5. MACHINE LEARNING

➢ Web Search and Recommendation Engines

➢ Text and Speech Recognition

➢ Social Networks and Advertisement

➢ Speech conversion from one language to other

➢ Medical diagnostics for detecting diseases

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.

The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension, machine learning uses that data to detect patterns in data and adjust program actions accordingly.

Face book's News Feed uses machine learning to personalize each member's feed. If a member frequently stops scrolling in order to read or "like" a particular friend's posts, the News Feed will start to show more of that friend's activity earlier in the feed.

Behind the scenes, the software is simply using statistical analysis and predictive analytics to identify patterns in the user's data and use to patterns to populate the News Feed.

Machine learning is particularly attractive in several real life problems because of the following reasons:

- Some tasks cannot be defined well except by example
- Working environment of machines may not be known at design time
- Explicit knowledge encoding may be difficult and not available
- Environments change over time
- Biological systems learn

Recently, learning is widely used in a number of application areas including,

- Data mining and knowledge discovery
- Speech/image/video (pattern) recognition
- Adaptive control
- Autonomous vehicles/robots
- Decision support systems

What is learning?

"Learning denotes changes in the system that is adaptive in the sense that they enable the system to do the same task (or tasks drawn from the same population) more effectively the next time." --Herbert Simon

"Learning is constructing or modifying representations of what is being experienced." --Richard Michal ski

"Learning is making useful changes in our minds." --Marvin Minsky

Supervised Learning

In supervised learning, the network is presented with inputs together with the target (teacher signal) outputs. Then, the neural network tries to produce an output as close as possible to the target signal by adjusting the values of internal weights. The most common supervised learning method is the "error correction method". Error correction method is used for networks which their neurons have discrete output functions. Neural networks are trained with this method in order to reduce the error (difference between the network's output and the desired output) to zero.

Unsupervised learning

The system is supplied with a set of training examples consisting only of inputs and is required to discover for itself what appropriate outputs should be, e. g. Self-Organizing Map.
Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups(clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bio informatics and data compression.

Reinforcement Learning

In reinforcement learning a teacher is available, but the teacher instead of directly providing the desired action corresponding to a perception, return reward and punishment to the learner for its action corresponding to a perception. Examples include a robot in an unknown topography where it gets a punishment when its hits an obstacle and reward when it moves smoothly. In order to design a learning system, the designer has to make the following choices based on the application.

Learning by Analogy

Reasoning by analogy generally involves abstracting details from a particular set of problems and resolving structural similarities between previously distinct problems. Analogical reasoning refers to this process of recognition and then applying the solution from the known problem to the new problem. Such a technique is often identified as case- based reasoning. Analogical learning generally involves developing a set of mappings between features of two instances.

I. **Transformational Analogy**: Suppose you are asked to prove a theorem in plane geometry. You might look for a previous theorem that is very similar and copy its proof, making substitutions when necessary. The idea is to transform a solution to a previous problem in to solution for the current problem. The following figure shows this process

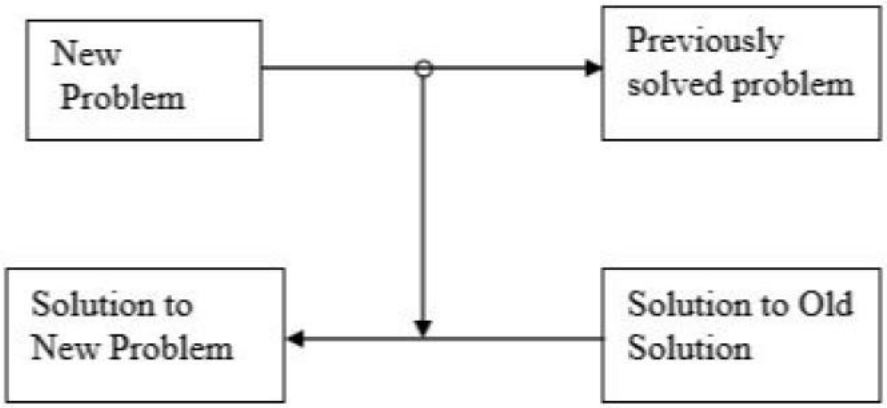


Fig: Transformational Analogy

II. Derivational Analogy

The transformational analogy does not look at how the old problem was solved: it only looks at the final solution. The detail of a problem-solving process is called derivation. And the analogical reasoning that incorporates these detailing is called derivational analogy. Derivational analogy is a necessary component in the transfer of skills in complex domains.

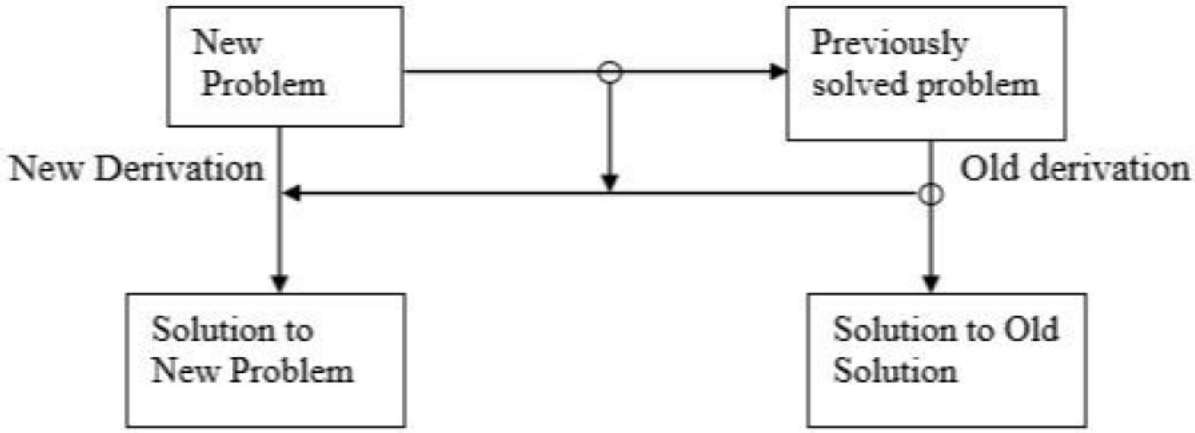


Fig: Derivational Analogy

Inductive Learning

Inductive learning is a process of learning where general principles, rules, or theories are derived from specific observations or examples. It emphasizes the transition from the particular to the general, making it a cornerstone of both human learning and artificial intelligence.

Examples of Inductive Learning

1. **Human Learning**:
    - A child sees several types of dogs and learns to generalize what features classify an animal as a dog.
    - Observing that plants need sunlight, water, and soil to grow, and forming a general principle of plant care.
2. **Machine Learning**:
    - Training a model to recognize spam emails based on labeled examples of spam and non-spam emails.
    - Building a recommendation system based on user preferences and past behavior.

Steps in Inductive Learning

1. **Observation**:
    - Gather specific data points or examples.
    - Example: Measure temperatures at different times of the day.
2. **Pattern Recognition**:
    - Identify similarities, trends, or recurring structures.
    - Example: Recognizing that mornings are cooler and afternoons are warmer.
3. **Generalization**:
    - Formulate a broader rule or theory that applies to similar situations.
    - Example: Concluding that temperature rises during the day due to the sun's position.
4. **Validation**:
    - Test the generalization on new data or scenarios to ensure accuracy.
    - Example: Predicting tomorrow’s temperature changes based on the rule and comparing with actual data.

Explanation based learning

Explanation-Based Learning (EBL) is a type of learning approach that focuses on understanding and internalizing the reasoning or process behind a solution or concept. Unlike trial-and-error or rote memorization, EBL emphasizes creating a deeper understanding by explaining *why* a certain conclusion is valid or why a specific process works.

Steps in Explanation-Based Learning

1. **Input**:
    - The system or learner receives a specific example or case.
    - Example: A math problem showing how to calculate the area of a rectangle.
2. **Explanation**:
    - Analyze and explain the reasoning or method that solves the problem.
    - Example: Understanding that the area is calculated by multiplying the length and width because it represents the total space covered.
3. **Generalization**:
    - Formulate a general rule or schema based on the explanation.

    - Example: Generalizing the area formula to apply to all rectangles, not just the one in the example.
4. **Validation**:
    - Test the explanation and generalization on new examples to ensure consistency.
    - Example: Applying the area formula to different rectangles to confirm correctness.

Examples of EBL

1. **In Human Learning**:
    - A student learns about gravity by understanding Newton's reasoning rather than memorizing the formula.
    - A medical trainee understands a diagnosis by analyzing the symptoms, test results, and reasoning of an experienced doctor.
2. **In Artificial Intelligence**:
    - AI learns to play chess by studying why certain strategies succeed in specific positions, then generalizes the reasoning to other scenarios.
    - In natural language processing, systems learn to parse sentences by analyzing grammar rules from examples.

Supervised Learning

Supervised Learning is a type of machine learning where a model is trained using labeled data. The algorithm learns to map input data to the correct output (label) based on examples provided during training. The goal is to generalize from these examples to make accurate predictions on new, unseen data.

Types of Supervised Learning

1. **Classification**:
    - The output is a discrete category or class.
    - Example: Classifying emails as spam or not spam, predicting whether an image contains a cat or a dog.
2. **Regression**:
    - The output is a continuous value.
    - Example: Predicting house prices based on size, location, and features.

Nearest Neighbor Algorithm

- K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Example: Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors
Step-2: Calculate the Euclidean distance of **K number of neighbors**
Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.
Step-4: Among these k neighbors, count the number of the data points in each category.
Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.
Step-6: Our model is ready.

Numerical Example

Example: Classification with $k = 3$

Suppose we have a dataset with two features and two classes: **Red** and **Blue**.

Point	Feature 1	Feature 2	Class
A	1.0	1.0	Red
B	1.5	2.0	Red
C	2.0	2.5	Blue
D	3.0	3.0	Blue

Query Point: (2.0, 2.0)

Step 1: Compute Distances (Euclidean)

$$\text{Distance to A} = \sqrt{(2.0 - 1.0)^2 + (2.0 - 1.0)^2} = \sqrt{1 + 1} = \sqrt{2} \approx 1.41$$

$$\text{Distance to B} = \sqrt{(2.0 - 1.5)^2 + (2.0 - 2.0)^2} = \sqrt{0.25} = 0.5$$

$$\text{Distance to C} = \sqrt{(2.0 - 2.0)^2 + (2.0 - 2.5)^2} = \sqrt{0.25} = 0.5$$

$$\text{Distance to D} = \sqrt{(2.0 - 3.0)^2 + (2.0 - 3.0)^2} = \sqrt{1 + 1} = \sqrt{2} \approx 1.41$$

Step 2: Select k=3 Nearest Neighbors

Sorting distances:

- B: 0.5 (Red)
- C: 0.5 (Blue)

- A: 1.41 (Red)

The k=3 nearest neighbors are B (Red), C (Blue), and A (Red).

Step 3: Majority Voting
The classes of the neighbors are:

- Red: 2
- Blue: 1

Since **Red** is the majority, the query point is classified as **Red**.

Example: Regression with $k = 2$

Suppose we want to predict the price of a house based on its size (in square feet). Here's our dataset:

House	Size (sq ft)	Price ($)
A	1200	300,000
B	1400	350,000
C	1600	400,000
D	2000	500,000

Query Point: Size = 1500 sq ft (we want to predict the price).

Step 1: Compute Distances

We calculate the distance between the query point and each house in the dataset (distance is the absolute difference since we're using one feature):

$$\text{Distance to A} = |1500 - 1200| = 300$$

$$\text{Distance to B} = |1500 - 1400| = 100$$

$$\text{Distance to C} = |1500 - 1600| = 100$$

$$\text{Distance to D} = |1500 - 2000| = 500$$

Step 2: Select $k = 2$ Nearest Neighbors

Sorting distances:

- B: 100 (Price = 350,000)
- C: 100 (Price = 400,000)

The $k = 2$ nearest neighbors are B and C.

Step 3: Predict Price

For regression, we take the **average** price of the neighbors:

$$\text{Predicted Price} = \frac{350,000 + 400,000}{2} = 375,000$$

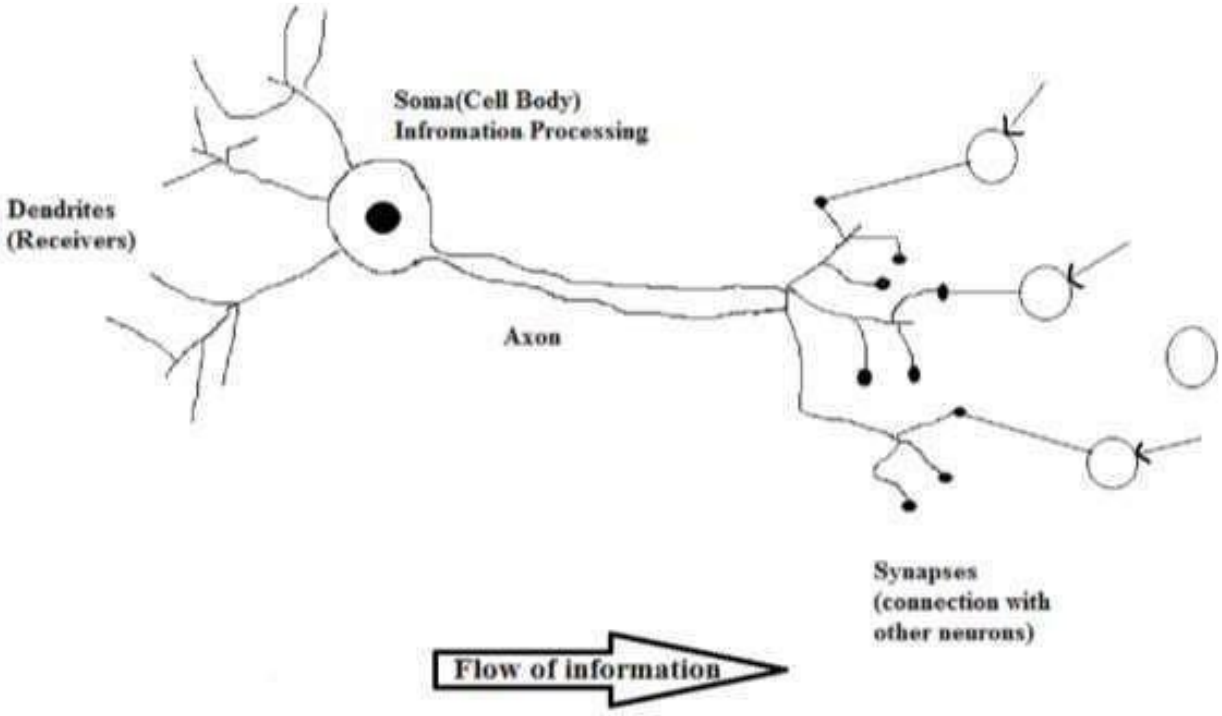
The predicted price for a 1500 sqft house is $375,000**.**

Neural Network

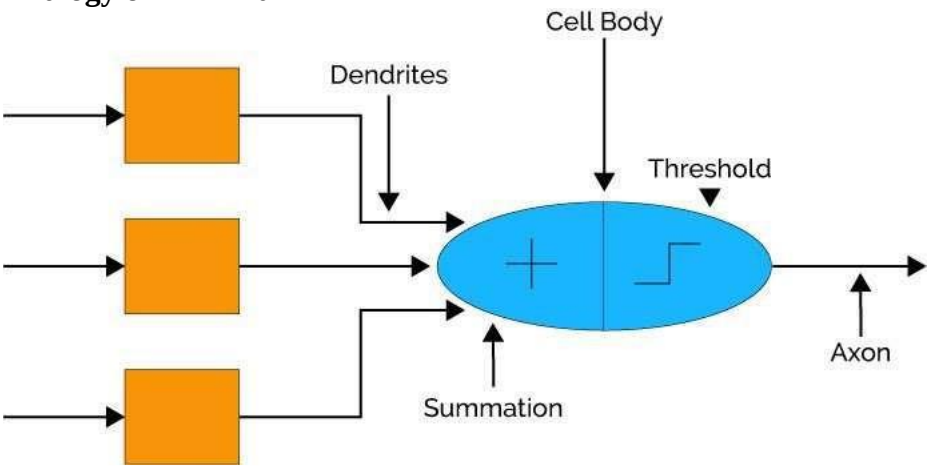
- Artificial Neural Network (ANN) is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks.
- ANNs are also named as “artificial neural systems,” or “parallel distributed processing systems,” or “connectionist systems.”
- ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units.
- These units, also referred to as nodes or neurons, are simple processors which operate in parallel.
- Every neuron is connected with other neuron through a connection link. Each connection link is associated with a weight that has information about the input signal.
- This is the most useful information for neurons to solve a particular problem because the weight usually excites or inhibits the signal that is being communicated.
- Each neuron has an internal state, which is called an activation signal. Output signals, which are produced after combining the input signals and activation rule, may be sent to other units.

Following table Shows some differences between ANN and BNN :-

Criteria	BNN	ANN
Processing	Massively parallel, slow but superior than ANN	Massively parallel, fast but inferior than BNN
Size	1011 neurons and 1015 interconnections	102 to 104 nodes (mainly depends on the type of application and network designer)
Learning	They can tolerate ambiguity	Very precise, structured and formatted data is required to tolerate ambiguity
Fault tolerance	Performance degrades with even partial damage	It is capable of robust performance, hence has the potential to be fault tolerant
Storage capacity	Stores the information in the synapse	Stores the information in continuous memory locations

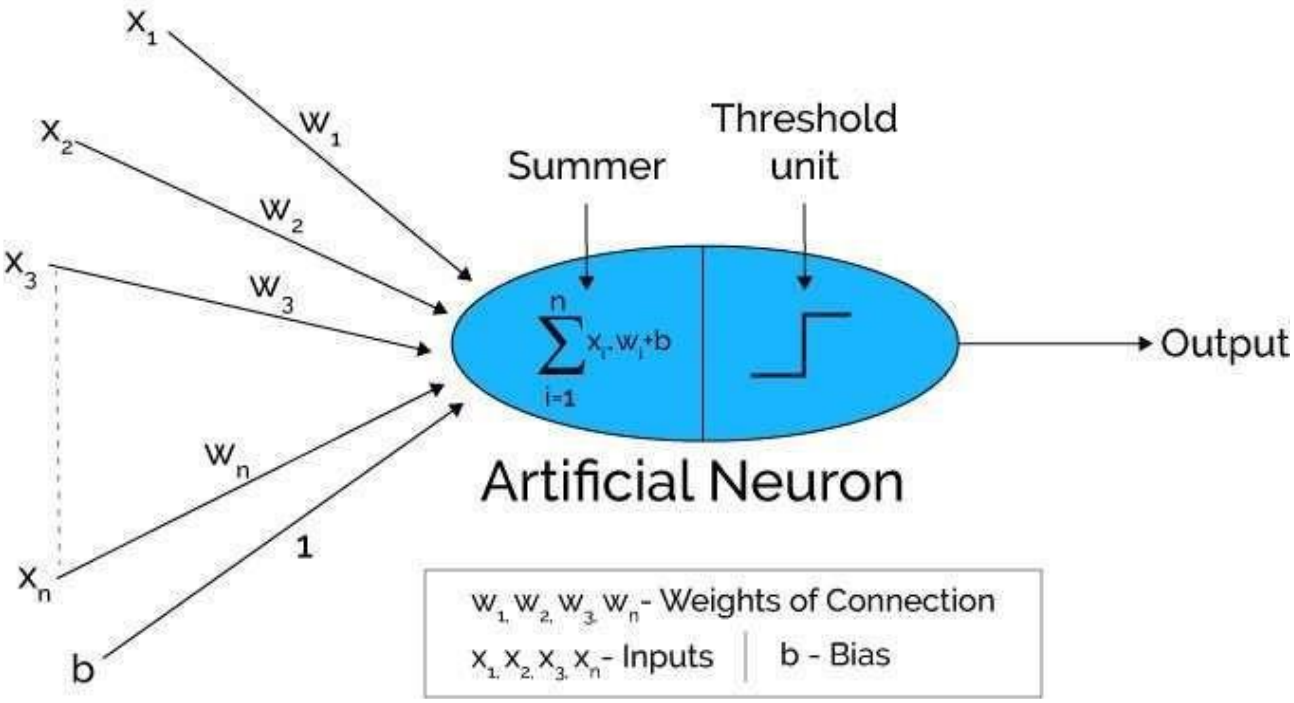


Analogy of ANN with BNN



- The dendrites in biological neural network is analogous to the weighted inputs based on their synaptic interconnection in artificial neural network.
- Cell body is analogous to the artificial neuron unit in artificial neural network which also comprises of summation and threshold unit.
- Axon carry output that is analogous to the output unit in case of artificial neural network. So, ANN are modelled using the working of basic biological neurons.

Model of Artificial Neural Network

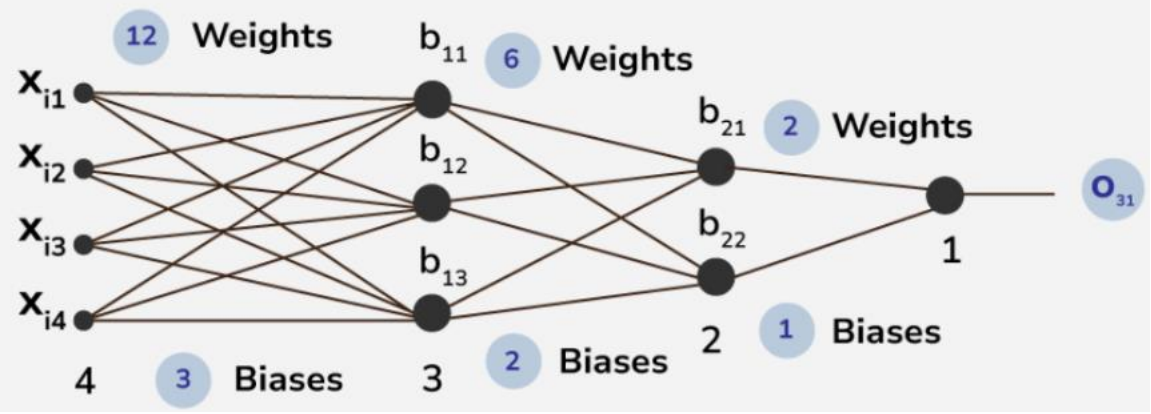


- Artificial neural networks can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges with weights are connections between neuron outputs and neuron inputs.
- The Artificial Neural Network receives input from the external world in the form of pattern and image in vector form. These inputs are mathematically designated by the notation x(n) for n number of inputs.
- Each input is multiplied by its corresponding weights. Weights are the information used by the neural network to solve a problem. Typically, weight represents the strength of the interconnection between neurons inside the neural network.
- The weighted inputs are all summed up inside computing unit (artificial neuron). In case the weighted sum is zero, bias is added to make the output not- zero or to scale up the system response. Bias has the weight and input always equal to '1'.
- The sum corresponds to any numerical value ranging from 0 to infinity.
- In order to limit the response to arrive at desired value, the threshold value is set up. For this, the sum is passed through activation function.
- The activation function is set of the transfer function used to get desired output. There are linear as well as the non-linear activation function.
- Some of the commonly used activation function are — binary, sigmoidal (linear) and tan hyperbolic sigmoidal functions(nonlinear).
- Binary — The output has only two values either 0 and 1. For this, the threshold value is set up. If the net weighted input is greater than 1, an output is assumed 1 otherwise zero.
- Sigmoidal Hyperbolic — This function has 'S' shaped curve. Here tan hyperbolic function is used to approximate output from net input. The function is defined as — f (x) = (1/1+ exp(-σx)) where σ— steepness parameter.

Training Neural Network

1. **Forward propagation**
   Forward propagation is the process in a neural network where the input data is passed through the network's layers to generate an output. It involves the following steps:
   - Input Layer: The input data is fed into the input layer of the neural network.
   - Hidden Layers: The input data is processed through one or more hidden layers. Each neuron in a hidden layer receives inputs from the previous layer, applies an activation function to the weighted sum of these inputs, and passes the result to the next layer.
   - Output Layer: The processed data moves through the output layer, where the final output of the network is generated. The output layer typically applies an activation function suitable for the task, such as softmax for classification or linear activation for regression.
   - Prediction: The final output of the network is the prediction or classification result for the input data.

Forward propagation is essential for making predictions in neural networks. It calculates the output of the network for a given input based on the current values of the weights and biases. The output is then compared to the actual target value to calculate the loss, which is used to update the weights and biases during the training process.



2. Loss function

The **loss function** is a critical component in neural network training. It measures the difference between the predicted output $(\hat{y})$ of the network and the true target values (y). The goal of training is to minimize the loss function, which ensures the network learns the correct patterns from the data.

i. **Regression Tasks**

For predicting continuous values:

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

  - Penalizes larger errors more than smaller ones.
  - Common in regression tasks.

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

  - Less sensitive to outliers than MSE.

ii. **Classification Tasks**

For predicting discrete categories:

- **Binary Cross-Entropy (Log Loss)** (for binary classification):

$$\text{Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

  - Used when the output layer has a sigmoid activation function.

- **Categorical Cross-Entropy** (for multi-class classification):

$$\text{Loss} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log(\hat{y}_{ij})$$

  - Used with a softmax activation in the output layer.
  - k is the number of classes.

3. Backpropagation

Training a neural network using **backpropagation** involves several key steps. Backpropagation is a supervised learning algorithm and relies on the gradient descent optimization method to minimize the error or loss.
Steps:

i. **Initialize the Network**

- **Define the architecture**: Choose the number of input nodes, hidden layers, neurons in each layer, and output nodes.
- **Initialize weights and biases**: Assign small random values to the weights and biases in the network.

ii. Feedforward Propagation

- Compute the output of each neuron in the network layer by layer:
  - **Linear Combination**: $z = W \cdot X + b$
  - **Activation**: $a = f(z)$, where f is the activation function (e.g., sigmoid, ReLU, tanh).
- Continue until the output layer is reached, producing predictions ($\hat{y}$).

iii. **Compute the loss**

- Compare the network's predictions ($\hat{y}$) with the true values (y).
- Use a loss function like:
  - Mean Squared Error (MSE) for regression: $\text{Loss} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
  - Cross-Entropy Loss for classification tasks.

iv. **Backpropagation**

Backpropagation calculates the gradients of the loss function with respect to the weights and biases using the **chain rule**.

1. **Compute Gradients for the Output Layer**:
   - Calculate the error at the output layer.
   - $\delta^{(L)} = \nabla_a \text{Loss} \cdot f'(z^{(L)})$
   - $f'(z^{(L)})$ is the derivative of the activation function in the output layer.
2. **Propagate the Error Backward**:
   - For each layer l, compute the error $\delta^{(l)}$: $\delta^{(l)} = (\delta^{(l+1)} \cdot W^{(l+1)}) \cdot f'(z^{(l)})$
3. **Compute Weight and Bias Gradients**:
   - Weight gradient: $\frac{\partial \text{Loss}}{\partial W^{(l)}} = a^{(l-1)} \cdot \delta^{(l)}$
   - Bias gradient: $\frac{\partial \text{Loss}}{\partial b^{(l)}} = \delta^{(l)}$

v. **Update Weights and Biases**

- Use an optimization algorithm (e.g., **Gradient Descent**) to update parameters:
  - $W^{(l)} = W^{(l)} - \eta \cdot \frac{\partial \text{Loss}}{\partial W^{(l)}}$
  - $b^{(l)} = b^{(l)} - \eta \cdot \frac{\partial \text{Loss}}{\partial b^{(l)}}$
- η is the learning rate.

vi. **Repeat for Multiple Epochs**

Perform steps 2-5 for all training data over several epochs until the loss converges or a stopping criterion is met.

4. Learning Rate

- The **learning rate** (η) is a crucial hyper-parameter in training a neural network. It controls how much the weights and biases of the network are updated during each step of optimization.
- Determines the size of the steps taken in the direction of the negative gradient of the loss function.
- Affects the speed and quality of convergence.

Mathematical Representation

In gradient descent, weights (W) and biases (b) are updated as follows:

$$W = W - \eta \cdot \frac{\partial \text{Loss}}{\partial W}$$

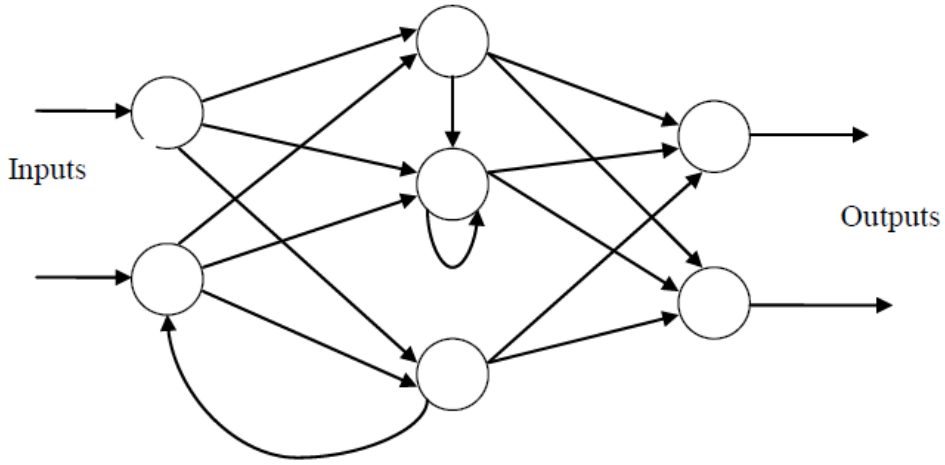
$$b = b - \eta \cdot \frac{\partial \text{Loss}}{\partial b}$$

Where:

- η: Learning rate
- $\frac{\partial \text{Loss}}{\partial W}$: Gradient of the loss with respect to the weights.

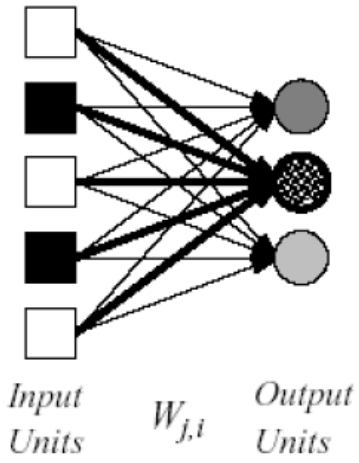
Recurrent neural network (RNN)

Feedback networks (figure 1) can have signals traveling in both directions
by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent.



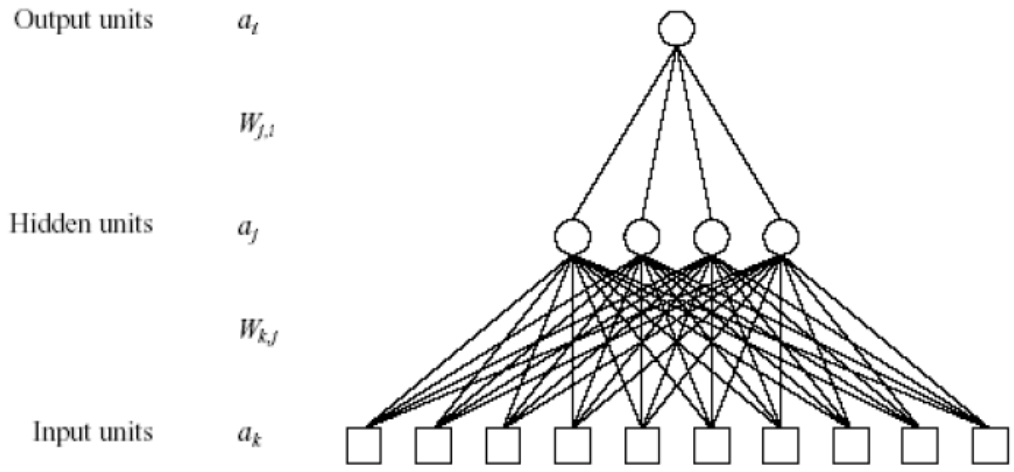
Single layered neural network

A neural network in which all the inputs connected directly to the outputs is called a single-layer neural network, or a perceptron network. Since each output unit is independent of the others each weight affects only one of the outputs.



Multilayered neural network

The neural network which contains input layers, output layers and some hidden layers also is called multilayer neural network. The advantage of adding hidden layers is that it enlarges the space of hypothesis. Layers of the network are normally fully connected.



Application of neural network

i. Speech Recognition

Speech occupies a prominent role in human-human interaction. Therefore, it is natural for people to expect speech interfaces with computers. In the present era, for communication with machines, humans still need sophisticated languages which are difficult to learn and use. To ease this communication barrier, a simple solution could be, communication in a spoken language that is possible for the machine to understand.

ii. Character Recognition

It is an interesting problem which falls under the general area of Pattern Recognition. Many neural networks have been developed for automatic recognition of handwritten characters, either letters or digits. Following are some ANNs which have been used for character recognition −

- Multilayer neural networks such as Backpropagation neural networks.

- Neocognitron

iii. Signature Verification Application

Signatures are one of the most useful ways to authorize and authenticate a person in legal transactions. Signature verification technique is a non-vision based technique.

For this application, the first approach is to extract the feature or rather the geometrical feature set representing the signature. With these feature sets, we have to train the neural networks using an efficient neural network algorithm. This trained neural network will classify the signature as being genuine or forged under the verification stage.

iv. Human Face Recognition

It is one of the biometric methods to identify the given face. It is a typical task because of the characterization of "non-face" images. However, if a neural network is well trained, then it can be divided into two classes namely images having faces and images that do not have faces.

First, all the input images must be preprocessed. Then, the dimensionality of that image must be reduced. And, at last it must be classified using neural network training algorithm.

Supervised vs. unsupervised learning

	SUPERVISED LEARNING	UNSUPERVISED LEARNING
Input Data	Uses Known and Labeled Data as input	Uses Unknown Data as input
Computational Complexity	Very Complex	Less Computational Complexity
Real Time	Uses off-line analysis	Uses Real Time Analysis of Data
Number of Classes	Number of Classes are known	Number of Classes are not known
Accuracy of Results	Accurate and Reliable Results	Moderate Accurate and Reliable Results

Learning by training ANN

Hebbian learning

From the point of view of artificial neurons and artificial neural networks, Hebb's principle can be described as a method of determining how to alter the weights between model neurons. **The weight between two neurons increases if the two neurons activate simultaneously—and reduces if they activate separately.** Nodes that

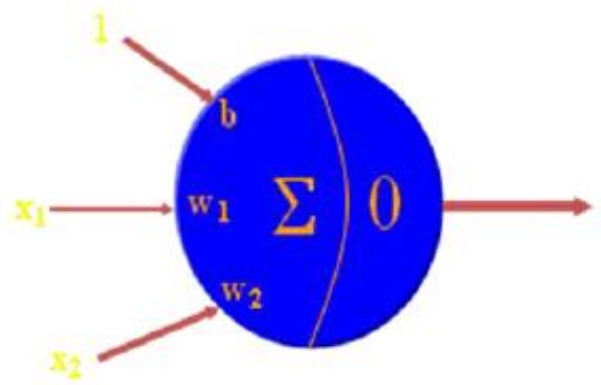
tend to be either both positive or both negative at the same time have strong positive weights, while those that tend to be opposite have strong negative weights.

Hebb's Algorithm:

Step 0: initialize all weights to 0
Step 1: Given a training input, s, with its target output, t, set the activations of the input units: $x_i = s_i$
Step 2: Set the activation of the output unit to the target value: y = t
Step 3: Adjust the weights: $w_i(new) = w_i(old) + x_i y$
Step 4: Adjust the bias (just like the weights): b(new) = b(old) + y

PROBLEM: Construct a Hebb Net which performs like an AND function, that is, only when both features are "active" will the data be in the target class.
TRAINING SET (with the bias input always at 1):

x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



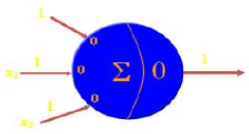
Training-First Input:

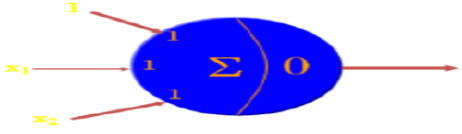
- Initialize the weights to 0

Present the first input:
(1 1 1) with a target of 1

Update the weights:

$w_1(new) = w_1(old) + x_1 t$
$= 0 + 1 = 1$
$w_2(new) = w_2(old) + x_2 t$
$= 0 + 1 = 1$
$b(new) = b(old) + t$
$= 0 + 1 = 1$





Training second input

- **Present the second input: (1 -1 1) with a target of -1**

Update the weights:

$w_1(new) = w_1(old) + x_1 t$

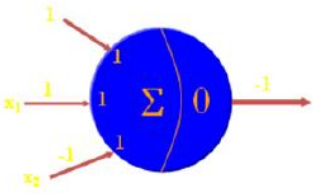
$= 1 + 1(-1) = 0$

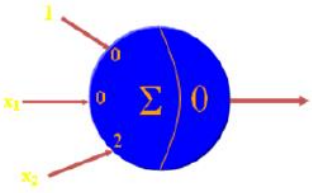
$w_2(new) = w_2(old) + x_2 t$

$= 1 + (-1)(-1) = 2$

$b(new) = b(old) + t$

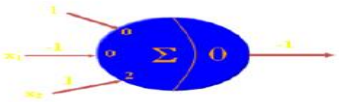
$= 1 + (-1) = 0$





Training third input

- **Present the third input: (-1 1 1) with a target of -1**



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

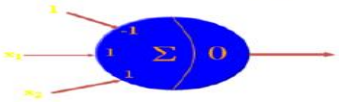
$= 0 + (-1)(-1) = 1$

$w_2(new) = w_2(old) + x_2 t$

$= 2 + 1(-1) = 1$

$b(new) = b(old) + t$

$= 0 + (-1) = -1$



Training fourth input

- **Present the fourth input: (-1 -1 1) with a target of -1**

Update the weights:

$w_1(new) = w_1(old) + x_1 t$

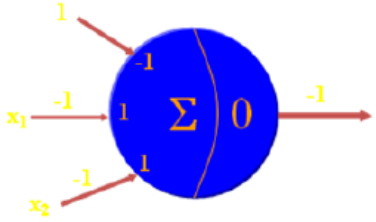
$= 1 + (-1)(-1) = 2$

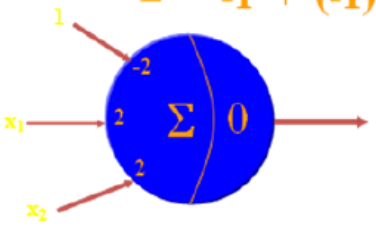
$w_2(new) = w_2(old) + x_2 t$

$= 1 + (-1)(-1) = 1$

$b(new) = b(old) + t$

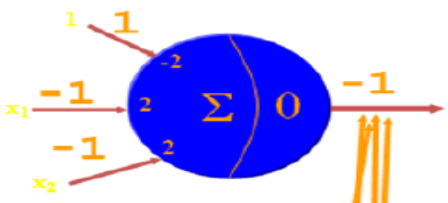
$= -1 + (-1) = -2$





Final neuron

x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
=1	1	1	=1
=1	=1	1	=1



$1*2 + 1*2 + 1*(-2) = 2 > 0$

$(-1)*2 + 1*2 + 1*(-2) = -2 < 0$

$1*2 + (-1)*2 + 1*(-2) = -2 < 0$

$(-1)*2 + (-1)*2 + 1*(-2) = -6 < 0$

Unsupervised learning

Unsupervised learning is a type of machine learning where the goal is to find patterns or structure in data without using labeled examples. Unlike supervised learning, where the model is trained on a dataset that includes input-output pairs (like images labeled with the objects they contain), unsupervised learning works with data that has no explicit labels or categories.

Example

You are given a bag of marbles of different colors, sizes, and textures. No one tells you what the categories are or what to do. Your job is to figure out a way to group the marbles based on their similarities, such as by color, size, or texture. Here,

Data = marbles (with no labels).
Model's goal = group similar marbles together.

Types of Unsupervised Learning:

1. Clustering: Grouping data into clusters based on similarity.
    - Example: Organizing customers into groups based on purchasing behavior.
    - Algorithms: K-Means, DBSCAN (Density-based spatial clustering of applications with noise).
2. Dimensionality Reduction: Simplifying data while retaining its important information.
    - Example: Summarizing a 1000-word essay into a few key sentences.
    - Algorithms: PCA (Principal Component Analysis), linear discrimant analysis.

K-means Clustering Algorithm

K-means clustering is an unsupervised machine learning algorithm used to group a dataset into k clusters. It is an iterative algorithm that starts by randomly selecting k centroids in the dataset. After selecting the centroids, the entire dataset is divided into clusters based on the distance of the data points from the centroid. In the new clusters, the centroids are calculated by taking the mean of the data points.

With the new centroids, we regroup the dataset into new clusters. This process continues until we get a stable cluster. K-means clustering is a partition clustering algorithm. We call it partition clustering because of the reason that the k-means clustering algorithm partitions the entire dataset into mutually exclusive clusters.

Given a dataset of N entries and a number K as the number of clusters that need to be formed, we will use the following steps to find the clusters using the k-means algorithm.

1. First, we will select K random entries from the dataset and use them as centroids.
2. Now, we will find the distance of each entry in the dataset from the centroids. You can use any distance metric such as Euclidean distance, Manhattan distance, or squared Euclidean distance.
3. After finding the distance of each data entry from the centroids, we will start assigning the data points to clusters. We will assign each data point to the cluster with the centroid to which it has the least distance.
4. After assigning the points to clusters, we will calculate the new centroid of the clusters. For this, we will use the mean of each data point in the same cluster as the new centroid. If the newly created centroids are the same as the centroids in the previous iteration, we will consider the current clusters to be final. Hence, we will stop the execution of the algorithm. If any of the newly created centroids is different from the centroids in the previous iteration, we will go to step 2.

K-means Clustering Numerical Example

You are given 15 points in the Cartesian coordinate system as follows with k=3.

Point	Coordinates
A1	(2,10)
A2	(2,6)

A3	(11,11)
A4	(6,9)
A5	(6,4)
A6	(1,2)
A7	(5,10)
A8	(4,9)
A9	(10,12)
A10	(7,5)
A11	(9,11)
A12	(4,6)
A13	(3,10)
A14	(3,8)
A15	(6,11)

First, we will randomly choose 3 centroids from the given data. Let us consider A2 (2,6), A7 (5,10), and A15 (6,11) as the centroids of the initial clusters. Hence, we will consider that

- Centroid 1= (2,6) is associated with cluster 1.
- Centroid 2= (5,10) is associated with cluster 2.
- Centroid 3= (6,11) is associated with cluster 3.

Now we will find the Euclidean distance between each point and the centroids. Based on the minimum distance of each point from the centroids, we will assign the points to a cluster. I have tabulated the distance of the given points from the clusters in the following table.

Point	Distance from Centroid 1 (2,6)	Distance from Centroid 2 (5,10)	Distance from Centroid 3 (6,11)	Assigned Cluster
A1 (2,10)	4	3	4.123106	Cluster 2
A2 (2,6)	0	5	6.403124	Cluster 1

A3 (11,11)	10.29563	6.082763	5	Cluster 3
A4 (6,9)	5	1.414214	2	Cluster 2
A5 (6,4)	4.472136	6.082763	7	Cluster 1
A6 (1,2)	4.123106	8.944272	10.29563	Cluster 1
A7 (5,10)	5	0	1.414214	Cluster 2
A8 (4,9)	3.605551	1.414214	2.828427	Cluster 2
A9 (10,12)	10	5.385165	4.123106	Cluster 3
A10 (7,5)	5.09902	5.385165	6.082763	Cluster 1
A11 (9,11)	8.602325	4.123106	3	Cluster 3
A12 (4,6)	2	4.123106	5.385165	Cluster 1
A13 (3,10)	4.123106	2	3.162278	Cluster 2
A14 (3,8)	2.236068	2.828427	4.242641	Cluster 1
A15 (6,11)	6.403124	1.414214	0	Cluster 3

At this point, we have completed the first iteration of the k-means clustering algorithm and assigned each point into a cluster.

In the above table, we can observe that the point that is closest to the centroid of a given cluster is assigned to the cluster.

Now, we will calculate the new centroid for each cluster.

- In cluster 1, we have 6 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), A12 (4,6), A14 (3,8). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (3.833, 5.167).

- In cluster 2, we have 5 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), and A13 (3,10). Hence, the new centroid for cluster 2 is (4, 9.6)
- In cluster 3, we have 4 points i.e. A3 (11,11), A9 (10,12), A11 (9,11), and A15 (6,11). Hence, the new centroid for cluster 3 is (9, 11.25).

Now that we have calculated new centroids for each cluster, we will calculate the distance of each data point from the new centroids. Then, we will assign the points to clusters based on their distance from the centroids. The results for this process have been given in the following table.

Point	Distance from Centroid 1 (3.833, 5.167)	Distance from centroid 2 (4, 9.6)	Distance from centroid 3 (9, 11.25)	Assigned Cluster
A1 (2,10)	5.169	2.040	7.111	Cluster 2
A2 (2,6)	2.013	4.118	8.750	Cluster 1
A3 (11,11)	9.241	7.139	2.016	Cluster 3
A4 (6,9)	4.403	2.088	3.750	Cluster 2
A5 (6,4)	2.461	5.946	7.846	Cluster 1
A6 (1,2)	4.249	8.171	12.230	Cluster 1
A7 (5,10)	4.972	1.077	4.191	Cluster 2
A8 (4,9)	3.837	0.600	5.483	Cluster 2
A9 (10,12)	9.204	6.462	1.250	Cluster 3
A10 (7,5)	3.171	5.492	6.562	Cluster 1
A11 (9,11)	7.792	5.192	0.250	Cluster 3
A12 (4,6)	0.850	3.600	7.250	Cluster 1
A13 (3,10)	4.904	1.077	6.129	Cluster 2
A14 (3,8)	2.953	1.887	6.824	Cluster 2

A15 (6,11)	6.223	2.441	3.010	Cluster 2

Now, we have completed the second iteration of the k-means clustering algorithm and assigned each point into an updated cluster. In the above table, you can observe that the point closest to the new centroid of a given cluster is assigned to the cluster.

Now, we will calculate the new centroid for each cluster for the third iteration.

- In cluster 1, we have 5 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), and A12 (4,6). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (4, 4.6).
- In cluster 2, we have 7 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), A13 (3,10), A14 (3,8), and A15 (6,11). Hence, the new centroid for cluster 2 is (4.143, 9.571)
- In cluster 3, we have 3 points i.e. A3 (11,11), A9 (10,12), and A11 (9,11). Hence, the new centroid for cluster 3 is (10, 11.333).

At this point, we have calculated new centroids for each cluster. Now, we will calculate the distance of each data point from the new centroids. Then, we will assign the points to clusters based on their distance from the centroids. The results for this process have been given in the following table.

Point	Distance from Centroid 1 (4, 4.6)	Distance from centroid 2 (4.143, 9.571)	Distance from centroid 3 (10, 11.333)	Assigned Cluster
A1 (2,10)	5.758	2.186	8.110	Cluster 2
A2 (2,6)	2.441	4.165	9.615	Cluster 1
A3 (11,11)	9.485	7.004	1.054	Cluster 3
A4 (6,9)	4.833	1.943	4.631	Cluster 2
A5 (6,4)	2.088	5.872	8.353	Cluster 1
A6 (1,2)	3.970	8.197	12.966	Cluster 1
A7 (5,10)	5.492	0.958	5.175	Cluster 2
A8 (4,9)	4.400	0.589	6.438	Cluster 2
A9 (10,12)	9.527	6.341	0.667	Cluster 3
A10 (7,5)	3.027	5.390	7.008	Cluster 1
A11 (9,11)	8.122	5.063	1.054	Cluster 3

A12 (4,6)	1.400	3.574	8.028	Cluster 1
A13 (3,10)	5.492	1.221	7.126	Cluster 2
A14 (3,8)	3.544	1.943	7.753	Cluster 2
A15 (6,11)	6.705	2.343	4.014	Cluster 2

Now, we will calculate the new centroid for each cluster for the third iteration.

- In cluster 1, we have 5 points i.e. A2 (2,6), A5 (6,4), A6 (1,2), A10 (7,5), and A12 (4,6). To calculate the new centroid for cluster 1, we will find the mean of the x and y coordinates of each point in the cluster. Hence, the new centroid for cluster 1 is (4, 4.6).
- In cluster 2, we have 7 points i.e. A1 (2,10), A4 (6,9), A7 (5,10) , A8 (4,9), A13 (3,10), A14 (3,8), and A15 (6,11). Hence, the new centroid for cluster 2 is (4.143, 9.571)
- In cluster 3, we have 3 points i.e. A3 (11,11), A9 (10,12), and A11 (9,11). Hence, the new centroid for cluster 3 is (10, 11.333).

Here, you can observe that no point has changed its cluster compared to the previous iteration. Due to this, the centroid also remains constant. Therefore, we will say that the clusters have been stabilized. Hence, the clusters obtained after the third iteration are the final clusters made from the given dataset.

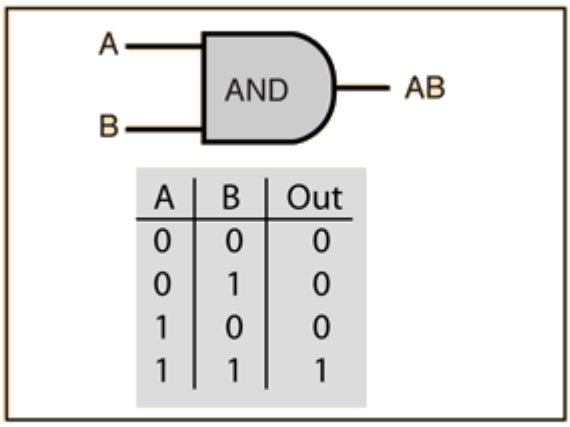
Training Neural Network

Perceptron Learning Theory:

The term "Perceptron" was coined by Frank Rosen Blatt in 1962 and is used to describe the connection of simple neurons into networks. These networks are simplified versions of the real nervous system where some properties are exaggerated and others are ignored. For the moment we will concentrate on Single Layer Perceptron.
So how can we achieve learning in our model neuron? We need to train them so they can do things that are useful. To do this we must allow the neuron to learn from its mistakes. There is in fact a learning paradigm that achieves this, it is known as supervised learning and works in the following
manner.
i. set the weight and thresholds of the neuron to random values.
ii. present an input.
iii. calculate the output of the neuron.

iv. alter the weights to reinforce correct decisions and discourage wrong decisions, hence reducing the error. So for the network to learn we shall increase the weights on the active inputs when we want the output to be active, and to decrease them when we want the output to be inactive.
v. Now present the next input and repeat steps iii. - v.

Training AND gate logic using perceptron



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate

First, we need to understand that the output of an AND gate is 1 only if both inputs (in this case, x1 and x2) are 1. So, following the steps listed above.

Row 1

- From w1x1+w2x2+b, initializing w1, w2, as 1 and b as -1, we get;

x1(1)+x2(1)-1

- Passing the first row of the AND logic table (x1=0, x2=0), we get;

0+0–1 = -1

- From the Perceptron rule, if Wx+b<0, then y`=0. Therefore, this row is correct, and no need for Backpropagation.

Row 2

- Passing (x1=0 and x2=1), we get;

0+1–1 = 0

- From the Perceptron rule, if Wx+b >= 0, then y`=1. This row is incorrect, as the output is 0 for the AND gate.
- So we want values that will make the combination of x1=0 and x2=1 to give y` a value of 0. If we change b to -1.5, we have;

0+1–1.5 = -0.5

- From the Perceptron rule, this works (for both row 1, row 2 and 3).

Row 4

- Passing (x1=1 and x2=1), we get;

1+1–1.5 = 0.5

- Again, from the perceptron rule, this is still valid.

Therefore, we can conclude that the model to achieve an AND gate, using the Perceptron algorithm is;

x1+x2–1.5