NLP, EVOLUTIONARY ALGORITHMS, FUZZY LOGIC, NEURAL NETWORKS

NATURAL LANGUAGE PROCESSING (NLP)

THE DREAM

- It'd be great if machines could
  - Process our email (usefully)
  - Translate languages accurately
  - Help us manage, summarize, and aggregate information
  - Use speech as a UI (when needed)
  - Talk to us / listen to us

- But they can't
  - Language is complex, ambiguous, flexible, and subtle
  - Good solutions need linguistics and machine learning knowledge

WHAT IS NLP?

- Natural language processing helps computer to understand human language as it is spoken.
- Real world use of natural languages such as English, Hindi, German, French etc. doesn't have a formulated structure and keeps on evolving.
- Natural language processing is an ongoing attempt to capture all the details from the natural languages.
- It sits at the intersection of computer science, artificial intelligence, and computational linguistics *(Wikipedia)*.

WE USE NLP EVERYDAY

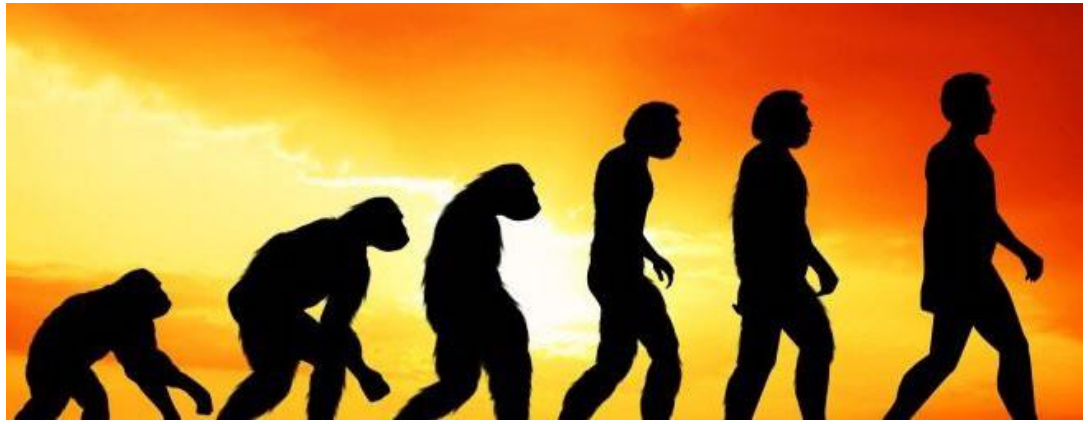
- **Word counters** (wc in UNIX)
- **Autocomplete** helps you to suggest rest of the word.
- Google search's **predictive typing** helps you by suggesting the next word.
- **Spell checker** in your email application saves you from stupid typing errors.
- **Spam detection** in your mail box separates spam mails from important ones.

MAJOR TASKS IN NLP

- Syntax
  - Part-of-speech tagging, parsing, stemming etc.
- Semantics
  - What is the computational meaning of individual words in context?
- Discourse
  - Automatic summarization, coreference resolution
- Speech
  - Speech recognition, text-to-speech

BIGGER APPLICATIONS

- Intelligent computer systems
- Computer aided instruction
- Information retrieval
- Intelligent web searching
- Data mining
- Machine translation
- Speech recognition
- Natural language generation
- Question answering



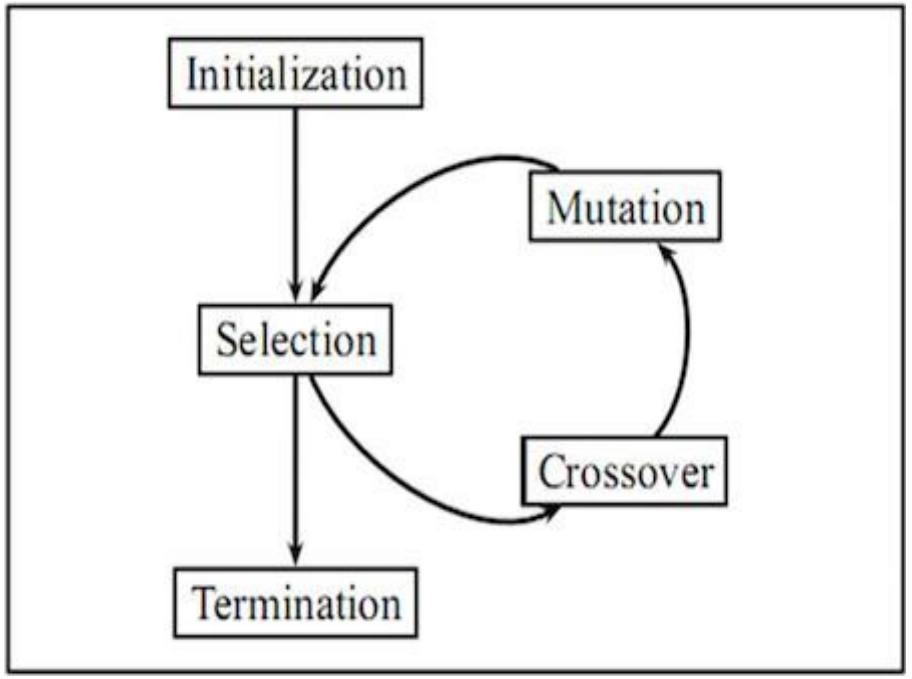
EVOLUTIONARY ALGORITHMS (EAs)

DEFINITION

- Evolutionary algorithms are a heuristic-based approach to solving problems that cannot be easily solved in polynomial time, such as classically NP-Hard problems, and anything else that would take far too long to exhaustively process.

STEPS IN EA

- The premise of an evolutionary algorithm is quite simple given that you are familiar with the process of natural selection.
- An EA contains four overall steps:
  - Initialization
  - Selection
  - Genetic operators
  - Termination
- These steps each correspond, roughly, to a particular facet of natural selection, and provide easy ways to modularize implementations of this algorithm category.
- Simply put, in an EA, fitter members will survive and proliferate, while unfit members will die off and not contribute to the gene pool of further generations, much like in natural selection.



PROBLEM DEFINITION

- We wish to find the best combination of elements that maximizes some fitness function, and we will accept a final solution once we have either ran the algorithm for some maximum number of iterations, or we have reached some fitness threshold.
- This scenario is clearly not the only way to use an EA, but it does encompass many common applications in the discrete case.

FIRST STEP: INITIALIZATION

- In order to begin our algorithm, we must first create an initial **population** of solutions.
- The population will contain an arbitrary number of possible solutions to the problem, oftentimes called **members**.
- It will often be created randomly (within the constraints of the problem) or, if some prior knowledge of the task is known, roughly centered around what is believed to be ideal.
- It is important that the population encompasses a wide range of solutions, because it essentially represents a gene pool; ergo, if we wish to explore many different possibilities over the course of the algorithm, we should aim to have many different genes present.

SECOND STEP: SELECTION

- Once a population is created, members of the population must now be evaluated according to a **fitness function**.
- A fitness function is a function that takes in the characteristics of a member, and outputs a numerical representation of how viable of a solution it is.
- Creating the fitness function can often be very difficult, and it is important to find a good function that accurately represents the data; it is very problem-specific.
- Now, we calculate the fitness of all members, and select a portion of the top-scoring members.

THIRD STEP: GENETIC OPERATORS

- This step really includes two sub-steps: **crossover and mutation**.
- Crossover:
  - After selecting the top members (typically top 2, but this number can vary), these members are now used to create the next generation in the algorithm.
  - Using the characteristics of the selected parents, new children are created that are a mixture of the parents' qualities.
  - Doing this can often be difficult depending on the type of data, but typically in combinatorial problems, it is possible to mix combinations and output valid combinations from these inputs.
- Mutation:
  - Now, we must introduce **new genetic material** into the generation.
  - If we do not do this crucial step, we will become stuck in local extrema very quickly, and will not obtain optimal results. This step is mutation, and we do this, quite simply, by changing a small portion of the children such that they no longer perfectly mirror subsets of the parents' genes.

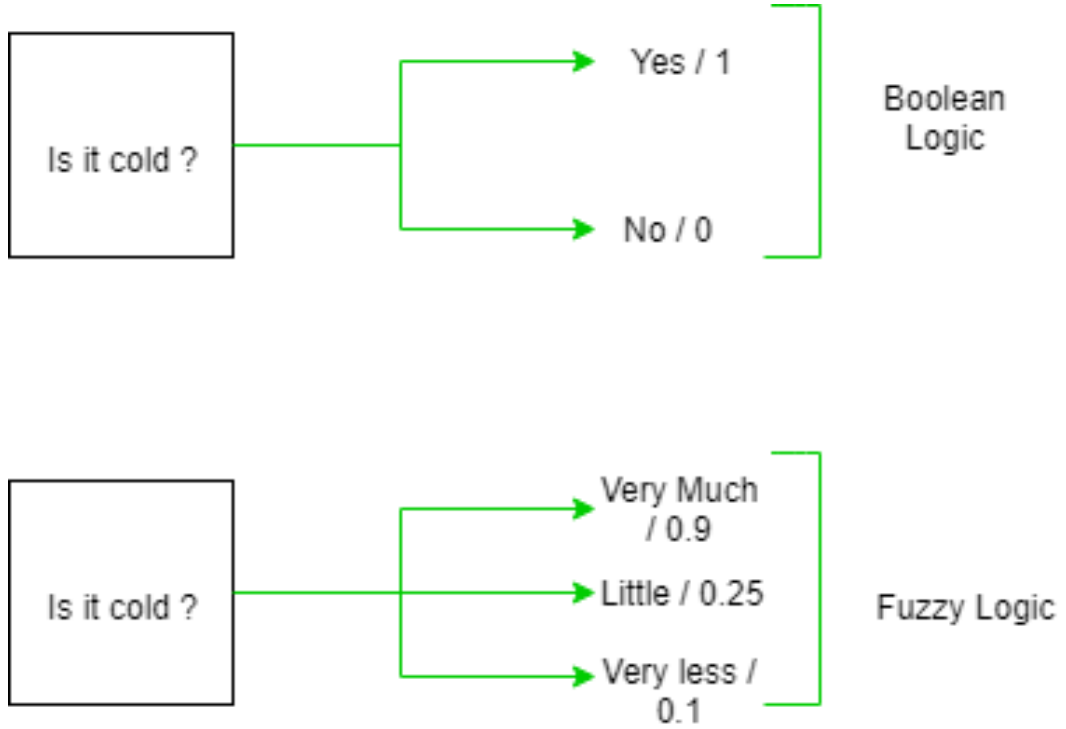
FOURTH STEP: TERMINATION

- Eventually, the algorithm must end.
- There are two cases in which this usually occurs:
  - either the algorithm has reached some **maximum runtime**, or
  - the algorithm has reached some **threshold of performance**.
- At this point a final solution is selected and returned.

FUZZY LOGIC

DEFINITION

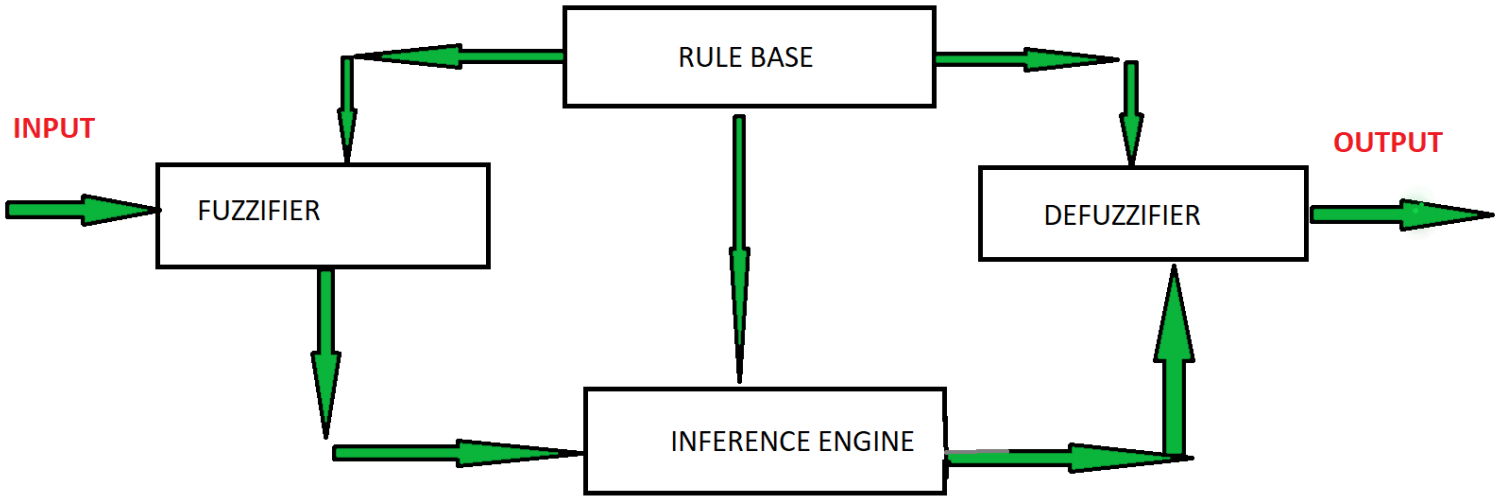
- **Fuzzy logic** is a form of many-valued logic in which the truth values of variables may be any real number between 0 and 1.
- It is employed to handle the concept of partial truth, where the truth value may range between completely true and completely false.
- By contrast, in Boolean logic, the truth values of variables may only be the integer values 0 or 1.
- For example, we might say that 'President Clinton is tall,' with degree of truth of 0.9.



ARCHITECTURE

- RULE BASE:
  - It contains the set of rules and the IF-THEN conditions provided by the experts to govern the decision making system, on the basis of linguistic information.
- FUZZIFICATION:
  - It is used to convert inputs i.e. crisp numbers into fuzzy sets.
  - Crisp inputs are basically the exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm's, etc.
- INFERENCE ENGINE:
  - It determines the matching degree of the current fuzzy input with respect to each rule and decides which rules are to be fired according to the input field.
  - Next, the fired rules are combined to form the control actions.
- DEFUZZIFICATION:
  - It is used to convert the fuzzy sets obtained by inference engine into a crisp value.

ARCHITECTURE



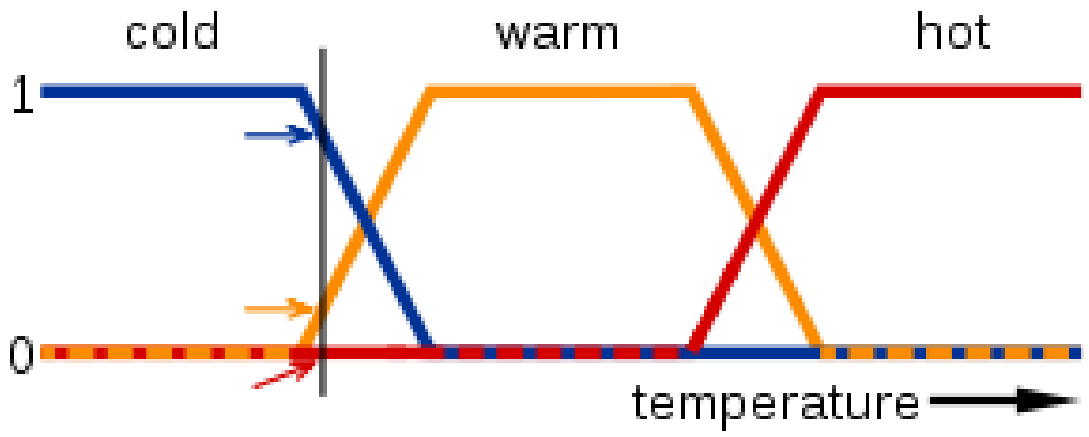
FUZZY LOGIC ARCHITECTURE

MEMBERSHIP FUNCTION

- Membership functions allow you to quantify linguistic term and represent a fuzzy set graphically.
- A **membership function** for a fuzzy set *A* on the universe of discourse *X* is defined as $\mu_A: X \rightarrow [0,1]$.
- Here, each element of *X* is mapped to a value between 0 and 1.
- It is called **membership value** or **degree of membership**.
- It quantifies the degree of membership of the element in *X* to the fuzzy set *A*.

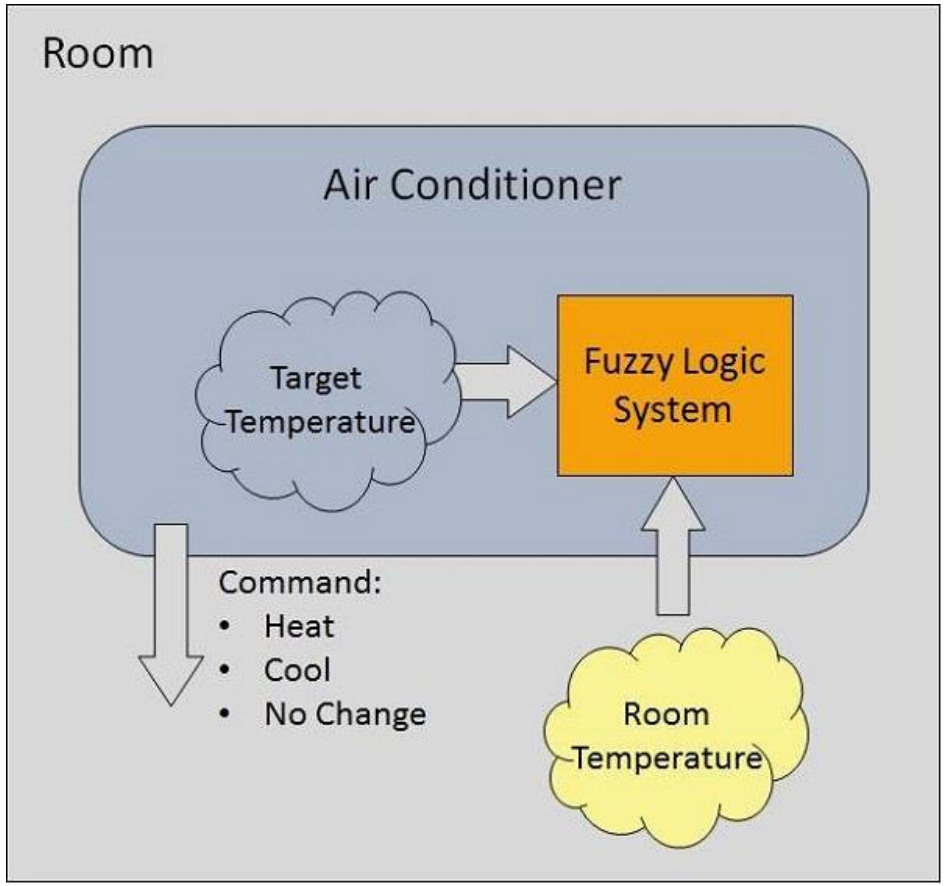
EXAMPLE

- In this image, the meanings of the expressions *cold*, *warm*, and *hot* are represented by functions mapping a temperature scale.
- A point on that scale has three "truth values"—one for each of the three functions.
- The vertical line in the image represents a particular temperature that the three arrows (truth values) gauge.
- Since the red arrow points to zero, this temperature may be interpreted as "not hot".
- The orange arrow (pointing at 0.2) may describe it as "slightly warm" and the blue arrow (pointing at 0.8) "fairly cold".



ANOTHER EXAMPLE

- Let us consider an air conditioning system with 5-level fuzzy logic system.
- This system adjusts the temperature of air conditioner by comparing the room temperature and the target temperature value.



ALGORITHM

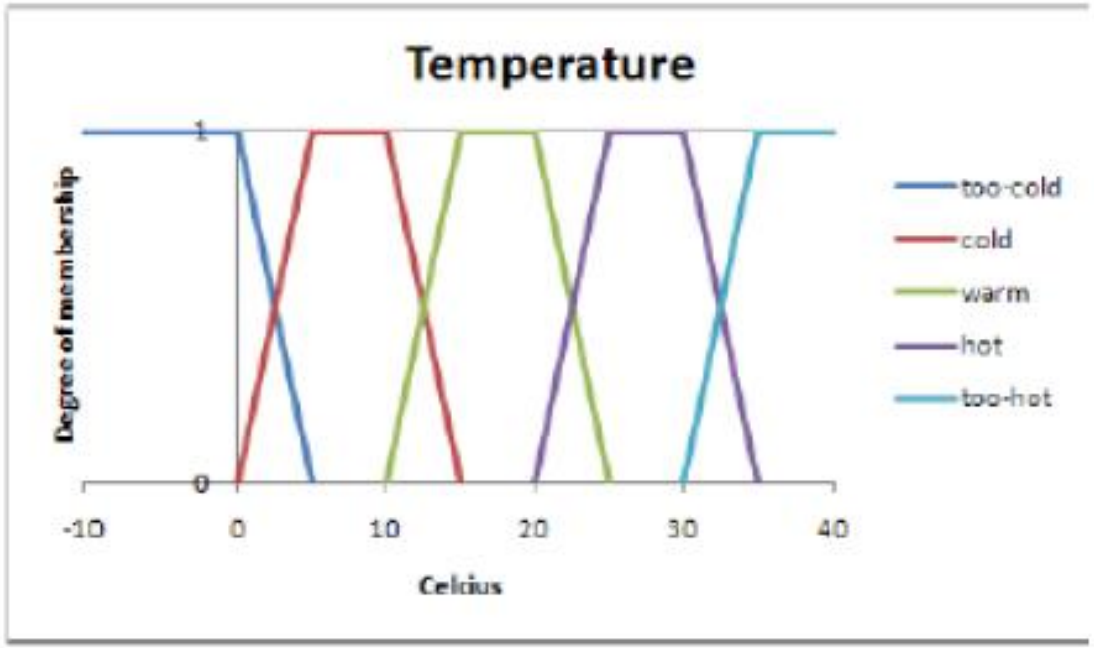
- Define linguistic variables and terms.
- Construct membership functions for them.
- Construct knowledge base of rules.
- Convert crisp data into fuzzy data sets using membership functions (fuzzification).
- Evaluate rules in the rule base (inference engine).
- Combine results from each rule (inference engine).
- Convert output data into non-fuzzy values (defuzzification).

STEP 1: DEFINE LINGUISTIC VARIABLES AND TERMS

- Linguistic variables are input and output variables in the form of simple words or sentences.
- For room temperature, cold, warm, hot, etc., are linguistic terms.
- Temperature (t) = {very-cold, cold, warm, very-warm, hot}
- Every member of this set is a linguistic term and it can cover some portion of overall temperature values.

STEP 2: CONSTRUCT MEMBERSHIP FUNCTIONS FOR THEM

- Membership functions are used in the fuzzification and defuzzification steps of a FLS, to map the non-fuzzy input values to fuzzy linguistic terms and vice versa.
- Note that, an important characteristic of fuzzy logic is that a numerical value does not have to be fuzzified using only one membership function. In other words, a value can belong to multiple sets at the same time. For example, according to Figure 3, a temperature value can be considered as \cold" and \too-cold" at the same time, with different degree of memberships.
- The membership functions of temperature variable are as shown:



STEP3: CONSTRUCT KNOWLEDGE BASE RULES

- Create a matrix of room temperature values versus target temperature values that an air conditioning system is expected to provide.

RoomTemp. /Target	Very_Cold	Cold	Warm	Hot	Very_Hot
Very_Cold	No_Change	Heat	Heat	Heat	Heat
Cold	Cool	No_Change	Heat	Heat	Heat
Warm	Cool	Cool	No_Change	Heat	Heat
Hot	Cool	Cool	Cool	No_Change	Heat
Very_Hot	Cool	Cool	Cool	Cool	No_Change

STEP3: CONSTRUCT KNOWLEDGE BASE RULES

- Build a set of rules into the knowledge base in the form of IF-THEN-ELSE structures.
- Sample fuzzy rules for the air conditioner system are listed:

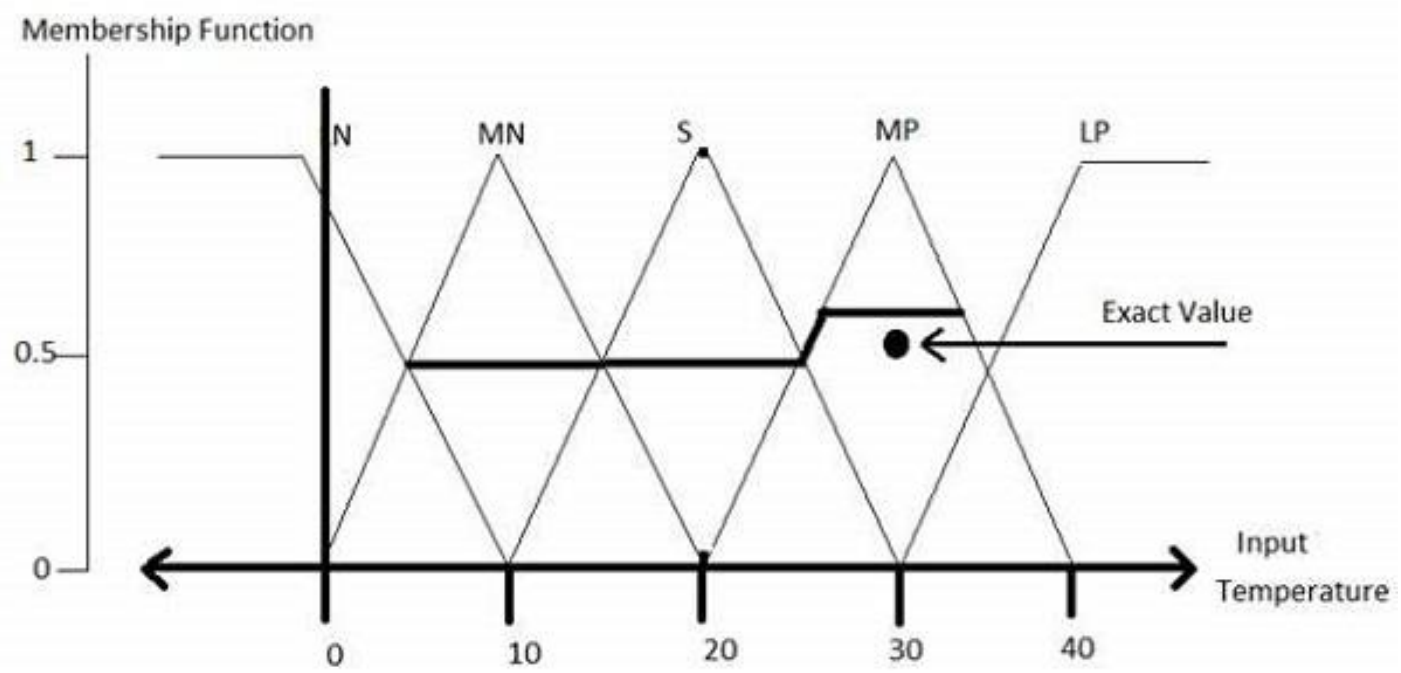
Sr. No.	Condition	Action
1	IF temperature=(Cold OR Very_Cold) AND target=Warm THEN	Heat
2	IF temperature=(Hot OR Very_Hot) AND target=Warm THEN	Cool
3	IF (temperature=Warm) AND (target=Warm) THEN	No_Change

STEP 4: OBTAIN FUZZY VALUE

- Fuzzy set operations perform evaluation of rules.
- The operations used for OR and AND are Max and Min respectively.
- Combine all results of evaluation to form a final result.
- This result is a fuzzy value.

STEP 5: PERFORM DEFUZZIFICATION

- Defuzzification is then performed according to membership function for output variable.



APPLICATION AREAS OF FUZZY LOGIC

- **Automotive Systems**
  - Automatic Gearboxes
  - Four-Wheel Steering
  - Vehicle environment control
- **Consumer Electronic Goods**
  - Hi-Fi Systems
  - Photocopiers
  - Still and Video Cameras
  - Television
- **Domestic Goods**
  - Microwave Ovens
  - Refrigerators
  - Toasters
  - Vacuum Cleaners
  - Washing Machines
- **Environment Control**
  - Air Conditioners/Dryers/Heaters
  - Humidifiers

ADVANTAGES

- Mathematical concepts within fuzzy reasoning are very simple.
- You can modify a fuzzy logic system by just adding or deleting rules due to flexibility of fuzzy logic.
- Fuzzy logic systems can take imprecise, distorted, noisy input information.
- Fuzzy logic systems are easy to construct and understand.
- Fuzzy logic is a solution to complex problems in all fields of life, including medicine, as it resembles human reasoning and decision making.

DISADVANTAGES

- There is no systematic approach to fuzzy system designing.
- They are understandable only when simple.
- They are suitable for the problems which do not need high accuracy.

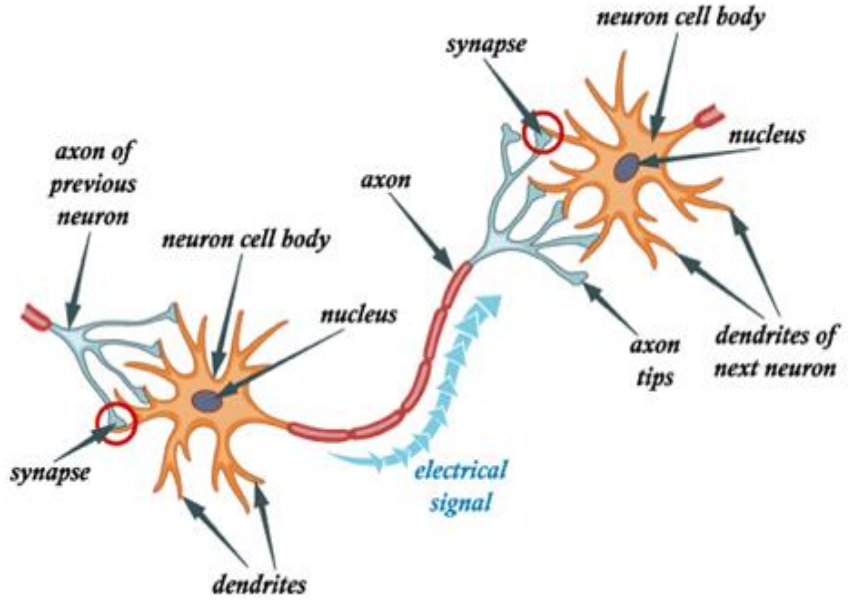
NEURAL NETWORKS

DEFINITION

- More properly referred to as an 'artificial' neural network (ANN)
- The inventor of the first neurocomputer, Dr. Robert Hecht-Nielsen, defines a neural network as -
- *"…A computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."*
- ANNs are processing devices (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the mamalian cerebral cortex but on much smaller scales.
- A large ANN might have hundreds or thousands of processor units, whereas a mamalian brain has billions of neurons with a corresponding increase in magnitude of their overall interaction and emergent behavior.
- Although ANN researchers are generally not concerned with whether their networks accurately resemble biological systems, some have.
  - For example, researchers have accurately simulated the function of the retina and modeled the eye rather well.

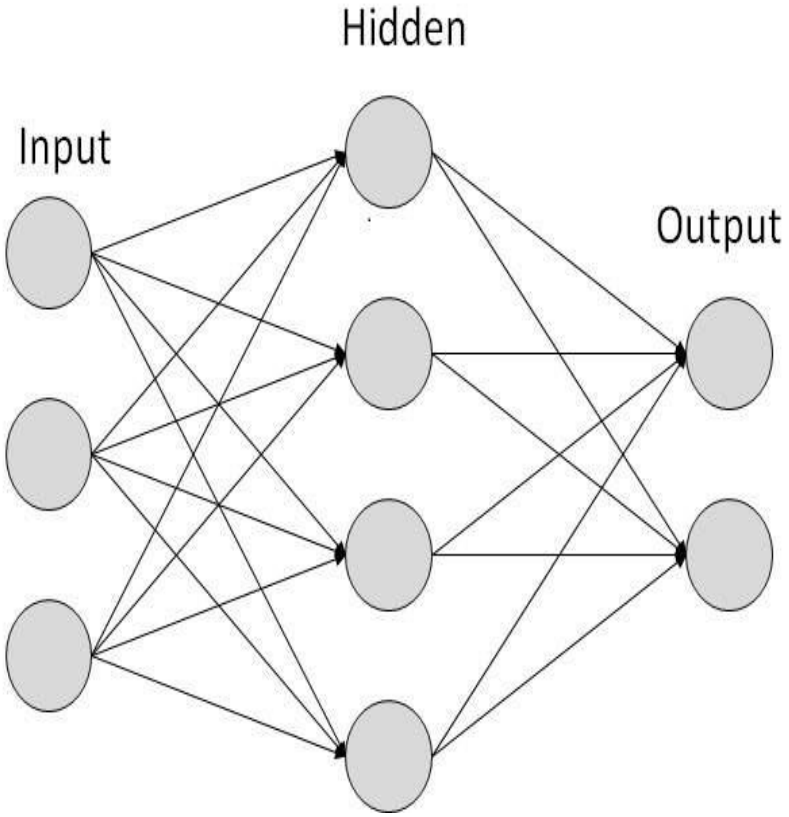
BASIC STRUCTURE OF ANNs

- The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living **neurons** and **dendrites**.
- The human brain is composed of 86 billion nerve cells called **neurons.**
- They are connected to other thousand cells by **axons.**
- Stimuli from external environment or inputs from sensory organs are accepted by dendrites.
- These inputs create electric impulses, which quickly travel through the neural network.
- A neuron can then send the message to other neuron to handle the issue or does not send it forward.



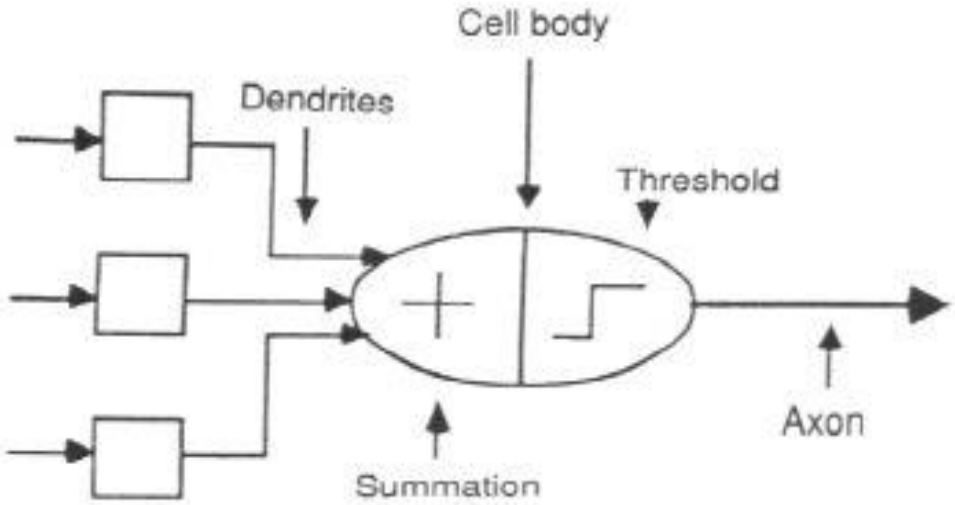
BASIC STRUCTURE OF ANNs

- ANNs are composed of multiple **nodes**, which imitate biological **neurons** of human brain.
- The neurons are connected by links and they interact with each other.
- The nodes can take input data and perform simple operations on the data.
- The result of these operations is passed to other neurons.
- The output at each node is called its **activation** or **node value.**
- Each link is associated with **weight.**
- ANNs are capable of learning, which takes place by altering weight values.



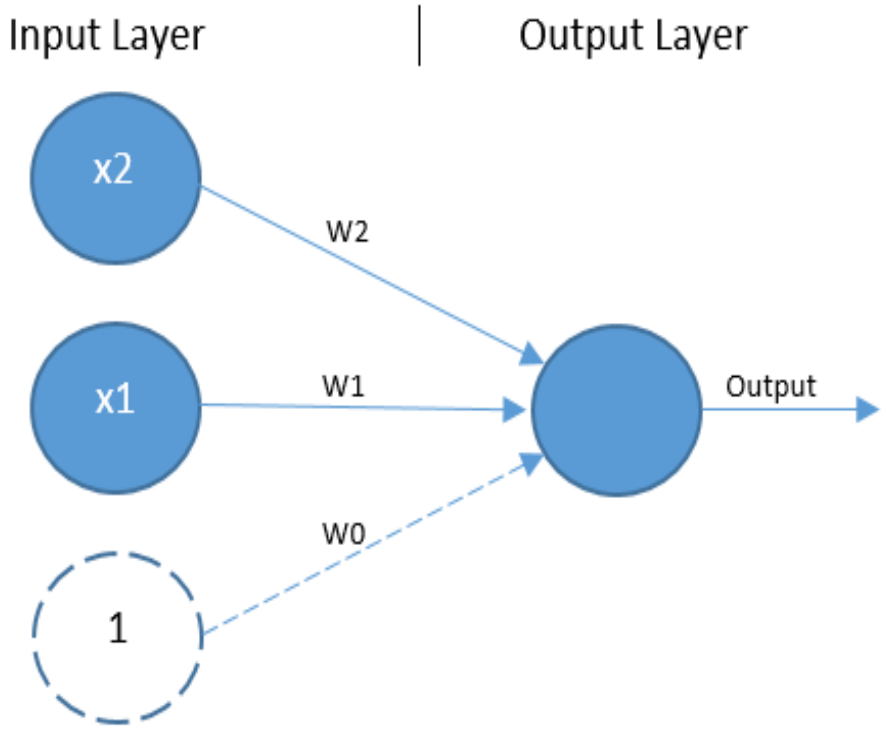
WORKING OF ANNs

- In the topology diagrams shown, each arrow represents a connection between two neurons and indicates the pathway for the flow of information.
- Each connection has a weight, an integer number that controls the signal between the two neurons.
- If the network generates a "good or desired" output, there is no need to adjust the weights.
- However, if the network generates a "poor or undesired" output or an error, then the system alters the weights in order to improve subsequent results.



A SINGLE NEURON: PERCEPTRON

- Perceptrons include a single layer of input units.
- This network takes numerical inputs **X1** and **X2** and has weights **w1** and **w2** associated with those inputs.
- Additionally, there is another input **1** with weight **b** (called the **bias**) associated with it.
- The output **Y** from the neuron is computed as shown in the figure.
- The function ***f*** is non-linear and is called the **activation function**.
- The activation function uses some means or other to reduce the sum of input values to a 1 or a 0 (or a value very close to a 1 or 0) in order to represent activation or lack thereof.
- Another form of unit, known as a bias unit, always activates, typically sending a hard coded 1 to all units to which it is connected.
- Here a bias unit is depicted by a dashed circle, while other units are shown as blue circles.

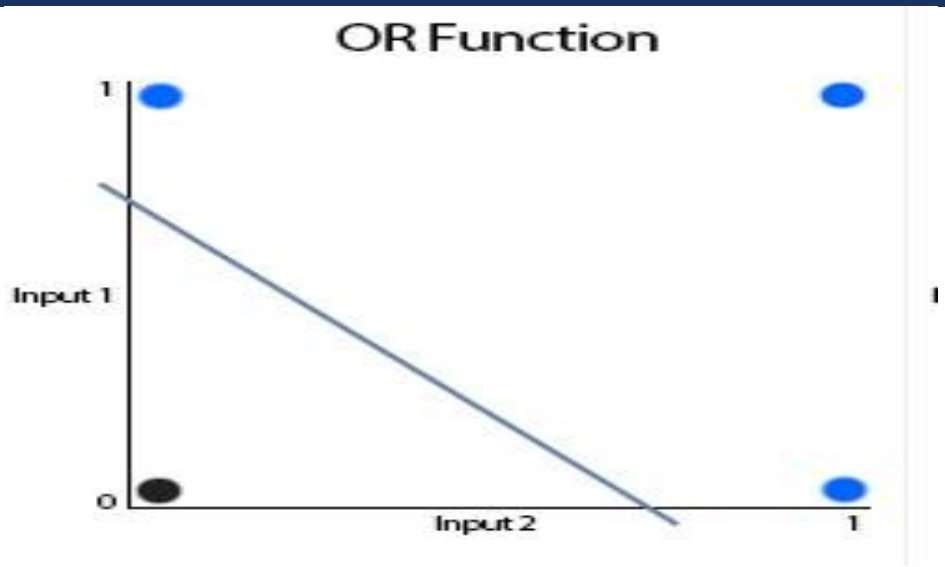


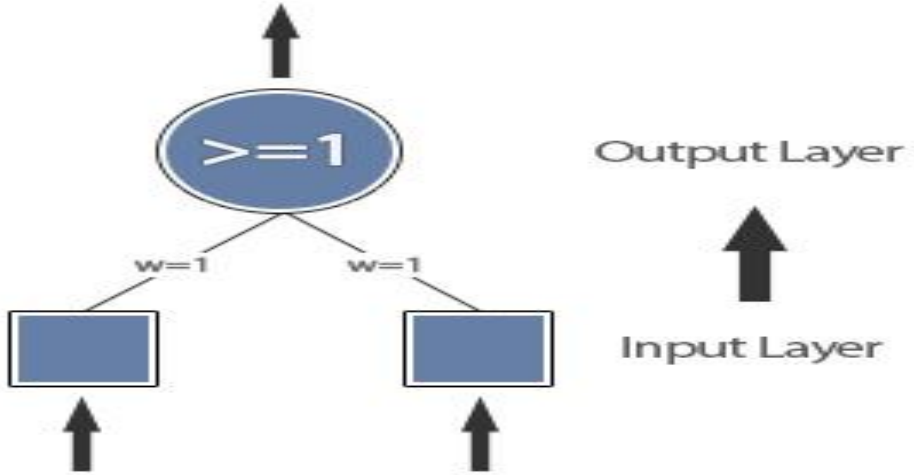
FEED-FORWARD NETWORKS

- The perceptron is a type of feed-forward network, which means the process of generating an output—known as forward propagation—flows in one direction from the input layer to the output layer.
- There are no connections between units in the input layer.
- Instead, all units in the input layer are connected directly to the output unit.

OR FUNCTION

- Notice how the OR function can be separated on the graph with a single straight line, this means the function is "linearly separable" and can be modelled within our neural network without implementing a hidden layer, for example, the OR function can be modeled with a single perceptron like this:





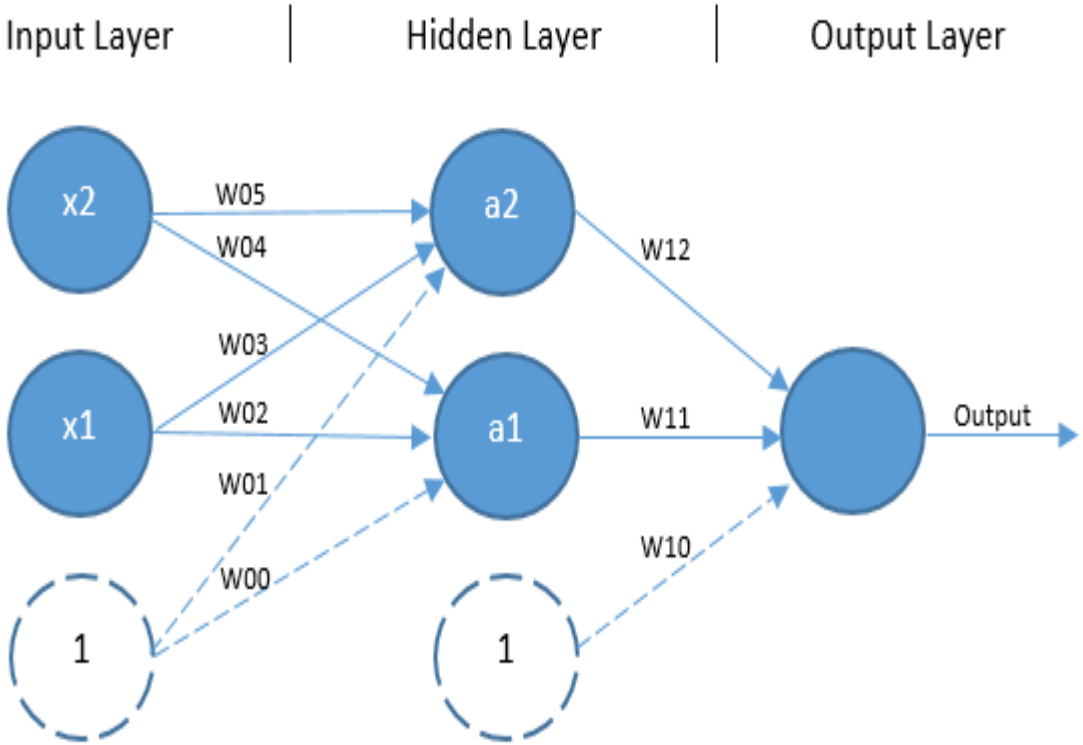
THE XOR PROBLEM

- On the surface, XOr appears to be a very simple problem, however, Minksy and Papert (1969) showed that this was a big problem for neural network architectures of the 1960s, known as perceptrons.
- A limitation of perceptron architecture is that it is only capable of separating data points with a single line.
- This is unfortunate because the XOR inputs are not linearly separable. This is particularly visible if you plot the XOR input values to a graph.
- There is no way to separate the 1 and 0 predictions with a single classification line.

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

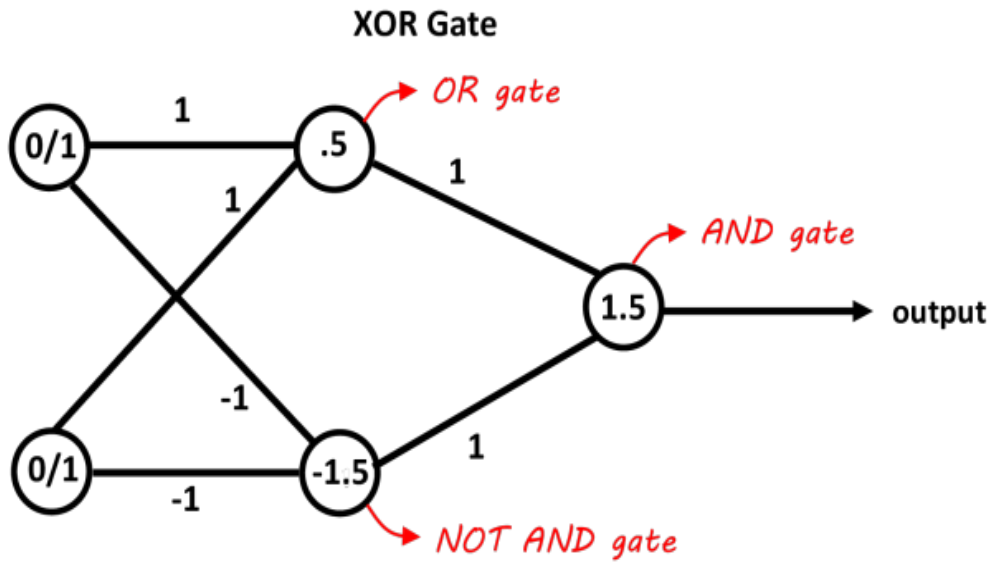
MULTILAYER PERCEPTRONS

- The solution to this problem is to expand beyond the single-layer architecture by adding an additional layer of units without any direct access to the outside world, known as a hidden layer.
- This kind of architecture is another feed-forward network known as a multilayer perceptron (MLP).
- MLP can have any number of units in its input, hidden and output layers. There can also be any number of hidden layers.



SOLUTION TO THE XOR PROBLEM

- A network with one hidden layer containing two neurons should be enough to separate the XOR problem. Follow these steps :-
- The first neuron acts as an OR gate and the second one as a NOT AND gate.
- Add both the neurons and if they pass the threshold it's positive. You can just use linear decision neurons for this with adjusting the biases for the thresholds.
- The inputs of the NOT AND gate should be negative for the 0/1 inputs.

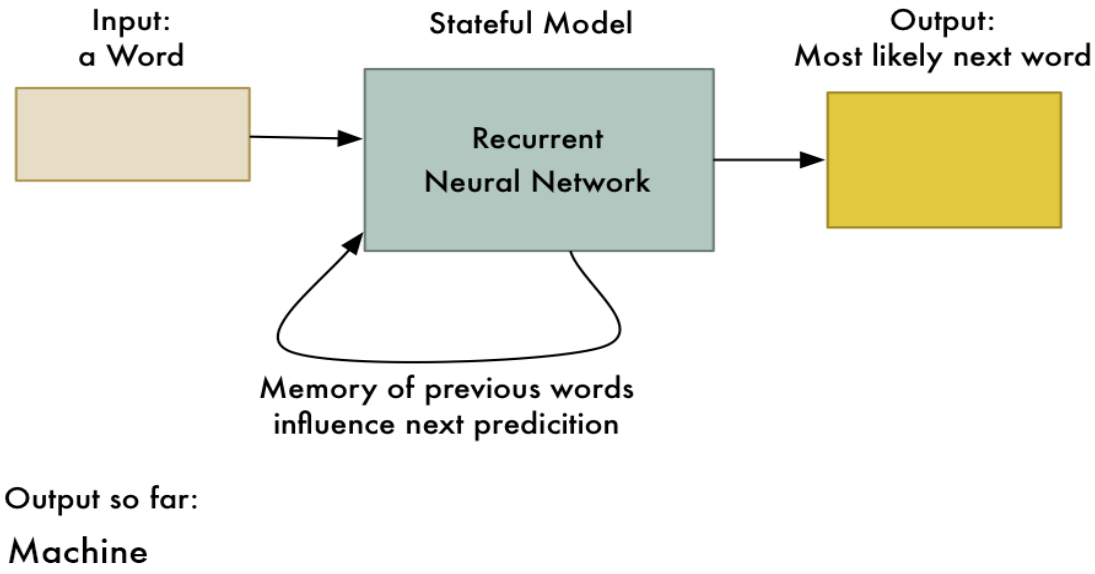


FEED-FORWARD NEURAL NETWORKS

- This neural network is one of the simplest form of ANN, where the data or the input travels in one direction.
- The data passes through the input nodes and exit on the output nodes.
- This neural network may or may not have the hidden layers.
- In simple words, it has a front propagated wave and no back propagation by using a classifying activation function usually.
- Application of feed-forward neural networks are found in computer vision and speech recognition where classifying the target classes are complicated.
- These kind of neural networks are responsive to noisy data and easy to maintain.

RECURRENT NEURAL NETWORKS (RNNs) – LONG SHORT TERM MEMORY

- The Recurrent Neural Network works on the principle of saving the output of a layer and feeding this back to the input to help in predicting the outcome of the layer.
- Here, the first layer is formed similar to the feed forward neural network with the product of the sum of the weights and the features.
- The recurrent neural network process starts once this is computed, this means that from one time step to the next each neuron will remember some information it had in the previous time-step.
- This makes each neuron act like a memory cell in performing computations.
- In this process, we need to let the neural network to work on the front propagation and remember what information it needs for later use.



THANK YOU.