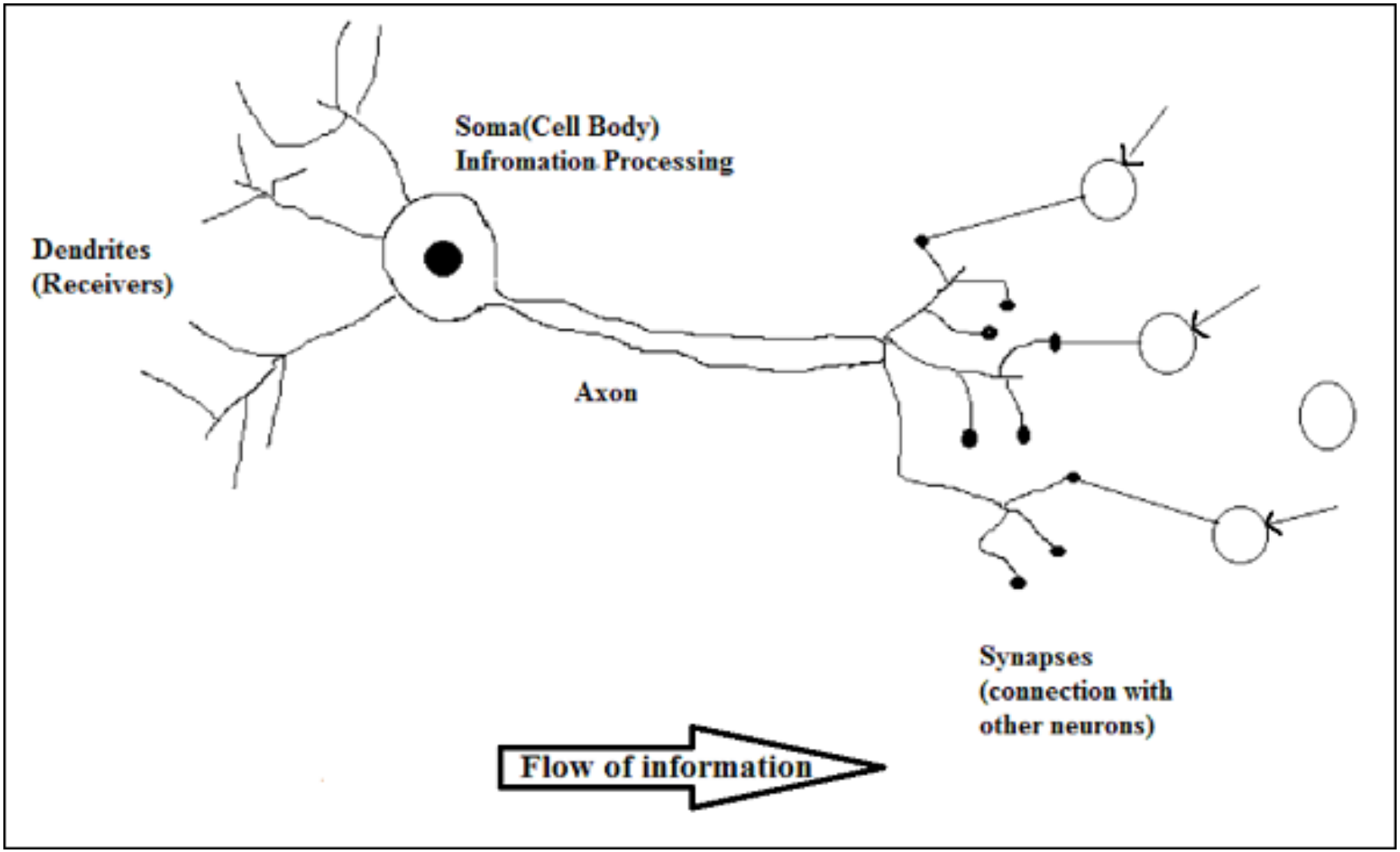
Rule	*Support of BCD*	*Frequency of LHS*	*Confidence*
$B \rightarrow CD$	8	13	0.61
$C \rightarrow BD$	8	11	0.72
$D \rightarrow BC$	8	9	0.89
$CD \rightarrow B$	8	9	0.89
$BD \rightarrow C$	8	8	1.0
$BC \rightarrow D$	8	8	1.0

Table 2.23 Confidence of association rules from {Chocolate, Donuts, Juice}

Rule	*Support*	*Frequency of LHS*	*Confidence*
$N \rightarrow MP$	7	9	0.78
$M \rightarrow NP$	7	10	0.70
$P \rightarrow NM$	7	11	0.64
$MP \rightarrow N$	7	9	0.78
$NP \rightarrow M$	7	7	1.0
$NM \rightarrow P$	7	7	1.0

Artificial Neural Network (ANN)

Biological Neuron



https://www.tutorialspoint.com/artificial_neural_network/artificial_neural_network_basic_concepts.htm

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.

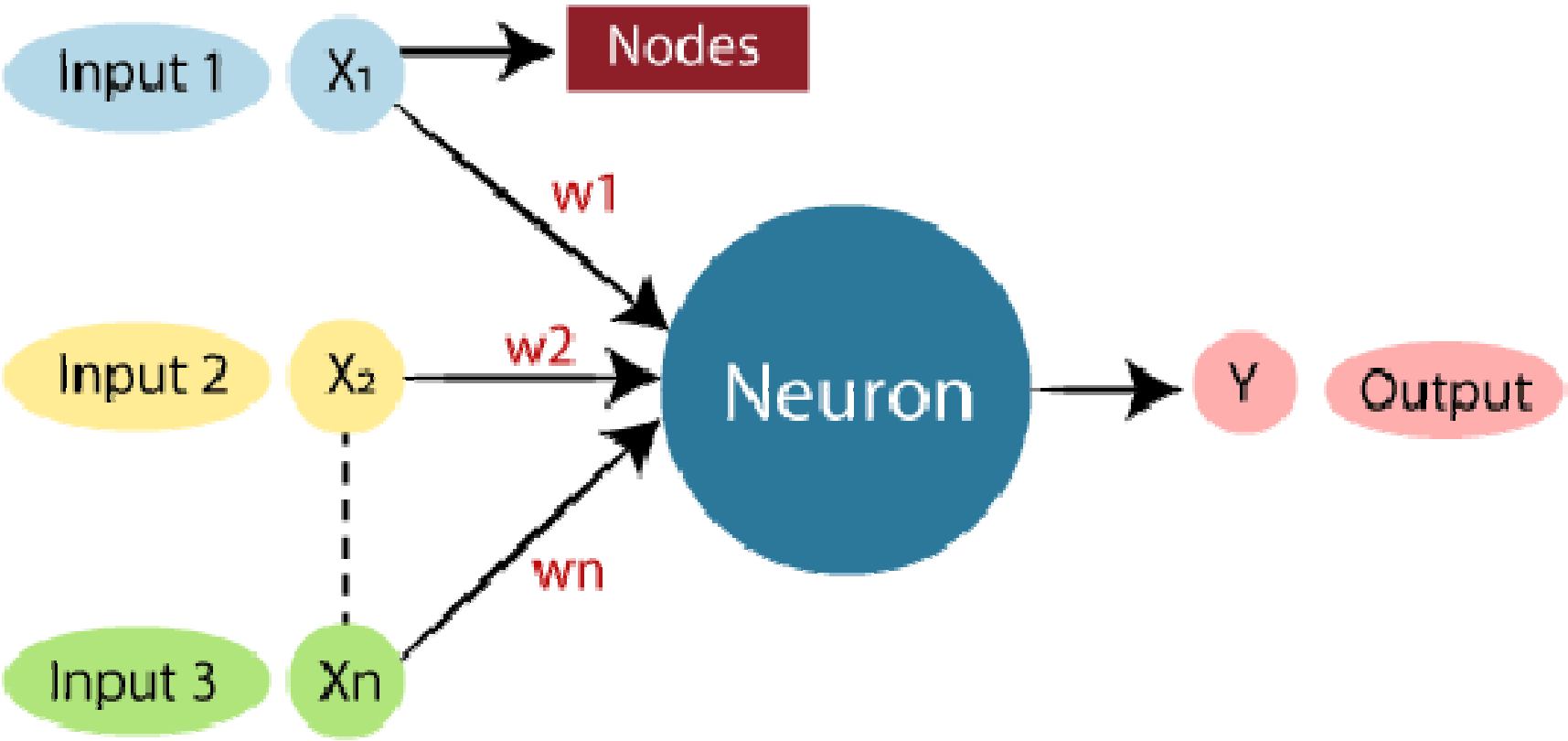
https://www.javatpoint.com/artificial-neural-network

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.

Advantages of Artificial Neural Network (ANN)

- Parallel processing capability.
- Storing data on the entire network.
- Capability to work with incomplete knowledge.
- Having fault tolerance.

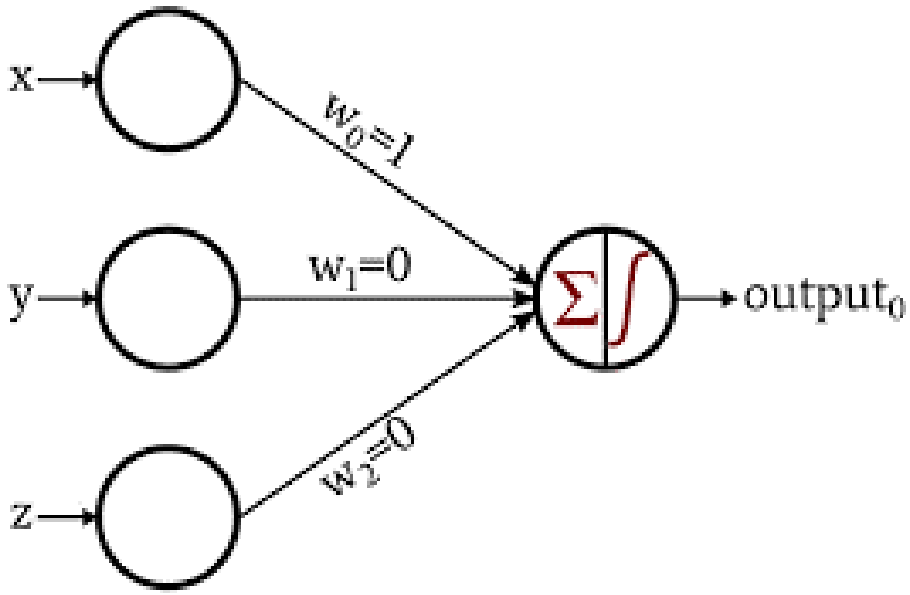
Typical Neural Network



https://www.javatpoint.com/artificial-neural-network

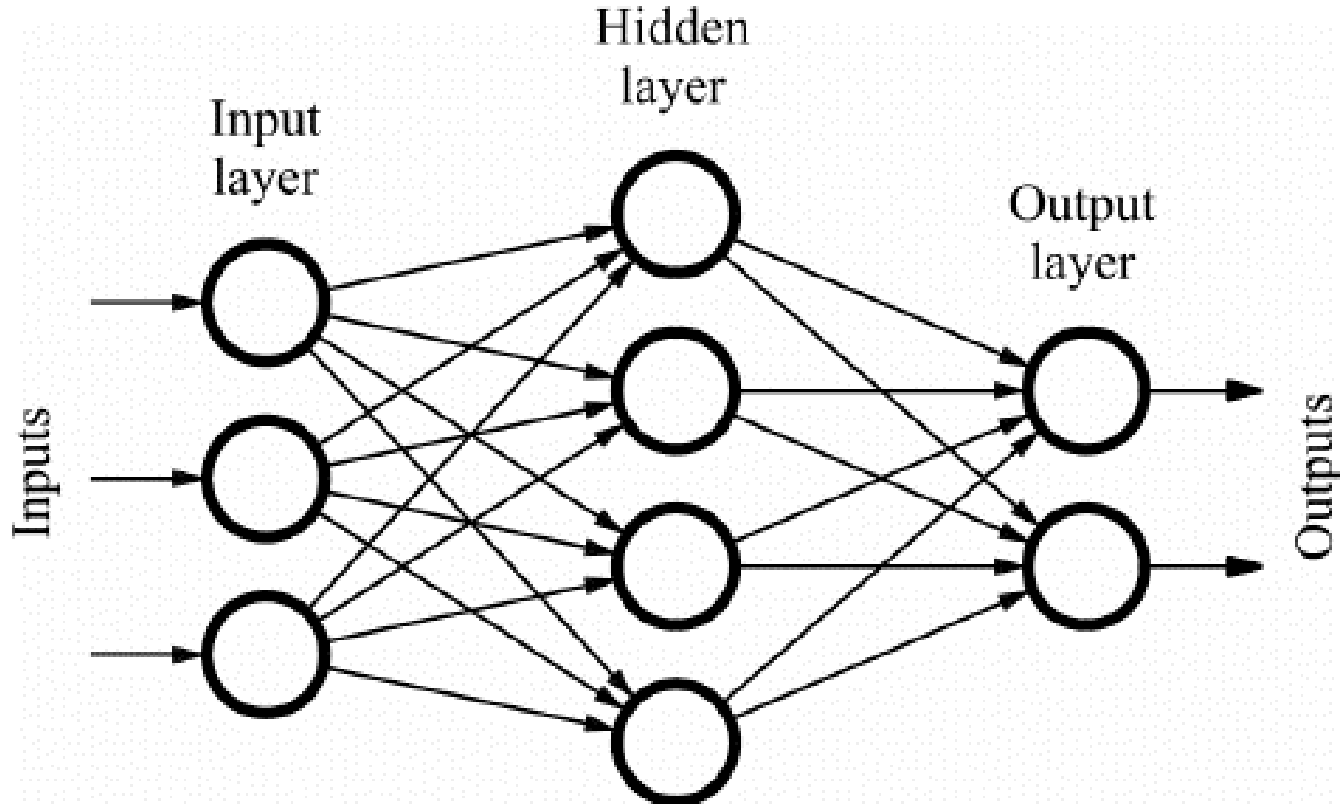
Types of Neural Network

1. Perceptron



The Perceptron is the most basic and oldest form of neural networks. It consists of just 1 neuron which takes the input and applies activation function on it to produce a binary output. It doesn't contain any hidden layers and can only be used for binary classification tasks.

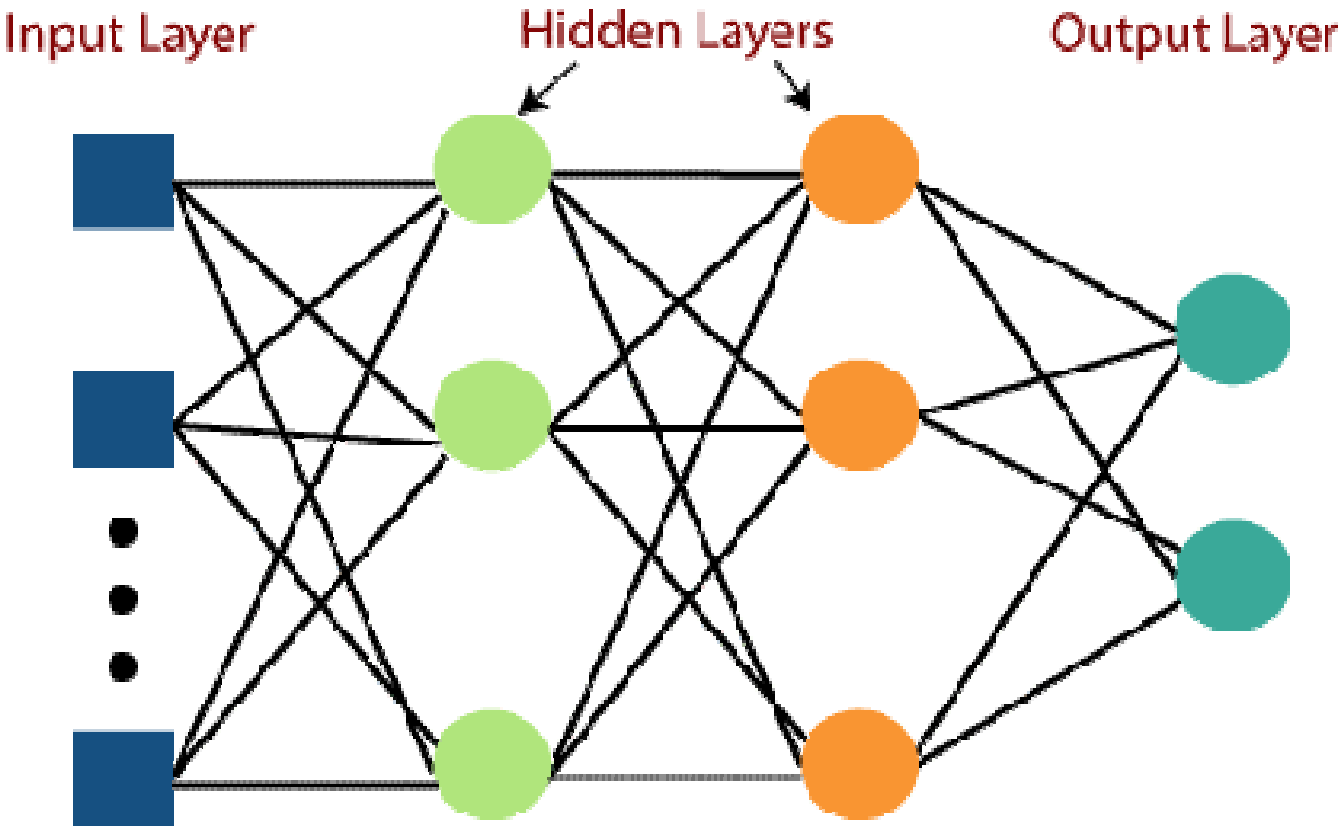
2. Feed Forward Network



The Feed Forward (FF) networks consist of multiple neurons and hidden layers which are connected to each other. These are called “feed-forward” because the data flow in the forward direction only, and there is no backward propagation. Hidden layers might not be necessarily present in the network depending upon the application.

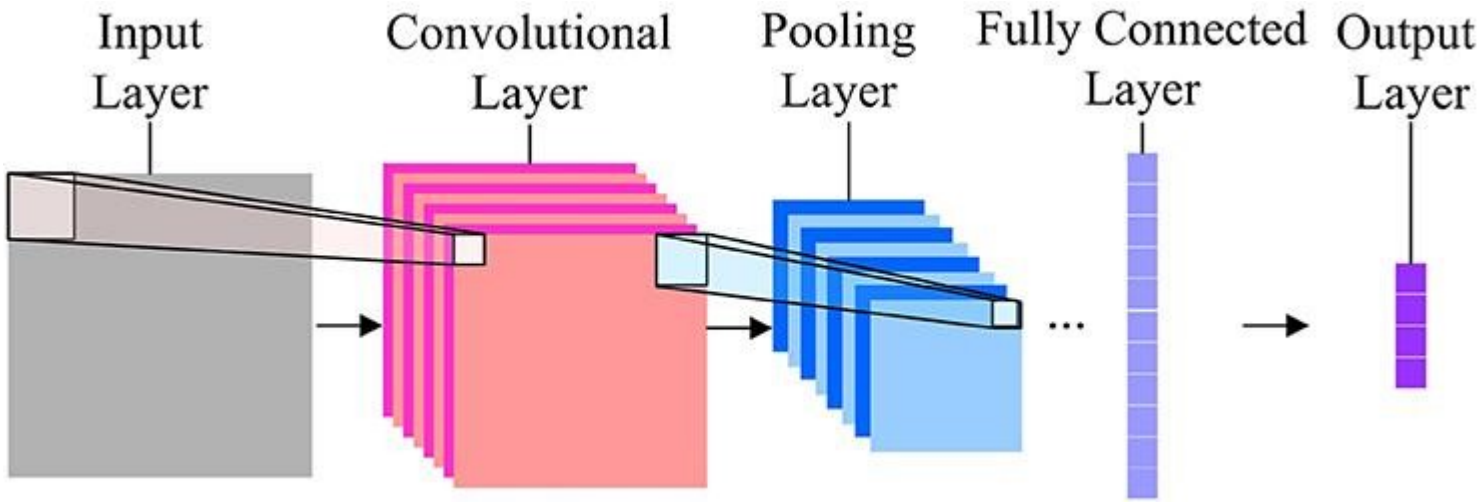
More the number of layers more can be the customization of the weights. And hence, more will be the ability of the network to learn. Weights are not updated as there is no backpropagation. The output of multiplication of weights with the inputs is fed to the activation function which acts as a threshold value.

3. Multi-Layer Perceptron



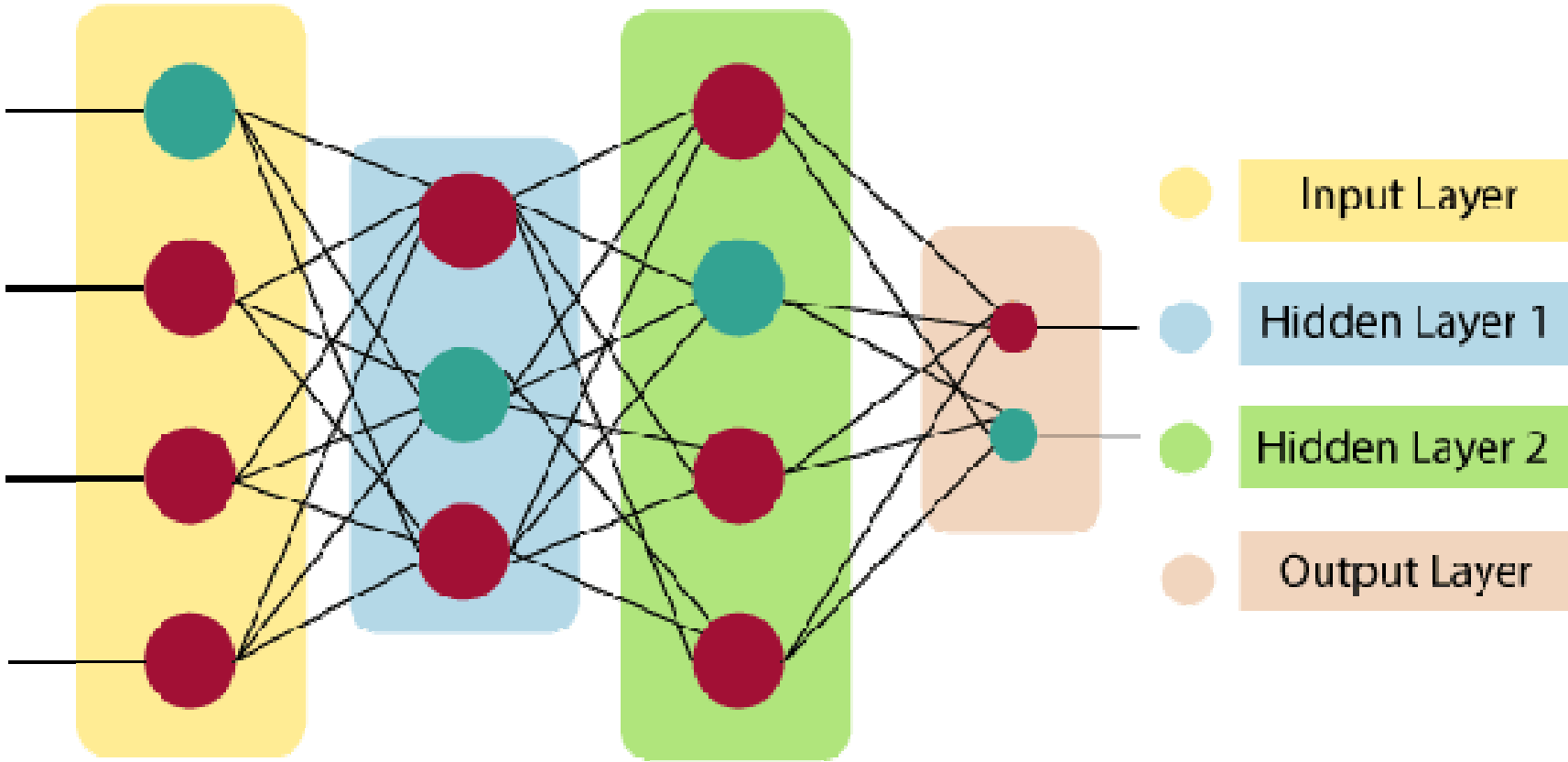
Multi-layer Perceptron is bi-directional, i.e., Forward propagation of the inputs, and the backward propagation of the weight updates. The activation functions can be changes with respect to the type of target. These are also called dense networks because all the neurons in a layer are connected to all the neurons in the next layer.
They are used in Deep Learning based applications but are generally slow due to their complex structure.

4. Convolutional Neural Networks (CNN)



When it comes to image classification, the most used neural networks are Convolution Neural Networks (CNN). CNN contain multiple convolution layers which are responsible for the extraction of important features from the image. The earlier layers are responsible for low-level details and the later layers are responsible for more high-level features.

Architecture of an artificial neural network



https://www.javatpoint.com/artificial-neural-network

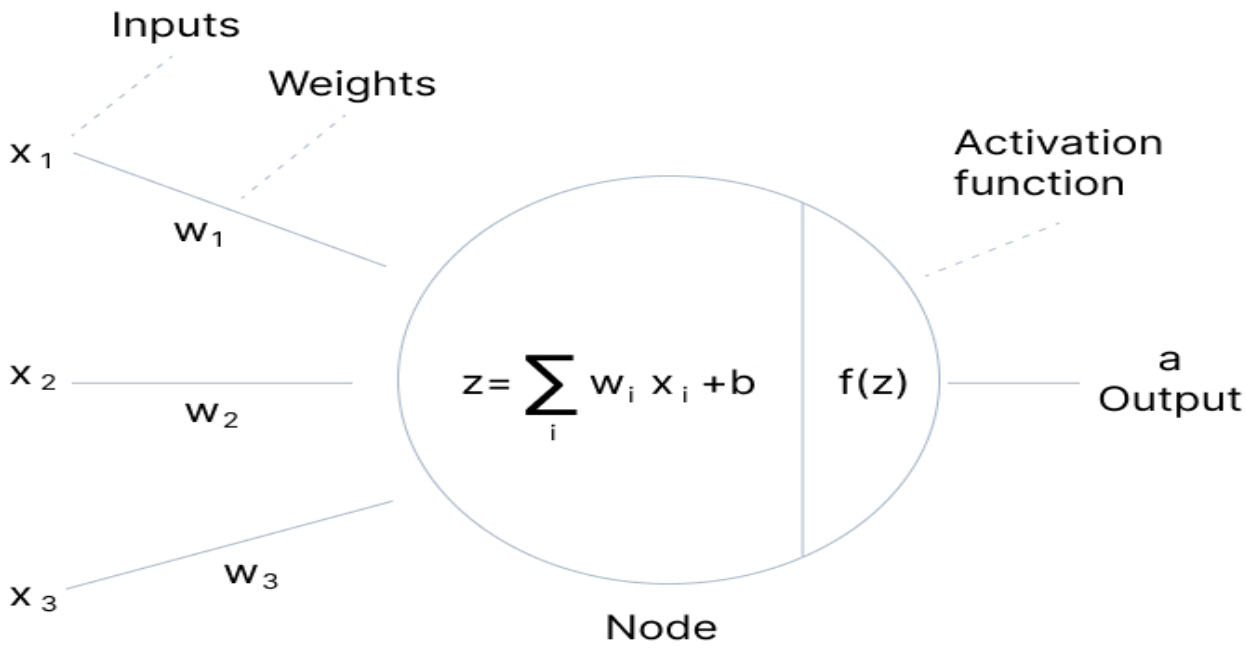
Supervised vs Unsupervised Learning

- Neural networks learn via supervised learning; Supervised machine learning involves an input variable *x* and output variable *y*. The algorithm learns from a training dataset. With each correct answers, algorithms iteratively make predictions on the data. The learning stops when the algorithm reaches an acceptable level of performance.
- Unsupervised machine learning has input data X and no corresponding output variables. The goal is to model the underlying structure of the data for understanding more about the data. The keywords for supervised machine learning are classification and regression. For unsupervised machine learning, the keywords are clustering and association.

Activation **Function**

Activation Function

- An activation function in a neural network defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network.

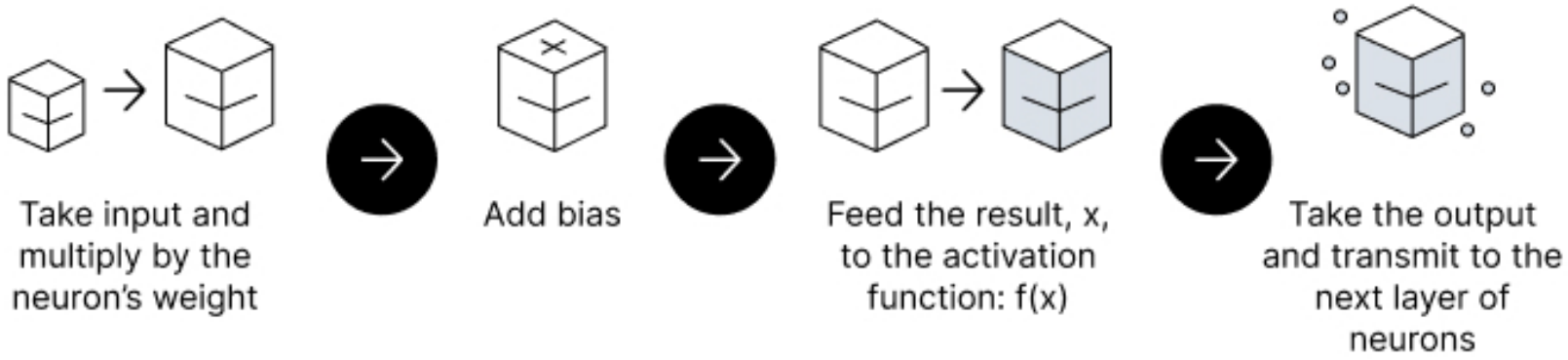


V7 Labs

$$Y = \sum (weight * input) + bias$$

*https://www.v7labs.com/blog/neural-networks-activation-functions

In the feedforward propagation, the Activation Function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.



V7 Labs

Activation functions introduce an additional step at each layer during the forward propagation, but its computation is worth it. Here is why*—

Let's suppose we have a neural network working *without* the activation functions.

In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. It's because it doesn't matter how many hidden layers we attach in the neural network; all layers will behave in the same way because the composition of two linear functions is a linear function itself.

Although the neural network becomes simpler, learning any complex task is impossible, and our model would be just a linear regression model.

*https://www.v7labs.com/blog/neural-networks-activation-functions

Sometimes the activation function is called a "*transfer function*." If the output range of the activation function is limited, then it may be called a "*squashing function*." Many activation functions are nonlinear and may be referred to as the "*nonlinearity*" in the layer or the network design.

The choice of activation function has a large impact on the capability and performance of the neural network, and different activation functions may be used in different parts of the model.
Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer.

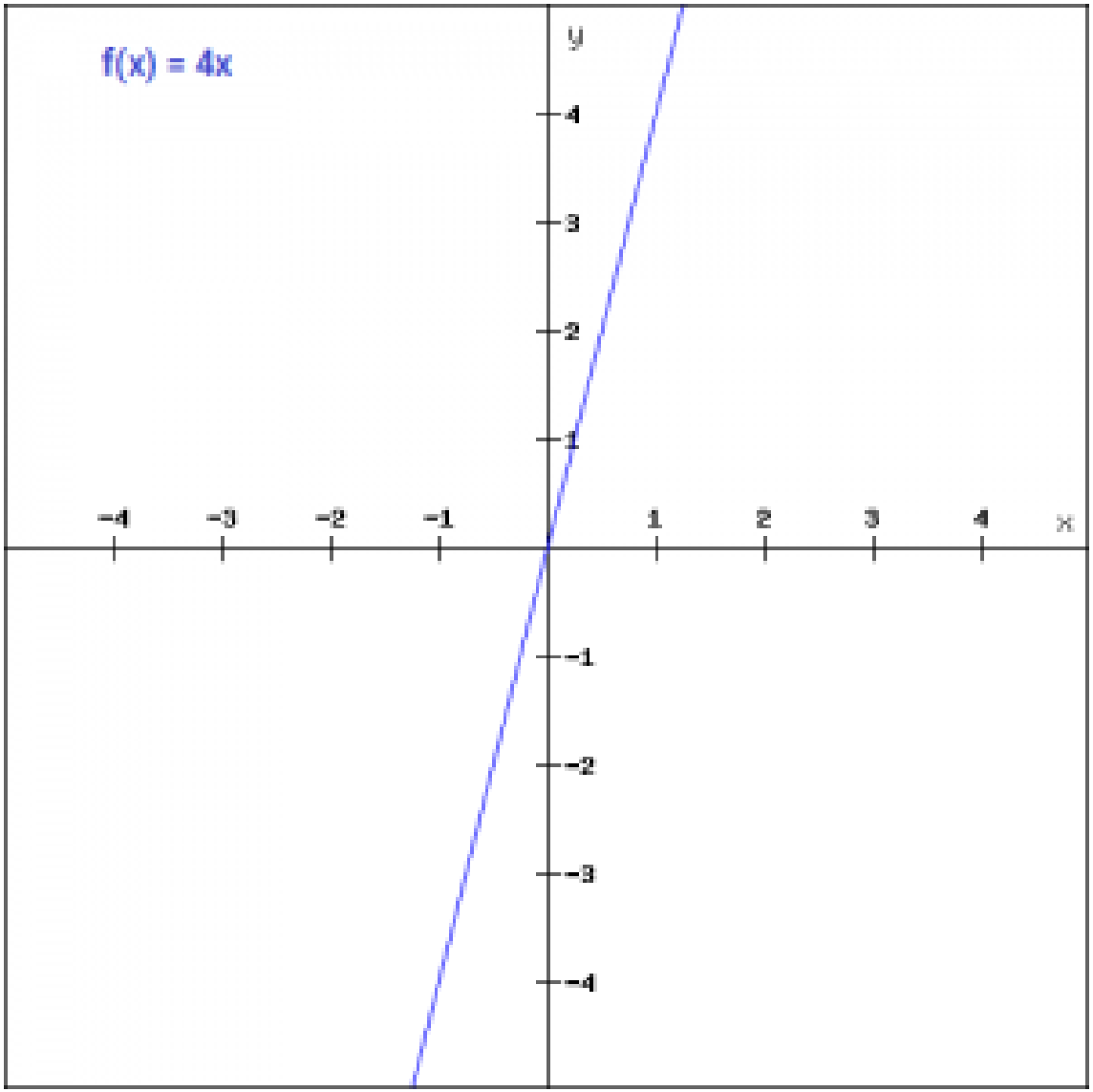
Types of Activation Function

- Linear Function
- Sigmoid
- Tanh — Hyperbolic tangent
- ReLu - Rectified linear units

Linear Function

- Linear function has the equation similar to as of a straight line i.e. **y = ax**
- No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer.
- **Range :** -inf to +inf

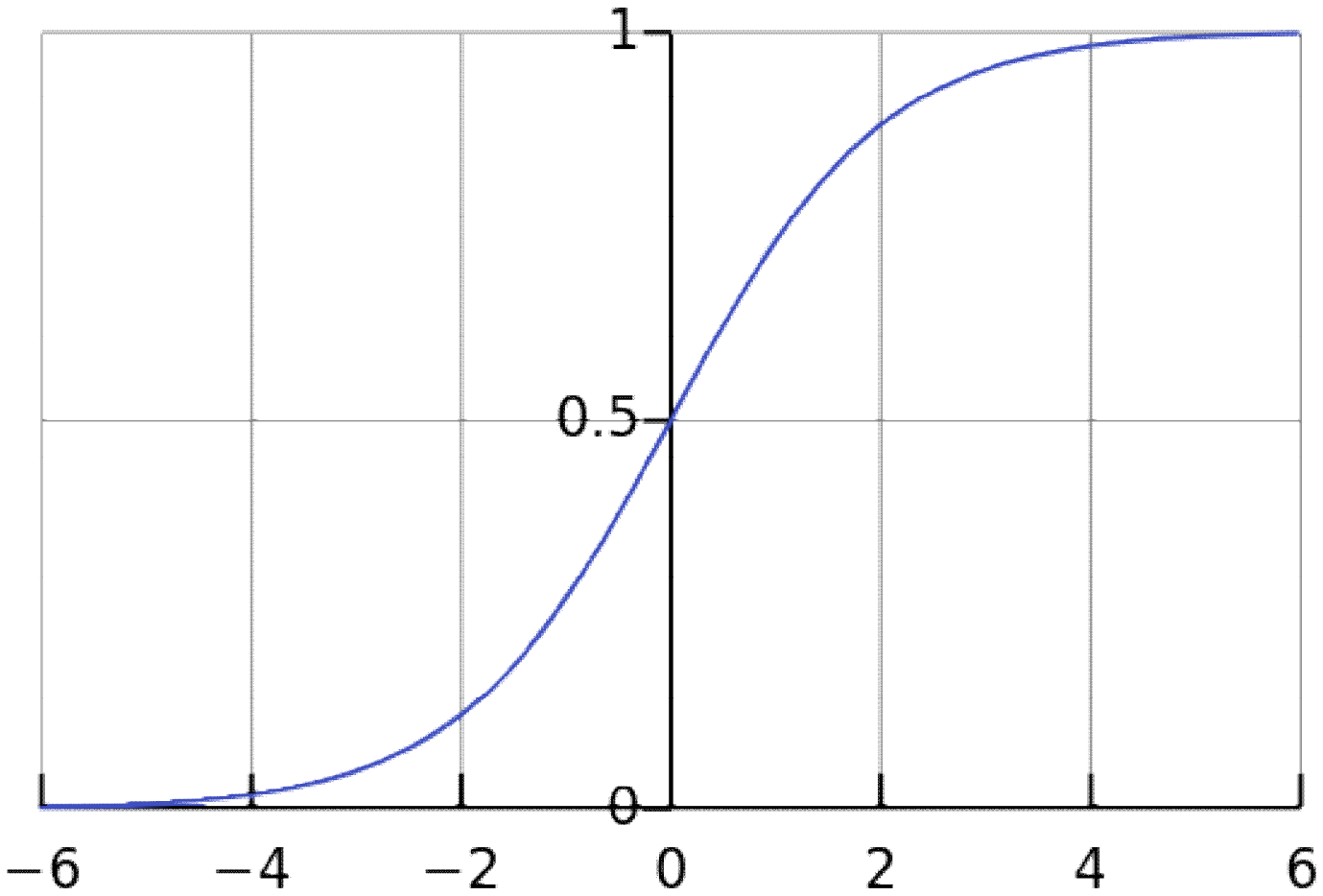
Linear Function



Sigmoid Function

- It is a function which is plotted as **‘S’** shaped graph.
- **Equation :**
  $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1

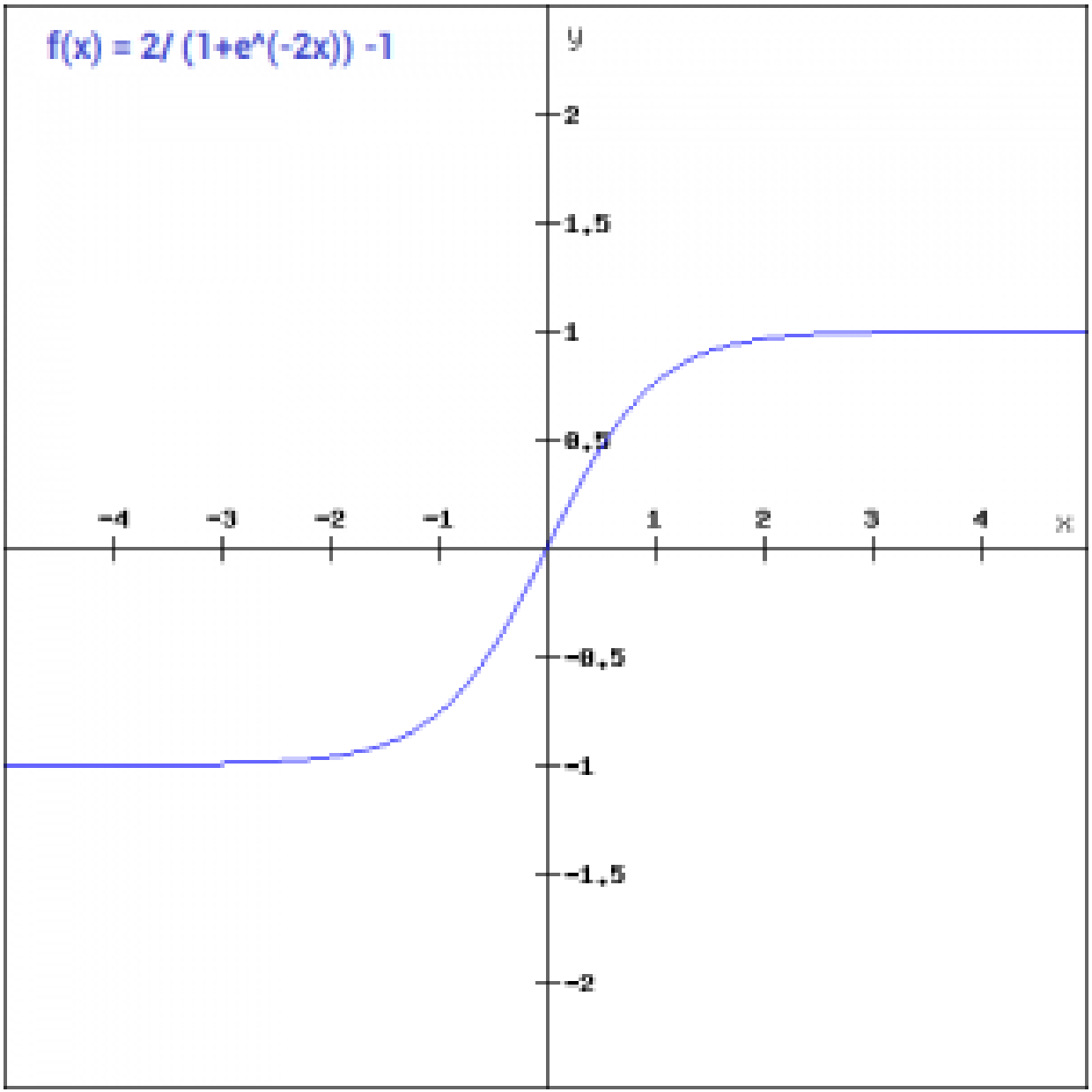
Sigmoid Function



Tanh

- The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. Thus the inputs to the next layers will not always be of the same sign.

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


f(x) = 2/ (1+e^(-2x)) -1
y
2
1,5
1
0,5
-4
-3
-2
-1
1
2
3
4
x
-0,5
-1
-1,5
-2

ReLU

- Stands for *Rectified linear unit*. It is the most widely used activation function. Mainly implemented in *hidden layers* of Neural network.
- **Equation :- *A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.

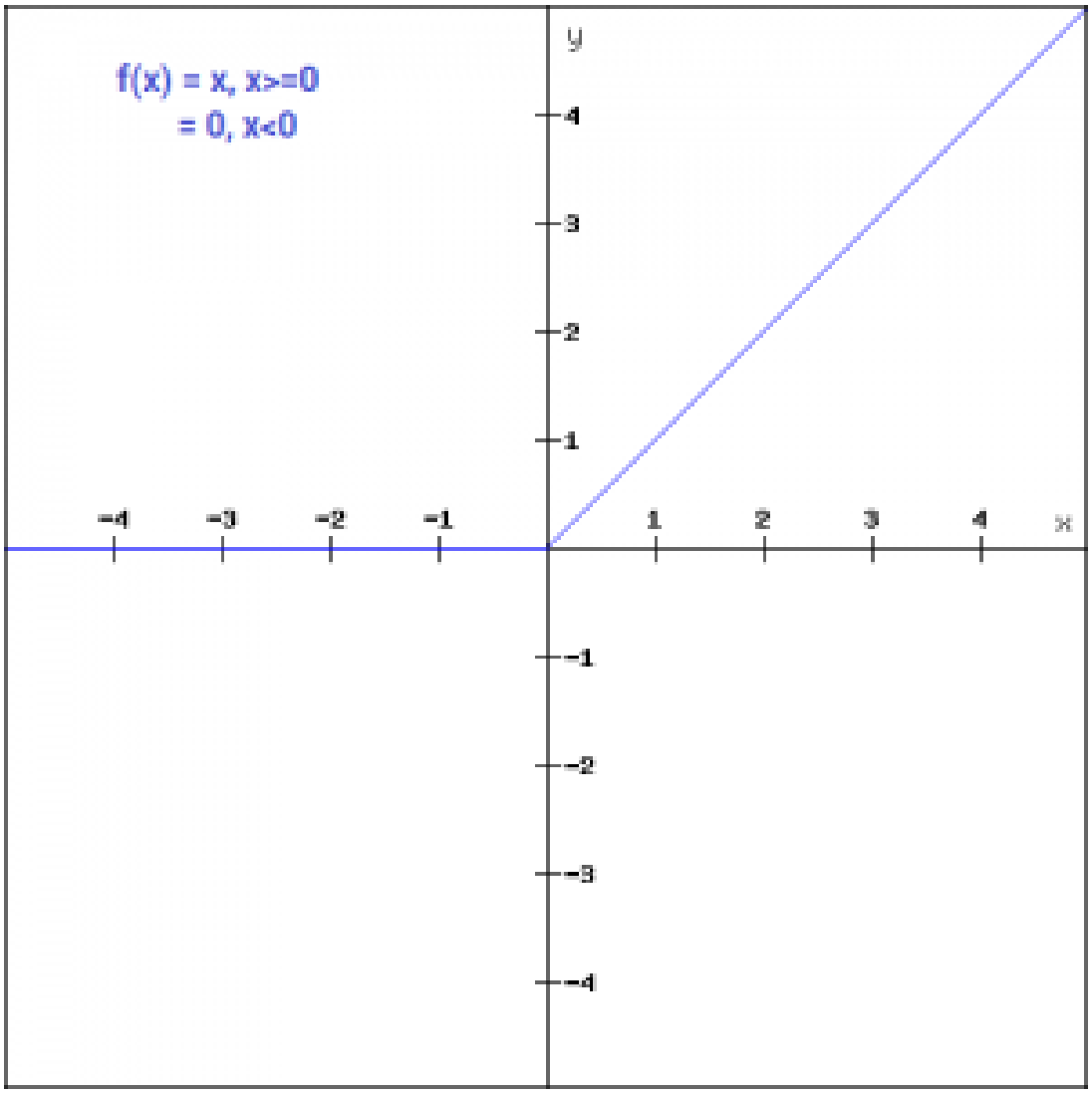
ReLU stands for Rectified Linear Unit.

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.

ReLU



Python Code Example

Input 1	Input 2	Input 3	Output
0	1	1	1
1	0	0	0
1	0	1	1

Fig 2: Training Examples

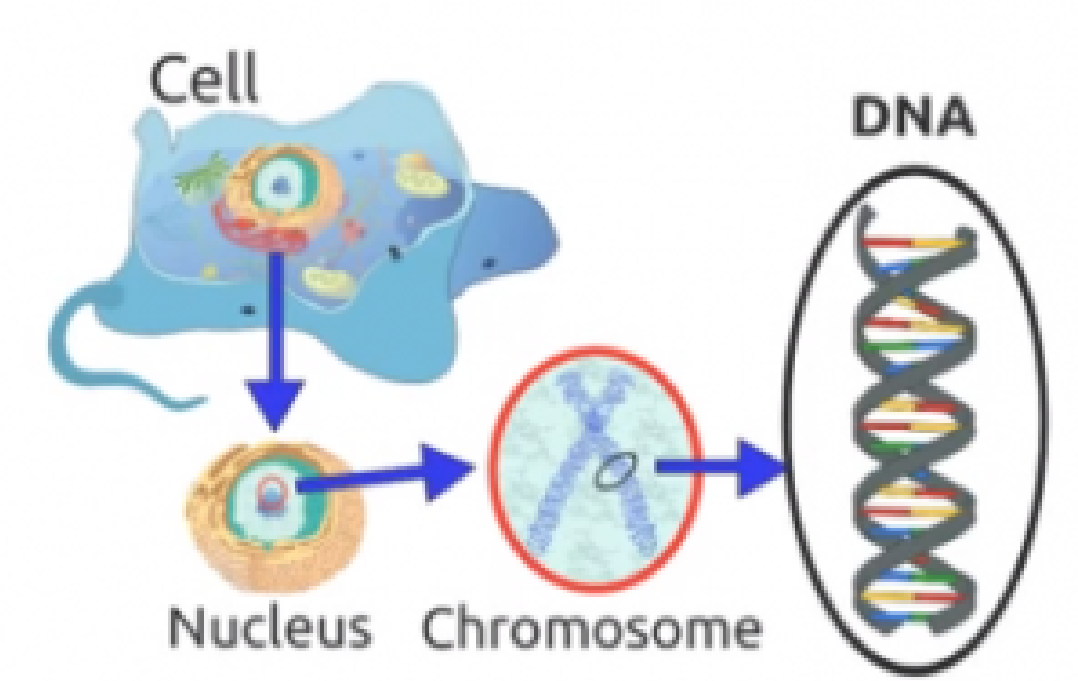
Now we want to predict the output the following set of inputs:

1	0	1	?

Fig 3: Test Example

Genetic Algorithms

Biological Inspiration*



Cells are the basic building block of all living things.
Therefore in each cell, there is the same set of chromosomes. Chromosome are basically the strings of DNA.
A chromosome consists of genes, commonly referred as blocks of DNA, where each gene encodes a specific trait, for example hair color or eye color.

*https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/

What is Genetic Algorithm

- Genetic Algorithm (GA) is a search-based optimization technique based on the principles of **Genetics and Natural Selection**.
- It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve.
- It is frequently used to solve optimization problems, in research, and in machine learning.

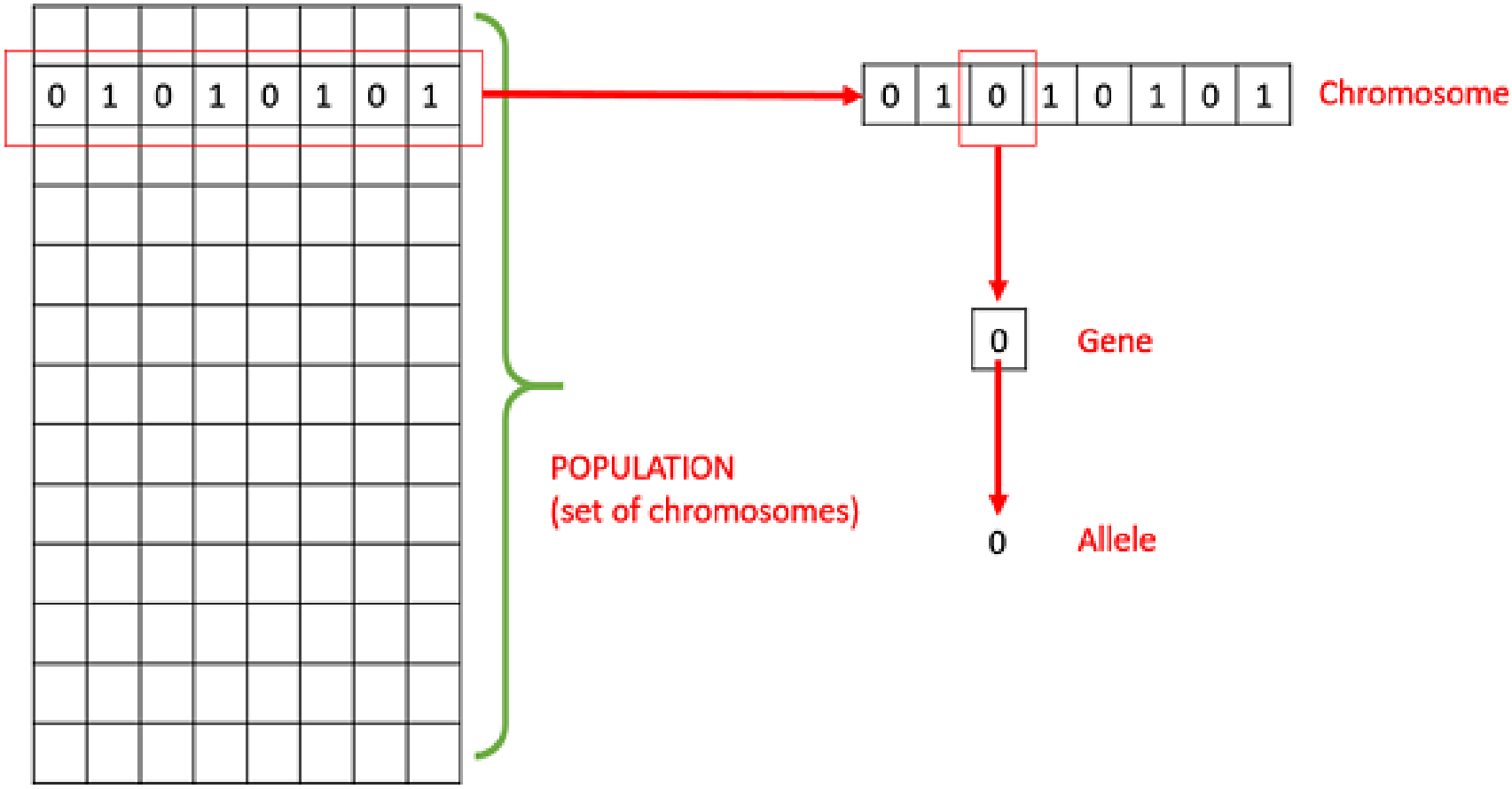
- GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success.

Advantages of GA

- Is faster and more efficient as compared to the traditional methods.
- Has very good parallel capabilities.
- Optimizes both continuous and discrete functions and also multi-objective problems.
- Provides a list of “good” solutions and not just a single solution.
- Always gets an answer to the problem, which gets better over the time.
- Useful when the search space is very large and there are a large number of parameters involved.

Basic Terminology

- **Population** – It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** – A chromosome is one such solution to the given problem.
- **Gene** – A gene is one element position of a chromosome.
- **Allele** – It is the value a gene takes for a particular chromosome.


0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1
Chromosome
0
Gene
POPULATION
(set of chromosomes)
0
Allele

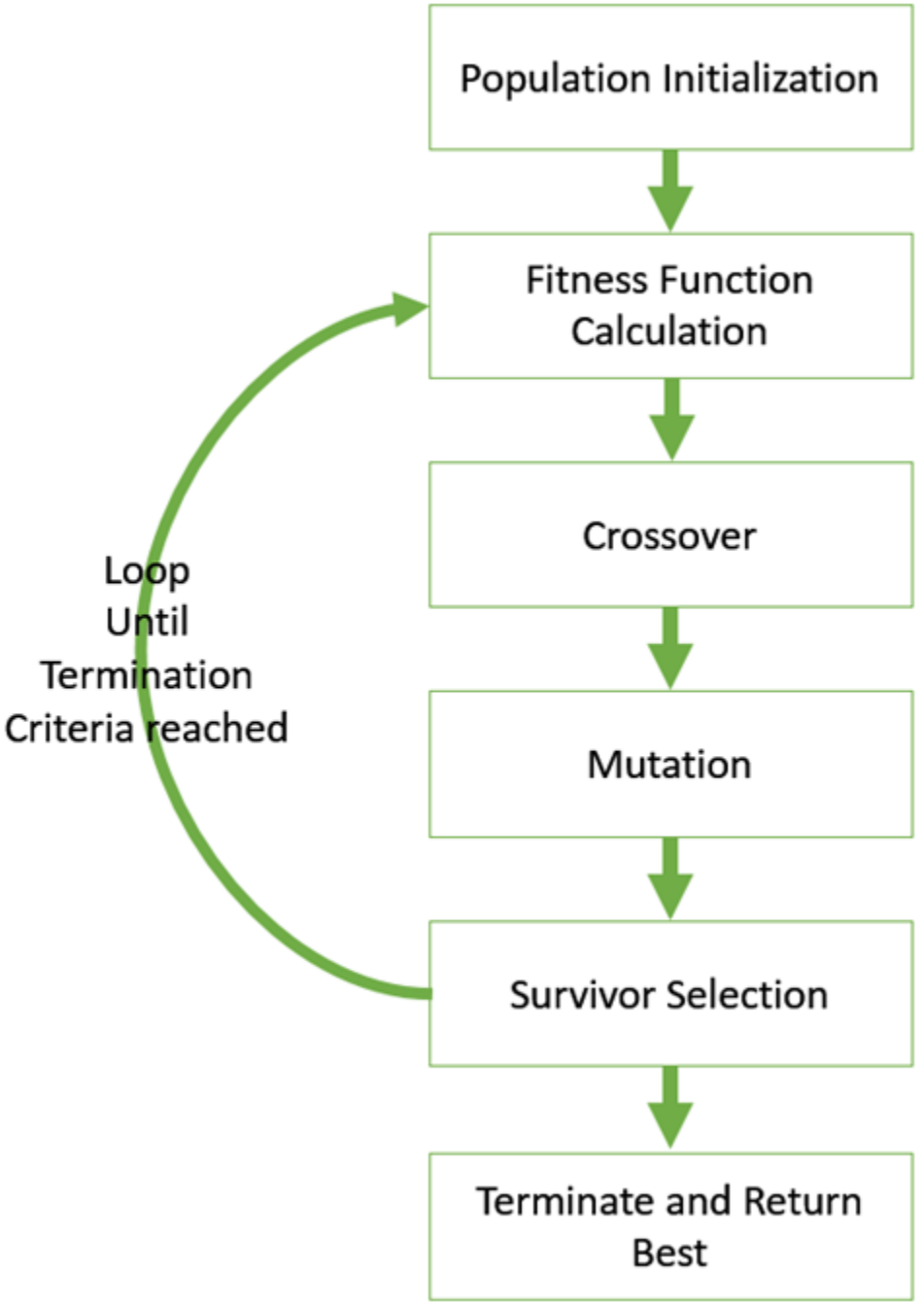
Genotype & Phenotype

- **Genotype** – Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.
- **Phenotype** – Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

Decoding and Encoding

- **Decoding and Encoding** – For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

Genetic Algorithm



Population Initialization

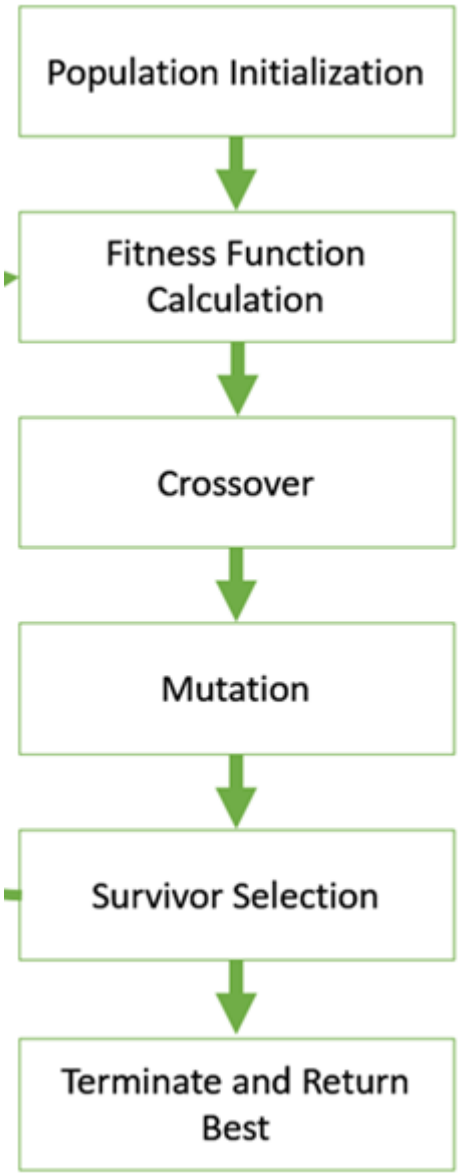
Population is a subset of solutions in the current generation. It can also be defined as a set of chromosomes

Random Initialization – Populate the initial population with completely random solutions.

Heuristic initialization – Populate the initial population using a known heuristic for the problem.

It has been observed that the entire population should not be initialized using a heuristic, as it can result in the population having similar solutions and very little diversity. It has been experimentally observed that the random solutions are the ones to drive the population to optimality.

It has also been observed that heuristic initialization in some cases, only effects the initial fitness of the population, but in the end, it is the diversity of the solutions which lead to optimality.



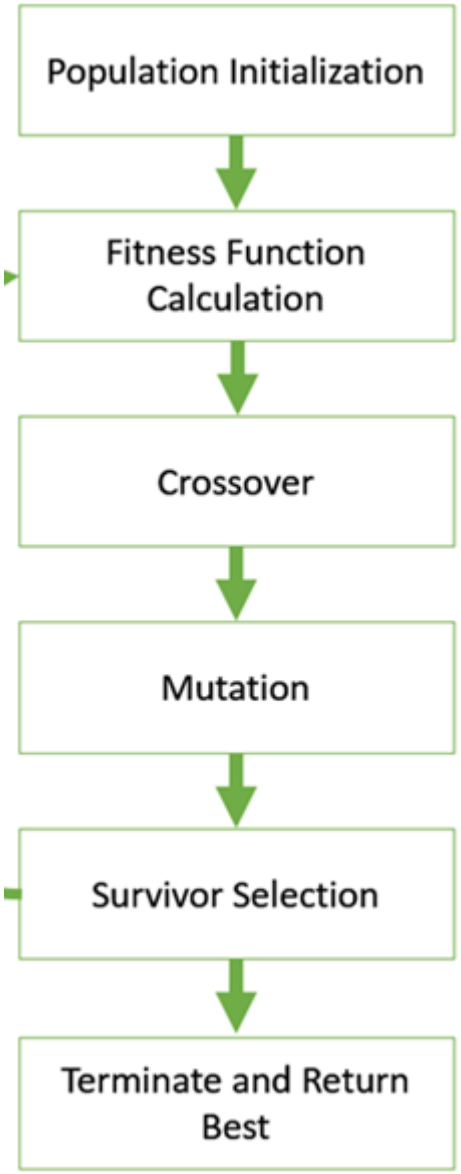
Fitness Function

The fitness function simply defined is a function which takes a **candidate solution to the problem as input and produces as output** how "fit" our how "good" the solution is with respect to the problem in consideration.

A fitness function should possess the following characteristics –

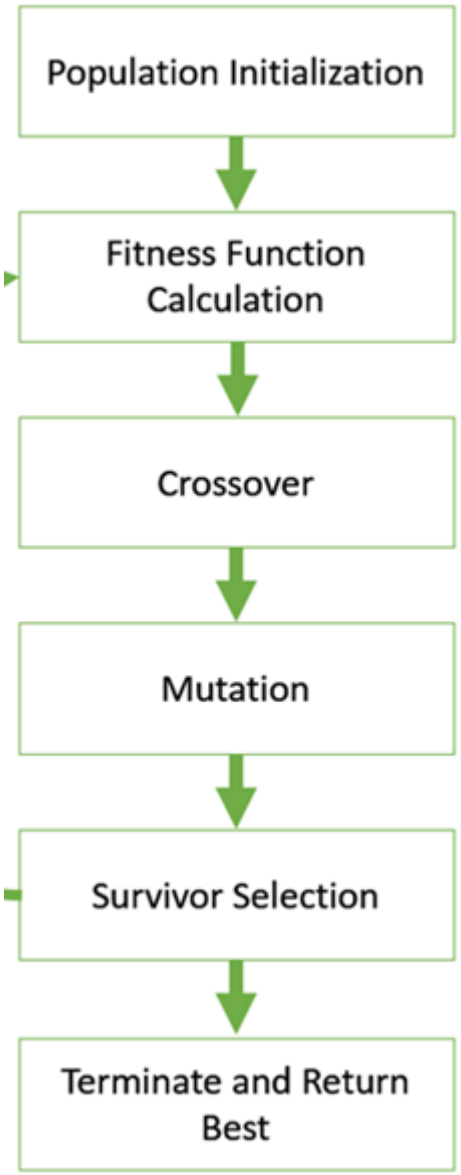
•The fitness function should be sufficiently fast to compute.

• It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

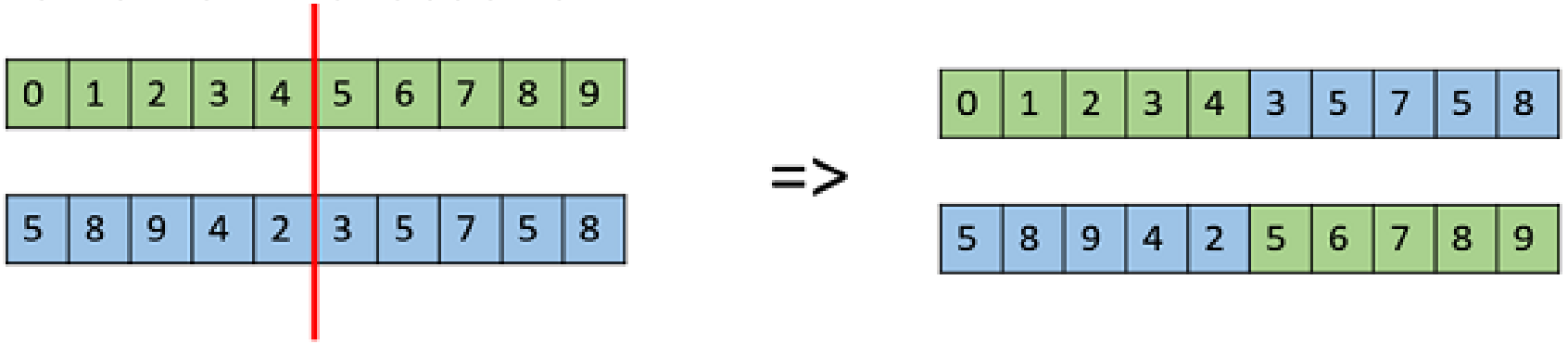


Crossover

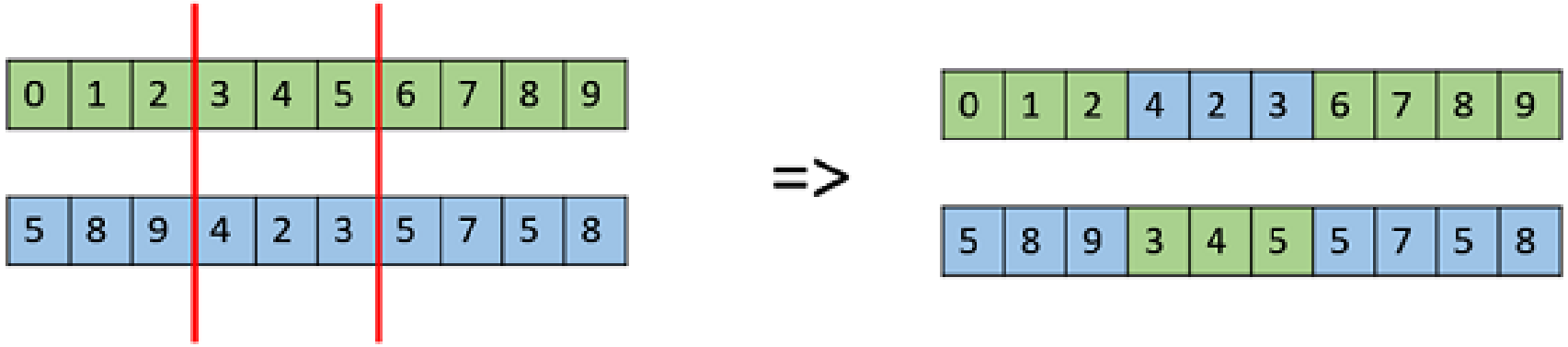
The crossover plays a most significant role in the reproduction phase of the genetic algorithm. In this process, a crossover point is selected at random within the genes. Then the crossover operator swaps genetic information of two parents from the current generation to produce a new individual representing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



One Point Crossover



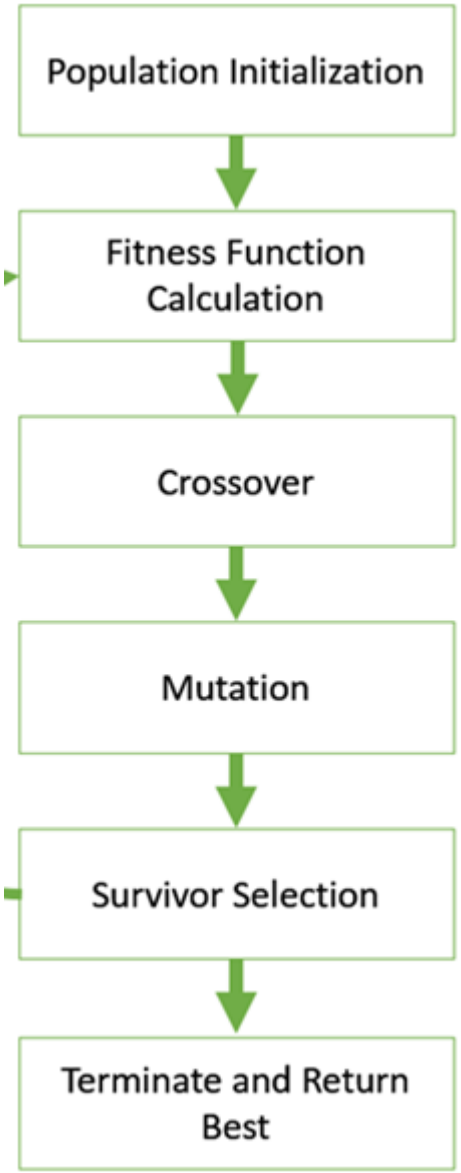
Multi Point Crossover



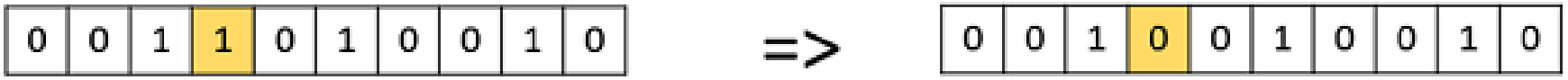
Uniform Crossover

Mutation

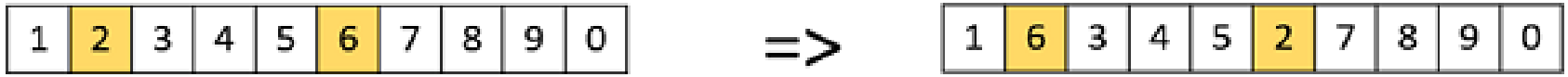
In simple terms, mutation may be defined as a small random tweak in the chromosome, to get a new solution. It can be done by flipping some bits in the chromosomes.



Bit Flip Mutation

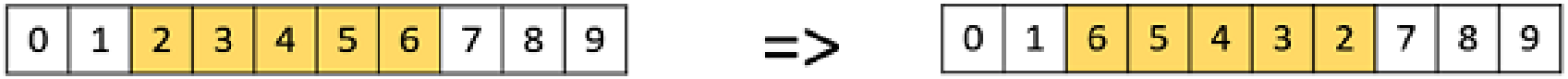


Swap Mutation



Scramble Mutation

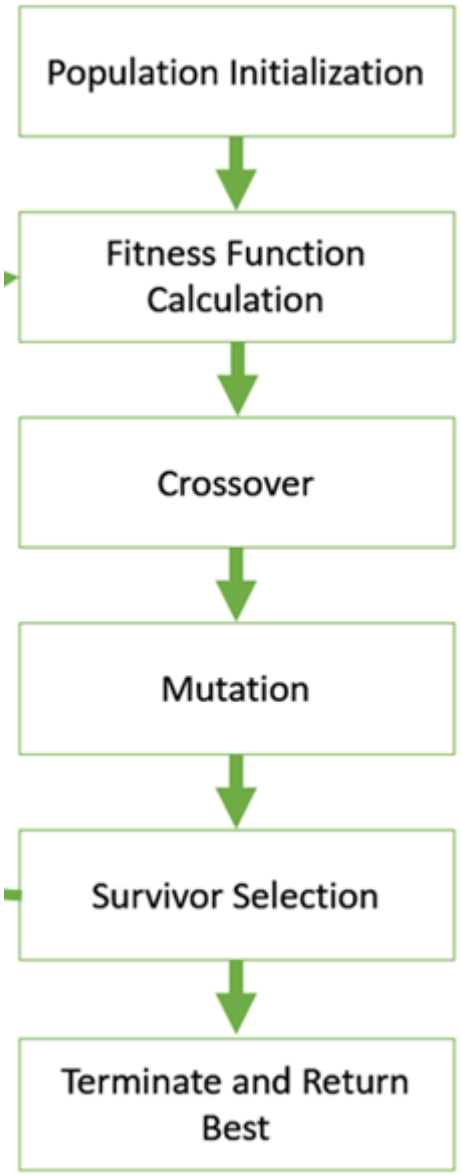
Inversion Mutation



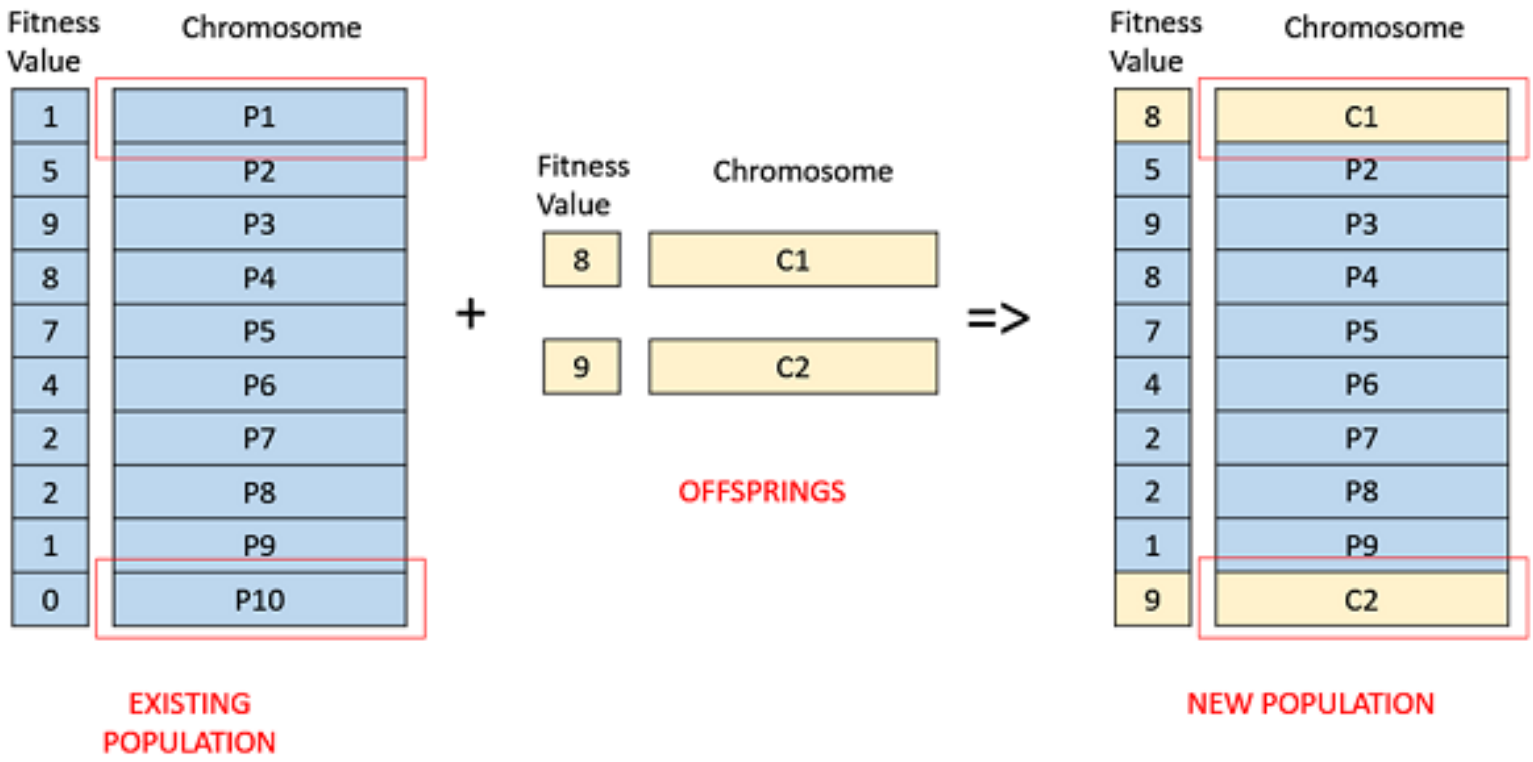
Survivor Selection

The Survivor Selection Policy determines which individuals are to be kicked out and which are to be kept in the next generation. It is crucial as it should ensure that the fitter individuals are not kicked out of the population, while at the same time diversity should be maintained in the population.

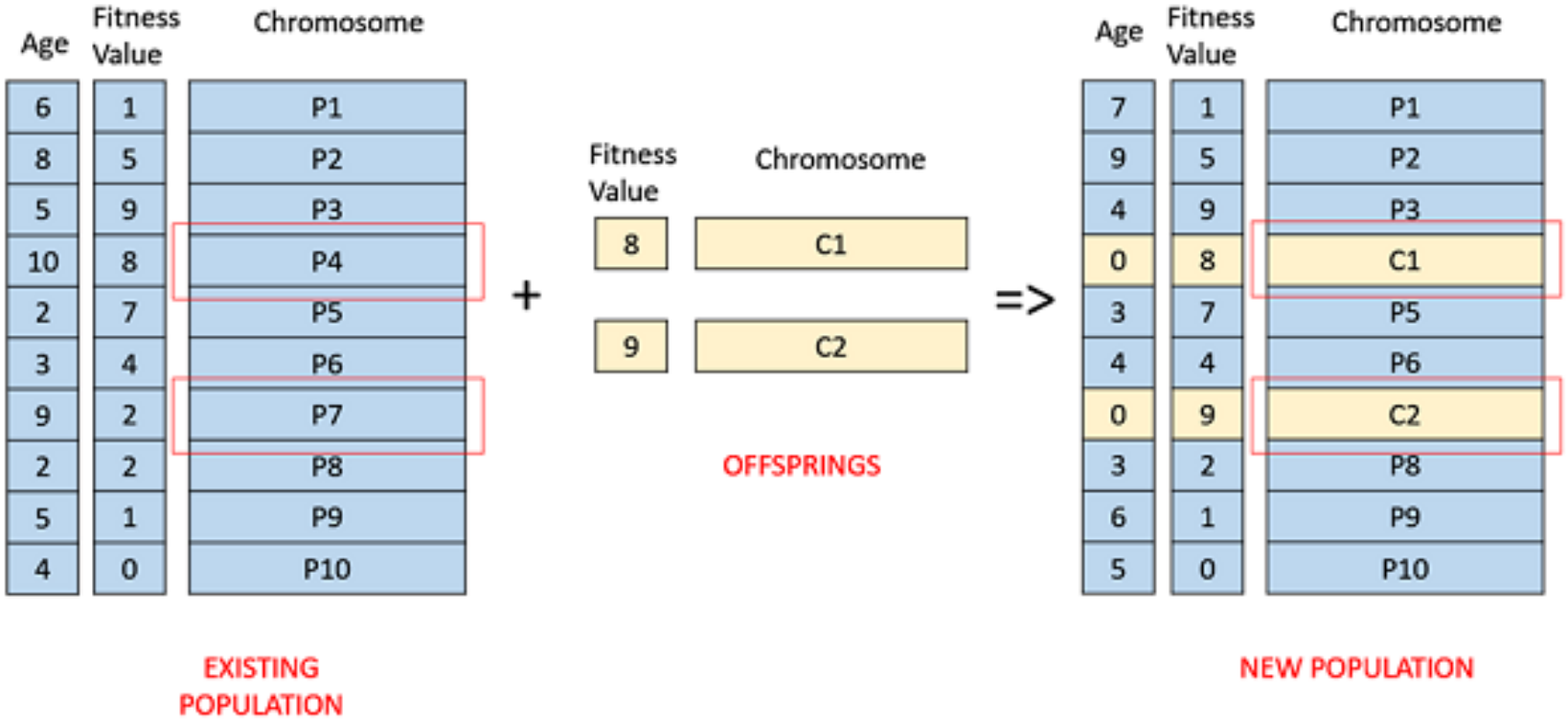
Some GAs employ **Elitism**. In simple terms, it means the current fittest member of the population is always propagated to the next generation. Therefore, under no circumstance can the fittest member of the current population be replaced.



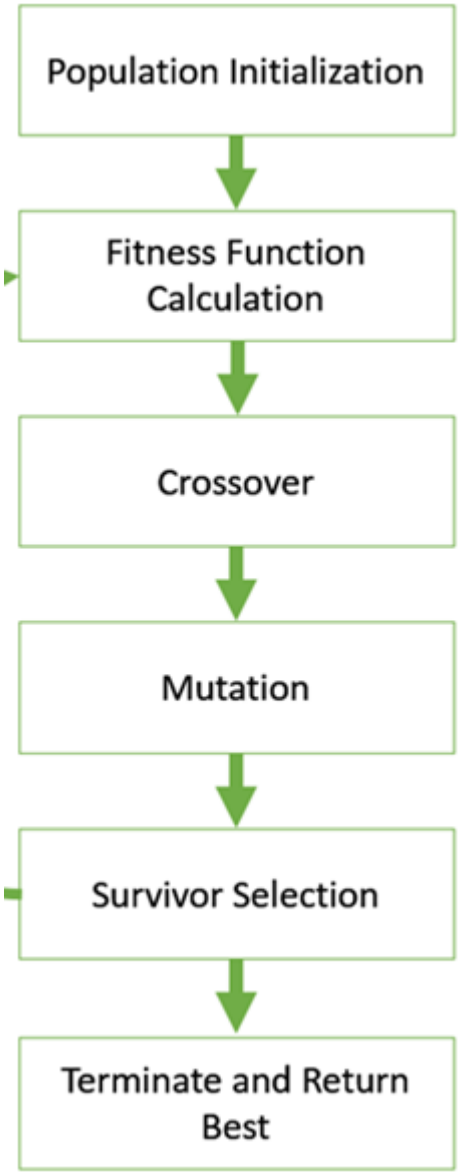
Fitness Based Selection



Age Based Selection



Terminate



The termination condition of a Genetic Algorithm is important in determining when a GA run will end. It has been observed that initially, the GA progresses very fast with better solutions coming in every few iterations, but this tends to saturate in the later stages where the improvements are very small. We usually want a termination condition such that our solution is close to the optimal, at the end of the run.
Usually, we keep one of the following termination conditions –

- When there has been no improvement in the population for X iterations.
- When the objective function value has reached a certain pre-defined value.

https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_termination_condition.htm

Traditional Algorithm	Genetic Algorithm
It selects the next point in the series by a deterministic computation.	It selects the next population by computation, which utilizes random number generators.
It creates an individual point at each iteration. The sequence of points approaches an optimal solution.	It creates a population of points at every iteration. The best point in the population approaches an optimal solution.
Advancement in each iteration is problem specific.	Concurrence in each iteration is a problem independent.

Tutorial

- Refer This tutorial for GA study.
- https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_genotype_representation.htm