

- Artificial Neural Network (ANN) is a computational system inspired by the structure, processing method, learning ability of a biological brain. An artificial neural network is composed of many artificial neurons that are linked together according to specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs.
- ANNs do not execute programmed instructions; they respond in parallel to the pattern of inputs presented to it. There are also no separate memory addresses for storing data. Instead, information is contained in the overall activation 'state' of the network. 'Knowledge' is thus represented by the network itself, which is quite literally more than the sum of its individual components.

1. **Neurons (Nodes):**

Neurons are the basic processing units in an artificial neural network. They receive input signals, perform computations using activation functions, and produce output signals. Neurons are organized in layers within the network.

2. **Layers:**

ANNs are typically organized into layers, including input, hidden, and output layers. The number of layers and the number of neurons in each layer can vary based on the specific architecture of the network.

- **Input Layer:** The input layer receives raw input data or features from the external environment or a previous layer in the network. Each neuron in the input layer represents a feature or input variable.
- **Hidden Layers:** Hidden layers are intermediate layers between the input and output layers. They perform complex transformations and feature extraction from the input data. Deep neural networks (DNNs) have multiple hidden layers, allowing them to learn hierarchical representations of data.
- **Output Layer:** The output layer produces the final output of the network based on the computations and transformations performed in the hidden layers. The number of neurons in the output layer depends on the type of task (e.g., binary classification, multi-class classification, regression).

3. **Connections (Edges):**

Connections between neurons carry weighted signals from one layer to another, allowing information to flow through the network during the forward pass (input to output). Each connection is associated with a weight parameter that determines the strength of the connection.

4. **Activation Functions:**

Activation functions introduce non-linearities into the network, enabling it to learn complex relationships in data. Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and softmax (for multi-class classification in the output layer).

5. **Weights and Biases:**

Weights represent the strength of connections between neurons, influencing the impact of input signals on the neuron's activation. Biases are additional parameters added to neurons that allow them to adjust the decision boundary and improve model fitting during training.

What are the types of Artificial Neural Networks?

- **Feedforward Neural Network:** The feedforward neural network is one of the most basic artificial neural networks. In this ANN, the data or the input provided travels in a single direction. It enters into the ANN through the input layer and exits through the output layer while hidden layers may or may not exist. So the feedforward neural network has a front-propagated wave only and usually does not have backpropagation.
- **Convolutional Neural Network:** A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit. But a CNN has one or more than one convolutional layer that uses a convolution operation on the input and then passes the result obtained in the form of output to the next layer. CNN has applications in speech and image processing which is particularly useful in computer vision.
- **Recurrent Neural Network:** The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer. The first layer in the RNN is quite similar to the feed-forward neural network and the recurrent neural network starts once the output of the first layer is computed. After this layer, each unit will remember some information from the previous step so that it can act as a memory cell in performing computations.

- The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell [right] that uses biochemical reactions to receive, process and transmit information. Fig. 1.3.1 shows biological neural systems.

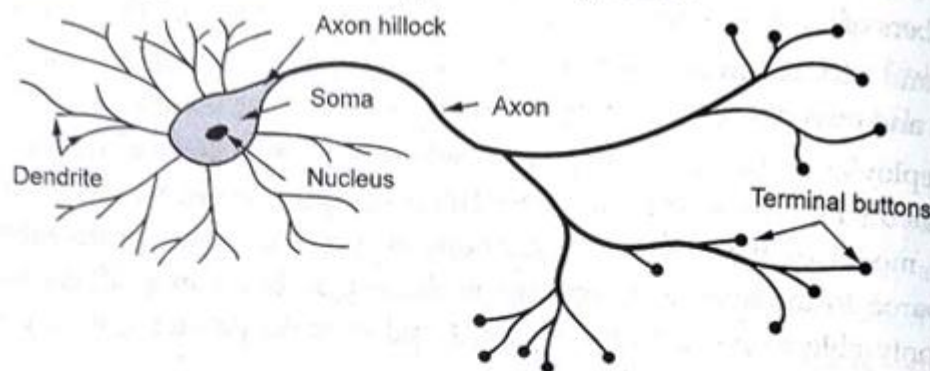


Fig. 1.3.1 Schematic of biological neuron

- The single cell neuron consists of the cell body or soma, the dendrites and the axon. The dendrites receive signals from the axons of other neurons. The small space between the axon of one neuron and the dendrite of another is the synapse. The afferent dendrites conduct impulses toward the soma. The efferent axon conducts impulses away from the soma.

Basic Components of Biological Neurons

1. The majority of *neurons* encode their activations or outputs as a series of brief electrical pulses (i.e. spikes or action potentials).
2. The neuron's *cell body (soma)* processes the incoming activations and converts them into output activations.
3. The neuron's *nucleus* contains the genetic material in the form of DNA. This exists in most types of cells, not just neurons.
4. *Dendrites* are fibres which emanate from the cell body and provide the receptive zones that receive activation from other neurons.
5. *Axons* are fibres acting as transmission lines that send activation to other neurons.
6. The junctions that allow signal transmission between the axons and dendrites are called *synapses*. The process of transmission is by diffusion of chemicals called *neurotransmitters* across the synaptic cleft.

Useful properties and capabilities of neural network

1. **Nonlinearity** : An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear.
2. **Adaptivity** : Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.
3. **Contextual information** : Knowledge is represented by the very structured and activation state of a neural network.
4. **Evidential response** : In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made.
5. **Uniformity of analysis and design** : Neural networks enjoy universality as information processors.
6. **VLSI implement-ability** : The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks.

1.4.1 Advantages of Neural Network

- The advantages of neural networks are due to its adaptive and generalization ability.
- a) Neural networks are adaptive methods that can learn without any prior assumption of the underlying data.
- b) Neural network, namely the feed forward multilayer perception and radial basis function network have been proven to be universal functional approximations.
- c) Neural networks are non-linear model with good generalization ability.

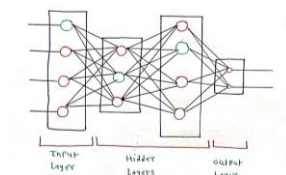
4.2 Application of Neural Network

Neural network applications can be grouped in following categories :

1. **Clustering** : A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.
2. **Classification/Pattern recognition** : The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.
3. **Function approximation** : The tasks of function approximation is to find an estimate of the unknown function $f()$ subject to noise. Various engineering and scientific disciplines require function approximation.
4. **Prediction/Dynamical systems** : The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from function approximation by considering time factor.

See in the notebook:

Neural Net Architecture:



Types-

- 1- Single layer feed forward network
 - a. Feed forward
 - b. Feedback neural network
- 2- Multi-layer feed forward network
- 3- Recurrent neural network

See the topologies from the notebook;

McCulloch Pitts model:

1. Neuron Representation:

- In the M-P model, a neuron is represented as a binary threshold logic unit (TLU). It receives binary inputs (0 or 1) from other neurons or external sources and produces a binary output (0 or 1) based on its internal processing.

2. Inputs and Weights:

- Each input to the neuron is associated with a weight that represents the strength of the connection. The weights can be either positive or negative.
- Inputs are typically denoted as x_1, x_2, \dots, x_n , and corresponding weights are denoted as w_1, w_2, \dots, w_n .

3. Activation Function:

- The activation function in the M-P model is a threshold function that determines the neuron's output based on the weighted sum of inputs.
- Mathematically, the activation function is defined as follows:

$$\text{Output} = \begin{cases} 1 & \text{if } \sum_{i=1}^n (w_i \cdot x_i) \geq \text{Threshold} \\ 0 & \text{otherwise} \end{cases}$$

↓

- Here, the threshold is a parameter that determines the neuron's firing behavior. If the weighted sum of inputs exceeds the threshold, the neuron fires and produces an output of 1; otherwise, it remains inactive with an output of 0.

Perceptron Model and learning:

Refere the notebook for perceptron model has the two types single and multi layer perceptron model and below only theory and for the learning use the training process from below

The Perceptron model is one of the simplest types of artificial neural networks (ANNs), specifically designed for binary classification tasks. It was introduced by Frank Rosenblatt in the late 1950s and laid the groundwork for more complex neural network architectures. The Perceptron model consists of a single layer of computational units called perceptrons, each capable of making binary decisions based on input features.

Let's delve deeper into the components and workings of the Perceptron model:

Perceptron Structure:

- A perceptron takes multiple input values x_1, x_2, \dots, x_n , each associated with a weight w_1, w_2, \dots, w_n .
- The perceptron computes the weighted sum of inputs and applies an activation function to produce an output.
- Mathematically, the output y of a perceptron can be expressed as:

$$y = \text{step}(w_1 \cdot x_1 + w_2 \cdot x_2 + \dots + w_n \cdot x_n + b)$$

where b is the bias term, and step is a step function (commonly a threshold function) that converts the weighted sum into a binary output (0 or 1).

Activation Function:

- The step function used in the perceptron model is a simple threshold function. It outputs 1 if the weighted sum of inputs plus the bias is greater than or equal to zero; otherwise, it outputs 0.
- Mathematically, the step function can be defined as:

$$\text{step}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Training the Perceptron:

- The training of a perceptron involves adjusting its weights and bias to correctly classify input data.
- The perceptron learning rule, also known as the delta rule or Rosenblatt's rule, is a simple algorithm used for training perceptrons.
- Steps in the training process:
 - a. Initialize weights w_i and bias b to small random values or zeros.
 - b. For each training example (x_1, x_2, \dots, x_n) , compute the predicted output \hat{y} using the current weights and bias.
 - c. Update weights and bias using the perceptron learning rule:

$$\Delta w_i = \eta \cdot (y - \hat{y}) \cdot x_i$$

$$\Delta b = \eta \cdot (y - \hat{y})$$

where η is the learning rate, y is the true output (ground truth), \hat{y} is the predicted output, and Δw_i and Δb are the updates to weights and bias, respectively.

- d. Repeat the training process for multiple epochs or until the model converges (if separable).

Adaline model:Refere the notebook for this

The Adaptive Linear Neuron (Adaline) model is a type of artificial neural network (ANN) introduced by Bernard Widrow and his graduate student Tedd Hoff in 1960. Adaline is closely related to the perceptron model but differs in its learning rule and activation function. It is designed for linear regression and classification tasks, and its key feature is the use of continuous-valued outputs rather than binary outputs like in perceptrons.

Let's delve deeper into the components and workings of the Adaline model:

Architecture:

- Similar to the perceptron, Adaline consists of a single layer of computational units (neurons).
- Each neuron in Adaline takes multiple input values x_1, x_2, \dots, x_n along with corresponding weights w_1, w_2, \dots, w_n .
- The weighted sum of inputs is computed along with a bias term to produce the net input z .
- The key difference lies in the activation function used and the learning rule applied.

Activation Function:

- The activation function in Adaline is a linear activation function (identity function), which outputs the net input directly without thresholding:
$$y = z$$
- Unlike perceptrons with a step function, Adaline's continuous output allows for real-valued predictions rather than binary classifications.

Learning Rule (Widrow-Hoff Rule):

- Adaline uses a variant of the Widrow-Hoff or Delta rule for learning and weight updates.
- The learning objective in Adaline is to minimize the difference between the actual output y and the target output d using a suitable error metric (often mean squared error, MSE).
- The weight update rule in Adaline is based on gradient descent, aiming to minimize the cost function (error) by adjusting weights and bias incrementally:

$$\Delta w_i = \eta \cdot (d - y) \cdot x_i$$

$$\Delta b = \eta \cdot (d - y)$$

- Here, Δw_i represents the change in weight for input x_i , η is the learning rate, d is the target output, y is the actual output, and Δb is the change in bias.

Associative Memory is a type of memory system in ANN that enables the network to recall the patterns or association between inputs.

Associative Memory is also known as Content Addressable Memory (CAM).

Types of Associative Memory

① Auto Associative Memory

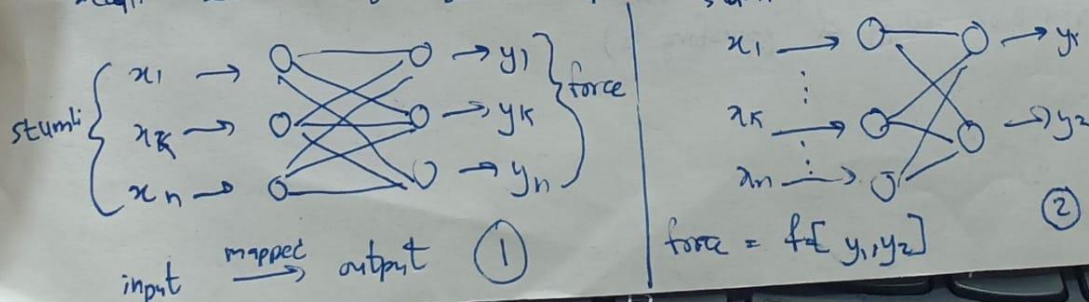
- The network is trained to recall patterns from noisy or incomplete versions of the those patterns.
- It associates an input pattern itself.
- Useful for task such as noise reduction, pattern completion or data reconstruction.
- One common type of autoassociative memory is Hopfield network, which is type RNN that can store and retrieve pattern.

② Hetero Associative Memory

- In heteroassociative memory, the network is trained to associate one set of pattern with another set of pattern.
(input) (output)

- Useful for pattern recognition, classification and mapping between different types of data.

- Example is Willshaw network, which is capable of associative recall between two set of patterns.



in the associated memory the input is associated with the output.



You

Explain the architecture of Multilayered neural network



ChatGPT

A multilayered neural network, also known as a multilayer perceptron (MLP), is a type of artificial neural network (ANN) characterized by multiple layers of interconnected neurons. The architecture of a multilayered neural network consists of three main types of layers: input layer, hidden layers, and output layer. Each layer contains one or more neurons, and neurons within adjacent layers are connected by weighted edges.

1. **Input Layer:**

- The input layer is the first layer of the network where input data is fed into the network. Each neuron in the input layer represents a feature or attribute of the input data. For example, in image recognition tasks, each neuron may represent a pixel value.
- Neurons in the input layer do not perform any computation; they simply pass the input data forward to the next layer.

2. **Hidden Layers:**

- Hidden layers are intermediary layers between the input and output layers. These layers are called "hidden" because their outputs are not observed directly from the outside and are used for internal processing.
- A multilayered neural network can have one or more hidden layers, and each hidden layer consists of multiple neurons. The number of hidden layers and neurons per layer is configurable based on the complexity of the problem and desired model performance.
- Neurons in hidden layers apply an activation function to the weighted sum of inputs received from the previous layer. Common activation functions include sigmoid, tanh, ReLU (Rectified Linear Unit), and others.

3. **Output Layer:**

- The output layer is the final layer of the network that produces the predicted outputs or responses based on the input data and the network's learned parameters.
- The number of neurons in the output layer depends on the nature of the problem. For instance, in binary classification tasks, there may be one neuron representing the output class probability, while in multi-class classification, there are multiple neurons, typically one for each class.
- Neurons in the output layer also apply an activation function appropriate for the task. For example, in binary classification, a sigmoid activation function is often used to produce probabilities between 0 and 1, while in multi-class classification, softmax activation is commonly used to generate class probabilities that sum up to 1.

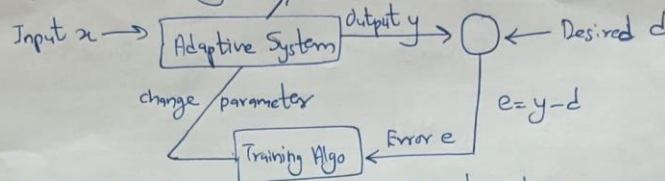
4. **Connections and Weights:**

- Each neuron in a layer is connected to every neuron in the subsequent layer through weighted connections. These weights represent the strength of the connection between neurons and are learned during the training process.
- During forward propagation, input data is multiplied by these weights and passed through activation functions in each layer to produce output values. During backpropagation (training phase), these weights are adjusted based on the calculated error, using optimization techniques like gradient descent and its variants, to minimize the loss function and improve the network's predictive accuracy.



Error Correction :-

Belong to supervised learning paradigm.



The goal of error correction algorithm attempt to minimize this error signal at each training iteration.

The most popular learning algorithm for use with error-correction learning is the Backpropagation.

A frequent used error function is mean squared error:-

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



ChatGPT

Error correction in the context of machine learning and neural networks refers to the process of reducing the discrepancy between predicted outputs and actual target values during training. Minimizing errors is crucial for improving the accuracy and performance of machine learning models. Several techniques and strategies can be employed to minimize errors in machine learning models:

1. Loss Functions:

- Use appropriate loss functions that quantify the difference between predicted and actual values. Common loss functions include:
 - Mean Squared Error (MSE) for regression tasks.
 - Cross-Entropy Loss for binary or multi-class classification tasks.
 - Mean Absolute Error (MAE) for robustness to outliers in regression tasks.

2. Gradient Descent Optimization:

- Employ optimization algorithms such as gradient descent and its variants (e.g., stochastic gradient descent, mini-batch gradient descent) to update model parameters (weights and biases) iteratively.
- Gradient descent algorithms minimize the loss function by adjusting parameters in the direction of steepest descent of the loss surface.

3. Backpropagation:

- Utilize the backpropagation algorithm in neural networks to compute gradients of the loss function with respect to network parameters layer by layer.
- Backpropagation propagates error gradients backward through the network, enabling efficient weight updates based on prediction errors.

4. Learning Rate Tuning:

- Adjust the learning rate hyperparameter carefully. A high learning rate may lead to oscillations or divergence, while a low learning rate may result in slow convergence.
- Techniques like learning rate schedules (e.g., decreasing learning rate over time) or adaptive learning rate methods (e.g., Adam, RMSprop) can help optimize the learning rate during training.

2.7 Backpropagation

- Backpropagation is a training method used for a multi-layer neural network. It is also called the generalized delta rule. It is a gradient descent method which minimizes the total squared error of the output computed by the net.
- The backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.
- Backpropagation is a systematic method for training multiple layer ANN. It is a generalization of Widrow-Hoff error correction rule. 80 % of ANN applications use backpropagation.

- Consider a simple neuron :

- a. Neuron has a summing junction and activation function.
- b. Any non linear function which differentiable everywhere and increases everywhere with sum can be used as activation function.
- c. Examples : Logistic function, Arc tangent function, Hyperbolic tangent activation function.

- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.

- **Need of hidden layers :**

1. A network with only two layers (input and output) can only represent the input with whatever representation already exists in the input data.
2. If the data is discontinuous or non-linearly separable, the innate representation is inconsistent, and the mapping cannot be learned using two layers (Input and Output).
3. Therefore, hidden layer(s) are used between input and output layers.

- **Weights** connects unit (neuron) in one layer only to those in the next higher layer. The output of the unit is scaled by the value of the connecting weight and it is fed forward to provide a portion of the activation for the units in the next higher layer.

- Backpropagation can be applied to an artificial neural network with any number of hidden layers. The training objective is to adjust the weights so that the application of a set of inputs produces the desired outputs.

- **Training procedure :** The network is usually trained with a large number of input-output pairs.

1. Generate weights randomly to small random values (both positive and negative) to ensure that the network is not saturated by large values of weights.
2. Choose a training pair from the training set.
3. Apply the input vector to network input.
4. Calculate the network output.
5. Calculate the error, the difference between the network output and the desired output.
6. Adjust the weights of the network in a way that minimizes this error.
7. Repeat steps 2 - 6 for each pair of input-output in the training set until the error for the entire system is acceptably low.

Forward pass and backward pass :

- Backpropagation neural network training involves two passes.

1. In the forward pass, the input signals moves forward from the network input to the output.
 2. In the backward pass, the calculated error signals propagate backward through the network, where they are used to adjust the weights.
 3. In the forward pass, the calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer.
- In the reverse pass,
- a. The weights of the output neuron layer are adjusted first since the target value of each output neuron is available to guide the adjustment of the associated weights, using the delta rule.
 - b. Next, we adjust the weights of the middle layers. As the middle layer neurons have no target values, it makes the problem complex.

* Difference between ANN & BNN ← Imp

ANN	BNN
1) It is mathematical model which is mainly inspired by biological neuron system in human brain <u>Definition</u>	It is also composed of several processing pieces known as neurons that are included linked together via synapses.
2) It is small in size <u>Size</u>	2) It is large in size
3) Its processing was sequential and centralized <u>Processing</u>	3) It processes the information in a parallel and distributive manner.
4) It processes at faster speed the info <u>Rate</u>	4) It processes the info at slow rate
5) It cannot perform complex pattern recognition <u>Complexity</u>	5) The large quantity and complexity of connection allows the brain to perform complicated tasks.
6) There is no fault tolerance <u>Fault tolerance</u>	6) It has fault tolerance
7) Its response time is measured in micro milliseconds. <u>response time</u>	7) Its response time is measured in nanoseconds.
8) It doesn't any feedback has Provide <u>Feedback</u>	8) It has provides feedback Provides
9) operating Environment well defined well-constrained. <u>operating env</u> <u>Control mechanism</u>	9) operating environment is poorly defined & un-constrained.
10) Its Control unit keeps track of all computer-related operations. <u>Learning</u>	10) keep All Processing is managed centrally.
11) Processing is sequential & centralized 1) Learning requires training	11) Processing is is done Parallel & distributed. Learning is natural.
12) Application:- Image Recognition, classification, regression <u>ex</u>	Robotics, sensorimotor control, natural language processing.

Memory

Memory is separate from a processor, localized, & non-content addressable

memory is integrated into the processor, distributed, & content addressable

Comparison Attribute	Feed-forward N.N.	Feedback (Recurrent) N.N.
single flow dir ⁿ	Forward only	Bidirectional.
Delay	No	Yes
Complexity	Low	High
Neuron independent in same layer	Yes	No
Speed	High	slow
Commonly used for	Pattern recognition, speech recognition, character recognition	Language translation, speech to text conversion & robotic control.
Adjustment of variable	Based on knowledge	Based on error
loop type	open loop	closed loop.
Importance of I/O	Input (important) more essential	output is (important) essential.

Table

Aspect	McCulloch-Pitts Neuron	Perceptron	Adaline (Adaptive Linear Neuron)
Model Type	Binary threshold unit	Single-layer neural network	Improvement over Perceptron
Activation Function	Step function	Step function	Linear activation function
Learning Rule	None (no learning)	Iterative learning	Continuous response values
Weights/Thresholds	Identical weights	Adjusted based on class labels	Stochastic gradient descent
Decision Boundary	Linear	Linear	Linear
Use Cases	Linearly separable data	Binary classification	Regression and binary classification

Remember, these models paved the way for more sophisticated neural networks! 🧠💡

Table

Aspect	Auto-Associative Memory	Hetero-Associative Memory
Purpose	Recalls the same pattern given an input pattern ($Y = X$)	Associates one set of patterns with another ($Y \neq X$)
Application	Speech recognition, image recognition	Data compression, data retrieval
Input-Output Relation	$Y = X$	$Y \neq X$
Robustness	Handles partial or degraded inputs	More flexible associations
Examples	Storing personal memories, recognizing familiar faces	Associating keywords with relevant documents
Training Process	- Auto-associative networks learn by iteratively updating weights based on input patterns. The network adjusts itself to minimize the difference between the input and the recalled pattern.	- Hetero-associative networks are trained using Hebbian learning or other associative learning rules. These rules allow the network to form associations between different patterns.
Capacity	- Auto-associative memory can store a limited number of patterns due to its recurrent nature.	- Hetero-associative memory has a larger capacity because it associates different patterns.
Biological Analogy	- Auto-associative memory is akin to the hippocampus in the brain, which plays a role in memory consolidation and recall.	- Hetero-associative memory corresponds to neocortical regions , where complex associations are formed.
Use Cases	- Auto-associative networks are useful for reconstructing missing or noisy data . For example, in image denoising, they can restore corrupted images.	- Hetero-associative networks excel in tasks like semantic linking , where they connect related concepts.
Network Architecture	- Auto-associative networks often have feedback loops (recurrent connections) that allow them to maintain internal states.	- Hetero-associative networks can be feedforward or recurrent , depending on the specific architecture.
Pattern Completion	- Auto-associative memory can complete partial patterns . If given a partial input, it recalls the full pattern.	- Hetero-associative memory does not necessarily complete partial patterns; it retrieves associated patterns.

An auto-associative memory retrieves the same pattern.

Hetero-associative memory retrieves the stored pattern.

Algorithm Back Propagation ()

Forward :

Initialize weight vector.

Broadcast the input to the neurons in level -1

$$Z_{in} = V_{0j} + \sum x_i v_i$$

$$z_j = f(Z_{in}) \quad // \text{Apply activation function.}$$

// For Each output unit compute.

$$y_{in} = W_{0k} + \sum z_j w_{jk}$$

$$y_k = f(y_{in}) \quad // \text{Apply Activation function.}$$

Backward (error) : Each op unit $y_k = (k = 1, 2, \dots, m)$ receives a target pattern to input training pattern & compute the error correction using.

$$\delta_k = (t_k - y_k) \cdot f'(y_{in})$$

// Do weight correction in w

$$\Delta w_{jk} = \alpha \delta_k z_j$$

// Calculate change in bias

$$\Delta w_{0k} = \alpha \delta_k$$

// Calculate hidden unit sum of δ

$$\delta_{in} = \sum \delta_k w_{jk}$$

$$\delta_j = \delta_{in} f'(z_{in})$$

// Calculate weight Adjustment in v

$$\Delta v_{ij} = \alpha \delta_j x_i$$

// Calculate ~~bias~~

$$\Delta v_{0j} = \alpha \delta_j$$

// Do weight adjustment

$$w_{jk}(\text{new}) = w_{jk}(\text{old}) - \Delta w_{jk}$$

	Supervised Learning	Unsupervised Learning
Input Data	Uses Known and Labeled Data as input	Uses Unknown Data as input
Computational Complexity	Less Computational Complexity	More Computational Complex
Real-Time	Uses off-line analysis	Uses Real-Time Analysis of Data
Number of Classes	The number of Classes is known	The number of Classes is not known
Accuracy of Results	Accurate and Reliable Results	Moderate Accurate and Reliable Results
Output data	The desired output is given.	The desired, output is not given.
Model	In supervised learning it is not possible to learn larger and more complex models than in, supervised learning	In unsupervised learning it is possible to learn larger and more complex models than with unsupervised learning
Training data	In supervised learning training data is used to infer model	In unsupervised learning training data is not used.
Another name	Supervised learning is also called classification.	Unsupervised learning is also called clustering.
Test of model	We can test our model.	We can not test our model.
Example	Optical Character Recognition	Find a face in an image.

Basic Learning Law :-

- Learning laws describe the weight vector for the i th processing unit at time instant $(t+1)$ in terms of the weight vector at time instant (t) as follows:

$$W_i(t+1) = W_i(t) + \Delta W_i(t)$$

where $\Delta W_i(t)$ is the change in the weight vector.

1. Hebb's Law

Here the change in the weight vector is given by.

$$\Delta W_i(t) = \eta f(W_i^T a) a$$

where η is learning rate parameter.

W is the weight.

a is input.

Therefore, the j th component of ΔW_i is given by

$$\Delta W_{ij} = \eta f(W_i^T a) a_j$$

$$= \eta S_i a_j \quad \text{for } j=1, 2, \dots, M$$

where S_i = output signal of the i th unit.

- The law states that the weight increment is proportional to the product of the input data and the resulting output signal of the unit.
- This law requires weight initialization to small random values around $W_{ij} = 0$ prior to learning.
- This law represents an unsupervised learning.

Features of ANN: →

1. ANN are extremely powerful computational devices.
2. ANN are modeled on the basis of current brain theories, in which information is represented by weight.
3. ANNs have massive parallelism, which makes them very efficient.
4. They can learn and generalize from training data. So there is no need for enormous feats of programming.
5. Storage is fault tolerant i.e. Some portion of the neural net can be removed and there will be only a small degradation in the quality of stored data.
6. They are particularly fault tolerant which is equivalent to the "graceful degradation" found in biological systems.
7. Data are naturally stored in the form of associative memory which contrasts with conventional memory, in which data are recalled by specifying address of that data.
- 8) They are very noise tolerant, so they can cope with situations where normal symbolic systems would have difficulty.
- 9) In practice they can do anything a symbolic/logic system can do and more.
- 10) Neural networks can extrapolate and interpolate from their stored information. The neural networks can also be trained. special training teaches the net to look for significant features or relationships of data.