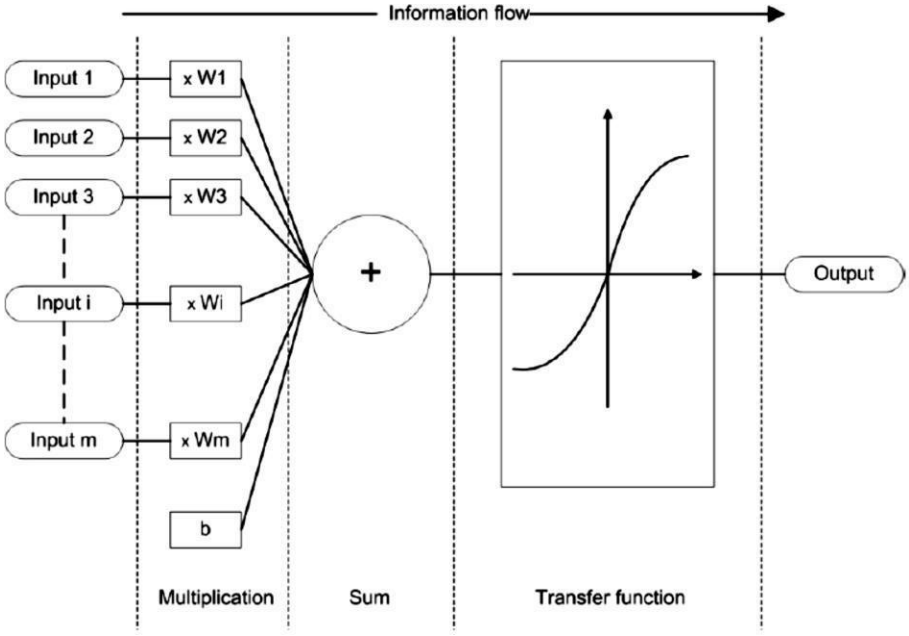
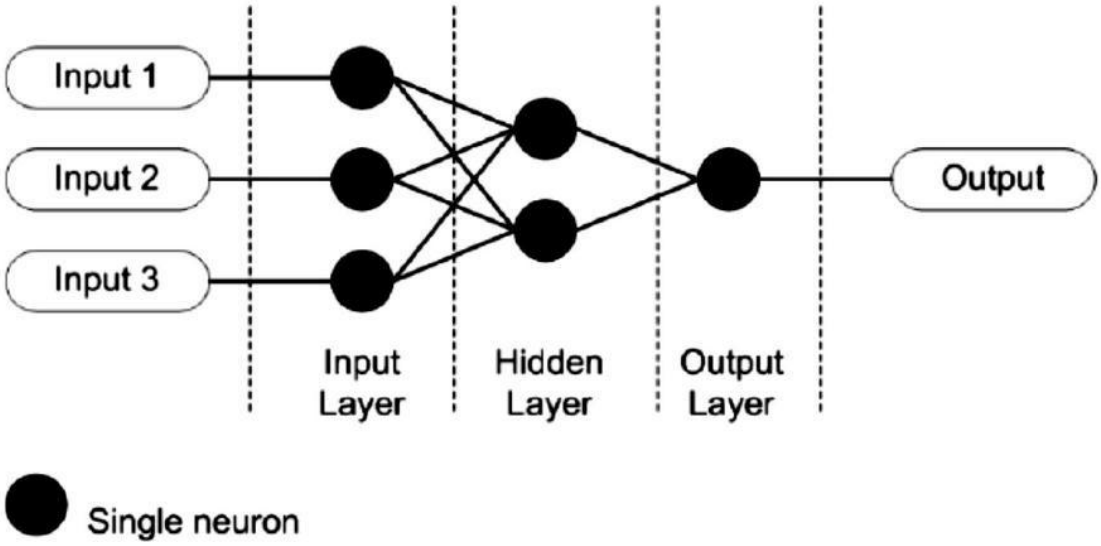
UNIT-1

NEURAL NETWORKS-1

WHAT IS ARTIFICIAL NEURAL NETWORK?

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function). Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron the sum of previously weighted inputs and bias is passing through activation function that is also called transfer function.





BIOLOGICAL NEURON STRUCTURE AND FUNCTIONS.

A neuron, or nerve cell, is an electrically excitable cell that communicates with other cells via specialized connections called synapses. It is the main component of nervous tissue. Neurons are typically classified into three types based on their function. Sensory neurons respond to stimuli such as touch, sound, or light that affect the cells of the sensory organs, and they send signals to the spinal cord or brain. Motor neurons receive signals from the brain and spinal cord to control everything from muscle contractions to glandular output. Interneurons connect neurons to other neurons within the same region of the brain or spinal cord. A group of connected neurons is called a neural circuit.

A typical neuron consists of a cell body (soma), dendrites, and a single axon. The soma is usually compact. The axon and dendrites are filaments that extrude from it. Dendrites typically branch profusely and extend a few hundred micrometers from the soma. The axon leaves the soma at a swelling called the axon hillock, and travels for as far as 1 meter in humans or more in other species. It branches but usually maintains a constant diameter. At the farthest tip of the axon's branches are axon terminals, where the neuron can transmit a signal across the synapse to another cell. Neurons may lack dendrites or have no axon. The term neurite is used to describe either a dendrite or an axon, particularly when the cell is undifferentiated.

The soma is the body of the neuron. As it contains the nucleus, most protein synthesis occurs here. The nucleus can range from 3 to 18 micrometers in diameter.

The dendrites of a neuron are cellular extensions with many branches. This overall shape and structure is referred to metaphorically as a dendritic tree. This is where the majority of input to the neuron occurs via the dendritic spine.

The axon is a finer, cable-like projection that can extend tens, hundreds, or even tens of thousands of times the diameter of the soma in length. The axon primarily carries nerve signals away from the soma, and carries some types of information back to it. Many neurons have only one axon, but this axon may—and usually will—undergo extensive branching, enabling communication with many target cells. The part of the axon where it emerges from the soma is called the axon hillock. Besides being an anatomical structure, the axon hillock also has the greatest density of voltage-dependent sodium channels. This makes it the most easily excited part of the neuron and the spike initiation zone for the axon. In electrophysiological terms, it has the most negative threshold potential.

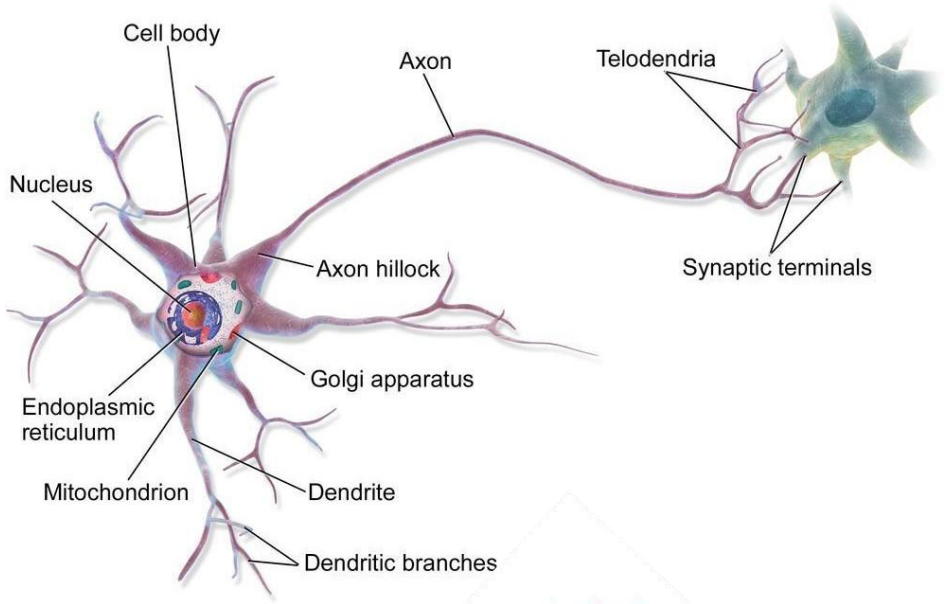
While the axon and axon hillock are generally involved in information outflow, this region can also receive input from other neurons.

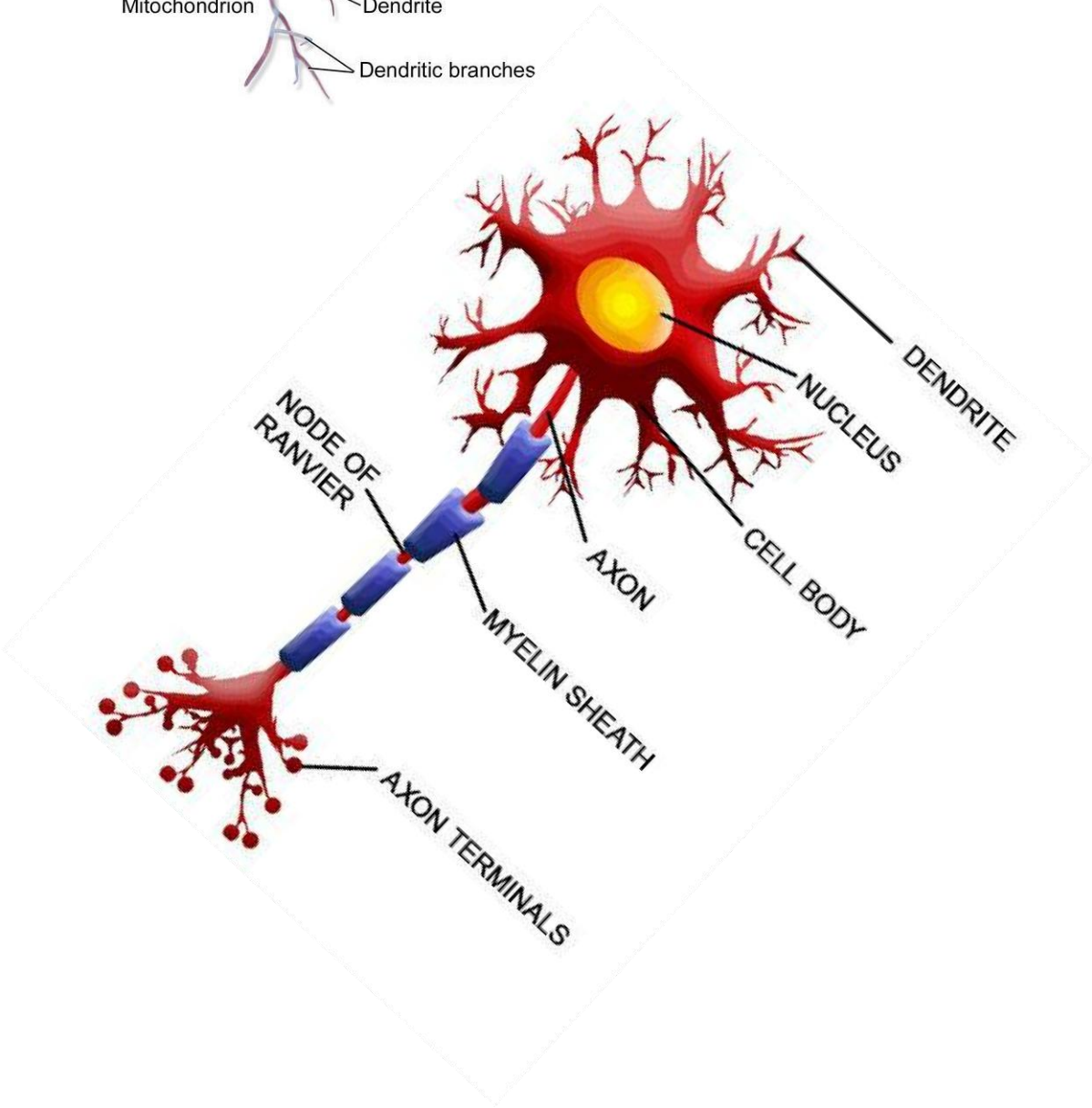
The axon terminal is found at the end of the axon farthest from the soma and contains synapses. Synaptic boutons are specialized structures where neurotransmitter chemicals are released to communicate with target neurons. In addition to synaptic boutons at the axon terminal, a neuron may have en passant boutons, which are located along the length of the axon.

Most neurons receive signals via the dendrites and soma and send out signals down the axon. At the majority of synapses, signals cross from the axon of one neuron to a dendrite of another. However, synapses can connect an axon to another axon or a dendrite to another dendrite. The signaling process is partly electrical and partly chemical. Neurons are electrically excitable, due to maintenance of voltage gradients across their membranes. If the voltage changes by a large amount over a short interval, the neuron generates an all-or-nothing electrochemical pulse called an action potential. This potential travels

rapidly along the axon, and activates synaptic connections as it reaches them. Synaptic signals may be excitatory or inhibitory, increasing or reducing the net voltage that reaches the soma.

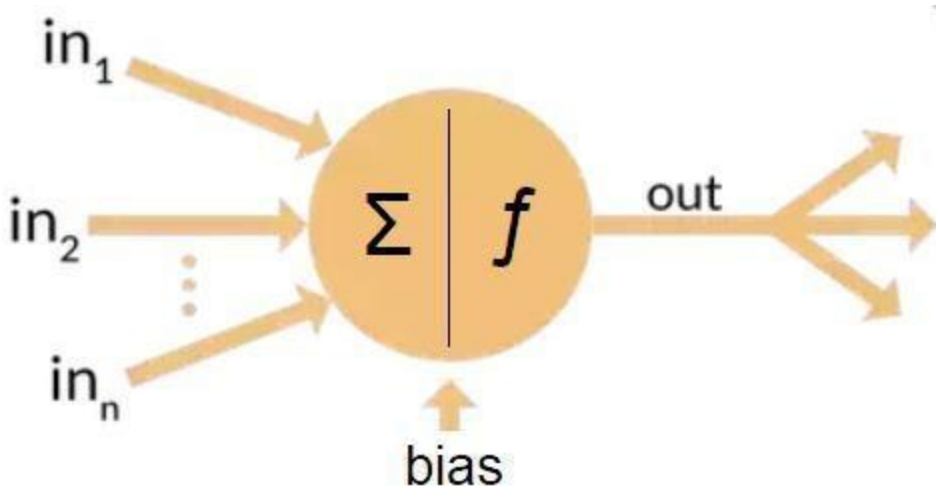
In most cases, neurons are generated by neural stem cells during brain development and childhood. Neurogenesis largely ceases during adulthood in most areas of the brain. However, strong evidence supports generation of substantial numbers of new neurons in the hippocampus and olfactory bulb.

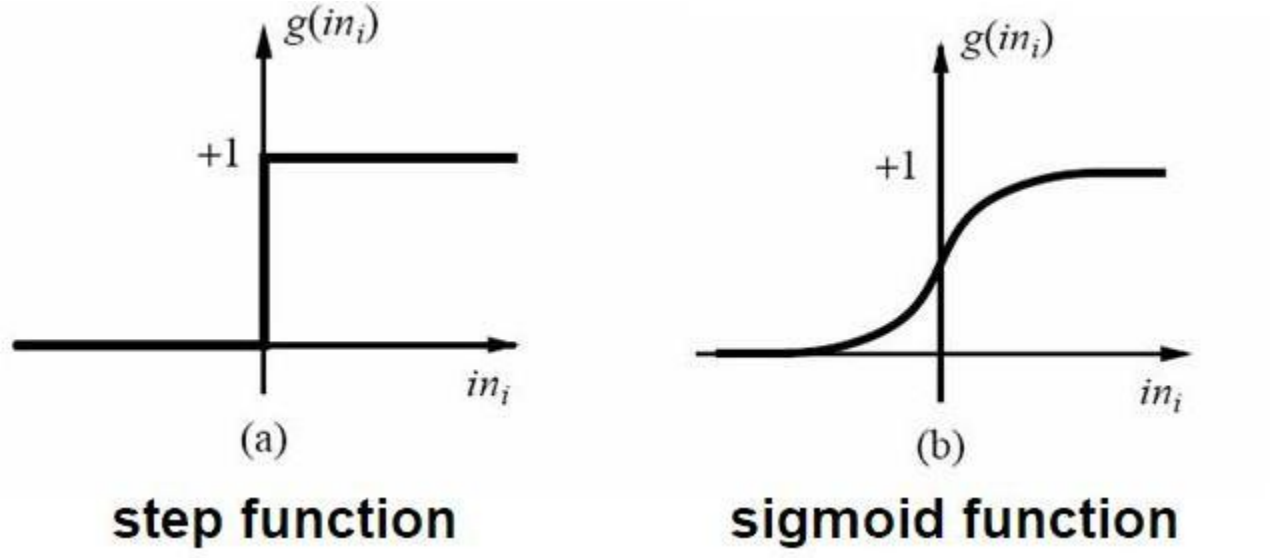




STRUCTURE AND FUNCTIONS OF ARTIFICIAL NEURON.

An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network. Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs (representing excitatory postsynaptic potentials and inhibitory postsynaptic potentials at neural dendrites) and sums them to produce an output (or activation, representing a neuron's action potential which is transmitted along its axon). Usually each input is separately weighted, and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. They are also often monotonically increasing, continuous, differentiable and bounded. The thresholding function has inspired building logic gates referred to as threshold logic; applicable to building logic circuits resembling brain processing. For example, new devices such as memristors have been extensively used to develop such logic in recent times.





STATE THE MAJOR DIFFERENCES BETWEEN BIOLOGICAL AND ARTIFICIANEURAL NETWORKS

1. Size: Our brain contains about 86 billion neurons and more than a 100 synapses (connections). The number of "neurons" in artificial networks is much less than that.

2. Signal transport and processing: The human brain works asynchronously, ANNs work synchronously.

3. Processing speed: Single biological neurons are slow, while standard neurons in ANNs are fast.

4. Topology: Biological neural networks have complicated topologies, while ANNs are often in a tree structure.

5. Speed: certain biological neurons can fire around 200 times a second on average. Signals travel at different speeds depending on the type of the nerve impulse, ranging from 0.61 m/s up to 119 m/s. Signal travel speeds also vary from person to person depending on their sex, age, height, temperature, medical condition, lack of sleep etc. Information in artificial neurons is carried over by the continuous, floating point number values of synaptic weights. There are no refractory periods for artificial neural networks (periods while it is impossible to send another action potential, due to the sodium channels being lock shut) and artificial neurons do not experience "fatigue": they are functions that can be calculated as many times and as fast as the computer architecture would allow.

6. Fault-tolerance: biological neuron networks due to their topology are also fault-tolerant. Artificial neural networks are not modeled for fault tolerance or self regeneration (similarly to fatigue, these ideas are not applicable to matrix operations), though recovery is possible by saving the current state (weight values) of the model and continuing the training from that save state.

7. Power consumption: the brain consumes about 20% of all the human body's energy — despite it's large cut, an adult brain operates on about 20 watts (barely enough to dimly light a bulb) being extremely efficient. Taking into account how humans can still operate for a while, when only given some c-vitamin rich lemon juice and beef tallow, this is quite remarkable. For benchmark: a single Nvidia GeForce Titan X GPU runs on 250 watts alone, and requires a power supply. Our machines are way less efficient than biological systems. Computers also generate a lot of heat when used, with consumer GPUs operating safely between 50–80°Celsius instead of 36.5–37.5 °C.

8. Learning: we still do not understand how brains learn, or how redundant connections store and recall information. By learning, we are building on information that is already stored in the brain. Our knowledge deepens by repetition and during sleep, and tasks that once required a focus can be executed automatically once mastered. Artificial neural networks in the other hand, have a predefined model, where no further neurons or connections can be added or removed. Only the weights of the connections (and biases representing thresholds) can change during training. The networks start with random weight values and will slowly try to reach a point where further changes in the weights would no longer improve performance. Biological networks usually don't stop / start learning. ANNs have different fitting (train) and prediction (evaluate) phases.

9. Field of application: ANNs are specialized. They can perform one task. They might be perfect at playing chess, but they fail at playing go (or vice versa). Biological neural networks can learn completely new tasks.

10. Training algorithm: ANNs use Gradient Descent for learning. Human brains use something different (but we don't know what).

BRIEFLY EXPLAIN THE BASIC BUILDING BLOCKS OF ARTIFICIAL NEURAL NETWORKS.

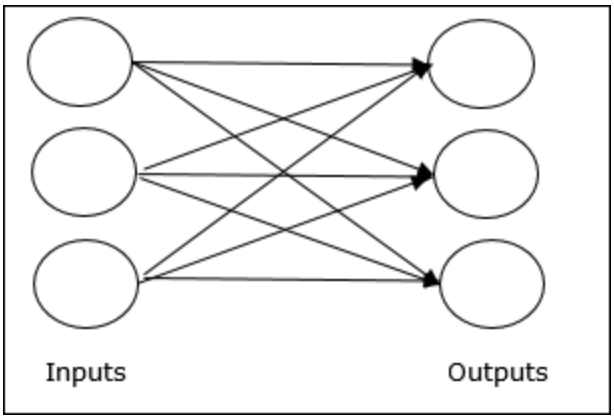
Processing of ANN depends upon the following three building blocks:

1. Network Topology
2. Adjustments of Weights or Learning
3. Activation Functions

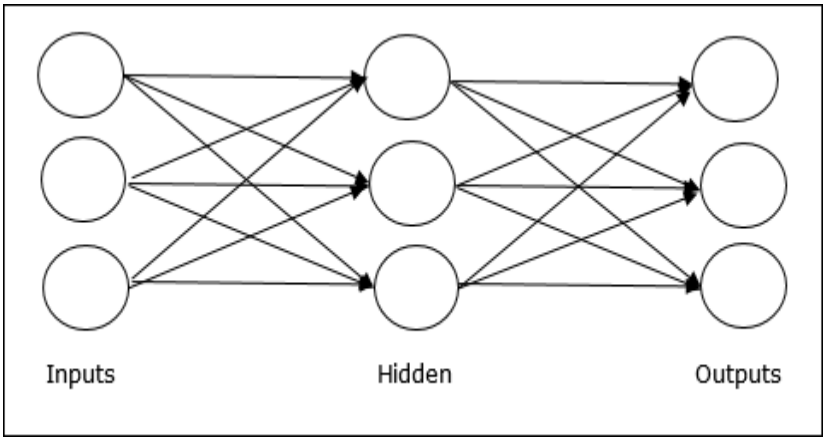
1. Network Topology: A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds:

A. Feed forward Network: It is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer are connected with the nodes of the previous layers. The connection has different weights upon them. There is no feedback loop means the signal can only flow in one direction, from input to output. It may be divided into the following two types:

- **Single layer feed forward network:** The concept is of feed forward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.

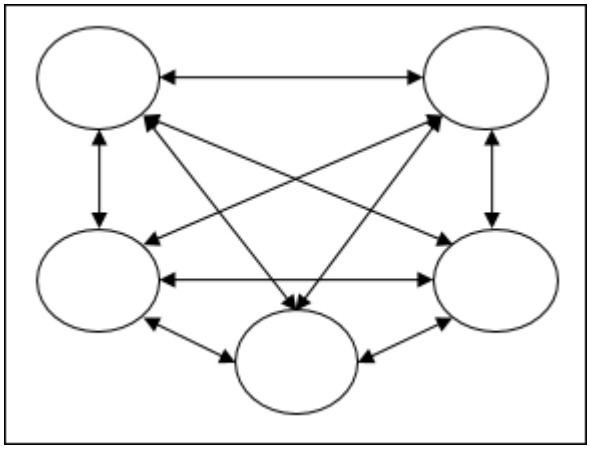


- **Multilayer feed forward network:** The concept is of feed forward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.

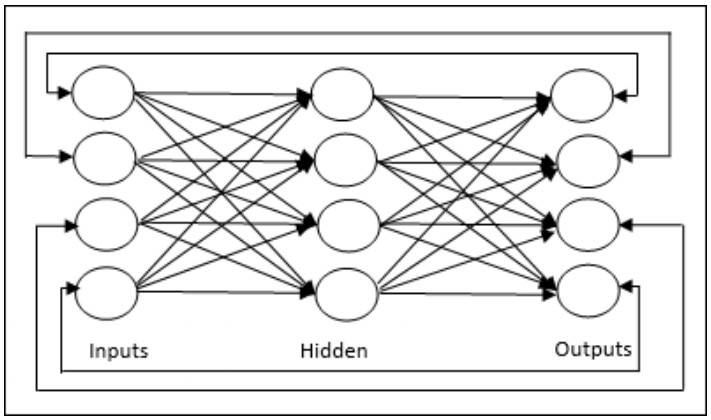


B. Feedback Network: As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types:

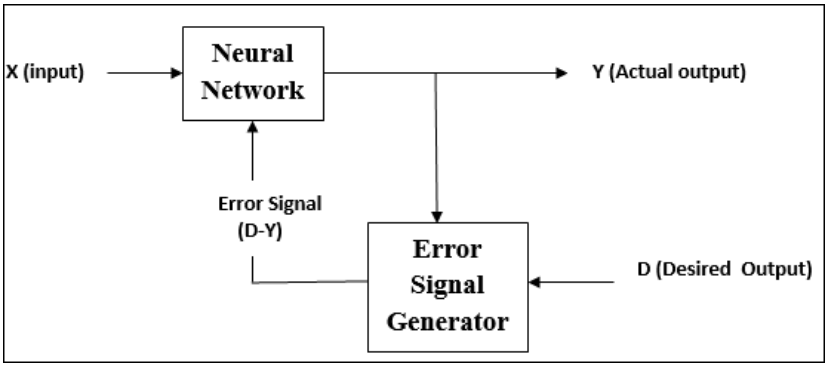
- **Recurrent networks:** They are feedback networks with closed loops. Following are the two types of recurrent networks.
- **Fully recurrent network:** It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.



- **Jordan network** – It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.

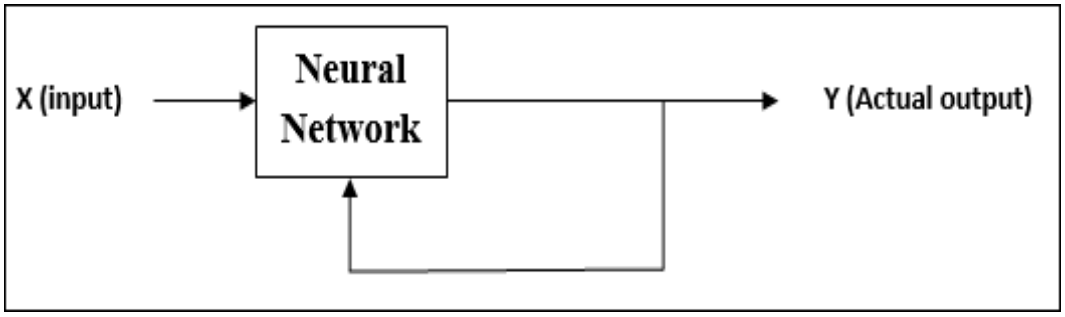


2. Adjustments of Weights or Learning: Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network. Learning in ANN can be classified into three categories namely supervised learning, unsupervised learning, and reinforcement learning.

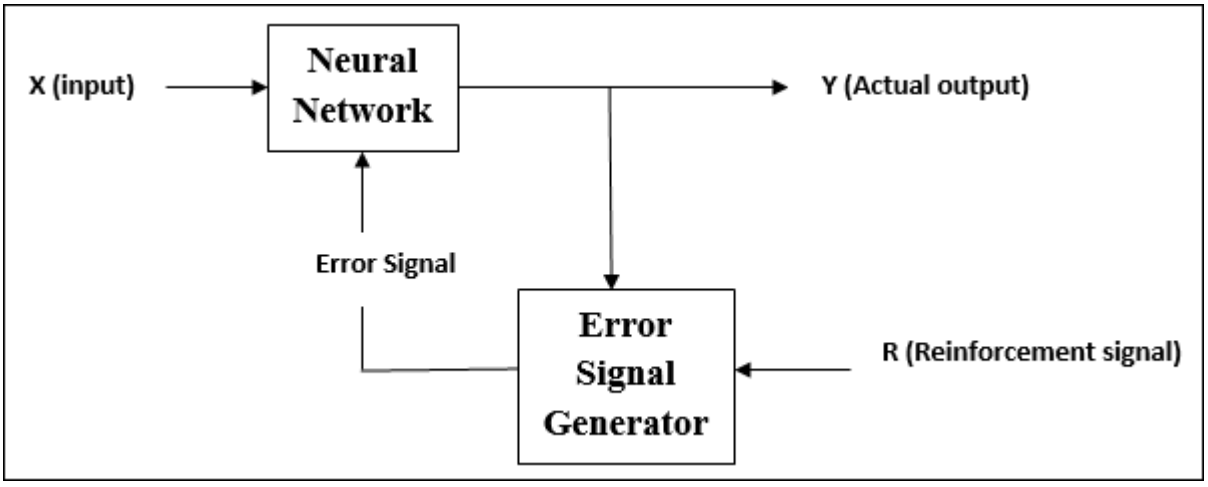


Supervised Learning: As the name suggests, this type of learning is done under the supervision of a teacher. This learning process is dependent. During the training of ANN under supervised learning, the input vector is presented to the network, which will give an output vector. This output vector is compared with the desired output vector. An error signal is generated, if there is a difference between the actual output and the desired output vector. On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output.

Unsupervised Learning: As the name suggests, this type of learning is done without the supervision of a teacher. This learning process is independent. During the training of ANN under unsupervised learning, the input vectors of similar type are combined to form clusters. When a new input pattern is applied, then the neural network gives an output response indicating the class to which the input pattern belongs. There is no feedback from the environment as to what should be the desired output and if it is correct or incorrect. Hence, in this type of learning, the network itself must discover the patterns and features from the input data, and the relation for the input data over the output.



Reinforcement Learning: As the name suggests, this type of learning is used to reinforce or strengthen the network over some critic information. This learning process is similar to supervised learning, however we might have very less information. During the training of network under reinforcement learning, the network receives some feedback from the environment. This makes it somewhat similar to supervised learning. However, the feedback obtained here is evaluative not instructive, which means there is no teacher as in supervised learning. After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.



3. **Activation Functions:** An activation function is a mathematical equation that determines the output of each element (perceptron or neuron) in the neural network. It takes in the input from each neuron and transforms it into an output, usually between one and zero or between -1 and one. It may be defined as the extra force or effort applied over the input to obtain an exact output. In ANN, we can also apply activation functions over the input to get the exact output. Followings are some activation functions of interest:

i) Linear Activation Function: It is also called the identity function as it performs no input editing. It can be defined as: $F(x) = x$

ii) Sigmoid Activation Function: It is of two type as follows –

- **Binary sigmoidal function:** This activation function performs input editing between 0 and 1. It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1. It is also strictly increasing in nature, which means more the input higher would be the output. It can be defined as

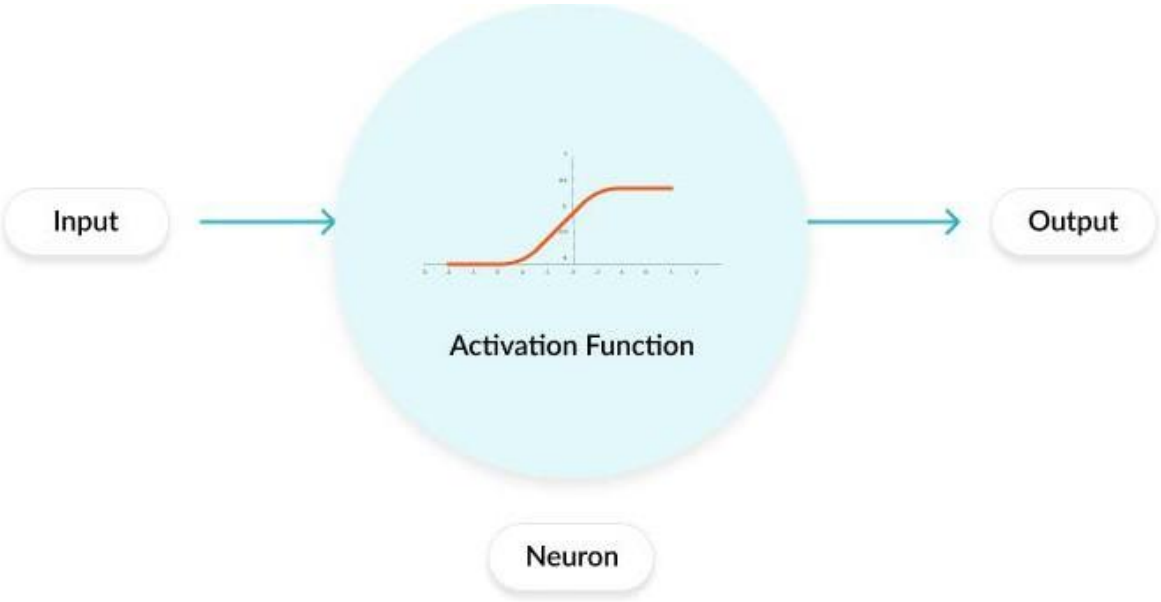
$$F(x)=sigm(x)=11+exp(−x)F(x)=sigm(x)=11+exp(−x)$$

- **Bipolar sigmoidal function:** This activation function performs input editing between -1 and 1. It can be positive or negative in nature. It is always bounded, which means its output cannot be less than -1 and more than 1. It is also strictly increasing in nature like sigmoid function. It can be defined as

$$F(x)=sigm(x)=21+exp(−x)−1=1−exp(x)1+exp(x)$$

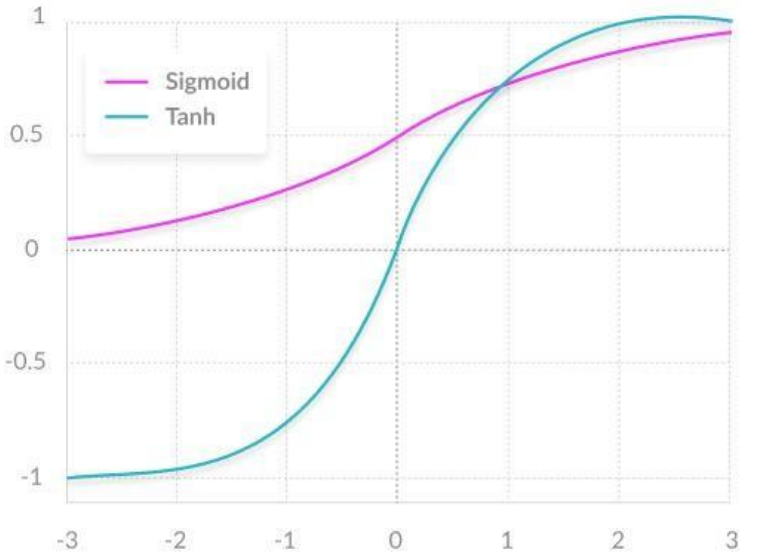
WHAT IS A NEURAL NETWORK ACTIVATION FUNCTION?

In a neural network, inputs, which are typically real values, are fed into the neurons in the network. Each neuron has a weight, and the inputs are multiplied by the weight and fed into the activation function. Each neuron's output is the input of the neurons in the next layer of the network, and so the inputs cascade through multiple activation functions until eventually, the output layer generates a prediction. Neural networks rely on nonlinear activation functions—the derivative of the activation function helps the network learn through the backpropagation process.



SOME COMMON ACTIVATION FUNCTIONS INCLUDE THE FOLLOWING:

1. **The sigmoid function** has a smooth gradient and outputs values between zero and one. For very high or low values of the input parameters, the network can be very slow to reach a prediction, called the *vanishing gradient* problem.
2. **The TanH function** is zero-centered making it easier to model inputs that are strongly negative strongly positive or neutral.
3. **The ReLu function** is highly computationally efficient but is not able to process inputs that approach zero or negative.
4. **The Leaky ReLu** function has a small positive slope in its negative area, enabling it to process zero or negative values.
5. **The Parametric ReLu** function allows the negative slope to be learned, performing backpropagation to learn the most effective slope for zero and negative input values.
6. **Softmax** is a special activation function use for output neurons. It normalizes outputs for each class between 0 and 1, and returns the probability that the input belongs to a specific class.
7. **Swish** is a new activation function discovered by Google researchers. It performs better than ReLu with a similar level of computational efficiency.



Two common neural network activation functions - Sigmoid and Tanh

APPLICATIONS OF ANN

1. Data Mining: Discovery of meaningful patterns (knowledge) from large volumes of data.
2. Expert Systems: A computer program for decision making that simulates thought process of a human expert.
3. Fuzzy Logic: Theory of approximate reasoning.
4. Artificial Life: Evolutionary Computation, Swarm Intelligence.
5. Artificial Immune System: A computer program based on the biological immune system.
6. Medical: At the moment, the research is mostly on modelling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.).Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quantity'. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

7. Computer Science: Researchers in quest of artificial intelligence have created spin offs like dynamic programming, object oriented programming, symbolic programming, intelligent storage management systems and many more such tools. The primary goal of creating an artificial intelligence still remains a distant dream but people are getting an idea of the ultimate path, which could lead to it.
8. Aviation: Airlines use expert systems in planes to monitor atmospheric conditions and system status. The plane can be put on autopilot once a course is set for the destination.
9. Weather Forecast: Neural networks are used for predicting weather conditions. Previous data is fed to a neural network, which learns the pattern and uses that knowledge to predict weather patterns.
10. Neural Networks in business: Business is a diverted field with several general areas of specialization such as accounting or financial analysis. Almost any neural network application would fit into one business area or financial analysis.
11. There is some potential for using neural networks for business purposes, including resource allocation and scheduling.
12. There is also a strong potential for using neural networks for database mining, which is, searching for patterns implicit within the explicitly stored information in databases. Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling.
13. Marketing: There is a marketing application which has been integrated with a neural network system. The Airline Marketing Tactician (a trademark abbreviated as AMT) is a computer system made of various intelligent technologies including expert systems. A feed forward neural network is integrated with the AMT and was trained using back-propagation to assist the marketing control of airline seat allocations. The adaptive neural approach was amenable to rule expression. Additionally, the application's environment changed rapidly and constantly, which required a continuously adaptive solution.
14. Credit Evaluation: The HNC company, founded by Robert Hecht-Nielsen, has developed several neural network applications. One of them is the Credit Scoring system which increases the profitability of the existing model up to 27%. The HNC neural systems were also applied to mortgage screening. A neural network automated mortgage insurance under writing system was developed by the Nestor Company. This system was trained with 5048 applications of which 2597 were certified. The data related to property and borrower qualifications. In a conservative mode the system agreed on the under writers on 97% of the cases. In the liberal model the system agreed 84% of the cases. This is system run on an Apollo DN3000 and used 250K memory while processing a case file in approximately 1 sec.

ADVANTAGES OF ANN

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Pattern recognition: is a powerful technique for harnessing the information in the data and generalizing about it. Neural nets learn to recognize the patterns which exist in the data set.
5. The system is developed through learning rather than programming.. Neural nets teach themselves the patterns in the data freeing the analyst for more interesting work.

6. Neural networks are flexible in a changing environment. Although neural networks may take some time to learn a sudden drastic change they are excellent at adapting to constantly changing information.
7. Neural networks can build informative models whenever conventional approaches fail. Because neural networks can handle very complex interactions they can easily model data which is too difficult to model with traditional approaches such as inferential statistics or programming logic.
8. Performance of neural networks is at least as good as classical statistical modelling, and better on most problems. The neural networks build models that are more reflective of the structure of the data in significantly less time.

LIMITATIONS OF ANN

In this technological era everything has Merits and some Demerits in others words there is a Limitation with every system which makes this ANN technology weak in some points. The various Limitations of ANN are:-

1) ANN is not a daily life general purpose problem solver.
2) There is no structured methodology available in ANN.
3) There is no single standardized paradigm for ANN development.
4) The Output Quality of an ANN may be unpredictable.
5) Many ANN Systems does not describe how they solve problems.
6) Black box Nature
7) Greater computational burden.
8) Proneness to over fitting.
9) Empirical nature of model development.

ARTIFICIAL NEURAL NETWORK CONCEPTS/TERMINOLOGY

Here is a glossary of basic terms you should be familiar with before learning the details of neural networks.

Inputs: Source data fed into the neural network, with the goal of making a decision or prediction about the data. Inputs to a neural network are typically a set of real values; each value is fed into one of the neurons in the input layer.

Training Set: A set of inputs for which the correct outputs are known, used to train the neural network.

Outputs : Neural networks generate their predictions in the form of a set of real values or boolean decisions. Each output value is generated by one of the neurons in the output layer.

Neuron/perceptron: The basic unit of the neural network. Accepts an input and generates a prediction.

Each neuron accepts part of the input and passes it through the activation function. Common activation functions are sigmoid, TanH and ReLu. Activation functions help generate output values within an acceptable range, and their non-linear form is crucial for training the network.

Weight Space: Each neuron is given a numeric weight. The weights, together with the activation function, define each neuron's output. Neural networks are trained by fine-tuning weights, to discover the optimal set of weights that generates the most accurate prediction.

Forward Pass: The forward pass takes the inputs, passes them through the network and allows each neuron to react to a fraction of the input. Neurons generate their outputs and pass them on to the next layer, until eventually the network generates an output.

Error Function: Defines how far the actual output of the current model is from the correct output. When training the model, the objective is to minimize the error function and bring output as close as possible to the correct value.

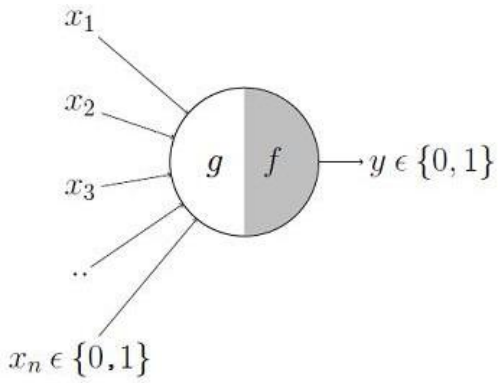
Backpropagation: In order to discover the optimal weights for the neurons, we perform a backward pass, moving back from the network's prediction to the neurons that generated that prediction. This is called backpropagation. Backpropagation tracks the derivatives of the activation functions in each successive neuron, to find weights that bring the loss function to a minimum, which will generate the best prediction. This is a mathematical process called *gradient descent*.

Bias and Variance: When training neural networks, like in other machine learning techniques, we try to balance between bias and variance. Bias measures how well the model fits the training set—able to correctly predict the known outputs of the training examples. Variance measures how well the model works with unknown inputs that were not available during training. Another meaning of bias is a "bias neuron" which is used in every layer of the neural network. The bias neuron holds the number 1, and makes it possible to move the activation function up, down, left and right on the number graph.

Hyperparameters: A hyper parameter is a setting that affects the structure or operation of the neural network. In real deep learning projects, tuning hyper parameters is the primary way to build a network that provides accurate predictions for a certain problem. Common hyper parameters include the number of hidden layers, the activation function, and how many times (epochs) training should be repeated.

MCCULLOGH-PITTS MODEL

In 1943 two electrical engineers, Warren McCullogh and Walter Pitts, published the first paper describing what we would call a neural network.



It may be divided into 2 parts. The first part, g takes an input, performs an aggregation and based on the aggregated value the second part, f makes a decision. Let us suppose that I want to predict my own decision, whether to watch a random football game or not on TV. The inputs are all boolean i.e., {0,1} and my output variable is also boolean {0: Will watch it, 1: Won't watch it}.

So, x_1 could be 'is Indian Premier League On' (I like Premier League more)

x_2 could be 'is it a knockout game (I tend to care less about the league level matches)

x_3 could be 'is Not Home' (Can't watch it when I'm in College. Can I?)

x_4 could be 'is my favorite team playing' and so on.

These inputs can either be excitatory or inhibitory. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if x_3 is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so x_3 is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together. Formally, this is what is going on:

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

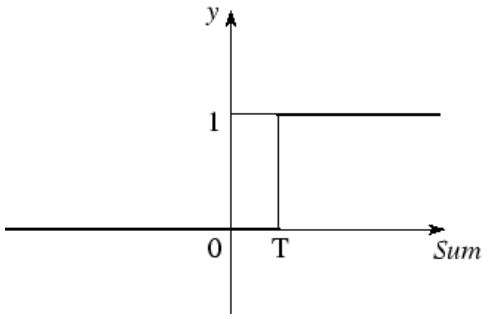
$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

We can see that g(x) is just doing a sum of the inputs — a simple aggregation. And theta here is called thresholding parameter. For example, if I always watch the game when the sum turns out to be 2 or more, the theta is 2 here. This is called the Thresholding Logic.

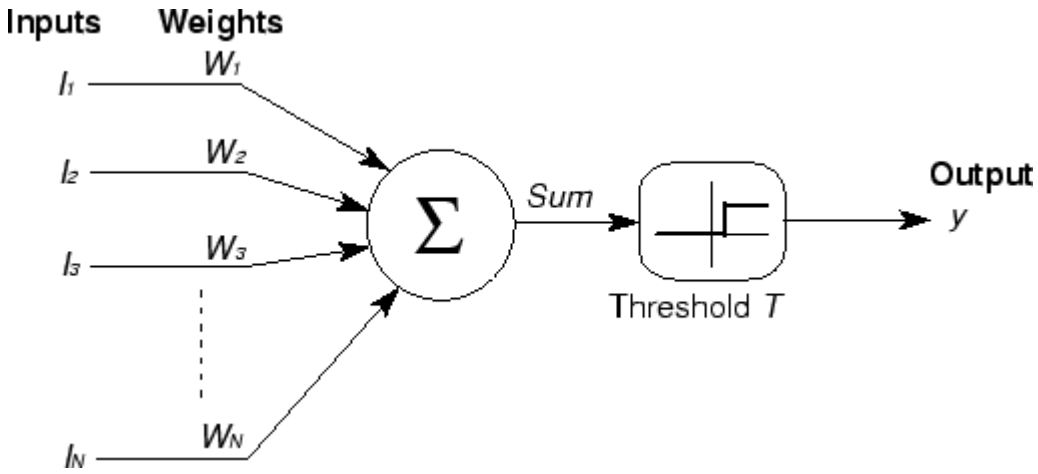
The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs I_1, I_2,I_3,...I_m and one output 'y' . The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output y is binary. Such a function can be described mathematically using these equations:

$$Sum = \sum_{i=1}^{N} I_i W_i, \qquad y = f(Sum).$$

Where, $W_1, W_2, W_3, ..., W_m$ are weight values normalized in the range of either $(0,1)$ or $(-1,1)$ and associated with each input line, Sum is the weighted sum, and T is a threshold constant. The function f is a linear step function at threshold T as shown in figure 2.3. The symbolic representation of the linear threshold gate is shown in figure below.



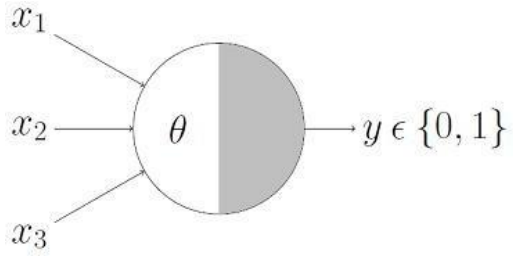
Linear Threshold Function



Symbolic Illustration of Linear Threshold Gate

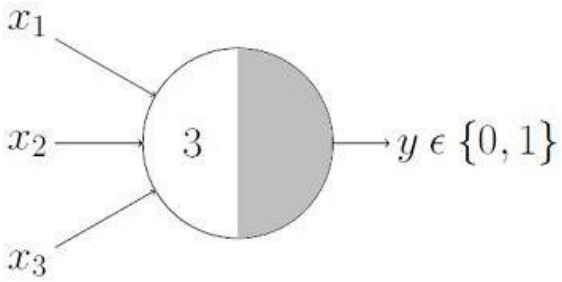
BOOLEAN FUNCTIONS USING McCULLOGH-PITTS NEURON

In any Boolean function, all inputs are Boolean and the output is also Boolean. So essentially, the neuron is just trying to learn a Boolean function.



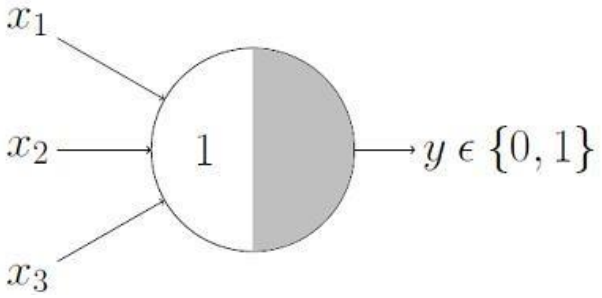
This representation just denotes that, for the boolean inputs x_1, x_2 and x_3 if the g(x) i.e., sum ≥ theta, the neuron will fire otherwise, it won't.

AND Function



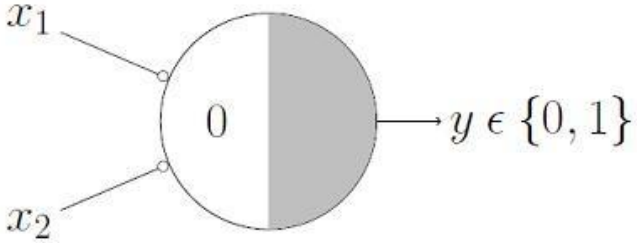
An AND function neuron would only fire when ALL the inputs are ON i.e., g(x) ≥ 3 here.

OR Function



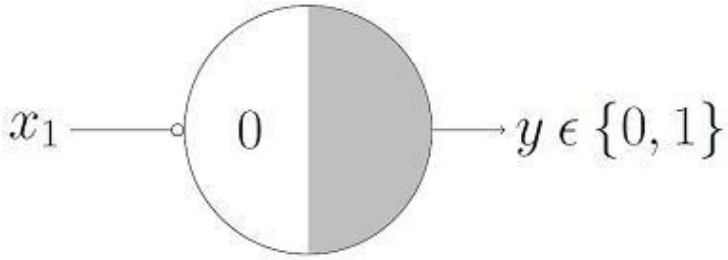
For an OR function neuron would fire if ANY of the inputs is ON i.e., g(x) ≥ 1 here.

NOR Function



For a NOR neuron to fire, we want ALL the inputs to be 0 so the thresholding parameter should also be 0 and we take them all as inhibitory input.

NOT Function



For a NOT neuron, 1 outputs 0 and 0 outputs 1. So we take the input as an inhibitory input and set the thresholding parameter to 0.

We can summarize these rules with the McCullough-Pitts output rule as:

The McCulloch-Pitts model of a neuron is simple yet has substantial computing potential. It also has a precise mathematical definition. However, this model is so simplistic that it only generates a binary output and also the weight and threshold values are fixed. The neural computing algorithm has diverse features for various applications. Thus, we need to obtain the neural model with more flexible computational features.

WHAT ARE THE LEARNING RULES IN ANN?

Learning rule is a method or a mathematical logic. It helps a Neural Network to learn from the existing conditions and improve its performance. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment. Applying learning rule is an iterative process. It helps a neural network to learn from the existing conditions and improve its performance.

The different learning rules in the Neural network are:

1. Hebbian learning rule – It identifies, how to modify the weights of nodes of a network.
2. Perceptron learning rule – Network starts its learning by assigning a random value to each weight.
3. Delta learning rule – Modification in sympatric weight of a node is equal to the multiplication of error and the input.
4. Correlation learning rule – The correlation rule is the supervised learning.
5. Outstar learning rule – We can use it when it assumes that nodes or neurons in a network arranged in a layer.

1. **Hebbian Learning Rule:** The Hebbian rule was the first learning rule. In 1949 Donald Hebb developed it as learning algorithm of the unsupervised neural network. We can use it to identify how to improve the weights of nodes of a network. The Hebb learning rule assumes that – If two neighbor neurons activated and deactivated at the same time, then the weight connecting these neurons should increase. At the start, values of all weights are set to zero. This learning rule can be used for both soft- and hard-activation functions. Since desired responses of neurons are not used in the learning procedure, this is the unsupervised learning rule. The absolute values of the weights are usually proportional to the learning time, which is undesired.

$$W_{ij} = x_i * x_j$$

Mathematical Formula of Hebb Learning Rule.

2. **Perceptron Learning Rule:** Each connection in a neural network has an associated weight, which changes in the course of learning. According to it, an example of supervised learning, the network starts its learning by assigning a random value to each weight. Calculate the output value on the basis of a set of records for which we can know the expected output value. This is the learning sample that indicates the entire definition. As a result, it is called a learning sample. The network then compares the calculated output value with the expected value. Next calculates an error function ∈, which can be the sum of squares of the errors occurring for each individual in the learning sample which can be computed as:

$$\sum_i \sum_j \left(E_{ij} - O_{ij} \right)^2$$

Mathematical Formula of Perceptron Learning Rule

Perform the first summation on the individuals of the learning set, and perform the second summation on the output units. E_{ij} and O_{ij} are the expected and obtained values of the j^{th} unit for the i^{th} individual. The network then adjusts the weights of the different units, checking each time to see if the error function has increased or decreased. As in a conventional regression, this is a matter of solving a problem of least squares. Since assigning the weights of nodes according to users, it is an example of supervised learning.

3. **Delta Learning Rule:** Developed by Widrow and Hoff, the delta rule, is one of the most common learning rules. It depends on supervised learning. This rule states that the modification in sympatric weight of a node is equal to the multiplication of error and the input. In Mathematical form the delta rule is as follows:

$$\Delta w = \eta (t - y) x_i$$

Mathematical Formula of Delta Learning Rule

For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference. The change in weight from ui to uj is: dwij = r* ai * ej. where r is the learning rate, ai represents the activation of ui and ej is the difference between the expected output and the actual output of uj. If the set of input patterns form an independent set then learn arbitrary associations using the delta rule.

It has seen that for networks with linear activation functions and with no hidden units. The error squared vs. the weight graph is a paraboloid in n-space. Since the proportionality constant is negative, the graph of such a function is concave upward and has the least value. The vertex of this paraboloid represents the point where it reduces the error. The weight vector corresponding to this point is then the ideal weight vector. We can use the delta learning rule with both single output unit and several output units. While applying the delta rule assume that the error can be directly measured. The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

4. **Correlation Learning Rule:** The correlation learning rule based on a similar principle as the Hebbian learning rule. It assumes that weights between responding neurons should be more positive, and weights between neurons with opposite reaction should be more negative. Contrary to the Hebbian rule, the correlation rule is the supervised learning, instead of an actual. The response, oj, the desired response, dj, uses for the weight-change calculation. In Mathematical form the correlation learning rule is as follows:

$$\Delta w_{ij} = \eta x_i d_j$$

Mathematical Formula of Correlation Learning Rule

Where d_j is the desired value of output signal. This training algorithm usually starts with the initialization of weights to zero. Since assigning the desired weight by users, the correlation learning rule is an example of supervised learning.

5. Out Star Learning Rule: We use the Out Star Learning Rule when we assume that nodes or neurons in a network arranged in a layer. Here the weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights. The out start rule produces the desired response t for the layer of n nodes. Apply this type of learning for all nodes in a particular layer. Update the weights for nodes are as in Kohonen neural networks. In Mathematical form, express the out star learning as follows:

$$w_{jk} = \begin{cases} \eta(y_k - w_{jk}) & \text{if node j wins the competition} \\ 0 & \text{if node j losses the competition} \end{cases}$$

Mathematical Formula of Out Star Learning Rule

This is a supervised training procedure because desired outputs must be known.

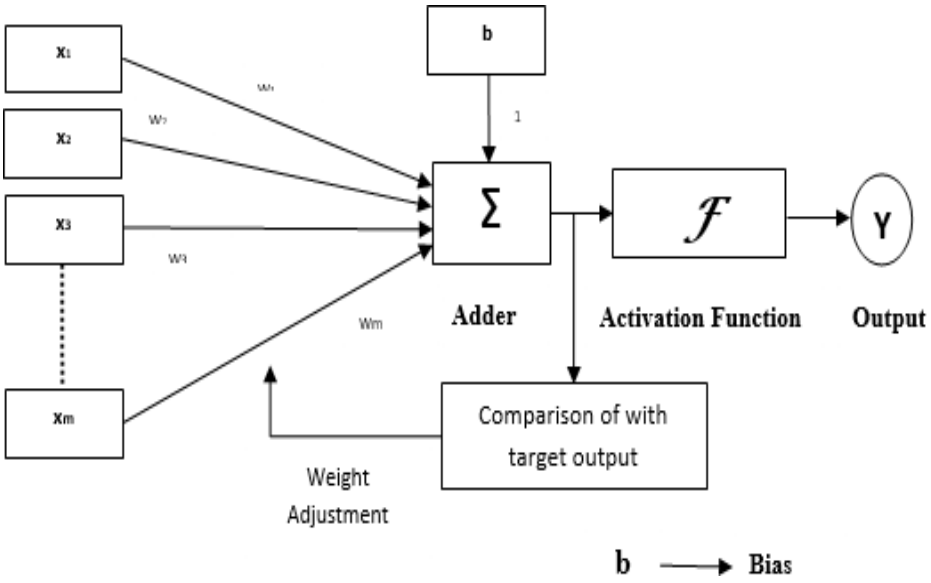
BRIEFLY EXPLAIN THE ADALINE MODEL OF ANN.

ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element) is an early single-layer artificial neural network and the name of the physical device that implemented this network. The network uses memistors. It was developed by Professor Bernard Widrow and his graduate student Ted Hoff at Stanford University in 1960. It is based on the McCulloch–Pitts neuron. It consists of a weight, a bias and a summation function. The difference between Adaline and the standard (McCulloch–Pitts) perceptron is that in the learning phase, the weights are adjusted according to the weighted sum of the inputs (the net). In the standard perceptron, the net is passed to the activation (transfer) function and the function's output is used for adjusting the weights. Some important points about Adaline are as follows:

- It uses bipolar activation function.
- It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.
- The weights and the bias are adjustable.

Architecture of ADALINE network: The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/target output. After comparison on the basis of training algorithm, the weights and bias will be updated.

Architecture of ADALINE: The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/target output. After comparison on the basis of training algorithm, the weights and bias will be updated.



Training Algorithm of ADALINE:

Step 1 – Initialize the following to start the training:

- Weights
- Bias
- Learning rate α

For easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

Step 2 – Continue step 3-8 when the stopping condition is not true.

Step 3 – Continue step 4-6 for every bipolar training pair s : t.

Step 4 – Activate each input unit as follows:

$$x_i = s_i \ (i=1 \text{ to } n)$$

Step 5 – Obtain the net input with the following relation:

$$y_{in} = b + \sum_{i}^{n} x_i \ w_i$$

Here 'b' is bias and 'n' is the total number of input neurons.

Step 6 – Apply the following activation function to obtain the final output:

$$f(y_{in}) = \begin{cases} 1, \text{if } y_{in} \geq 0 \\ -1, \text{if } y_{in} < 0 \end{cases}$$

Step 7 – Adjust the weight and bias as follows :

Case 1 – if $y \neq t$ then, $w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$

$$b(new) = b(old) + \alpha(t - y_{in})$$

Case 2 – if $y = t$ then, $w_i(new) = w_i(old)$

$$b(new) = b(old)$$

Here 'y' is the actual output and 't' is the desired/target output. $(t - y_{in})$ is the computed error.

Step 8 – Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

EXPLAIN MULTIPLE ADAPTIVE LINEAR NEURONS (MADALINE).

Madaline which stands for Multiple Adaptive Linear Neuron, is a network which consists of many Adalines in parallel. It will have a single output unit. Three different training algorithms for MADALINE networks called Rule I, Rule II and Rule III have been suggested, which cannot be learned using backpropagation. The first of these dates back to 1962 and cannot adapt the weights of the hidden-output connection.[10] The second training algorithm improved on Rule I and was described in 1988.[8] The third "Rule" applied to a modified network with sigmoid activations instead of signum; it was later found to be equivalent to backpropagation. The Rule II training algorithm is based on a principle called "minimal disturbance". It proceeds by looping over training examples, then for each example, it:

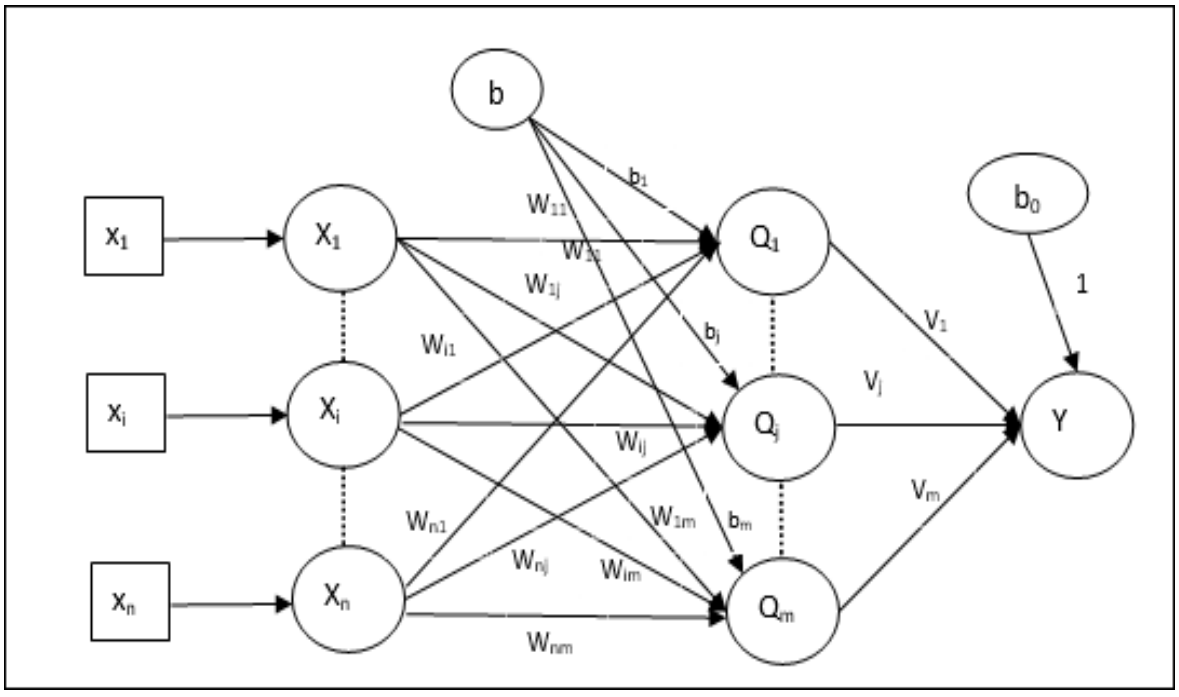
- finds the hidden layer unit (ADALINE classifier) with the lowest confidence in its prediction, tentatively flips the sign of the unit,
- accepts or rejects the change based on whether the network's error is reduced,
- stops when the error is zero.

Some important points about Madaline are as follows:

- It is just like a multilayer perceptron, where Adaline will act as a hidden unit between the input and the Madaline layer.
- The weights and the bias between the input and Adaline layers, as in we see in the Adaline architecture, are adjustable.
- The Adaline and Madaline layers have fixed weights and bias of 1.
- Training can be done with the help of Delta rule.

BRIEFLY EXPLAIN THE ARCHITECTURE OF MADALINE

MADALINE (Many ADALINE) is a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecture for classification that uses ADALINE units in its hidden and output layers, i.e. its activation function is the sign function. The three-layer network uses memistors. The architecture of Madaline consists of “n” neurons of the input layer, “m” neurons of the Adaline layer, and 1 neuron of the Madaline layer. The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.



Training Algorithm of MADALINE

By now we know that only the weights and bias between the input and the Adaline layer are to be adjusted, and the weights and bias between the Adaline and the Madaline layer are fixed.

Step 1 – Initialize the following to start the training:

- Weights
- Bias
- Learning rate α

For easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

Step 2 – Continue step 3-8 when the stopping condition is not true.
Step 3 – Continue step 4-6 for every bipolar training pair s:t.
Step 4 – Activate each input unit as follows:

$$x_i = s_i (I = 1 \text{ to } n)$$

Step 5 – Obtain the net input at each hidden layer, i.e. the Adaline layer with the following relation:

$$O_{inj} = b_j + \sum_{i}^{n} x_i \, w_{ij} \; j = 1 \; to \; m$$

Here 'b' is bias and 'n' is the total number of input neurons.

Step 6 – Apply the following activation function to obtain the final output at the Adaline and the Madaline Layer:

$$f(y_{in}) = \begin{cases} 1, if \; x \geq 0 \\ -1, if \; x < 0 \end{cases}$$

Output at the hidden Adaline unit $Q_j = f(Q_{inj})$
Final output of the network $y = f(y_{in})$
i.e.

$$y_{inj} = b_0 + \sum_{j=1}^{m} Q_j \, v_j$$

Step 7 – Calculate the error and adjust the weights as follows –

Case 1 – if y ≠ t and t = 1 then,

$$w_{ij}(new) = w_{ij}(old) + \alpha(1 - Q_{inj}) x_i$$
$$b_j(new) = b_j(old) + \alpha(1 - Q_{inj})$$

In this case, the weights would be updated on Qj where the net input is close to 0 because t = 1.

Case 2 – if y ≠ t and t = -1 then,

$$w_{ik}(new) = w_{ik}(old) + \alpha(-1 - Q_{ink}) x_i$$
$$b_k(new) = b_k(old) + \alpha(-1 - Q_{ink})$$

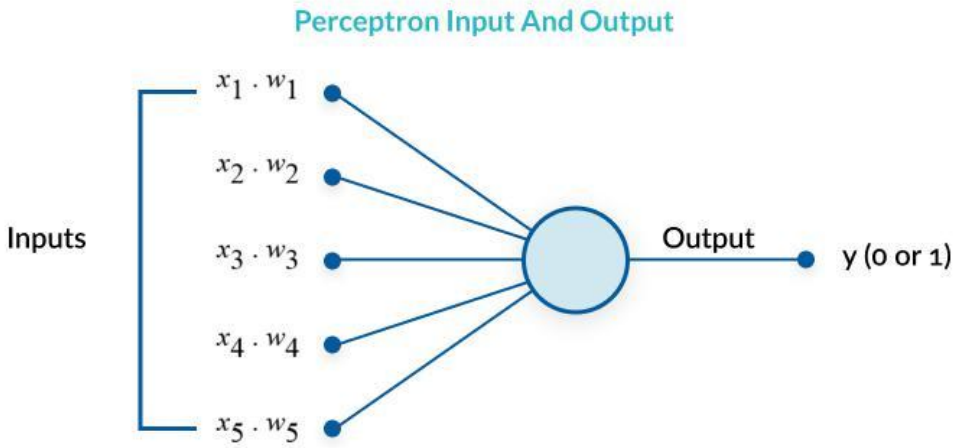
In this case, the weights would be updated on Qk where the net input is positive because t = -1. Here 'y' is the actual output and 't' is the desired/target output.

Case 3 – if y = t, then there would be no change in weights.

Step 8 – Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

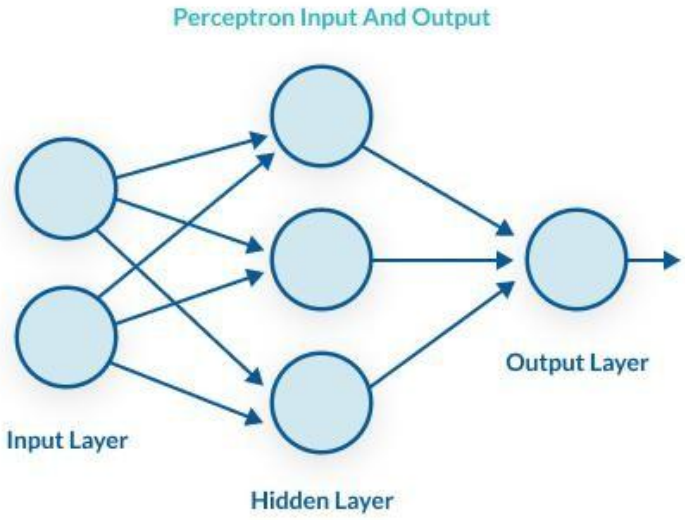
WHAT IS A PERCEPTRON?

A perceptron is a binary classification algorithm modeled after the functioning of the human brain—it was intended to emulate the neuron. The perceptron, while it has a simple structure, has the ability to learn a



What is Multilayer Perceptron?

A multilayer perceptron (MLP) is a group of perceptrons, organized in multiple layers, that can accurately answer complex questions. Each perceptron in the first layer (on the left) sends signals to all the perceptrons in the second layer, and so on. An MLP contains an input layer, at least one hidden layer, and an output layer.



The perceptron learns as follows:

1. Takes the inputs which are fed into the perceptrons in the input layer, multiplies them by their weights, and computes the sum.
2. Adds the number one, multiplied by a "bias weight". This is a technical step that makes it possible to move the output function of each perceptron (the activation function) up, down, left and right on the number graph.
3. Feeds the sum through the activation function—in a simple perceptron system, the activation function is a step function.
4. The result of the step function is the output.

A multilayer perceptron is quite similar to a modern neural network. By adding a few ingredients, the perceptron architecture becomes a full-fledged deep learning system:

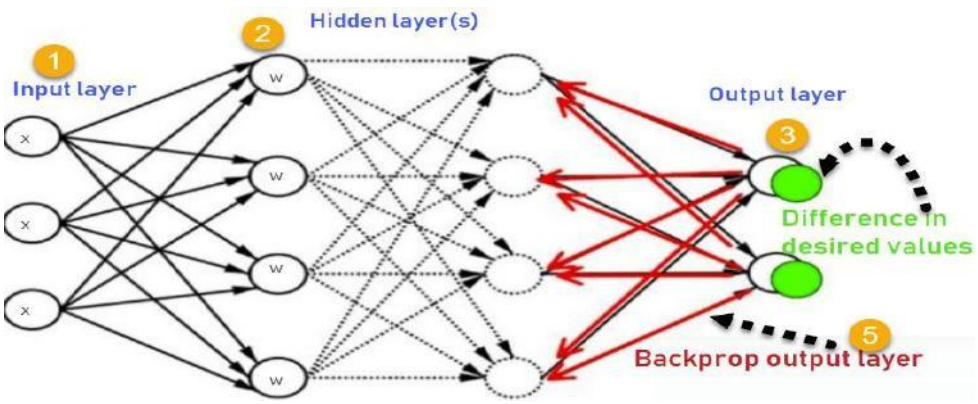
- **Activation functions and other** hyperparameters**:** a full neural network uses a variety of activation functions which output real values, not boolean values like in the classic perceptron. It is more flexible in terms of other details of the learning process, such as the number of training iterations (iterations and epochs), weight initialization schemes, regularization, and so on. All these can be tuned as hyperparameters.
- **Backpropagation**: a full neural network uses the backpropagation algorithm, to perform iterative backward passes which try to find the optimal values of perceptron weights, to generate the most accurate prediction.
- **Advanced architectures**: full neural networks can have a variety of architectures that can help solve specific problems. A few examples are Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), and Generative Adversarial Networks (GAN).

WHAT IS BACKPROPAGATION AND WHY IS IT IMPORTANT?

After a neural network is defined with initial weights, and a forward pass is performed to generate the initial prediction, there is an error function which defines how far away the model is from the true prediction. There are many possible algorithms that can minimize the error function—for example, one could do a brute force search to find the weights that generate the smallest error. However, for large neural networks, a training algorithm is needed that is very computationally efficient. Backpropagation is that algorithm—it can discover the optimal weights relatively quickly, even for a network with millions of weights.

HOW BACKPROPAGATION WORKS?

1. **Forward pass**—weights are initialized and inputs from the training set are fed into the network. The forward pass is carried out and the model generates its initial prediction.
2. **Error function**—the error function is computed by checking how far away the prediction is from the known true value.
3. **Backpropagation with gradient descent**—the backpropagation algorithm calculates how much the output values are affected by each of the weights in the model. To do this, it calculates partial derivatives, going back from the error function to a specific neuron and its weight. This provides complete traceability from total errors, back to a specific weight which contributed to that error. The result of backpropagation is a set of weights that minimize the error function.
4. **Weight update**—weights can be updated after every sample in the training set, but this is usually not practical. Typically, a batch of samples is run in one big forward pass, and then backpropagation performed on the aggregate result. The *batch size* and number of batches used in training, called *iterations*, are important hyperparameters that are tuned to get the best results. Running the entire training set through the backpropagation process is called an *epoch*.



Training algorithm of BPNN:

1. Inputs X, arrive through the pre connected path
2. Input is modeled using real weights W. The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs

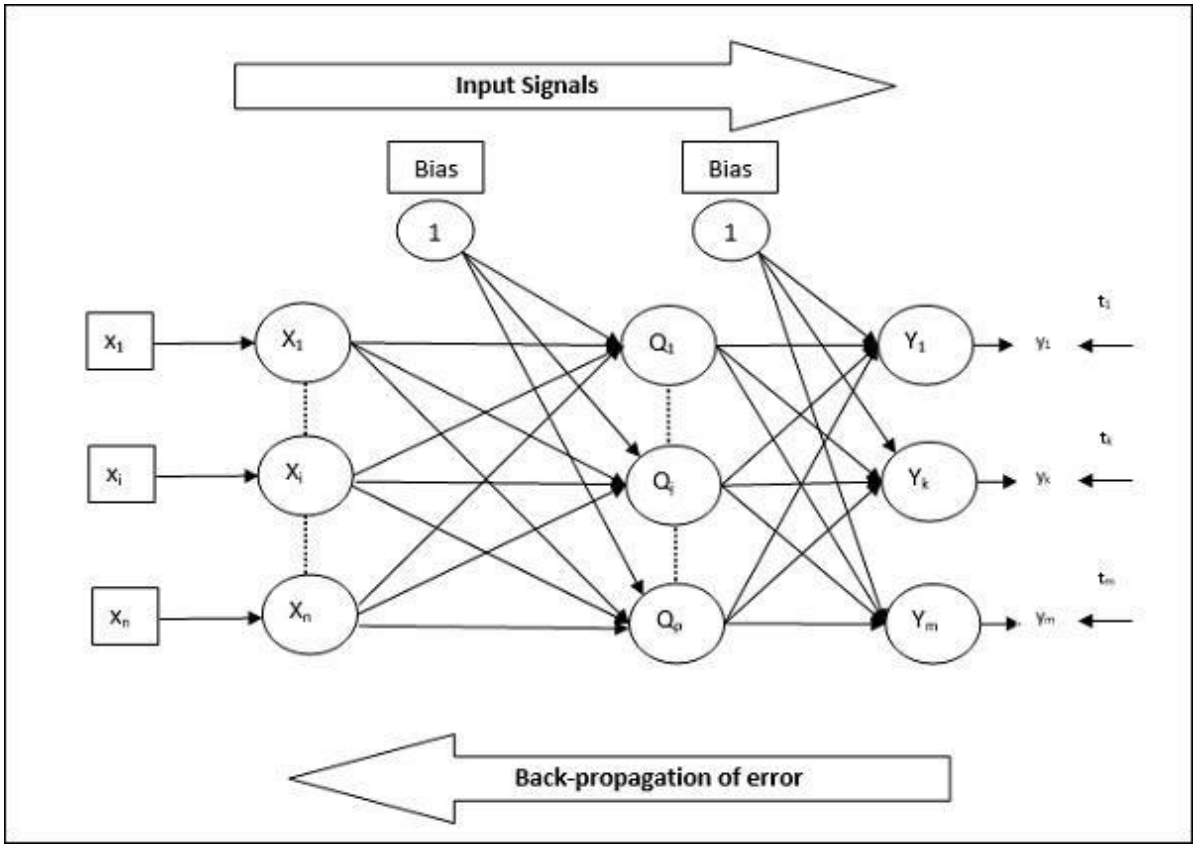
$$Error_B = \text{Actual Output} - \text{Desired Output}$$

5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Keep repeating the process until the desired output is achieved

Architecture of back propagation network:

As shown in the diagram, the architecture of BPN has three interconnected layers having weights on them. The hidden layer as well as the output layer also has bias, whose weight is always 1, on them. As is clear from the diagram, the working of BPN is in two phases. One phase sends the signal from the input layer to the output layer, and the other phase back propagates the error from the output layer to the input layer.



UNIT-2

WHAT ARE THE ANN LEARNING PARADIGMS?

Learning can refer to either acquiring or enhancing knowledge. As Herbert Simon says, Machine Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.
ANN learning paradigms can be classified as supervised, unsupervised and reinforcement learning. Supervised learning model assumes the availability of a teacher or supervisor who classifies the training examples into classes and utilizes the information on the class membership of each training instance, whereas, Unsupervised learning model identify the pattern class information heuristically and Reinforcement learning learns through trial and error interactions with its environment (reward/penalty assignment).
Though these models address learning in different ways, learning depends on the space of interconnection neurons. That is, supervised learning learns by adjusting its inter connection weight combinations with the help of error signals where as unsupervised learning uses information associated with a group of neurons and reinforcement learning uses reinforcement function to modify local weight parameters. Thus, learning occurs in an ANN by adjusting the free parameters of the network that are adapted where the ANN is embedded.

BRIEFLY EXPLAIN SUPERVISED LEARNING.

Supervised learning is based on training a data sample from data source with correct classification already assigned. Such techniques are utilized in feed forward or Multi Layer Perceptron (MLP) models. These MLP has three distinctive characteristics:
1. One or more layers of hidden neurons that are not part of the input or output layers of the network that enable the network to learn and solve any complex problems
2. The nonlinearity reflected in the neuronal activity is differentiable and,
3. The interconnection model of the network exhibits a high degree of connectivity These characteristics along with learning through training solve difficult and diverse problems. Learning through training in a supervised ANN model also called as error backpropagation algorithm. The error correction-learning algorithm trains the network based on the input-output samples and finds error signal, which is the difference of the output calculated and the desired output and adjusts the synaptic weights of the neurons that is proportional to the product of the error signal and the input instance of the synaptic weight. Based on this principle, error back propagation learning occurs in two passes:

Forward Pass: Here, input vector is presented to the network. This input signal propagates forward, neuron by neuron through the network and emerges at the output end of the network as output signal:
$y(n) = \varphi(v(n))$, where $v(n)$ is the induced local field of a neuron defined by $v(n) = \Sigma\, w(n)y(n)$.
The output that is calculated at the output layer $o(n)$ is compared with the desired response $d(n)$ and finds the error $e(n)$ for that neuron. The synaptic weights of the network during this pass are remains same.

Backward Pass: The error signal that is originated at the output neuron of that layer is propagated backward through network. This calculates the local gradient for each neuron in each layer and allows the synaptic weights of the network to undergo changes in accordance with the delta rule as:

$$\Delta w(n) = \eta * \delta(n) * y(n).$$

This recursive computation is continued, with forward pass followed by the backward pass for each input pattern till the network is converged. Supervised learning paradigm of an ANN is efficient and finds solutions to several linear and non-linear problems such as classification, plant control, forecasting, prediction, robotics etc.

BRIEFLY EXPLAIN UNSUPERVISED LEARNING.

Self-Organizing neural networks learn using unsupervised learning algorithm to identify hidden patterns in unlabeled input data. This unsupervised refers to the ability to learn and organize information without providing an error signal to evaluate the potential solution. The lack of direction for the learning algorithm in unsupervised learning can sometime be advantageous, since it lets the algorithm to look back for patterns that have not been previously considered. The main characteristics of Self-Organizing Maps (SOM) are:

1. It transforms an incoming signal pattern of arbitrary dimension into one or 2 dimensional map and perform this transformation adaptively
2. The network represents feed forward structure with a single computational layer consisting of neurons arranged in rows and columns.
3. At each stage of representation, each input signal is kept in its proper context and,
4. Neurons dealing with closely related pieces of information are close together and they communicate through synaptic connections.

The computational layer is also called as competitive layer since the neurons in the layer compete with each other to become active. Hence, this learning algorithm is called competitive algorithm. Unsupervised algorithm in SOM works in three phases:

Competition phase: for each input pattern x, presented to the network, inner product with synaptic weight w is calculated and the neurons in the competitive layer finds a discriminant function that induce competition among the neurons and the synaptic weight vector that is close to the input vector in the Euclidean distance is announced as winner in the competition. That neuron is called best matching neuron,

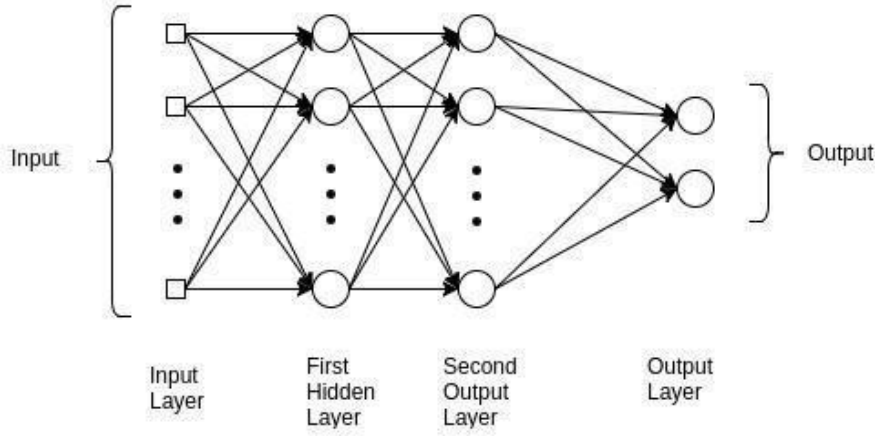
i.e. $x = \arg\min \|x - w\|$.

Cooperative phase: the winning neuron determines the center of a topological neighborhood h of cooperating neurons. This is performed by the lateral interaction d among the cooperative neurons. This topological neighborhood reduces its size over a time period.

Adaptive phase: enables the winning neuron and its neighborhood neurons to increase their individual values of the discriminant function in relation to the input pattern through suitable synaptic weight adjustments, $\Delta w = \eta h(x)(x - w)$. Upon repeated presentation of the training patterns, the synaptic weight vectors tend to follow the distribution of the input patterns due to the neighborhood updating and thus ANN learns without supervisor.

BRIEFLY EXPLAIN MULTI LAYER PERCEPTRON MODEL.

In the Multilayer perceptron, there can more than one linear layer (combinations of neurons). If we take the simple example the three-layer network, first layer will be the input layer and last will be output layer and middle layer will be called hidden layer. We feed our input data into the input layer and take the output from the output layer. We can increase the number of the hidden layer as much as we want, to make the model more complex according to our task.

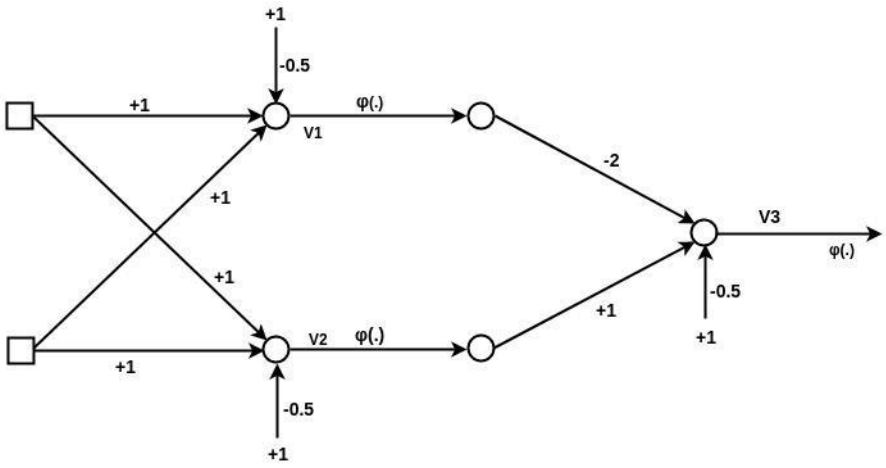


Feed Forward Network, is the most typical neural network model. Its goal is to approximate some function f (). Given, for example, a classifier y = f ∗ (x) that maps an input x to an output class y, the MLP find the best approximation to that classifier by defining a mapping, y = f(x; θ) and learning the best parameters θ for it. The MLP networks are composed of many functions that are chained together. A network with three functions or layers would form f(x) = f (3)(f (2)(f (1)(x))). Each of these layers is composed of units that perform a transformation of a linear sum of inputs. Each layer is represented as y = f(WxT + b). Where f is the activation function, W is the set of parameter, or weights, in the layer, x is the input vector, which can also be the output of the previous layer, b is the bias vector and T is the training function. The layers of an MLP consist of several fully connected layers because each unit in a layer is connected to all the units in the previous layer. In a fully connected layer, the parameters of each unit are independent of the rest of the units in the layer, that means each unit possess a unique set of weights.

Training the Model of MLP: There are basically three steps in the training of the model.

1. Forward pass
2. Calculate error or loss
3. Backward pass

1. Forward pass: In this step of training the model, we just pass the input to model and multiply with weights and add bias at every layer and find the calculated output of the model.



2. Calculate error / loss: When we pass the data instance (or one example) we will get some output from the model that is called Predicted output (pred_out) and we have the label with the data that is real output or expected output(Expect_out). Based upon these both we calculate the loss that we have to backpropagate(using Backpropagation algorithm). There is various Loss Function that we use based on our output and requirement.
3. Backward Pass: After calculating the loss, we back propagate the loss and update the weights of the model by using gradient. This is the main step in the training of the model. In this step, weights will adjust according to the gradient flow in that direction.

Applications of MLP:

1. MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation.
2. MLPs are universal function approximators and they can be used to create mathematical models by regression analysis.
3. MLPs are a popular machine learning solution in diverse fields such as speech recognition, image recognition, and machine translation software.

ELECTRIC LOAD FORECASTING USING ANN

ANNs were first applied to load forecasting in the late 1980's. ANNs have good performance in data classification and function fitting. Some examples of utilizing ANN in power system applications are: Load forecasting, fault classification, power system assessment, real time harmonic evaluation, power factor correction, load scheduling, design of transmission lines, and power system planning. Load forecast has been an attractive research topic for many decades and in many countries all over the world, especially in fast developing countries with higher load growth rate. Load forecast can be generally classified into four categories based on the forecasting time as detailed in the table below.

Load forecasting	Period	Importance
Long term	One year to ten Years	• To calculate and to allocate the required future capacity. • To plan for new power stations to face customer requirements. • Plays an essential role to determine future budget.
Medium term	One week to few months	Fuel allocation and maintenance schedules.
Short term	One hour to a week	• Accurate for power system operation. • To evaluate economic dispatch, hydrothermal co-ordination, unit commitment, transaction. • To analysis system security among other mandatory function.
Very short term	One minute to an hour	Energy management systems (EMS).

An ANN for load forecasting can be trained on a training set of data that consists of time-lagged load data and other non-load parameters such as weather data, time of day, day of week, month, and actual load data. Some ANNs are only trained against days with data similar to the forecast day. Once the network has been trained, it is tested by presenting it with predictor data inputs. The predictor data can be time-lagged load data and forecasted weather data (for the next 24 hours). The forecasted load output from the ANN is compared to the actual load to determine the forecast error. Forecast error is sometimes presented in terms of the root mean square error (RMSE) but

more commonly in terms of the mean absolute percent error (MAPE). An ANN trained on a specific power system's load and weather data will be system dependent. The ANN generated for that system will most likely not perform satisfactorily on another power system with differing characteristics. It is possible the same ANN architecture may be reused on the new system, but retraining will be required.

Training and Testing with ANN

The whole data set was divided into two sets: Training set and Test Set. Training set consists of 80% of whole data and Test set contains the rest data. The training set was used to make a model which, therefore, predicts the load in the future. The model is made by a MATLAB app Neural Net Fitting. The training set has got inputs which are as follows:

1. Temperature (in ^{o} C)
2. Humidity (in %)
3. Pressure (in mBar)
4. Time (in hours)
5. Global Horizontal (in W/m2)
6. Previous Day Same Hour Load (in kW)
7. Previous Week Same Day Same Hour Load (in kW)

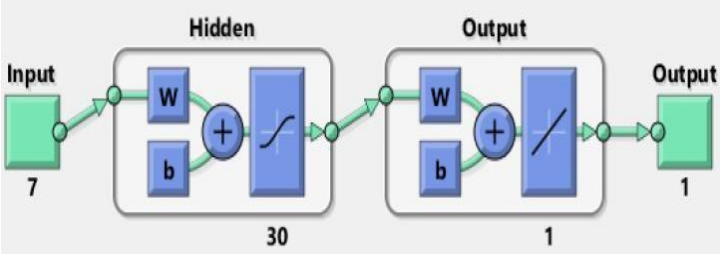
Data Collected: All the following data are collected from substation of feeders:
Maximum, Voltage Minimum, Voltage Maximum, Current Minimum, present MWH consumption, & Temperature.

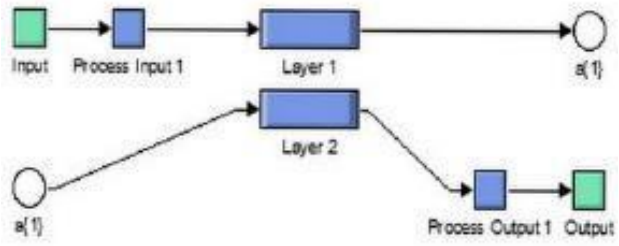
Steps for Implementation of load forecasting using ANN:

- Gathering and arranging the data in MS Excel spreadsheet.
- Tagging the data into groups.
- Analyse the data.
- SIMULINK / MATLAB simulation of data using ANN.

Procedure of testing load forecasting using ANN:

- ANN was created for the user-defined forecast day.
- The data was performed on the training and forecast predictor datasets, the number of hidden layers, or neurons, in the ANN was defined to be 30 neurons.
- The built-in MATLAB Levenberg - Marquardt optimization training function was used to perform the backpropagation training of the feed-forward ANN.
- This process iteratively updated the internal weight and bias values of the ANN to obtain a low error output when utilizing the training predictor dataset and a target dataset.
- The target dataset consists of the actual load values for a given predictor dataset.
- After testing, the ANN forecasted plot was plotted against test set data plot and MAPE was calculated. The results of this forecast were stored, and the entire ANN training, testing, and forecasting process was repeated a set number of times with the intention of reducing the forecast error. The Simulink Model was extracted from the net fitting toolbox is shown below.



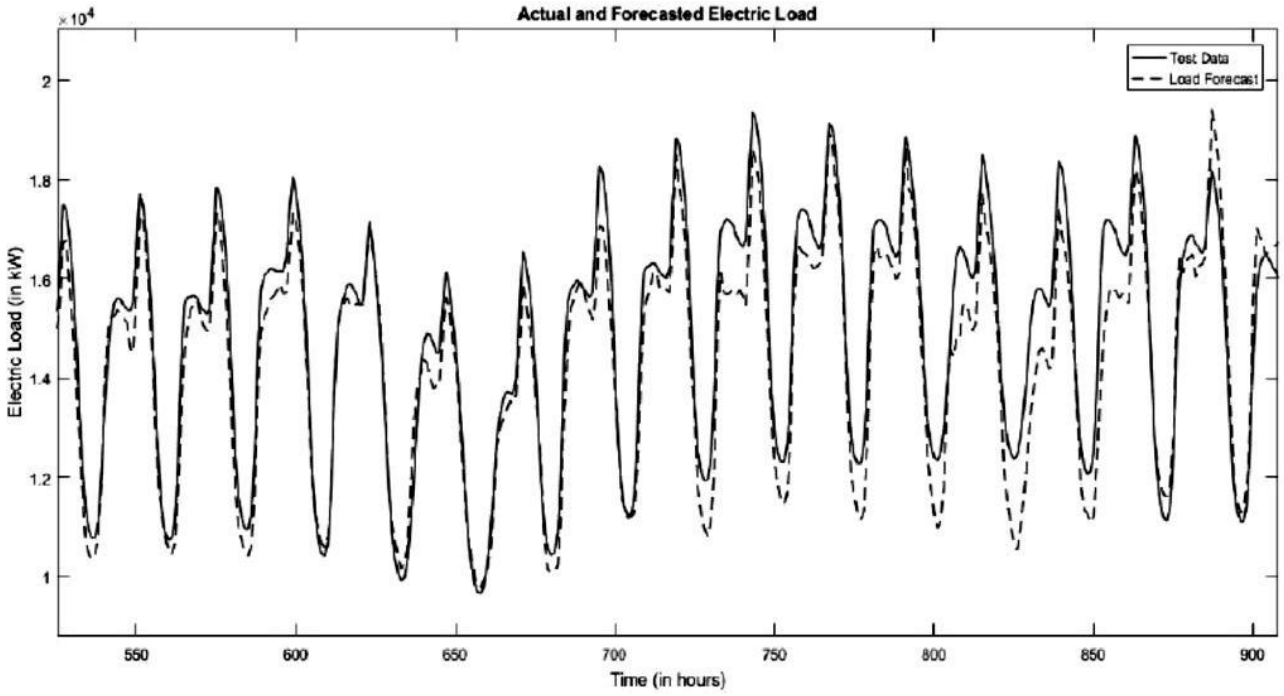


In comparing different models, the average percentage forecasting error is used as a measure of the performance. This is defined as:

$$E_{mv} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\left|\hat{L}_i - L_i\right|}{L_i}\right] \times 100\%$$

where,
N = the number of cases to be forecast (number of hours in
the case of hourly load forecasting).
L_i = the i^{th} load value
$\hat{L}_i$ = the i^{th} load forecast

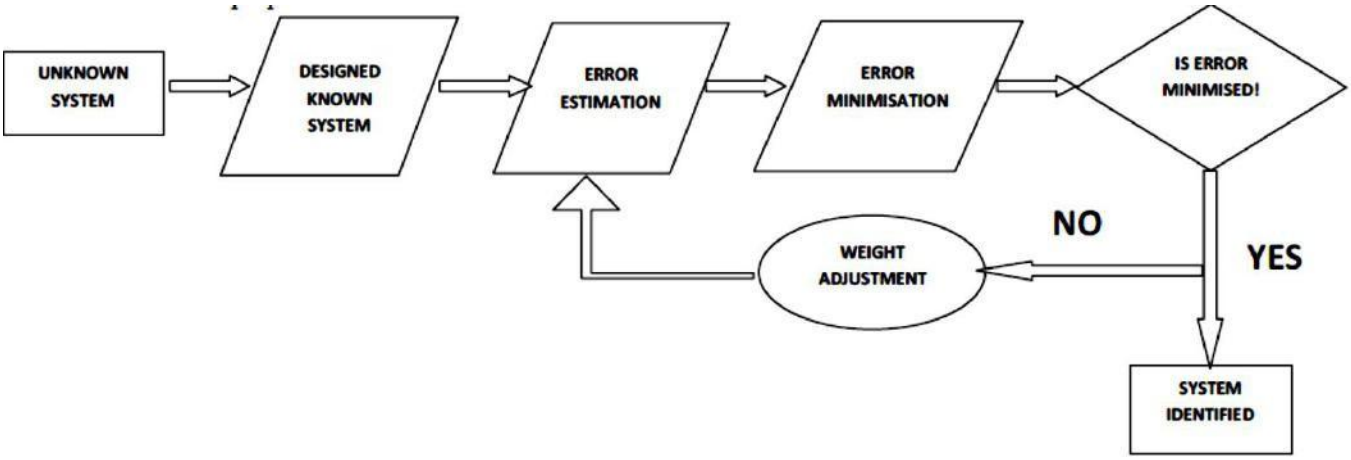
Result: A graph of forecasted load was plotted against the time (in hours) and comparison was made against the Actual Load (test data load). A part of this graph is shown below. The graph shows a little deviation of the forecasted plot from the Test data load. The MAPE (mean absolute percentage error) came out to be 5.1440 % which is bearable.



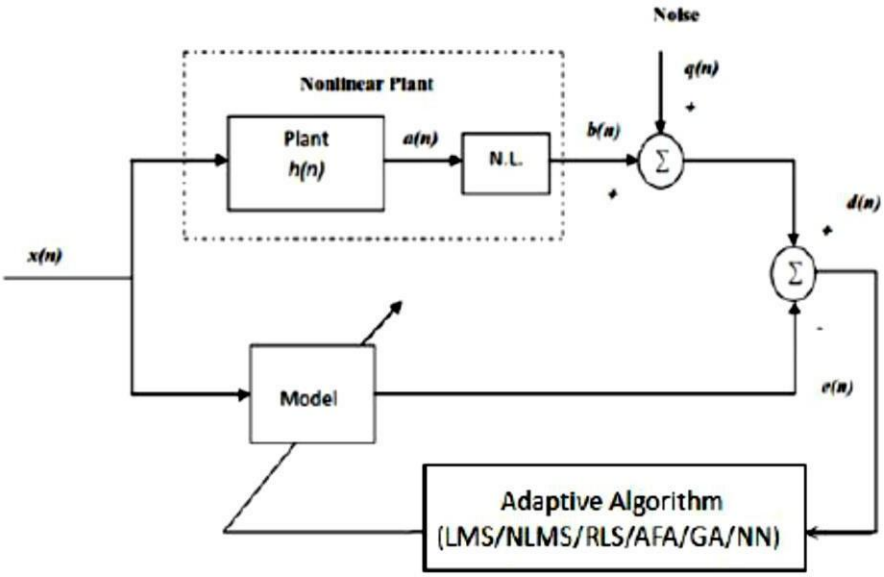
EXPLAIN IN DETAIL ABOUT SYSTEM IDENTIFICATION USING ANN:

A system identification problem can be formulated as an optimization task where the objective is to find a model and a set of parameters that minimize the prediction error between the measured data and the model output. Recurrent neural network (RNN) based adaptive algorithm, is now widely used in system identification due to its robustness and calculus simplicity. Based on the error signal, the filter's coefficients are updated and corrected, in order to adapt, so the output signal has the same values as the reference signal.

System identification is the process of deriving a mathematical model of a system using observed data. In system modeling three main principles have to be considered such as separation, selection and parsimony. System Identification is an essential requirement in areas such as control, communication, power system and instrumentation for obtaining a model of a system (plant) of interest or a new system to be developed. The identification task is to determine a suitable estimate of finite dimensional parameters which completely characterize the plant. The selection of the estimate is based on comparison between the actual output sample and a predicted value on the basis of input data up to that instant.



Basic flow chart of system identification using neural network adaptive algorithm

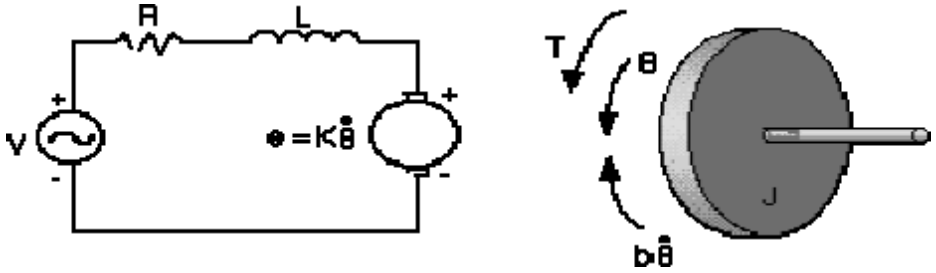


Basic System Identification Process

EXPLAIN IN DETAIL ABOUT ANN APPLICATION IN CONTROL SYSTEMS:

The process industry implements many techniques with certain parameters in its operations to control the working of several actuators on field. Amongst these actuators, DC motor is a very common machine. The angular position of DC motor can be controlled to drive many processes such as the arm of a robot. The most famous and well known controller for such applications is PID controller. It uses proportional, integral and derivative functions to control the input signal before sending it to the plant unit. Neural networks model human neural systems through computerized algorithm. They are capable of parallel computations and distributive storage of information like human brain. In recent years, they have been widely used for optimum calculations and processes in industrial controls, communications, chemistry and petroleum.

There are various types of control mechanisms that may be applied on the speed and angular position of a DC motor, depending upon the accuracy required.



Electric circuit of a DC motor

The electric circuit of a DC motor is shown that governs its rotation for desired velocity or position.
The dynamics of a DC motor may be explained by the following equations:

$$v = Ri + L\frac{di}{dt} + e$$

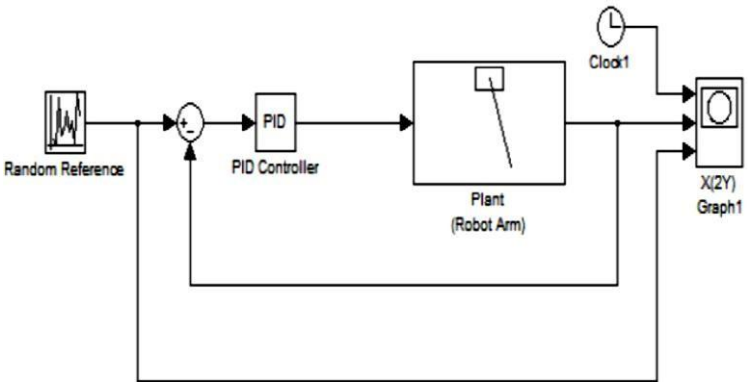
$$T = Ki$$

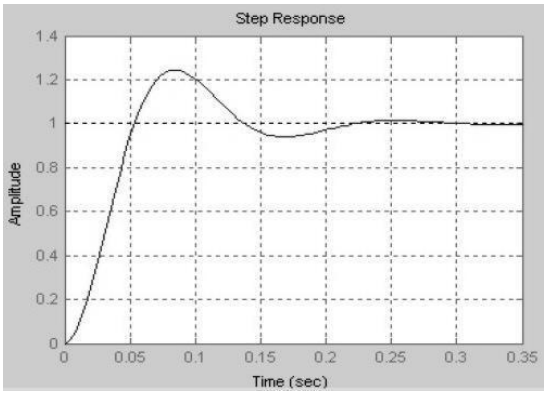
Where, v is voltage applied across armature, R is Armature Resistance, i is Armature Current, L is Armature Inductance and e is back electromotive force (emf) produced across the armature terminals upon its rotation. In second equation, T is Torque, K is motor constant representing torque constant and back emf constant, i is Armature Current.

The DC motor's torque is also represented by the following relation:

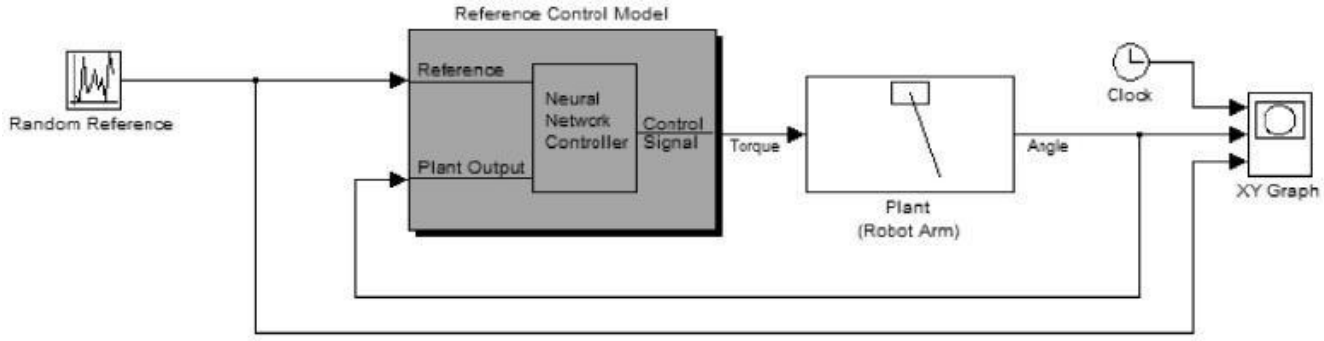
$$T = J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt}$$

Where T is Torque, J is moment of inertia of motor and its load, θ is angular displacement of motor's shaft and b is frictional constant of motor and its load. In order to control the velocity or position of a DC motor, a torque is applied across its armature with controlled parameters. This torque is controlled by a calculated voltage signal at the input. The most common control application for speed and position controls of DC motors with high accuracy is PID (Proportional-Integral-Derivative) control.

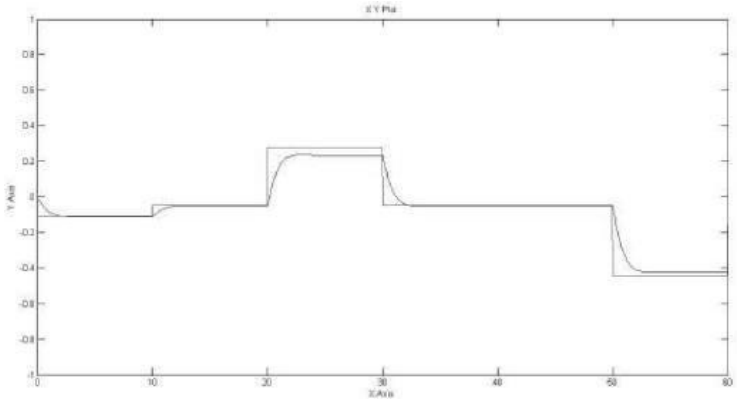




Artificial Neural Networks are famous learning models for their ability to cope with the demands of a changing environment. This network works with supervised learning where data set is presented to train the network before simulation is run to get output results. The block diagram below shows the implementation of ANN control for Robot Arm (DC motor position control) through the available model in MATLAB analysis.



In order to train the controller block of Artificial Neural Network controller, user is free to input the desired values as per the operational requirements before the start of controller's training. In the initial step, the data is generated to train the controller. While data generation process, plant response follows the reference model which is necessary for training's data set to be valid. If the response is not accurate, the data set may be regenerated. If data set is acceptable, the controller may be trained through 'Train Controller' option. The training of Artificial Neural Network controller then starts according to the given parameters. However, it is done after 'Plant Identification' i.e. training the plant unit of ANN controller through the same procedure. The training of ANN controller may take significant amount of time depending upon the given parameters and processing speed.



The simulation results in MATLAB showed that the output of plant in the examined ANN control follows the input reference signal with acceptable results in terms of time delay factor and system dynamics. Since ANN control learns from experience as it is trained through data set in supervised learning, ANN control is more responsive than PID to unknown dynamics of the system which makes it even more suitable for industrial control applications having uncertainties and unknown dynamics due to environmental noise.

BRIEFLY EXPLAIN PATTERN RECOGNITION USING ANN.

Pattern recognition is the automated recognition of patterns and regularities in data. Pattern recognition is closely related to artificial intelligence and machine learning, together with applications such as data mining and knowledge discovery in databases (KDD), and is often used interchangeably with these terms. However, these are distinguished: machine learning is one approach to pattern recognition, while other approaches include hand-crafted (not learned) rules or heuristics; and pattern recognition is one approach to artificial intelligence, while other approaches include symbolic artificial intelligence.

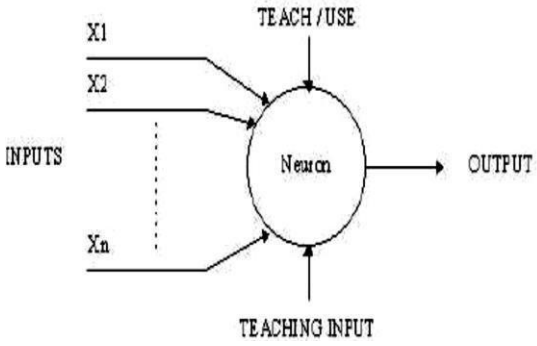
A modern definition of pattern recognition is: The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.

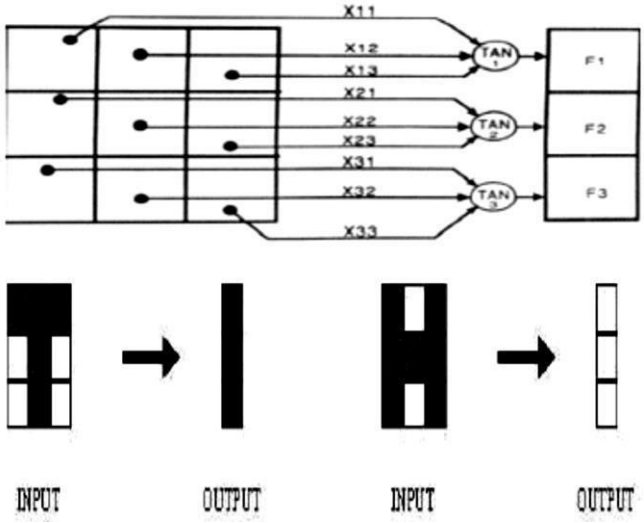
Pattern recognition is generally categorized according to the type of learning procedure used to generate the output value. Supervised learning assumes that a set of training data (the training set) has been provided, consisting of a set of instances that have been properly labeled by hand with the correct output.
Unsupervised learning, on the other hand, assumes training data that has not been hand-labeled, and attempts to find inherent patterns in the data that can then be used to determine the correct output value for new data instances.

Algorithms for pattern recognition based on statistical modelling of data. With statistical model in hand, one applies probability theory and decision theory to get an algorithm. This is opposed to using heuristics/"common sense" to design an algorithm. The following learning types are associated with pattern recognition using ANN.

1. Supervised learning
2. Unsupervised learning
3. Generative model
4. Discriminative model.

Pattern recognition can be implemented by using a feed-forward neural network that has been trained accordingly. During training, the network is trained to associate outputs with input patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural networks comes to life when a pattern that has no output associated with it, is given as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern. During training, the network is trained to associate outputs within put patterns. When the network is used, it identifies the input pattern and tries to output the associated output pattern. The power of neural network comes to life when a pattern that has no output associated with it, is give it as an input. In this case, the network gives the output that corresponds to a taught input pattern that is least different from the given pattern.





If we representblacksquareswith0 and white squares with1 then the truth tables forth e 3neurons after generalization are;

Top neuron:

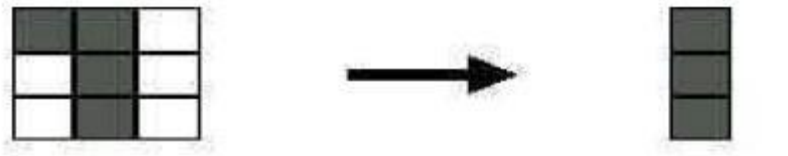
X11:		0	0	0	0	1	1	1	1
X12:		0	0	1	1	0	0	1	1
X13:		0	1	0	1	0	1	0	1

Middleneuron:

X21:		0	0	0	0	1	1	1	1
X22:		0	0	1	1	0	0	1	1
X23:		0	1	0	1	0	1	0	1
OUT:		1	0 /	1	0 /	0 /	0	0 /	0

Bottomneuron:

X21:		0	0	0	0	1	1	1	1
X22:		0	0	1	1	0	0	1	1
X23:		0	1	0	1	0	1	0	1
OUT:		1	0	1	1	0	0	1	0



INPUT OUTPUT

In this case, it is obvious that the output should be all blacks since the input pattern is almost the same as the 'T' pattern. Here also, it is obvious that the output should be all whites since the input pattern is almost the same as the 'H'pattern. Many common pattern recognition algorithms are probabilistic in nature, in that they use statistical inference to find the best label for a given instance. Unlike other algorithms, which simply output a "best" label, often probabilistic algorithms also output a probability of the instance being described by the given label. In addition, many probabilistic algorithms output a list of the N-best labels with associated probabilities, for some value of N, instead of simply a single best label. When the number of possible labels is fairly small (e.g., in the case of classification), N may be set so that the probability of all possible labels is output.

Classification of pattern recognition algorithms (supervised algorithms predicting categorical labels)

1. Parametric:
    - Linear discriminant analysis
    - Quadratic discriminant analysis
    - Maximum entropy classifier
2. Nonparametric:
    - Decision trees, decision lists
    - Kernel estimation and K-nearest-neighbor algorithms
    - Naive Bayes classifier
    - Neural networks (multi-layer perceptrons)
    - Perceptrons
    - Support vector machines
    - Gene expression programming

3. Clustering algorithms (unsupervised algorithms predicting categorical labels)
    - Categorical mixture models
    - Hierarchical clustering (agglomerative or divisive)
    - K-means clustering
    - Correlation clustering
    - Kernel principal component analysis (Kernel PCA)

4. Ensemble learning algorithms (supervised meta-algorithms for combining multiple learning algorithms together)
    - Boosting (meta-algorithm)
    - Bootstrap aggregating ("bagging")
    - Ensemble averaging
    - Mixture of experts, hierarchical mixture of experts

5. General algorithms for predicting arbitrarily-structured (sets of) labels
    - Bayesian networks
    - Markov random fields
6. Multilinear subspace learning algorithms (predicting labels of multidimensional data using tensor representations)

    Unsupervised:
    - Multilinear principal component analysis (MPCA)

7. Real-valued sequence labeling algorithms (predicting sequences of real-valued labels)

    Supervised:
    - Kalman filters
    - Particle filters
    - Regression algorithms (predicting real-valued labels)

8. Regression analysis

    Supervised:
    - Gaussian process regression (kriging)
    - Linear regression and extensions
    - Neural networks and Deep learning methods

    Unsupervised:
    - Independent component analysis (ICA)
    - Principal components analysis (PCA)

9. Sequence labeling algorithms (predicting sequences of categorical labels)

    Supervised:
    - Conditional random fields (CRFs)
    - Hidden Markov models (HMMs)
    - Maximum entropy Markov models (MEMMs)
    - Recurrent neural networks (RNNs)

    Unsupervised:
    - Hidden Markov models (HMMs)
    - Dynamic time warping (DTW)

FUZZY LOGIC

WHAT IS FUZZY LOGIC?

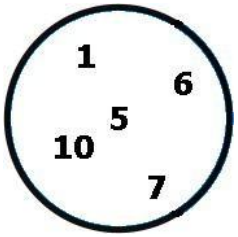
Fuzzy logic is an extension of Boolean logic by Lotfi Zadeh in 1965 based on the mathematical theory of fuzzy sets, which is a generalization of the classical set theory. Fuzzy logic is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1 both inclusive. It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false. Fuzzy logic is based on the observation that people make decisions based on imprecise and non-numerical information.

One advantage of fuzzy logic in order to formalize human reasoning is that the rules are set in natural language. For example, here are some rules of conduct that a driver follows, assuming that he does not want to lose his driver's licence:

If the light is red...	if my speed is high...	and if the light is close...	then I brake hard.
If the light is red...	if my speed is low...	and if the light is far...	then I maintain my speed.
If the light is orange...	if my speed is average...	and if the light is far...	then I brake gently.
If the light is green...	if my speed is low...	and if the light is close...	then I accelerate.

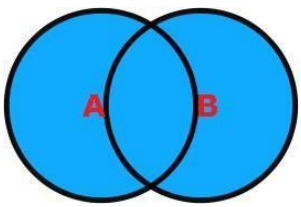
BRIEFLY EXPLAIN CLASSICAL / CRISP SETS

A classical set is a collection of objects in a given range with a sharp boundary. An object can either belong to the set or not belong to the set. For example, 5, 10, 7, 6, 9 is a set of integers. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 is the set of integers between 0 and 10. 's', 'd', 'z', 'a' is a set of characters. "Site", "of", "zero" is a set of words. We can also create sets of functions, assumptions, definitions, sets of individuals (that is to say, a population), etc. and even sets of sets. Sets are often represented in graphic form, typically by circles, as figure below illustrates.

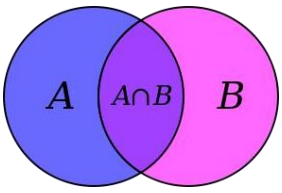


Graphical representation of the set {1, 5, 6, 7, 10}

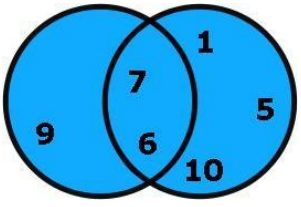
The concept of belonging is important in set theory: it refers to the fact that an element is part of a set or not. For example, the integer 7 belongs to the set 6, 7, 9. In contrast, the integer 5 does not belong to the set 6, 7, 9. Membership is symbolized by the character in the non-membership and by the same symbol, but barred possible. Thus, we have 7 ∈ {6, 7, 9} and 5 ∈/ {6, 7, 9}. Below are few examples of set theory operations.



Union of two sets, denoted A ∪ B



Intersection of two sets, denoted A ∩ B



Representation of the sets A = {6, 7, 9} and B = {1, 5, 6, 7, 10}.

It is clear that an element either belongs to a set or does not belong to that set in the classical set and its operation. There is a sharp boundary between different elements for different sets and these cannot be mixed with each other. However, for the fuzzy set, it has different laws.

WHAT ARE FUZZY SETS / MODELS?

A fuzzy set is a combination of the elements having a changing degree of membership in the set. Here "fuzzy" means vagueness, in other words, the transition among various degrees of the membership complies that the limits of the fuzzy sets are vague and ambiguous. Therefore, the membership of the elements from the universe in the set is measured against a function to identify the uncertainty and ambiguity. A fuzzy set is denoted by a text having tilde under strike. Now, a fuzzy set X would contain all the possible outcome from interval 0 to 1. Suppose a, is an element in the universe is a member of fuzzy set X, the function gives the mapping by X (a) = [0, 1]. The notion convention used for fuzzy sets when the universe of discourse U (set of input values for the fuzzy set X) is discrete and finite, for fuzzy set X is given by:

$$\mu \underset{\sim}{X}(a) = \in [0,1]$$

$$\underset{\sim}{X} = \{ \frac{\mu_A(a_1)}{a_1} + \frac{\mu_A(a_1)}{a_1} + \ldots\ldots\}$$

$$= \{ \sum_i \frac{\mu_X(a_i)}{a_i}$$

The fuzzy set theory was initially proposed by a computer scientist Lotfi A. Zadeh in the year of 1965. After that lot of theoretical development has been done in a similar field. Previously the theory of crisp sets based in dual logic is used in the computing and formal reasoning which involves the solutions in either of two form such as "yes or no" and "true or false".

Fuzzy logic: Unlike crisp logic, in fuzzy logic, approximate human reasoning capabilities are added in order to apply it to the knowledge-based systems. But, what was the need to develop such a theory? The fuzzy logic theory provides a mathematical method to apprehend the uncertainties related to the human cognitive process, for example, thinking and reasoning and it can also handle the issue of uncertainty and lexical imprecision.

Example: Let's take an example to understand fuzzy logic. Suppose we need to find whether the colour of the object is blue or not. But the object can have any of the shade of blue depending on the intensity of the primary colour. So, the answer would vary accordingly, such as royal blue, navy blue, sky blue, turquoise blue, azure blue, and so on. We are assigning the darkest shade of blue a value 1 and 0 to the white colour at the lowest end of the spectrum of values. Then the other shades will range in 0 to 1 according to intensities. Therefore, this kind of situation where any of the values can be accepted in a range of 0 to 1 is termed as fuzzy.

EXPLAIN THE PROPERTIES OF FUZZY SETS

Fuzzy set have a number of properties. Here are definitions of the most important properties, but they are not necessary for understanding of the course. If you want, you can go now directly to the next section. Let X be a set and A, a fuzzy subset of X and μA the membership function characterizing it. μA(x) is called the membership degree of x in A.

Definition 1: Let X be a set. A fuzzy subset A of X is characterized by a membership function.
fa : X → [0, 1]. (In theory, it is possible that the output is greater than 1, but in practice it is almost never used.)

Note: This membership function is equivalent to the identity function of a classical set.

Definition 2: The height of A, denoted h(A), corresponds to the upper bound of the codomain of its membership function: $h(A) = \sup\{\mu A(x) \mid x \in X\}$.

Definition 3: A is said to be normalized if and only if h(A) = 1. In practice, it is extremely rare to work on non-normalized fuzzy sets.

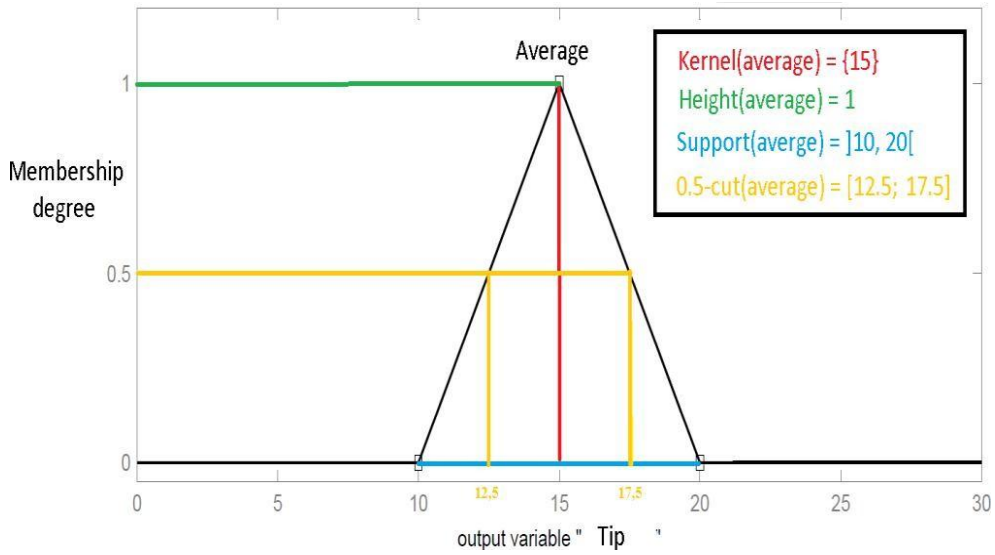
Definition 4: The support of A is the set of elements of X belonging to at least some A (i.e. the membership degree of x is strictly positive). In other words, the support is the set supp(A) = {x ∈ X | μA(x) > 0}.

Definition 5: The kernel of A is the set of elements of X belonging entirely to A. In other words, the kernel noy(A) = {x ∈ X | μA(x) = 1}. By construction, noy(A) ⊆ supp(A).

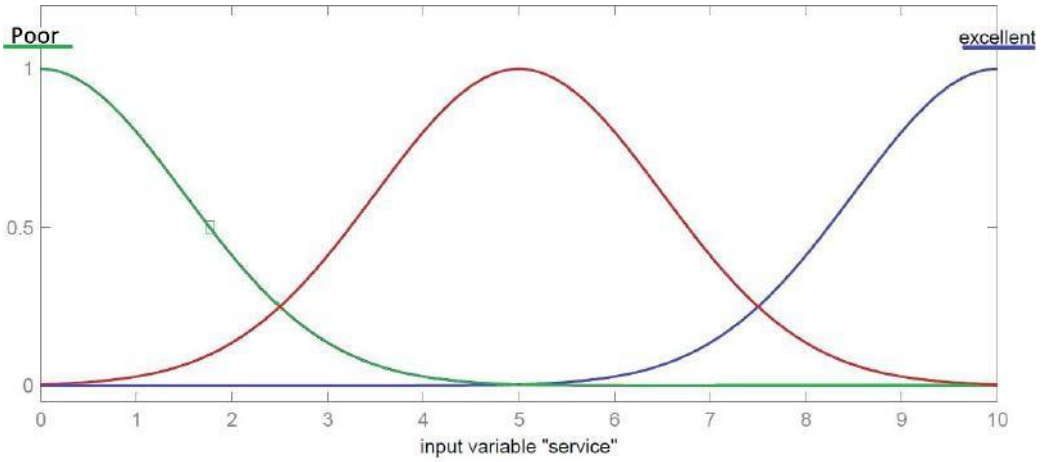
Definition 6: An α-cut of A is the classical subset of elements with a membership degree greater than or equal toα : α-cut(A) = {x ∈ X | μA(x) " α}.

Definition 7: Let V be a variable (quality of service, tip amount, etc.), X the range of values of the variable and TV a finite or infinite set of fuzzy sets. A linguistic variable
corresponds to the triplet (V, X, TV).

Another membership function for an average tip through which we have included the above properties is presented in Figure



A membership function with properties displayed

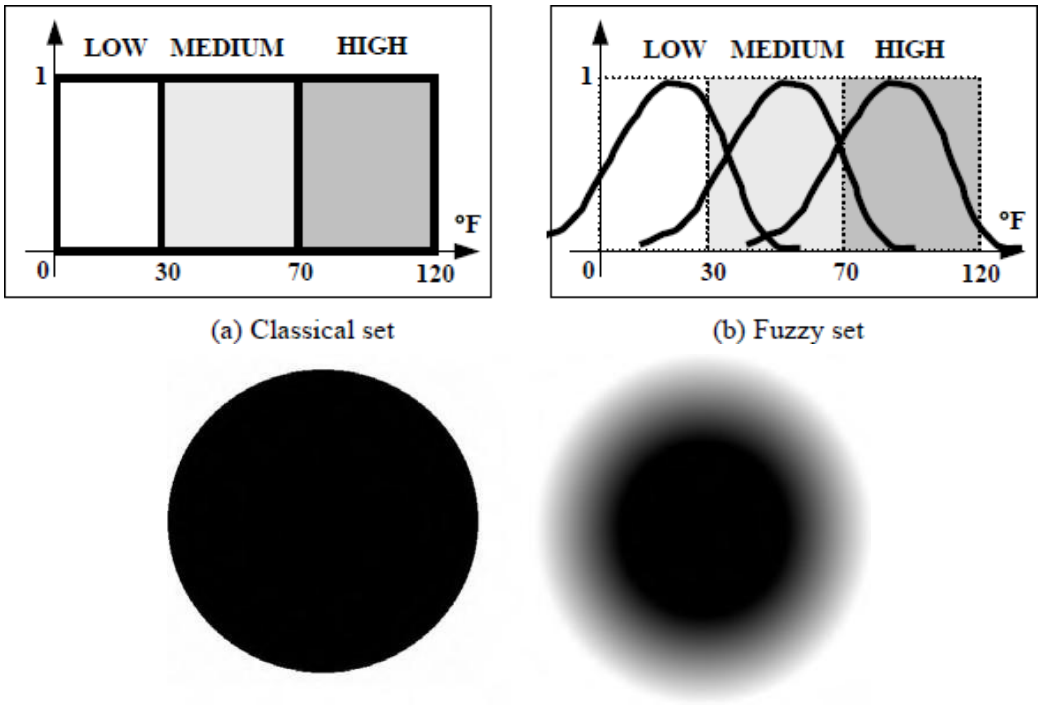


$V = service$
$X = \Re^{+}$
$T_V = \{$ Poor , Good , *excellent* $\}$

Linguistic variable 'quality of service'

COMPARE BETWEEN CRISP/CLASSICAL & FUZZY SETS.

Compared with a classical set, a fuzzy set allows members to have a smooth boundary. In other words, a fuzzy set allows a member to belong to a set to some partial degree. Previously the theory of crisp sets based in dual logic is used in the computing and formal reasoning which involves the solutions in either of two form such as "yes or no" and "true or false". For instance, still using the temperature as an example, the temperature can be divided into three categories: LOW (0 ~30°F), MEDIUM (30°F ~ 70°F) and HIGH (70 ~ 120°F) from the point of view of the classical set, which is shown in Figure below.



Graphical representation of a conventional set and a fuzzy set

In the classical set, any temperature can only be categorized into one subset, LOW, MEDIUM or HIGH, and the boundary is crystal clear. But in the fuzzy set such as shown in Figure (b), these boundaries become vague or smooth. One temperature can be categorized into two or maybe even three subsets simultaneously. For example, the temperature 40°F can be considered to belong to LOW to a certain degree, say 0.5 degree, but at the same time it can belong to MEDIUM to about 0.7 degree. Another interesting thing is the temperature 50°F, which can be considered to belong to LOW and HIGH to around degree and belong to MEDIUM to almost 1 degree. The dash-line in Figure b represents the classical set boundary. It is clear that a fuzzy set contains elements which have varying degrees of membership in the set, and this is contrasted with the classical or crisp sets because members of a classical set cannot be members unless their membership is full or complete in that set. A fuzzy set allows a member to have a partial degree of membership and this partial degree membership can be mapped into a function or a universe of membership values.

Comparison Chart

BASIS FOR COMPARISON	FUZZY SET	CRISP SET
Basic	Prescribed by vague or ambiguous properties.	Defined by precise and certain characteristics.
Property	Elements are allowed to be partially included in the set.	Element is either the member of a set or not.
Applications	Used in fuzzy controllers	Digital design
Logic	Infinite-valued	bi-valued

EXPLAIN FUZZY LOGIC SYSTEM:

The implementation of fuzzy logic technique to a real application, requires the following three steps:

1. Fuzzification – convert classical data or crisp data into fuzzy data or Membership Functions (MFs)
2. Fuzzy Inference Process – combine membership functions with the control rules to derive the fuzzy output
3. Defuzzification – use different methods to calculate each associated output and put them into a table: the lookup table. Pick up the output from the lookup table based on the current input during an application

As mentioned before, all machines can process crisp or classical data such as either '0' or '1'. In order to enable machines to handle vague language input such as 'Somehow Satisfied', the crisp input and output must be converted to linguistic variables with fuzzy components. For instance, to control an air conditioner system, the input temperature and the output control variables must be converted to the associated linguistic variables such as 'HIGH', 'MEDIUM', 'LOW' and 'FAST', 'MEDIUM' or 'SLOW'. The former is corresponding to the input temperature and the latter is associated with the rotation speed of the operating motor. Besides those conversions, both the input and the output must also be converted from crisp data to fuzzy data. All of these jobs are performed by the first step –fuzzification.

In the second step, to begin the fuzzy inference process, one need combine the Membership Functions with the control rules to derive the control output, and arrange those outputs into a table called the lookup table. The control rule is the core of the fuzzy inference process, and those rules are directly related to a human being's intuition and feeling. For example, still in the air conditioner control system, if the temperature is too high, the heater should be turned off, or the heat driving motor should be slowed down, which is a human being's intuition or common sense. Different methods such as Center of Gravity (COG) or Mean of Maximum (MOM) are utilized to calculate the associated control output, and each control output should be arranged into a table called lookup table.

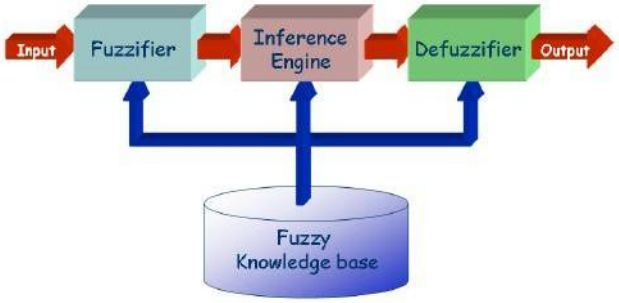
During an actual application, a control output should be selected from the lookup table developed from the last step based on the current input. Furthermore, that control output should be converted from the linguistic variable back to the crisp variable and output to the control operator. This process is called defuzzification or step 3.

In most cases the input variables are more than one dimension for real applications, and one needs to perform fuzzification or develop a Membership Function for each dimensional variable separately. Perform the same operation if the system has multiple output variables.

Summarily, a fuzzy process is a process of crisp-fuzzy-crisp for a real system. The original input and the terminal output must be crisp variables, but the intermediate process is a fuzzy inference process. The reason why one needs to change a crisp to a fuzzy variable is that, from the point of view of fuzzy control or a human being's intuition, no absolutely crisp variable is existed in our real world.

Any physical variable may contain some other components. For instance, if someone says: the temperature here is high. This high temperature contains some middle and even low temperature components. From this point of view, fuzzy control uses universal or global components, not just a limited range of components as the classical variables did.

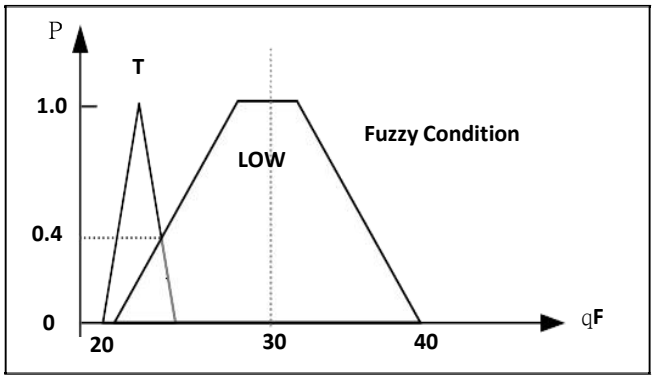
With the rapid development of fuzzy technologies, different fuzzy control strategies have been developed based on different classical control methods, such as PID-fuzzy control, sliding-mode fuzzy control, neural fuzzy control, adaptor fuzzy control and phase-plan mapping fuzzy control. More and more new fuzzy control strategies or combined crisp and fuzzy control techniques are being developed and will be applied to many areas in our society in the future.



Fuzzy Control Rules:

Fuzzy control rule can be considered as the knowledge of an expert in any related field of application. The fuzzy rule is represented by a sequence of the form IF- THEN, leading to algorithms describing what action or output should be taken in terms of the currently observed information, which includes both input and feedback if a closed-loop control system is applied. The law to design or build a set of fuzzy rules is based on a human being's knowledge or experience, which is dependent on each different actual application.

A fuzzy IF-THEN rule associates a condition described using linguistic variables and fuzzy sets to an output or a conclusion. The IF part is mainly used to capture knowledge by using the elastic conditions, and the THEN part can be utilized to give the conclusion or output in linguistic variable form. This IF-THEN rule is widely used by the fuzzy inference system to compute the degree to which the input data matches the condition of a rule. Figure below illustrates a way to calculate the degree between a fuzzy input *T* (temperature) and a fuzzy condition LOW. Here we still use the air conditioner system as an example.



Matching a fuzzy input with a fuzzy condition

This calculation can also be represented by the function

$$M(T, LOW) = \text{Support } min(P_T(x), P_{LOW}(x)) \quad (2.16)$$

Two types of fuzzy control rules are widely utilized for most real applications. One is fuzzy mapping rules and the other is called fuzzy implication rules.

Fuzzy Mapping Rules:

Fuzzy mapping rules provide a functional mapping between the input and the output using linguistic variables. The foundation of a fuzzy mapping rule is a fuzzy graph, which describes the relationship between the fuzzy input and the fuzzy output. Sometimes, in real applications, it is very hard to derive a certain relationship between the input and the output, or the relationship between those inputs and outputs are very complicated even when that relationship is developed. Fuzzy mapping rules are a good solution for those situations.

Fuzzy mapping rules work in a similar way to human intuition or insight, and each fuzzy mapping rule only approximates a limited number of elements of the function, so the entire function should be approximated by a set of fuzzy mapping rules. Still using our air conditioner system as an example, a fuzzy mapping rule can be derived as

IF the temperature is LOW, THEN the heater motor should be rotated FAST. For other input temperatures, different rules should be developed.

For most actual applications, the input variables are commonly more than one dimension. For example, in our air conditioner system, the inputs include both current temperature and the change rate of the temperature. The fuzzy control rules should also be extended to allow multiple inputs to be considered to derive the output. Table below is an example of fuzzy control rules applied in our air conditioner system.

An example of fuzzy rules

$\dot{T}$ \ T	**LOW**	**MEDIUM**	**HIGH**
LOW	FAST	MEDIUM	MEDIUM
MEDIUM	FAST	SLOW	SLOW
HIGH	MEDIUM	SLOW	SLOW

The rows and columns represent two inputs, the temperature input and the change rate of the temperature input, and those inputs are related to IF parts in IF- THEN rules. The conclusion or control output can be considered as a third dimensional variable that is located at the cross point of each row (temperature) and each column (change rate of the temperature), and that conclusion is associated with the THEN part in IF-THEN rules. For example, when the current temperature is LOW, and the current change rate of the temperature is also LOW, the heater motor's speed should be FAST to increase the temperature as soon as possible. This can be represented by the IF-THEN rule as

IF the temperature is LOW, and the change rate of the temperature is LOW, THEN the conclusion or output (heater motor speed) should be FAST. All other rules follow a similar strategy, which is very similar to a human being's intuition. In this air conditioner example, a total of nine rules are developed. For those applications that need high control accuracy, the input and output should be divided into more small segments, and more fuzzy rules should be applied.

Fuzzy Implication Rules

A fuzzy implication rule describes a generalized logic implication relationship between inputs and outputs. The foundation of a fuzzy implication rule is the narrow sense of fuzzy logic. Fuzzy implication rules are related to classical two-valued logic and multiple valued logic. Still using the air conditioner system as an example, the implication is IF the temperature is LOW, THEN the heater motor should be FAST. Based on this implication and a fact: the temperature is HIGH. The result that the heater motor should slow down or the SLOW can be inferred.

Defuzzification and the Lookup Table

The conclusion or control output derived from the combination of input, output membership functions and fuzzy rules is still a vague or fuzzy element, and this process in called fuzzy inference. To make that conclusion or fuzzy output available to real applications, a defuzzification process is needed. The defuzzification process is meant to convert the fuzzy output back to the crisp or classical output to the control objective. Remember, the fuzzy conclusion or output is still a linguistic variable, and this linguistic variable needs to be converted to the crisp variable via the defuzzification process. Three defuzzification techniques are commonly used, which are: Mean of Maximum method, Center of Gravity method and the Height method.

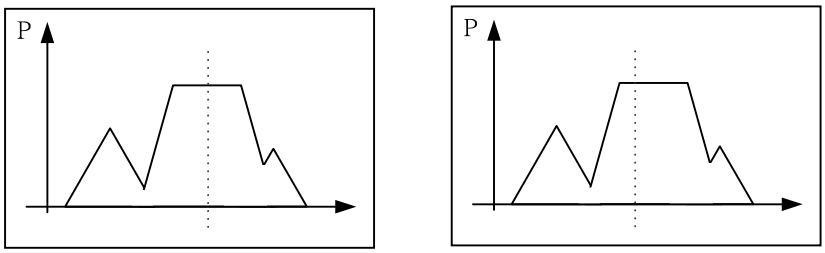
Mean of Maximum (MOM) Method

The Mean of Maximum (MOM) defuzzification method computes the average of those fuzzy conclusions or outputs that have the highest degrees. For example, the fuzzy conclusion is: the heater motor x is rotated FAST. By using the MOM method, this defuzzification can be expressed as

$$\text{MOM(FAST)} = \frac{\sum_{x \in T} x'}{T} \qquad T = \{x' \mid \mu_{\text{FAST}}(x') = \text{Support } \mu_{\text{FAST}}(x)\}$$

where T is the set of output x that has the highest degrees in the set FAST.

A graphic representation of the MOM method is shown in Figure below. A shortcoming of the MOM method is that it does not consider the entire shape of the output membership function, and it only takes care of the points that have the highest degrees in that function. For those membership functions that have different shapes but the same highest degrees, this method will produce the same result.



(a) MOM method example (b) COG method example

Graphic representation of defuzzification techniques

Center of Gravity (COG) Method

The Center of Gravity method (COG) is the most popular defuzzification technique and is widely utilized in actual applications. This method is similar to the formula for calculating the center of gravity in Physics. The weighted average of the membership function or the center of the gravity of the area bounded by the membership function curve is computed to be the most crisp value of the fuzzy quantity. For example, for the conclusion: the heater motor *x* is rotated FAST. The COG output can be represented as

$$\mathrm{COG(FAST)} = \frac{\sum_{x} \mu_{FAST}(x) \times x}{\sum_{x} \mu_{FAST}(x)}$$

The Lookup Table

The terminal product of defuzzification is the lookup table. Defuzzification needs to be performed for each subset of a membership function, both inputs and outputs. For instance, in the air conditioner system, one needs to perform defuzzification for each subset of temperature input such as LOW, MEDIUM and HIGH based on the associated fuzzy rules. The defuzzification result for each subset needs to be stored

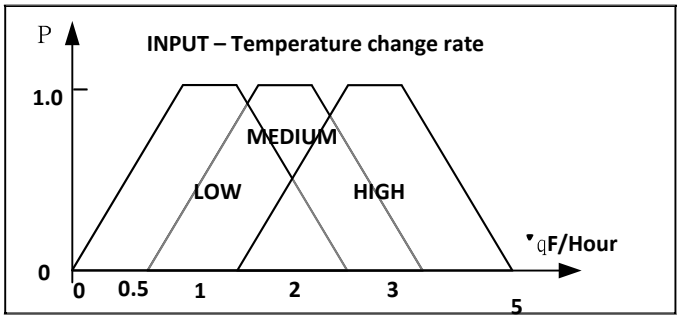
in the associated location in the lookup table according to the current temperature and temperature change rate. In the following we use the air conditioner system as an example to illustrate the defuzzification process and the creation of the lookup table.

To make this illustration simple, we make two assumptions:

I. assume that the membership function of the change rate of the temperature can be described as in Figure below;
II. only four rules are applied to this air conditioner system, which are

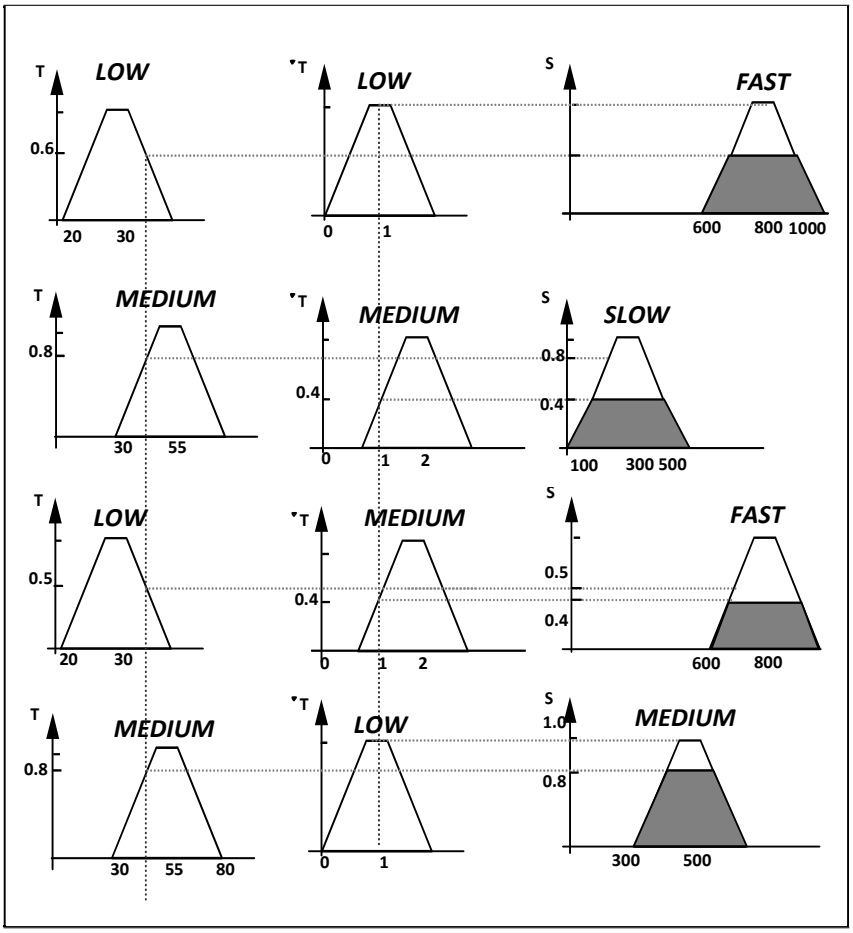
1) IF the temperature is LOW, and the change rate of the temperature is LOW, THEN the heater motor speed should be FAST
2) IF the temperature is MEDIUM, and the change rate of the temperature is MEDIUM, THEN the heater motor speed should be SLOW
3) IF the temperature is LOW, and the change rate of the temperature is MEDIUM, THEN the heater motor speed should be FAST
4) IF the temperature is MEDIUM, and the change rate of the temperature is LOW, THEN the heater motor speed should be MEDIUM

The membership function of the change rate of temperature



Based on the assumption made for the membership function and fuzzy rules, we can illustrate this defuzzification process using a graph. Four fuzzy rules can be interpreted as functional diagrams, as shown in Figure. As an example, consider the current input temperature is 35 qF and the change rate of the temperature is 1 qF per hour. From Figure, it can be found that the points of intersection between the temperature values of 35 qF and the graph in the first column (temperature input *T*) have the membership functions of 0.6, 0.8, 0.5 and 0.8. Likewise, the second column (temperature change rate ▾*T*) shows that a temperature change rate of 1 qF per hour has the membership functions of 1.0, 0.4, 0.4 and

The fuzzy output for the four rules is the intersection of the paired values obtained from the graph, or the AND result between the temperature input and the temperature change rate input. According to Equation, this operation result should be: min (0.6, 1.0), min (0.8, 0.4), min (0.5, 0.4) and min (0.8, 1.0), which produces to 0.6, 0.4, 0.4 and 0.8, respectively.



An illustration of fuzzy output calculation

Off-line and On-line Defuzzification

As mentioned , the defuzzification process is to derive the desired crisp output value using a defuzzification technique that combines the membership functions with the fuzzy rules. The defuzzification process can be further divided into two categories: off-line defuzzification and on-line defuzzification.

So-called off-line defuzzification means that all input and output membership functions, fuzzy rules and the lookup table should be developed based on the estimations of the real application prior to the implementation of the fuzzy logic technique to the actual application. This means that all input and output membership functions are developed based on the actual experience or the input and output parameter ranges of a specified application, and the lookup table is calculated in terms of those definitions of input and output membership functions. The advantage of this method is that most fuzzy inference related calculations are performed prior to the real implementation and therefore the fuzzy process is less time consuming. The disadvantage of this technique is that the fuzzy output is only based on the estimation of input and output parameters, so the control accuracy is not as high as that of on-line method.

The on-line method has real-time controllability. Both input and output membership functions are developed during the real-time processing of an actual application. Also the lookup table elements are calculated in real-time based on the current actual inputs and outputs. In this method, only fuzzy rules are developed prior to the real applications. The advantage of this method is that higher control accuracy can be obtained for a process and the fuzzy output can be computed in real-time. The disadvantage of this method is that a longer processing time is needed and it is a somewhat time-consuming process. However, with the development of new computer technologies, today much faster CPUs are available, and processing time is no longer a big deal for this method.

Architectures of Fuzzy Logic Controls

Combining the discussions we made in the previous sections, a structure or architecture of fuzzy logic control system is given here. As shown in Figure below, which is a typical fuzzy closed-loop control system, the inputs are error and error rate, which are combined by block *M* to input to the fuzzy inference system. The lookup table is derived based on the membership function of inputs, the output and the fuzzy control rules. A control gain factor *G* is used to tune the output of the lookup table to obtain different output values. The interpolation block S is used to smooth the output element of the lookup table. A feedback signal is obtained from the output of the system.

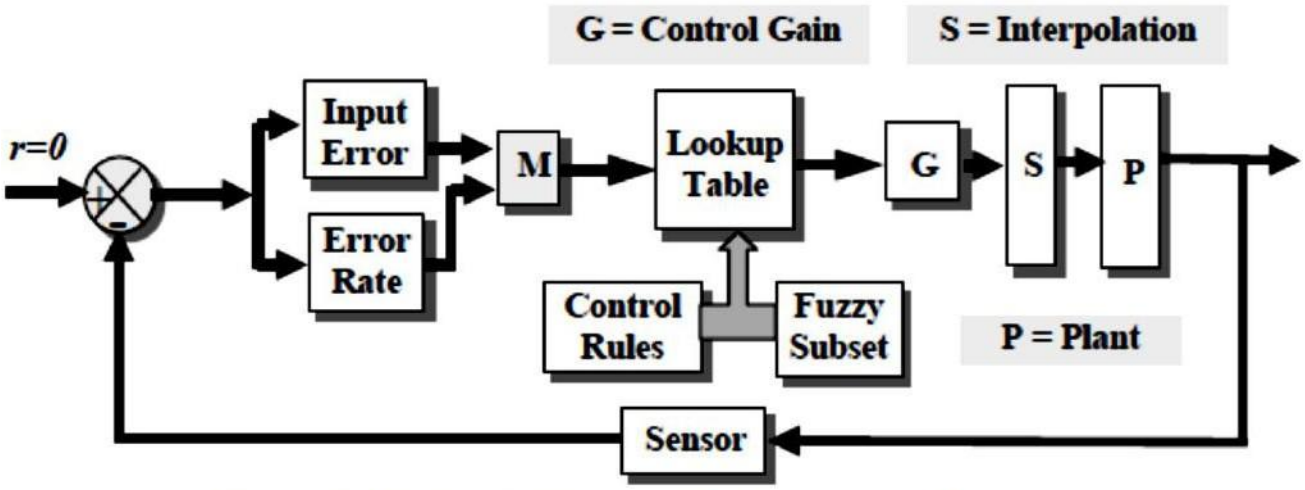


Figure 2.12. Block diagram of a fuzzy control system

For a system that needs higher control accuracy, a multiple lookup tables fuzzy control system is needed, which is shown in Figure 2.13.

Two lookup tables are developed in this control system, a coarse and a fine table. During the application, the switch between the coarse and the fine table is under the control of the input error limit. This limit value can be defined by the user based on the real application. Two-set membership functions and control rules are utilized in this system to satisfy the requirement of higher control accuracy. When the system needs quick responses or quick actions, the coarse table is used. When the system needs high control accuracy or small control errors, the fine lookup table is selected. The sacrifice for this method is that more memory is needed to store both coarse and fine tables, and a little longer time is needed to make decision in selecting table in terms of the input error limit value.

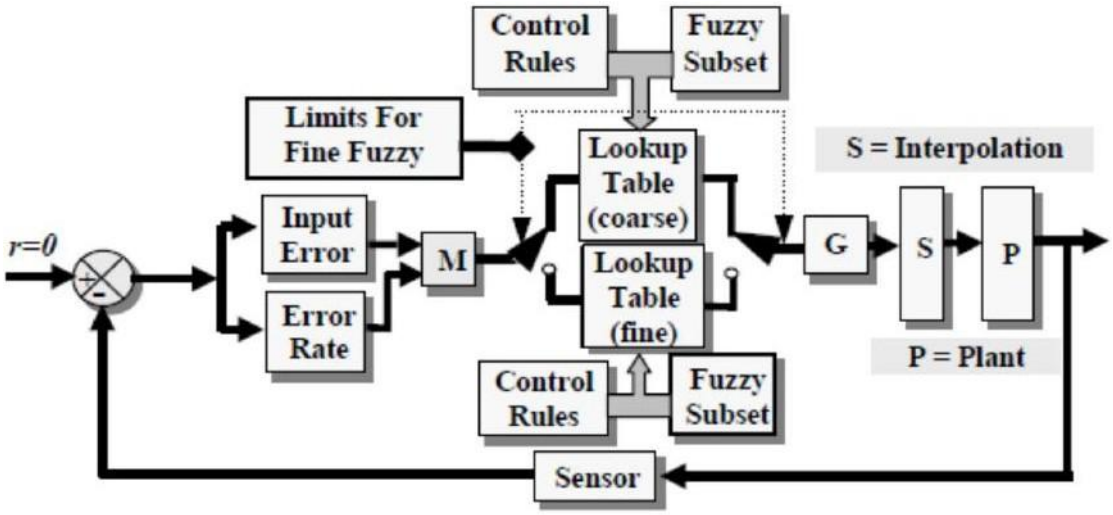


Figure 2.13. Block diagram of an accurate fuzzy control system

EXPLAIN THE ARCHITECTURE OF FUZZY LOGIC CONTROL DESIGN / CONTROLLER.

A control system is an arrangement of physical components designed to alter another physical system so that this system exhibits certain desired characteristics. Following are some reasons of using Fuzzy Logic in Control Systems:

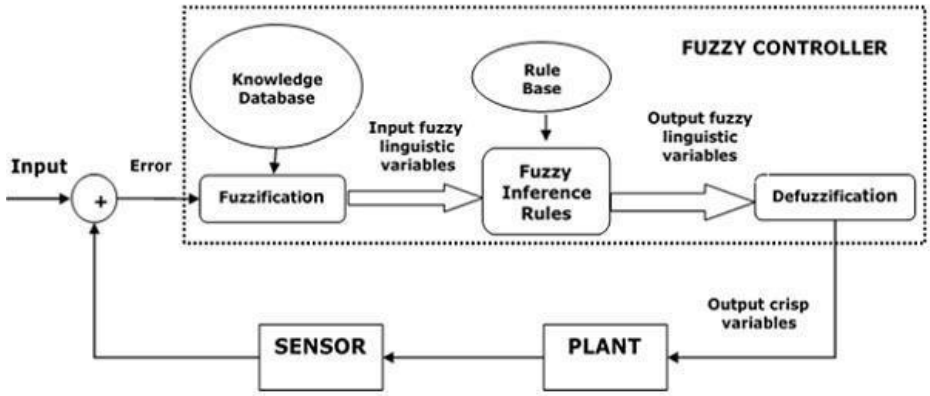
- While applying traditional control, one needs to know about the model and the objective function formulated in precise terms. This makes it very difficult to apply in many cases.
- By applying fuzzy logic for control we can utilize the human expertise and experience for designing a controller.
- The fuzzy control rules, basically the IF-THEN rules, can be best utilized in designing a controller.

Assumptions in Fuzzy Logic Control (FLC) Design: While designing fuzzy control system, the following six basic assumptions should be made:

- The plant is observable and controllable − It must be assumed that the input, output as well as state variables are available for observation and controlling purpose.
- Existence of a knowledge body − It must be assumed that there exist a knowledge body having linguistic rules and a set of input-output data set from which rules can be extracted.
- Existence of solution − It must be assumed that there exists a solution.
- 'Good enough' solution is enough − The control engineering must look for 'good enough' solution rather than an optimum one.
- Range of precision − Fuzzy logic controller must be designed within an acceptable range of precision.
- Issues regarding stability and optimality − The issues of stability and optimality must be open in designing Fuzzy logic controller rather than addressed explicitly.

Architecture of Fuzzy Logic Control:

The following diagram shows the architecture of Fuzzy Logic Control (FLC).



Major Components of FLC: Followings are the major components of the FLC as shown in the above figure –

1. Fuzzifier – The role of fuzzifier is to convert the crisp input values into fuzzy values.
2. Fuzzy Knowledge Base – It stores the knowledge about all the input-output fuzzy relationships. It also has the membership function which defines the input variables to the fuzzy rule base and the output variables to the plant under control.
3. Fuzzy Rule Base – It stores the knowledge about the operation of the process of domain.
4. Inference Engine – It acts as a kernel of any FLC. Basically it simulates human decisions by performing approximate reasoning.
5. Defuzzifier – The role of defuzzifier is to convert the fuzzy values into crisp values getting from fuzzy inference engine.

Steps in Designing FLC

Following are the steps involved in designing FLC:

- Identification of variables – Here, the input, output and state variables must be identified of the plant which is under consideration.
- Fuzzy subset configuration – The universe of information is divided into number of fuzzy subsets and each subset is assigned a linguistic label. Always make sure that these fuzzy subsets include all the elements of universe.
- Obtaining membership function – Now obtain the membership function for each fuzzy subset that we get in the above step.
- Fuzzy rule base configuration – Now formulate the fuzzy rule base by assigning relationship between fuzzy input and output.
- Fuzzification – the fuzzification process is initiated in this step.
- Combining fuzzy outputs – By applying fuzzy approximate reasoning, locate the fuzzy output and merge them.
- Defuzzification – finally, initiate defuzzification process to form a crisp output.

State the Advantages of Fuzzy Logic Control:

- Cheaper – Developing a FLC is comparatively cheaper than developing model based or other controller in terms of performance.
- Robust – FLCs are more robust than PID controllers because of their capability to cover a huge range of operating conditions.
- Customizable – FLCs are customizable.
- Emulate human deductive thinking – Basically FLC is designed to emulate human deductive thinking, the process people use to infer conclusion from what they know.
- Reliability – FLC is more reliable than conventional control system.
- Efficiency – Fuzzy logic provides more efficiency when applied in control system.

State the Disadvantages of Fuzzy Logic Control:

- Requires lots of data – FLC needs lots of data to be applied.
- Useful in case of moderate historical data – FLC is not useful for programs much smaller or larger than historical data.
- Needs high human expertise – This is one drawback, as the accuracy of the system depends on the knowledge and expertise of human beings.
- Needs regular updating of rules – The rules must be updated with time.

LIST THE APPLICATIONS OF FUZZY LOGIC:

Application in Aerospace: In aerospace, fuzzy logic is used in Altitude control of spacecraft, Satellite altitude control, Flow and mixture regulation in aircraft deicing vehicles.

Applications in Automotive Industry: In automotive, fuzzy logic is used for idle speed control, Shift scheduling method for automatic transmission, Intelligent highway systems, Traffic control, Improving efficiency of automatic transmissions

Applications in Business: In business, fuzzy logic is used in Decision-making support systems, Personnel evaluation in a large company.

Applications in Defense: In defense, fuzzy logic is used in Underwater target recognition, Automatic target recognition of thermal infrared images, Naval decision support aids, Control of a hypervelocity interceptor Fuzzy set modeling of NATO decision making.

Applications in Electronics: In electronics, fuzzy logic is used in Control of automatic exposure in video cameras, to maintain Humidity in a clean room, Air conditioning systems, Washing machine timing, Microwave ovens, Vacuum cleaners.

Applications in Finance: In the finance field, fuzzy logic is used in Banknote transfer control, Fund management, Stock market predictions.

Applications in Industrial Sector: In industrial, fuzzy logic is used in Cement kiln controls heat exchanger control, Activated sludge wastewater treatment process control, Water purification plant control, Quantitative pattern analysis for industrial quality assurance, Control of constraint satisfaction problems in structural design, Control of water purification plants

Applications in Manufacturing: In the manufacturing industry, fuzzy logic is used in Optimization of cheese production, Optimization of milk production.

Applications in Marine: In the marine field, fuzzy logic is used in Autopilot for ships, optimal route selection, Control of autonomous underwater vehicles, Ship steering.

Applications in Medical field: In the medical field, fuzzy logic is used in Medical diagnostic support system, Control of arterial pressure during anesthesia, Multivariable control of anesthesia, Modeling of neuropathological findings in Alzheimer's patients, Radiology diagnoses, Fuzzy inference diagnosis of diabetes and prostate cancer.

Applications in Transportation: In transportation, fuzzy logic is used in Automatic underground train operation, Train schedule control, Railway acceleration, Braking and stopping.

Applications in Pattern Recognition and Classification: In Pattern Recognition and Classification, fuzzy logic is used in Fuzzy logic based speech recognition, Fuzzy logic based, Handwriting recognition; Fuzzy logic based facial characteristic analysis, Command analysis, Fuzzy image search, Criminal investigation and prevention based on fuzzy logic reasoning.

LECTURE NOTES FOR NEURAL NETWORKS AND FUZZY LOGIC (15A02604)
UNIT- 5
FUZZY LOGIC APPLICATIONS TO ELECTRICAL SYSTEMS

FUZZY LOGIC IMPLEMENTATION FOR INDUCTION MOTOR CONTROL

Vector control, known as field-oriented control (FOC), is a variable-frequency drive (VFD) control scheme where the stator currents of a three-phase AC electric motor are acknowledged as two orthogonal components that can be visualized with a vector. One component defines the magnetic flux of the motor, the supplementary is the torque. The control system of the drive calculates from the flux and torque references specified by the drive's speed control the corresponding current component references. Typically proportional-integral (PI) controllers are employed to keep the measured current components at their reference values. The pulse-width modulation of the variable-frequency drive describes the transistor switching according to the stator voltage references that are the output of the PI current controllers. FOC is used to manage the AC synchronous and induction motors. It was originally developed for high performance motor applications that are required to operate smoothly over the bursting speed range, generate full torque at zero speed, and have high dynamic performance including fast acceleration and deceleration. However, it is becoming increasingly attractive for lower performance applications also due to FOC's motor size, cost and power consumption reduction dominance. It is expected that with increasing computational power of the microprocessors it will ultimately nearly universally relocate single-variable scalar volts-per-Hertz (V/f) control.

FUZZY LOGIC CONTROLLER

Fuzzy logic provides a sturdy framework for achieving robust and simple solutions amid different approaches of intelligent computation. Fuzzy model is a collection of IF - THEN rules with indistinguishable predicates that use a fuzzy reasoning such as Sugeno and Mamdani models. Sugeno type systems can be used to model any inference system in which the output membership functions are either linear or constant whereas Mamdani type produces either linear or nonlinear output. The fuzzy logic controller consists of four stages, fuzzification of inputs and derivation of rules, inference mechanism and de-fuzzification. Fuzzy logic systems are collective function approximations. In general, the goal of fuzzy logic system is to yield a set of outputs for given inputs in a non-linear system, lacking using any mathematical model but by using linguistic rules. A general block diagram of Fuzzy logic is shown Fig 1.

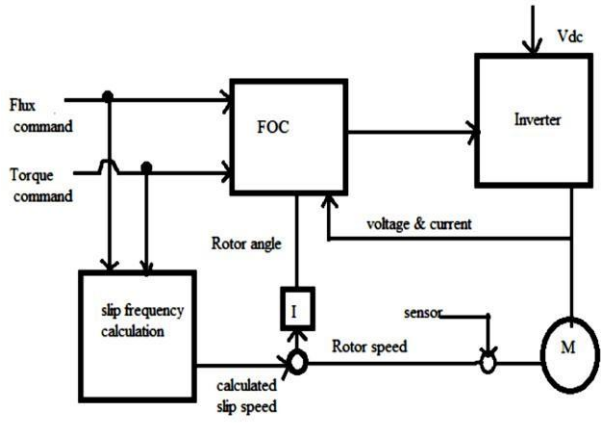


Fig 1. Field oriented control

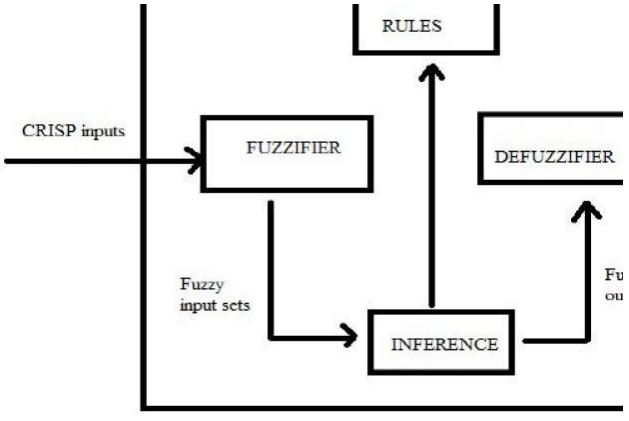


Fig 2 General block diagram of fuzzy logic controller

SYSTEM MODEL

The block model of the induction motor system with the controller be developed using the power system, power electronics, control system, signal processing toolboxes & from the fundamental functions available in the Simulink library in Matlab / Simulink. In this paper, plots of voltage, torque, speed, load & flux, etc are plotted as functions of time with the controller and the waveforms are observed on the equivalent scopes after running the simulations. The entire system modelled in Simulink is a closed loop feedback control system consisting of the plants, controllers, samplers, comparators, feedback systems, the mux, de-mux, summers, adders, gain blocks, multipliers, clocks, subsystems, integrators, state-space models, subsystems,

the output sinks (scopes), the input sources, etc. The developed Simulink model for the control of various parameters of the SCIM is shown in the Fig 2. A set of 49 fuzzy rules are written and called in the form of a file in the developed Simulink model with the controller. While the simulation is run, the 2 fuzzy inputs are then given to the controller (Takagi-Sugeno-fuzzy), where the output is obtained afterward. The response curves of flux, load, torque, terminal voltage, and speed & torques v/s time are observed on the respective From the simulation results shown in the Figure 4 below, it is observed that the stator current does not exhibit any overshoots, undershoots, the response of the flux, torque, terminal voltage, speed & stator Currents, etc. takes lesser time to settle & reach the desired value compared to the results using vector control. This shows the effectiveness of the developed controller. It is also observed that with the controller, the response characteristics curves take less time to settle & reach the final steady state value compared to that in. In the fig 5, it shows the graph of speed Vs time for vector control and fuzzy logic controller.

Table shows the rule base for controlling the speed

Δe \ e	NM	NS	ZE	PS	PM	PL
NL	NL	NL	NLM	NM	NMS	NS	ZE
NM	NL	NLM	NM	NMS	NS	ZE	PS
NS	NLM	NM	NMS	NS	ZE	PS	PMS
ZE	NM	NMS	NS	ZE	PS	PMS	PM
PS	NMS	NS	ZE	PS	PMS	PM	PLM
PM	NS	ZE	PS	PMS	PM	PLM	PL
PL	ZE	PS	PMS	PM	PLM	PL	PL

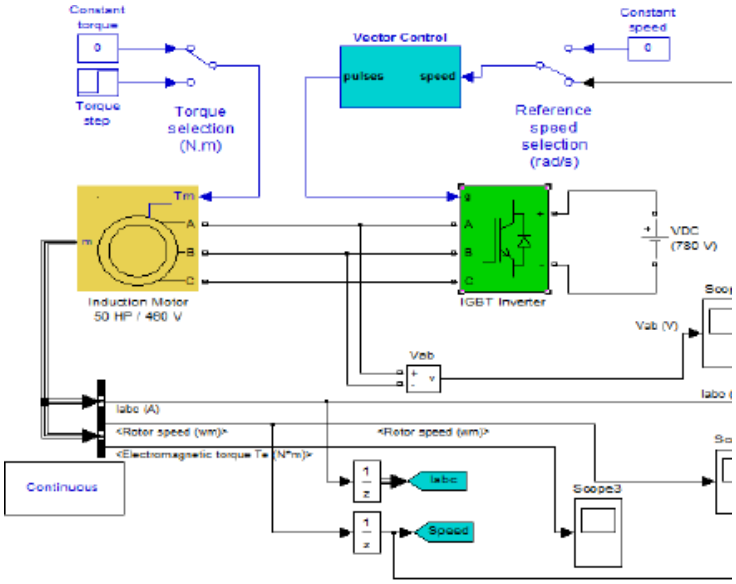


Fig 3 Vector control of induction m

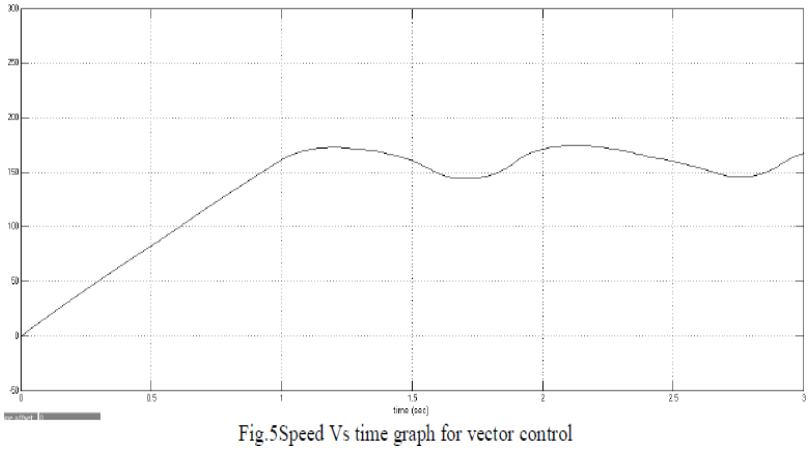


Fig.5Speed Vs time graph for vector control

FUZZY LOGIC IMPLEMENTATION FOR SWITCHED RELUCTANCE MOTOR CONTROL

The SRMs overall performance can be improved by two main ways. The first way, by improving the mechanical design and second one by the control techniques. There are different types of control strategy can be applied on SRM such as speed/position control, current control, and direct/indirect torque control. The torque ripples consider the main challenge of the SRM in many applications particularly in EVs applications, this problem is very complicated and affected by many factors and it is not easy to solve. Different strategies of control are used to overcome of this problem like traditional control strategy, torque distribution strategy, linearization control, intelligent control, and other control methods. So, by selecting the suitable control strategy for each application, the torque ripple can be efficiently decreased. The SRMs can be run in current or voltage control mode. The voltage controlled of SRM is high sensitive to voltage ripple on the supply side and its bandwidth control is lesser, so it is essential to use current control when SRM performance is desired to an accurate torque control.

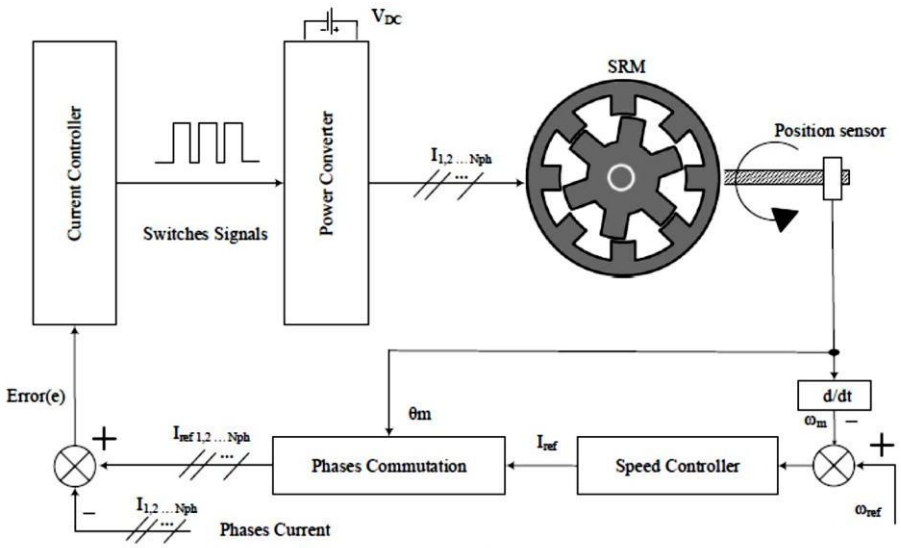
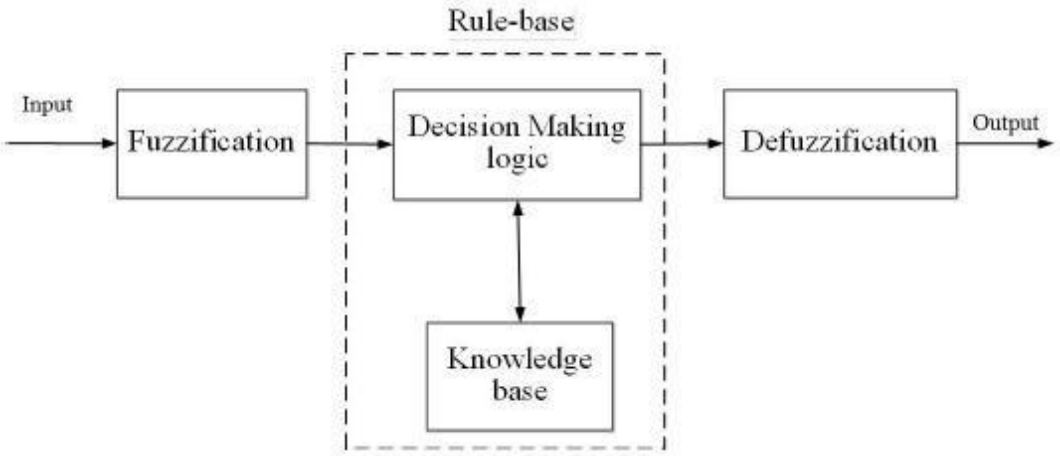


Fig. 1. *SRMs current control block diagram*

FUZZY LOGIC CONTROL APPROACH:

The fuzzy logic approach offers a simpler, quicker and more reliable solution than conventional techniques. The FLC has three main blocks. The first one is Fuzzification block, which modifies crisp inputs (input values from real world) into linguistic variables to enable the input physical signal to use the rule-base through membership functions. The second block is Rule-base, where fuzzy inputs are compared, and the controller makes the decision based on the membership functions of each input. The last one is Defuzzification block, which converts back the fuzzy outputs of the rule-base to crisp ones and selects membership functions for the different control outputs from the rule-base.



FLC Speed Control Algorithm:

Step 1: The Switched reluctance motor speed signal is sampled.
Step 2: Calculate speed error and change in speed error.
Step 3: Determine fuzzy sets for speed error.
Step 4: Determine membership for speed error.
Step 5: Determine fuzzy sets for change in speed error.
Step 6: Determine membership for change in speed error.
Step 7: Finding control action as per fuzzy rule and Calculate error.
Step 8: Sending control command to the system after calculation of error.

Input and output variables of fuzzy membership function are selected as follows,
PB-Positive Big
PM-Positive Medium
PS-Positive Small
NB-Negative Big
NM-Negative Medium
NS-Negative Small
Z-Zero
The triangular shaped function is chosen as membership functions Because of the best control performance and simplicity.

Table 1: Rule based table

E/CE	NB	NM	NS	ZO	PS	PS	PB
NB	NB	NB	NB	NB	NM	NS	ZO
NM	NB	NB	NB	NM	NS	ZO	PS
NS	NB	NB	NM	NS	ZO	PS	PM
ZO	NB	NM	NS	ZO	PS	PM	PB
PS	NM	NS	ZO	PS	PM	PB	PB
PM	NS	ZO	PS	PM	PB	PB	PB
PB	ZO	PS	PM	PB	PB	PB	PB

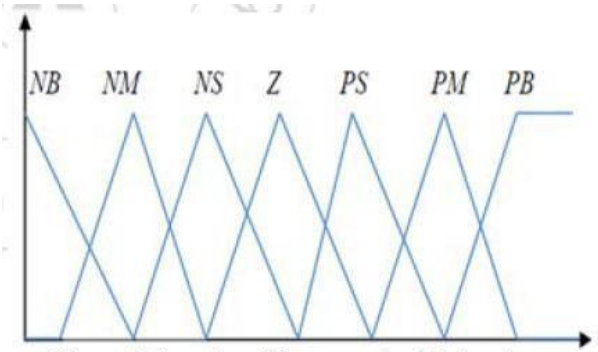


Figure 4: Seven level fuzzy membership functions

Fuzzification: This is considered the first step to be programmed, the FLC uses linguistic variables instead of numerical variables. So, the error input signals can be assigned as Negative Very Big (NVB), Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (ZE), Positive Small (PS), Positive Medium (PM), Positive Big (PB), Positive Very Big (PVB). The triangular membership function is used for fuzzification as shown in Figure. The process of fuzzification convert numerical variable (real number) to a linguistic variable (fuzzy set). The change of error, which used as the second input for fuzzy system also converted from numerical value to a linguistic variable according to the triangular membership.
Defuzzification: The fuzzy logic rules generate the demanded output in a linguistic variable; these variables must be transformed to crisp output (real number). This step is the defuzzification; the membership functions used in this study for defuzzification are shows in Fig. 7. The output signal can be assigned as: Extremely Low (EL), Very Low (VL), Low (L), Under Medium (UM), Medium (M), Above Medium (AM), High (H), Very High (VH), Extremely High (EH), which represent the modulation index (m), where ($0 \leq m \leq 1$). There are three different methods can be used for membership defuzzification, Center of Area(COA), Bisector, or Middle of Maximum (MOM). The center area (COA) is considered the most popular method, so it is used for defuzzification in this study, which is presented in Equation,

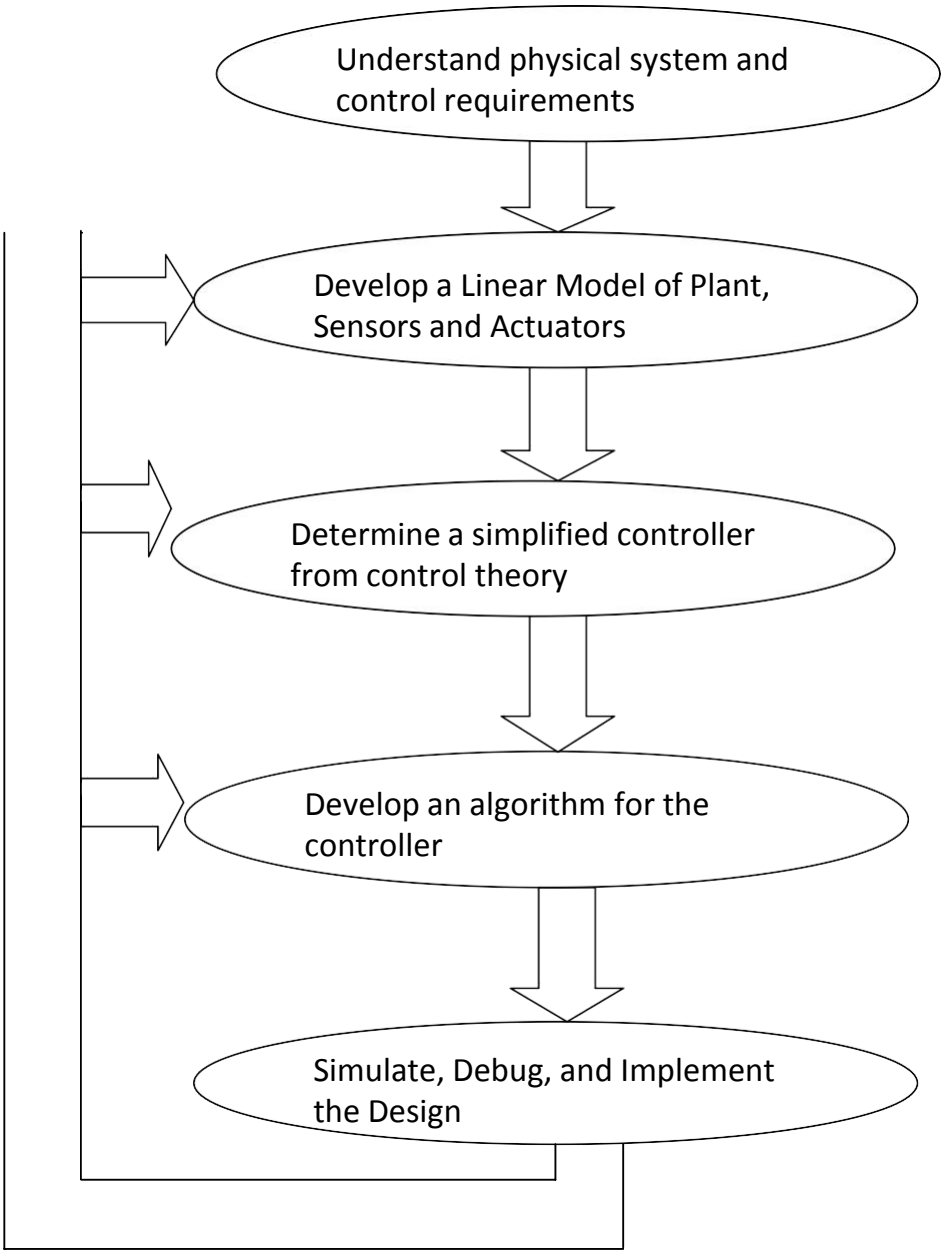
$$U(n) = \frac{\sum_{j=1}^{n} \mu(uj)\, \omega j}{\sum_{i=1}^{n} \mu(uj)}$$

By adopting the FLC, the SRM current track the reference signal with minimum values of current ripples comparing with traditional current control techniques, and hence, the torque ripples during the conduction period of each phase of the motor were reduced. The controller was tested at different load conditions and with different turn on angles.

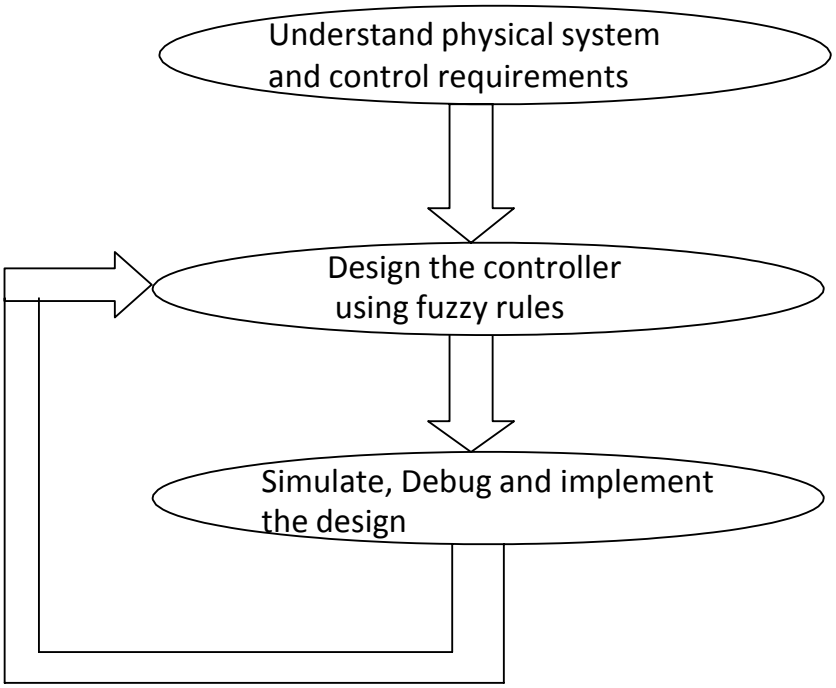
BRIEFLY EXPLAIN FUZZY BASED EXCITATION CONTROL FOR AVR.

Generating Unit Control is a complete closed-loop system and in the past a lot of effort has been dedicated to improve the performance of the controllers. The main problem for example with excitation control is that the control law is based on a linearized machine model and the control parameters are tuned to some nominal operating conditions. In case of a large disturbance, the system conditions will change in a highly non-linear manner and the controller parameters are no longer valid. In this case the controller may even add a destabilizing effect to the disturbance by for example adding negative damping. Fuzzy logic concept incorporates an alternative way which allows one to design a controller using a higher level of abstraction without knowing the plant model. Conventional modeling and control approaches based on differential equations are often insufficient, mainly due to the lack of precise formal knowledge about the process to be controlled. Unlike the conventional control, if the mathematical model of the process is unknown we can design fuzzy controllers in a manner that guarantees certain key performance criteria. Fuzzy logic control (FLC) reduces the time and complexity in analyzing the differential equations involved in the conventional control and hence in the overall design development cycle as depicted below.

Conventional Design Methodology

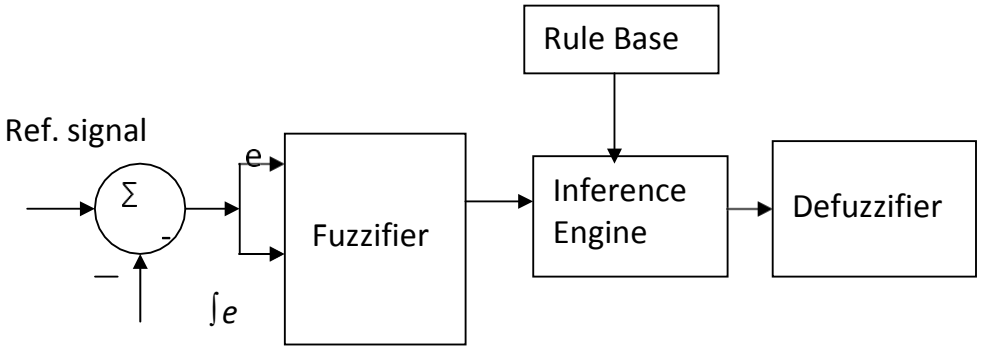


Fuzzy based Design Methodology



Fuzzy Logic Controller design is a three-stage process. It comprises of fuzzification, inference mechanism and defuzzification stages. To design the controller, firstly, membership functions for the input variables (error, and integral of error must be specified). Secondly, the fuzzy inference system must be defined which consists of a series of "If…..then….." linguistic rules. Then finally, the membership functions for the output must be selected. A structure of a fuzzy logic controller is shown in figure below.

Fuzzy controller architecture



The fuzzy logic controller has three main components:

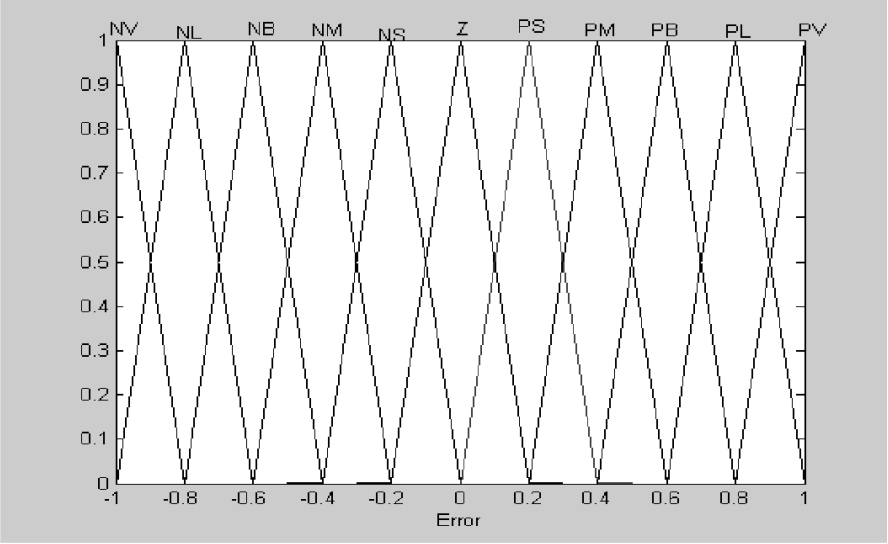
1. The inference mechanism incorporates the rule base which holds the knowledge in the form of a set of rules of how best to control the system, and evaluates which of the control rules are relevant at the current time and then decides what the input to the plant should be.
2. The fuzzification interface that simply modifies the inputs so that they can be interpreted and compared to the rules in the rule base.
3. The defuzzification interface that converts the conclusion reached by the interference mechanism into the inputs to the plant.

For the fuzzy logic AVR control, the error between the reference voltage Vref and the terminal voltage Vt i.e. Ve and the integral of the error V1 which is the difference between the immediate and previous voltage error values are considered as the inputs to the fuzzy controller while the output is the specified output voltage of the alternator Vt. In this controller, eleven fuzzy subsets are chosen.

These are:

"Negative Very large" (NV), "Negative Large" (NL), "Negative Big"(NB), "Negative Medium" (NM), "Negative Small" (NS), "Zero" (Z), "Positive Small" (PS), "Positive Medium" (PM), "Positive Big" (PB), "Positive Large" (PL), "Positive Very large" (PV), These input variables are assigned numerical values as (-1, -0.8, -0.6, -0.4, -0.2,0, 0.2, 0.4, 0.6, 0.8 and 1) which stands for Negative Very large, Negative Large, Negative Big, Negative Medium, Negative Small, Zero, Positive Small, Positive Medium, Positive Big, Positive Large and Positive Very large respectively. For Fuzzy-AVR control, triangular membership function is suitable and therefore used.

Membership functions for the error



With the two inputs for this FLC, an (11x11) decision table is constructed as shown in Table below. Every entity in the table represents a rule. The antecedent of each rule conjuncts Ve and V_1 fuzzy set values.

V_e

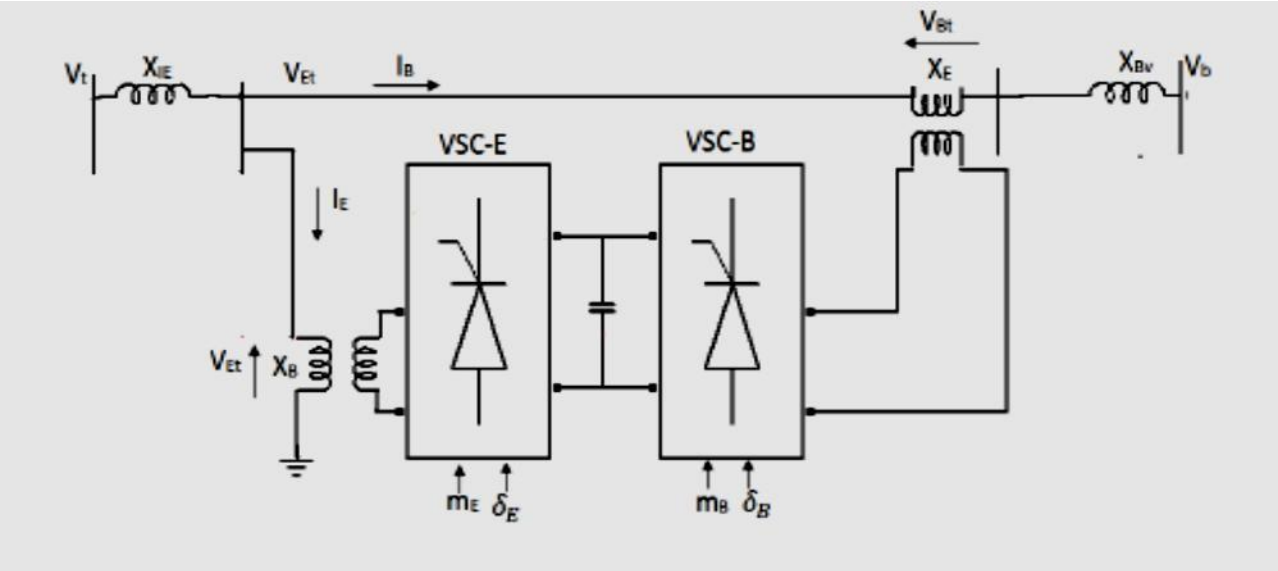
$\int V_e$	U	NV	NL	NB	NM	NS	Z	PS	PM	PB	PL	PV
	NV	NV	NV	NV	NV	NV	NV	NL	NB	NM	NS	Z
	NL	NV	NV	NV	NV	NV	NL	NB	NM	NS	Z	PS
	NB	NV	NV	NV	NV	NL	NB	NM	NS	Z	PS	PM
	NM	NV	NV	NV	NL	NB	NM	NS	Z	PS	PM	PB
	NS	NV	NV	NL	NB	NM	NS	Z	PS	PM	PB	PL
	Z	NV	NL	NB	NM	NS	Z	PS	PM	PB	PL	PV
	PS	NL	NB	NM	NS	Z	PS	PM	PB	PL	PV	PV
	PM	NB	NM	NS	Z	PS	PM	PB	PL	PV	PV	PV
	PB	NM	NS	Z	PS	PM	PB	PL	PV	PV	PV	PV
	PL	NS	Z	PS	PM	PB	PL	PV	PV	PV	PV	PV
	PV	Z	PS	PM	PB	PL	PV	PV	PV	PV	PV	PV

Defuzzification operates on the implied fuzzy sets produced by the inference mechanism and combines their effects to provide the most certain controller output which is the output of the plant. Widely, center of area method (centroid) is used for defuzzification according to the membership function of the output. The arrangement of the structure results in a computationally less intensive control algorithm. Another significant advantage of fuzzy logic is that it can easily accommodate additional input signals. Fuzzy Logic Control provides a convenient means to develop the controller which can accommodate the non-linear nature of the exciter-generator system. Fuzzy logic on the other hand is not without any set back, the set back is observed on the steady-state error which can be cleared by tuning the gains.

EXPLAIN IN DETAIL, THE DESIGN OF FUZZY LOGIC CONTROLLER FOR A 18 / INFINITE BUS BAR SYSTEM:

It is possible that as constrained by the operational requirements the single machine infinite bus bar system is frequently subjected to sudden real and reactive power loading and also as a result of sudden inclusion or exclusion of sources and loads connected to the grid, it may exhibit low frequency Power Oscillations. Though the traditional Proportional integral (PI) controller can handle this problem the associated non linearity and the lack of precise mathematical models of sub systems it warrants that some intelligent control techniques are necessary so that the system offers seamless stable operation. The adaptive Fuzzy inference system is an effective alternate, since it can handle problems associated with systems for which the exact mathematical model is either not available or what is available is approximate or too complex.

UPFC control architecture for an 18/infinite bus system

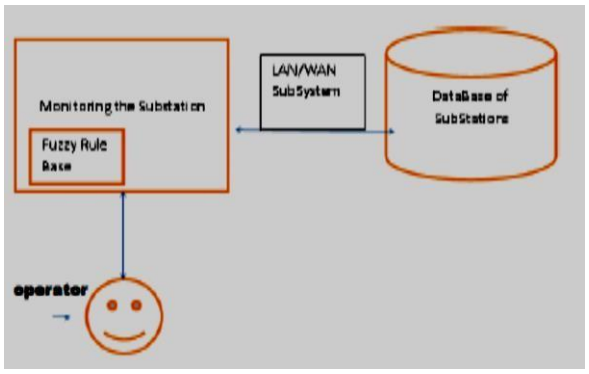


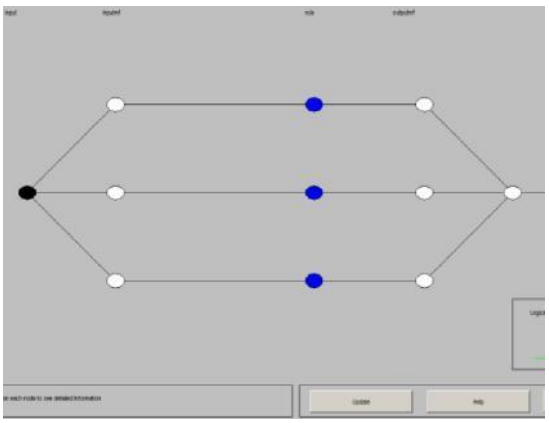
The FLC being a control technique to deal with approximate data to arrive at an agreeable result it can be put under service where there is an expectation with a certain range or degree of acceptability. In the management of UPFC, the FLC can offer a stable operation tracking the set points and keeping the actual values within close vicinity of the desired set points. The fuzzy logic system in its basic form has the following steps.

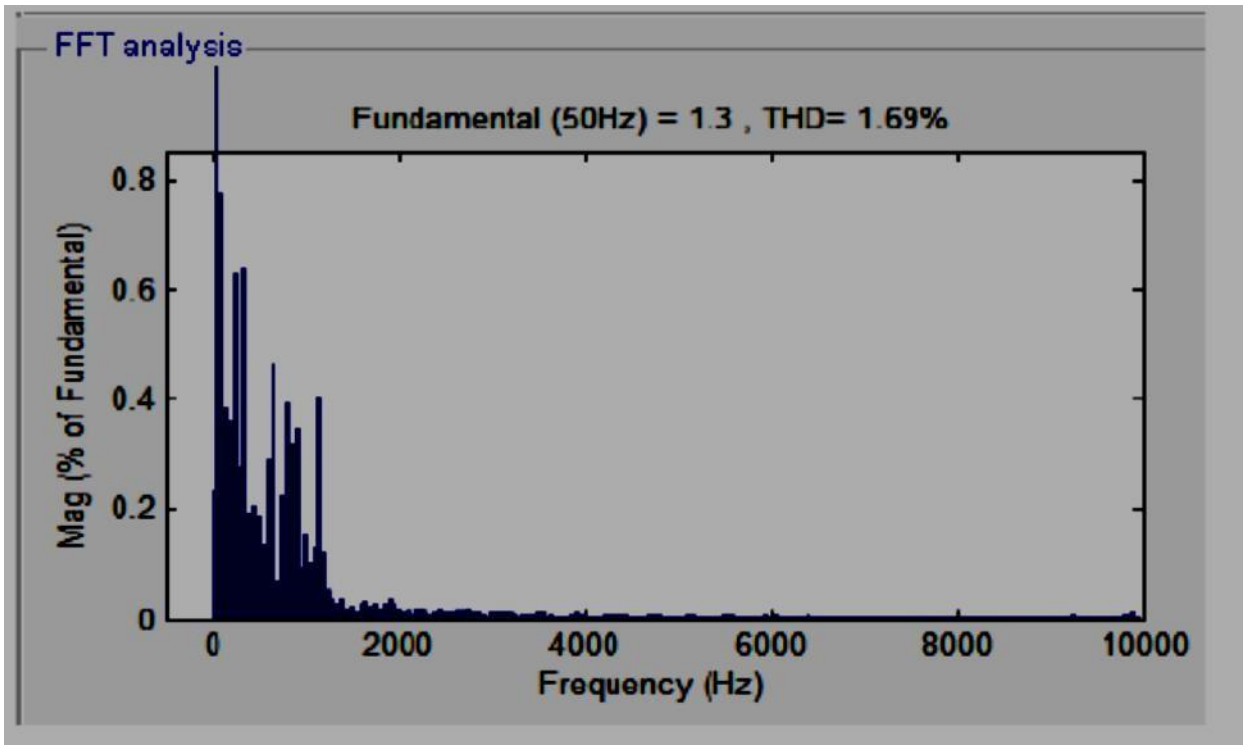
- Fuzzyfication,
- Inference and Decision making using the rule base
- Defuzzification.

Out of these three basic steps of the FLC the Fuzzification process and the Inference and decision making process depend largely on the perception and experience of the designer. Thus the performance of the FLC is a function of the experience of the operator and hence the entire control system is at the disposal of the designer. Further performance improvement is possible by integrating the FLC with an artificial neural network system (ANN) to form a adaptive neuro fuzzy inference system(ANFIS). In ANFIS, a neural network is framed to do the job of learning and gaining experience. Once the learning process is over the Neural Network can be set to decide on the ranges for various linguistic variables and their overlap. In ANFIS system the ANN is used to coin the rules based on the experimental data supplied to the ANN. Thus if the capability of an ANN is incorporated with the FLC the drawbacks of the conventional FLC of being reliant on the designer is alleviated. A. Training data collection In the present design four FLCs were first designed and the system was put under service. There were a set of four error-error rate outputs. The individual data sets of these four data sets were individually used to train an ANFIS unit and thus four units of ANFIS were developed. The related screen shots as they appear step after in MATLAB SIMULINK environment are shown in figures below.

ANFIS structure







Performance comparison of two controllers for UPFC

Parameter	Fuzzy	ANFIS
P delivered to Load	1.4	1.457
Q Delivered to Load	1.02	1.034
PF at Source	0.94	0.99
THD at PCC	1.69%	1.18%

Algorithm for ANFIS:

Step 1: Define the linguistic variables and terms (initialization)
Step 2: Construct the membership functions (initialization)
Step 3: Construct the rule base (initialization)
Step 4: Convert crisp input data to fuzzy values using the membership functions (fuzzification)
Step 5: Evaluate the rules in the rule base (inference)
Step 6: Combine the results of each rule (inference)
Step 7: Convert the output data to non-fuzzy values (defuzzification)

Linguistic variables

- Quantification variables (All, Many, None)
- Usability variables (Sometimes, Frequently, Always)
- Likelihood variables (Possible, Likely, Certain)

The two most important concepts within Fuzzy Logic (FL) are Linguistic variable and fuzzy if-then rule. The widely used fuzzy if-then rule is:

1. IF (load is more than too-high) THEN command is fault.
2. IF (load is too-high) THEN command is Reduce the Load.
3. IF (load is Normal) THEN command is no-change.

Table 1. Sample test results

Sub Station	A	B	C	Fuzzy Remarks
Amps	Too Low	Over Load	Normal	Substation B Connected with substation A
	Low	Low	Over Load	Substation C Connected with substation A
	Over Load	low	Too Low	Substation A Connected with substation C
	Over Load	Normal	high	Substation A Connected with substation B
	Normal	High	Over load	Substation C Connected with substation A
	High	Overload	Normal	Substation B Connected with substation C

Since by using FL method we can detect the location of fault exactly it saves the time of the operator and the fault can be quickly rectified and the power can be supplied to the affected area immediately from the area where there is excess of power available by means of resource sharing. As fuzzy logic is simple and quick in detecting the power fault in the power system it is widely used. Normally each and every power grid substation is connected with Load Control Centre. Load Control Centre will be monitoring those of Substations every second. If fault occurs in one Substation that will be obtained by Load control Centre through SCADA (Supervisory control and Data Acquisition) and informed to healthy substations which are connected with the faulty Substation, then only power will be shared to faulty substation. In which Data Traffic is a major drawback because at the same time many substations may fail when communication delay will occur and so the resource recovery and load balancing will be delayed, people will be affected without power, cost loss. The ANFIS will overcome the present status of power sharing and fault diagnosis power system method which eliminates the need for Load Control Centre.