Neural Networks

Biological neural network vs Artificial neural network

Features	Artificial Neural Network	Biological Neural Network
Definition	It is the mathematical model which is mainly inspired by the biological neuron system in the human brain.	It is also composed of several processing pieces known as neurons that are linked together via synapses.
Processing	Its processing was sequential and centralized.	It processes the information in a parallel and distributive manner.
Size	It is small in size.	It is large in size.
Control Mechanism	Its control unit keeps track of all computer-related operations.	All processing is managed centrally.
Rate	It processes the information at a faster speed.	It processes the information at a slow speed.

Complexity	It cannot perform complex pattern recognition.	The large quantity and complexity of the connections allow the brain to perform complicated tasks.
Feedback	It doesn't provide any feedback.	It provides feedback.
Fault tolerance	There is no fault tolerance.	It has fault tolerance.
Operating Environment	Its operating environment is well-defined and well-constrained	Its operating environment is poorly defined and unconstrained.
Memory	Its memory is separate from a processor, localized, and non-content addressable.	Its memory is integrated into the processor, distributed, and content-addressable.
Reliability	It is very vulnerable.	It is robust.
Learning	It has very accurate structures and formatted data.	They are tolerant to ambiguity.
Response time	Its response time is measured in milliseconds.	Its response time is measured in nanoseconds.

Perceptron

- Perceptron is one of the simplest Artificial neural network architectures.
- It was introduced **by Frank Rosenblatt in 1957s**.
- It is the simplest type of **feedforward neural network**, consisting of a single layer of input nodes that are fully connected to a layer of output nodes.
- It can learn the linearly separable patterns.
- **It uses slightly different types of artificial neurons known as threshold logic units (TLU).**
- It was first introduced by McCulloch and Walter Pitts in the 1940s.

Types of Perceptron

- **Single-Layer Perceptron**
  - This type of perceptron is limited to learning linearly separable patterns effective for tasks where the data can be divided into distinct categories through a straight line.

- **Multilayer Perceptron**
  - Multilayer perceptrons possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

Basic Components of Perceptron

- A perceptron, the basic unit of a neural network, comprises essential components that collaborate in information processing.
- **Input Features**
  - The perceptron takes multiple input features, each input feature represents a characteristic or attribute of the input data.
- **Weights**
  - Each input feature is associated with a weight, determining the significance of each input feature in influencing the perceptron’s output.
  - During training, these weights are adjusted to learn the optimal values.

- **Summation Function**
  - The perceptron calculates the weighted sum of its inputs using the summation function. The summation function combines the inputs with their respective weights to produce a weighted sum.

- **Activation Function**
  - The weighted sum is then passed through an activation function.
  - Perceptron uses Heaviside step function functions.
  - which take the summed values as input and compare with the threshold and provide the output as 0 or 1.
- **Output**
  - The final output of the perceptron, is determined by the activation function's result.
  - For example, in binary classification problems, the output might represent a predicted class (0 or 1).
- **Bias**
  - A bias term is often included in the perceptron model.
  - The bias allows the model to make adjustments that are independent of the input.
  - It is an additional parameter that is learned during training.

- **Learning Algorithm (Weight Update Rule)**
  - During training, the perceptron learns by adjusting its weights and bias based on a learning algorithm.
  - A common approach is the perceptron learning algorithm, which updates weights based on the difference between the predicted output and the true output.

Learning Rate

- The learning rate in neural network training is a crucial hyperparameter that determines the size of weight updates during optimization.
- It dictates the step size taken in the direction of the negative gradient during backpropagation.
- An optimal learning rate is essential for achieving high model performance.

- Too low a learning rate can lead to slow convergence and getting stuck in local optima, while too high a learning rate may cause overshooting of the ideal solution.
- Therefore, selecting an appropriate learning rate is critical for efficient and effective model training.

Impact of learning rate

- In neural network training, the optimization algorithm adjusts the network's weights based on the gradients of the loss function.
- The learning rate controls the size of these weight updates, impacting convergence speed and solution quality.
- A high learning rate can lead to overshooting and unstable training, while a low learning rate may cause slow convergence or getting stuck in local minima.

- Issues might arise for the model if the learning rate is too high −
  - **Oscillations** − If the weights update too quickly, the model could fluctuate around the ideal result.
  - **Divergence** − If the weights update too quickly, the model may depart from the ideal outcome.
  - **Bad performance** − The model could reach a less-than-ideal solution, which would result in poor performance.
- Issues might arise for the model if the learning rate is too low −
  - Slow convergence might result in sluggish training because the optimization technique may take too long to find a solution.
  - Poor performance might result from the optimization method becoming trapped at a local minimum.

Gradient descent

- Gradient descent is an optimization algorithm used to minimize the loss function of a model by iteratively updating the model parameters (weights) in the direction that reduces the loss.
- The basic idea is to adjust the parameters in proportion to the negative of the gradient of the loss function with respect to the parameters.

How gradient descent works:

- **Initialization**: The algorithm starts with an initial set of parameters (weights) for the model.
- **Compute Loss**: The loss function is evaluated using the current set of parameters on a training dataset.
- **Compute Gradient**: The gradient of the loss function with respect to each parameter is computed. This gradient indicates the direction of steepest ascent in the parameter space.

- **Update Parameters**: The parameters are updated by moving in the opposite direction of the gradient. This update is scaled by a factor called the learning rate, which determines the size of the step taken in parameter space.
- **Iterate**: Steps 2-4 are repeated iteratively until a stopping criterion is met, such as reaching a maximum number of iterations or achieving a desired level of loss.

Gradient descent comes in different variants, including:

- **Batch Gradient Descent**: Computes the gradient using the entire training dataset.
- **Stochastic Gradient Descent (SGD)**: Computes the gradient using only one randomly chosen sample from the training dataset at each iteration.
- **Mini-batch Gradient Descent**: Computes the gradient using a small random subset (mini-batch) of the training dataset.

Delta Learning Rule

- It was developed by Bernard Widrow and Marcian Hoff and It depends on supervised learning and has a continuous activation function.
- It is also known as the Least Mean Square method and it minimizes error over all the training patterns.
- It is based on a gradient descent approach which continues forever.
- It states that the modification in the weight of a node is equal to the product of the error and the input where the error is the difference between desired and actual output.

Hebbian Learning Rule

- Donald Hebb developed it in 1949 as an unsupervised learning algorithm in the neural network.
- We can use it to improve the weights of nodes of a network.

The following phenomenon occurs when

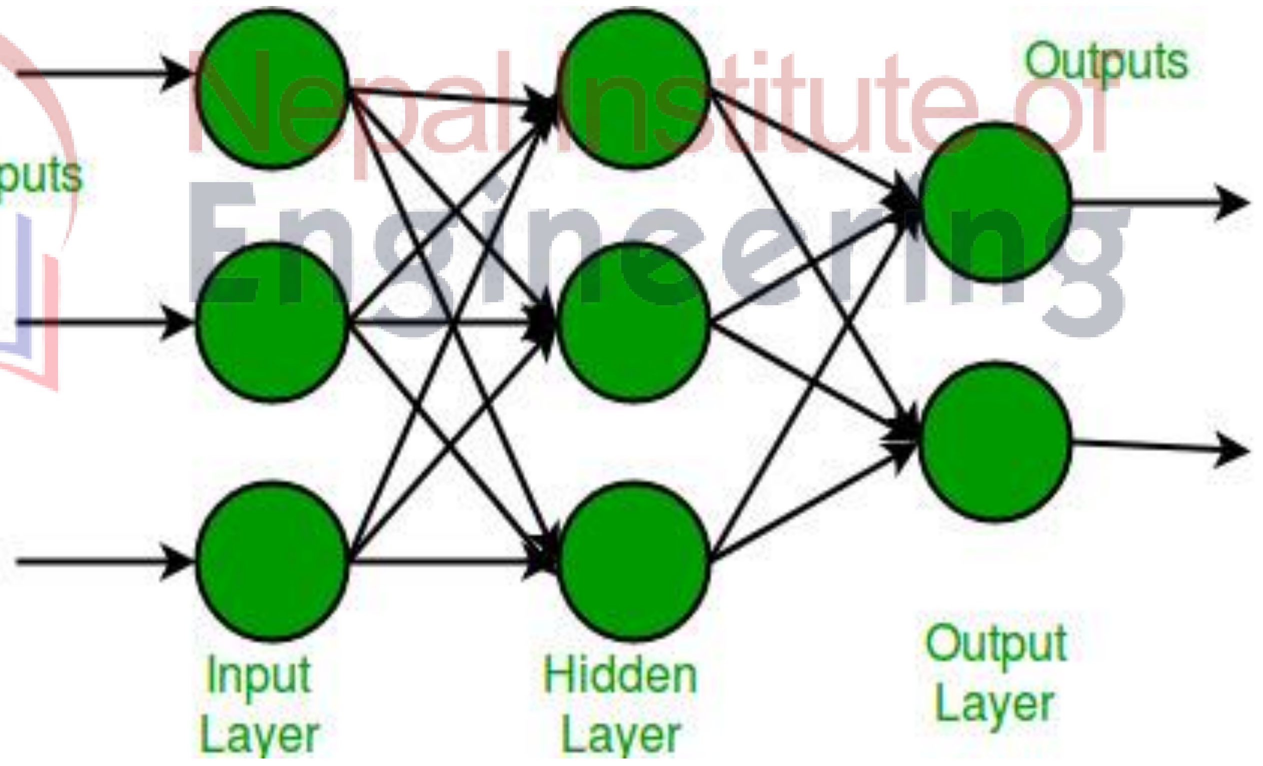
- If two neighbor neurons are operating in the same phase at the same period of time, then the weight between these neurons should increase.
- For neurons operating in the opposite phase, the weight between them should decrease.
- If there is no signal correlation, the weight does not change, the sign of the weight between two nodes depends on the sign of the input between those nodes

- When inputs of both the nodes are either positive or negative, it results in a strong positive weight.
- If the input of one node is positive and negative for the other, a strong negative weight is present.

Multi-layer Perceptron

- Multi-layer perception is also known as MLP.
- It is fully connected dense layers, which transform any input dimension to the desired dimension.
- A multi-layer perception is a neural network that has multiple layers.
- To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons.
- A gentle introduction to **neural networks and TensorFlow** can be found here:
  - Neural Networks
  - Introduction to TensorFlow

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



The Hopfield Neural Networks

- Invented by Dr John J. Hopfield consists of one layer of *'n'* fully connected recurrent neurons.
- It is generally used in performing auto-association and optimization tasks.
- It is calculated using a converging interactive process and it generates a different response than our normal neural nets.

Discrete Hopfield Network

- It is a fully interconnected neural network where each unit is connected to every other unit.
- It behaves in a discrete manner, i.e. it gives finite distinct output, generally of two types:
  - **Binary (0/1)**
  - **Bipolar (-1/1)**

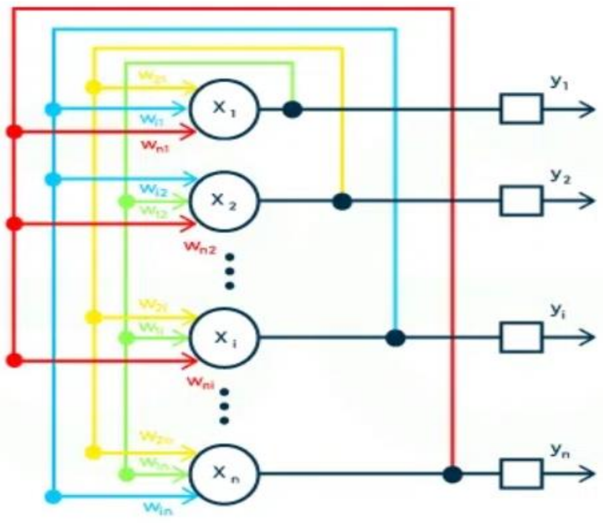
- The weights associated with this network are symmetric in nature and have the following properties.

  $wij=wji2$

  $wii=0$

Structure & Architecture of Hopfield Network

- **Each neuron has an inverting and a non-inverting output.**
- **Being fully connected, the output of each neuron is an input to all other neurons but not the self.**
- The below figure shows a sample representation of a Discrete Hopfield Neural Network architecture having the following elements.



Backpropagation

- Backpropagation is a widely used algorithm for training feedforward neural networks.
- It computes the gradient of the loss function with respect to the network weights.
- It is very efficient, rather than naively directly computing the gradient concerning each weight.
- This efficiency makes it possible to use gradient methods to train multi-layer networks and update weights to minimize loss; variants such as gradient descent or stochastic gradient descent are often used.
- The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight via the chain rule, computing the gradient layer by layer, and iterating backward from the last layer to avoid redundant computation of intermediate terms in the chain rule.

Features of Backpropagation

- It is the gradient descent method as used in the case of simple perceptron network with the differentiable unit.
- it is different from other networks in respect to the process by which the weights are calculated during the learning period of the network.
- Training is done in the three stages:
  - the feed-forward of input training pattern
  - the calculation and backpropagation of the error
  - updation of the weight

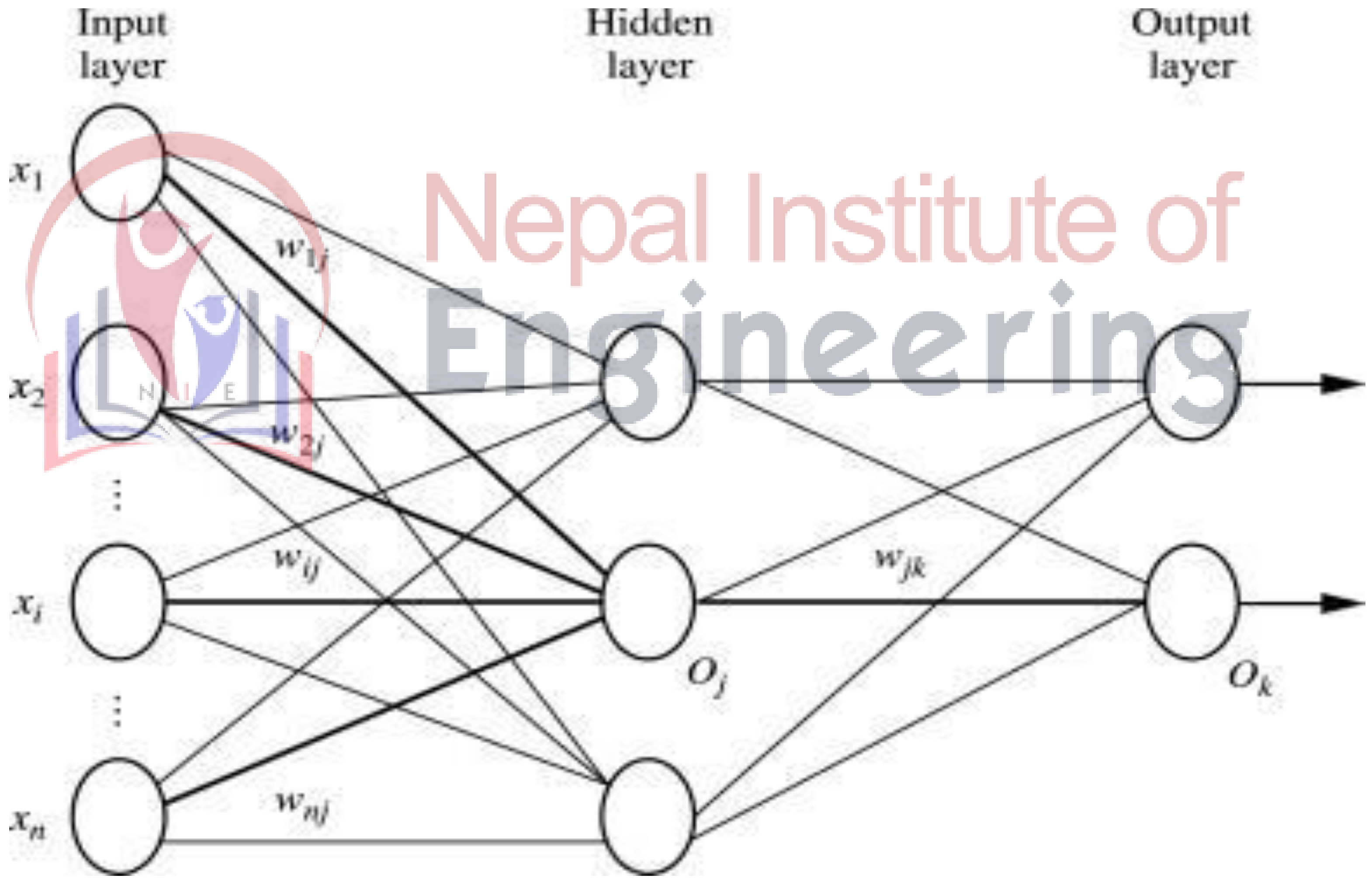
Working of Backpropagation

- Neural networks use supervised learning to generate output vectors from input vectors that the network operates on.
- It Compares generated output to the desired output and generates an error report if the result does not match the generated output vector.
- Then it adjusts the weights according to the bug report to get your desired output.

Backpropagation Algorithm:

- **Step 1:** Inputs X, arrive through the preconnected path.
- **Step 2:** The input is modeled using true weights W. Weights are usually chosen randomly.
- **Step 3:** Calculate the output of each neuron from the input layer to the hidden layer to the output layer.
- **Step 4:** Calculate the error in the outputs
- Backpropagation Error= Actual Output – Desired Output

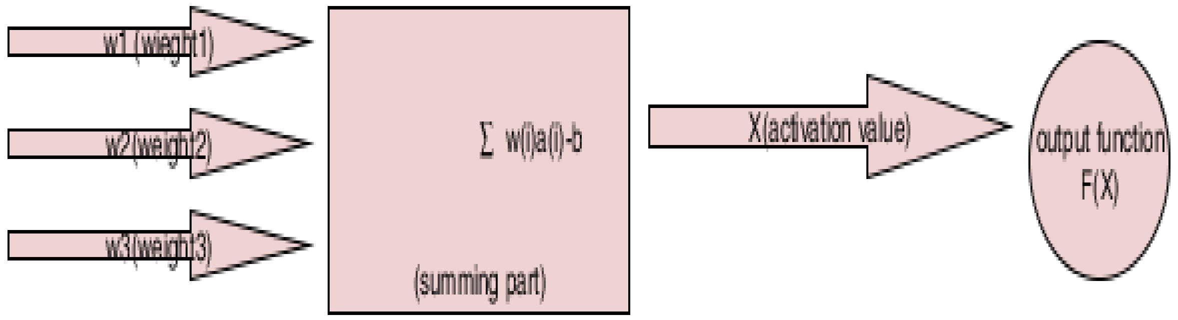
- **Step 5:** From the output layer, go back to the hidden layer to adjust the weights to reduce the error.
- **Step 6:** Repeat the process until the desired output is achieved.



Parameters :

- x = inputs training vector $x=(x_1,x_2,\ldots\ldots\ldots\ldots x_n)$.
- t = target vector $t=(t_1,t_2 \ldots\ldots\ldots\ldots\ldots\ t_n)$.
- δ_k = error at output unit.
- δ_j = error at hidden layer.
- α = learning rate.
- V_{0j} = bias of hidden unit j.

- What is the name of the model in figure below?



a) Rosenblatt perceptron model
b) McCulloch-pitts model
c) Widrow's Adaline model
d) None of the mentioned

- What is nature of function F(x) in the figure?

a) linear
b) non-linear
c) can be either linear or non-linear
d) none of the mentioned

- Continuous perceptron learning is also known as delta learning?
  a) yes
  b) no
- What is reinforcement learning?
  a) learning is based on evaluative signal
  b) learning is based o desired output for an input
  c) learning is based on both desired output & evaluative signal
  d) none of the mentioned
- What is hebbian learning?
  a) synaptic strength is proportional to correlation between firing of post & presynaptic neuron
  b) synaptic strength is proportional to correlation between firing of postsynaptic neuron only
  c) synaptic strength is proportional to correlation between firing of presynaptic neuron only
  d) none of the mentioned

- Correlation learning law is special case of?
  a) Hebb learning law
  b) Perceptron learning law
  c) Delta learning law
  d) LMS learning law
- The loss function is minimized by ___________
  a) Linear regression
  b) Polynomial regression
  c) PAC learning
  d) Gradient descent optimixation algorithm minimize th e loss function in machine
- What is the minimum number of parameters of the gradient descent algorithm?
  a) 1
  b) 2 learning rate and initial weights
  c) 3
  d) 4

- What happens when the learning rate is low?
  a) It always reaches the minima quickly
  b) It reaches the minima very slowly
  c) It overshoots the minima
  d) N
- How can states of units be updated in hopfield model?
  a) synchronously
  b) asynchronously
  c) synchronously and asynchronously
  d) none of the mentioned
- What is asynchronous update in hopfield model?
  a) all units are updated simultaneously
  b) a unit is selected at random and its new state is computed
  c) a predefined unit is selected and its new state is computed
  d) none of the mentioned

- The backpropagation law is also known as generalized delta rule, is it true?
  a) yes
  b) no
- What is true regarding backpropagation rule?
  a) it is also called generalized delta rule
  b) error in output is propagated backwards only to determine weight updates
  c) there is no feedback of signal at nay stage
  d) all of the mentioned
- What are general limitations of back propagation rule?
  a) local minima problem
  b) slow convergence
  c) scaling
  d) all of the mentioned

- Which of the following statements is true about the learning rate alpha in gradient descent?

a) If alpha is very small, gradient descent will be fast to converge. If alpha is too large, gradient descent will overshoot

b) If alpha is very small, gradient descent can be slow to converge. If alpha is too large, gradient descent will overshoot

c) If alpha is very small, gradient descent can be slow to converge. If alpha is too large, gradient descent can be slow too

d) If alpha is very small, gradient descent will be fast to converge. If alpha is too large, gradient descent will be slow

- Who invented gradient descent?

  a) Ross Quinlan

  b) Leslie Valiant

  c) Thomas Bayes

  d) Augustin-Louis Cauchy