Learning (Knowledge Acquired by Study)

According to Simons:

- Changes in the System in that are adaptive in the sense that they enable the system to do the same task or task drawn by the same population more efficiently the next time.
- Process by which one entity acquires knowledge

Learning Strategies:

- Learning by Advices.
- Learning by Problem solving
- Learning by Examples.

Learning Methods:

- MEMORIZATION(Rote Learning)
- DIRECT INSTRUCTION(By Being told)
- ANALOGY
- INDUCTION
- DEDUCTION

Rote Learning (Memorization)

- Simple
- Requires least amount of interference Simply coping the knowledge in the same form that it will be used directly into knowledge base.
- Ex : used for memorizing multiplication table
- In case of data caching , we store computed values we do not have to recomputed them later, thus saving a significant time .Caching is used in AI programs to produce some surprising performance improvements .Such caching is known as **Rote Learning**

DIRECT INSTRUCTION (By being told)

- Slightly complex from previous.
- Requires more inference.
- Knowledge must be transformed into an operational form being integrated into knowledge base.
- EX: As teacher presents a number of facts directly to us in well-organized manner.

ANALOGY

- Requires more inferencing.
- Process of learning new concept or solutions through the use of similar known concepts or solutions.
- EX : when we learn to drive a truck using our knowledge of car driving.

INDUCTION:

- Similar to analogy learning
- Requires more inferring than first two methods.
- Used when we formulate a general concept after seeing a number of instances of examples of the concept.
- EX : We learn the concept of color or sweet taste after experiencing the sensations associated with the several examples of colored objects or sweet foods

Deduction

- New facts are derived from the known facts.
- Requires more inference
- EX: we could learn deductively that Bob & Sam are cousin's ,if we have the knowledge of Bob and Sam's parents and rules for the cousin relationship.

INDUCTION LEARNING:

Definition: Process of acquiring generalized knowledge from examples or instance of some classes.

Terms:

- CLASS
- OBJECT
- CONCEPT
- HYPOTHESIS
- TARGET CONCEPT
- POSITIVE INSTANCE
- NEGATIVE INSTANCE

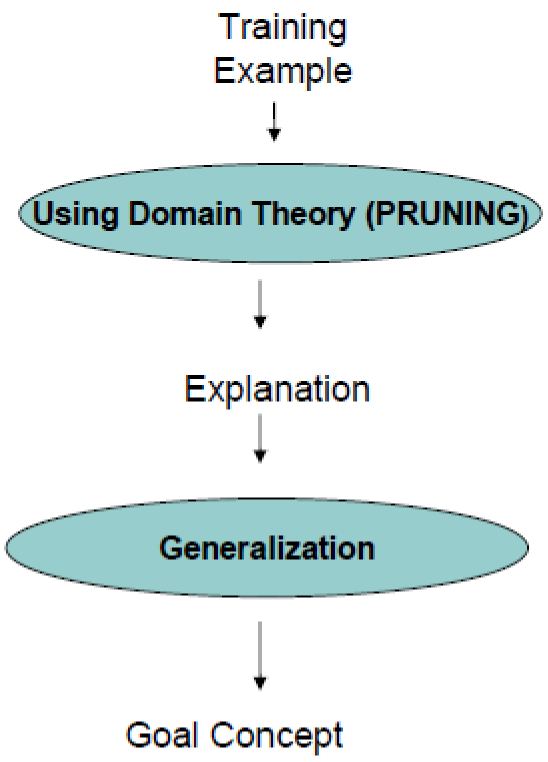
Explanation BASED LEARNING

- Known as Explanation Based Generalization.
- Form of Deductive Generalization.

INPUTS USED BY EBL PROGRAM:

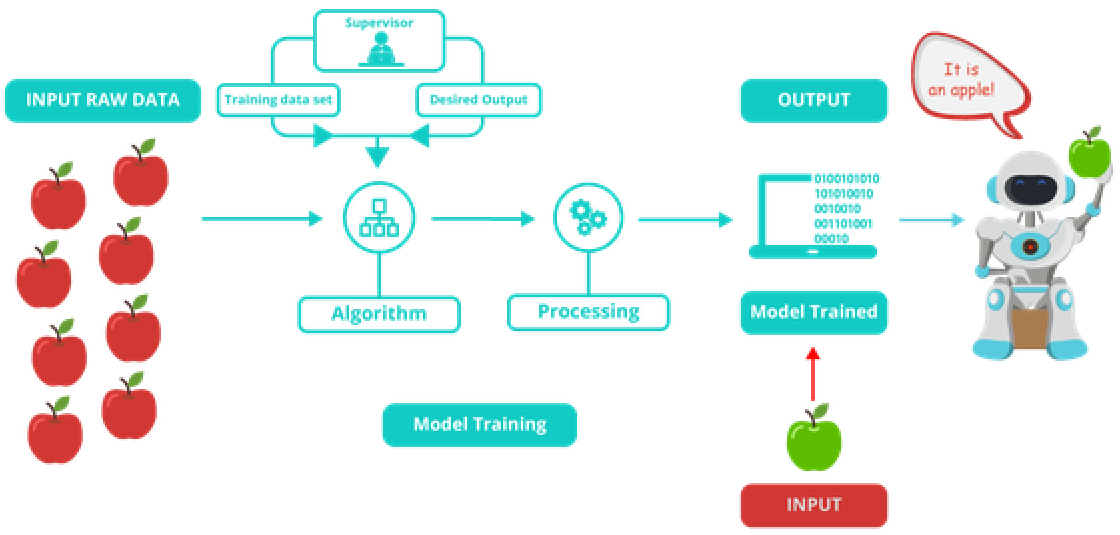
- Training example.
- Goal concept.
- Operational criteria
- Domain Theory

Steps for EBL

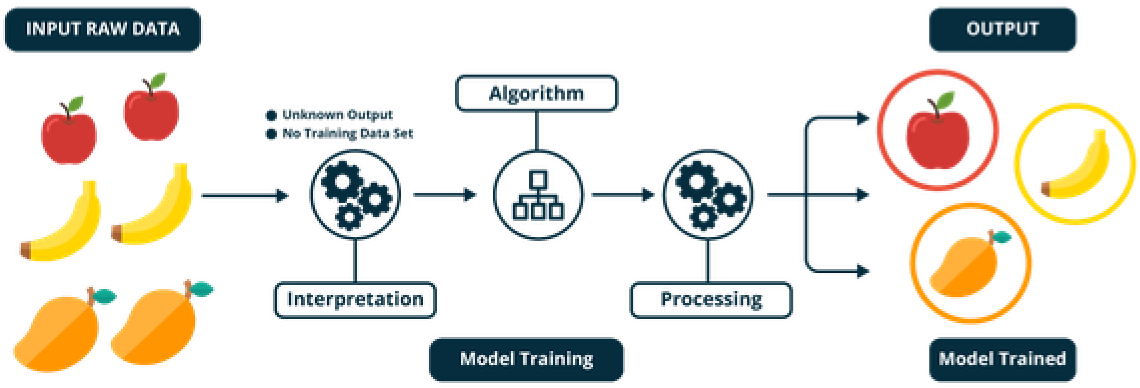


Learning Types:

a) Supervised learning: Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labelled training data consisting of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value (also called the *supervisory signal*). A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples



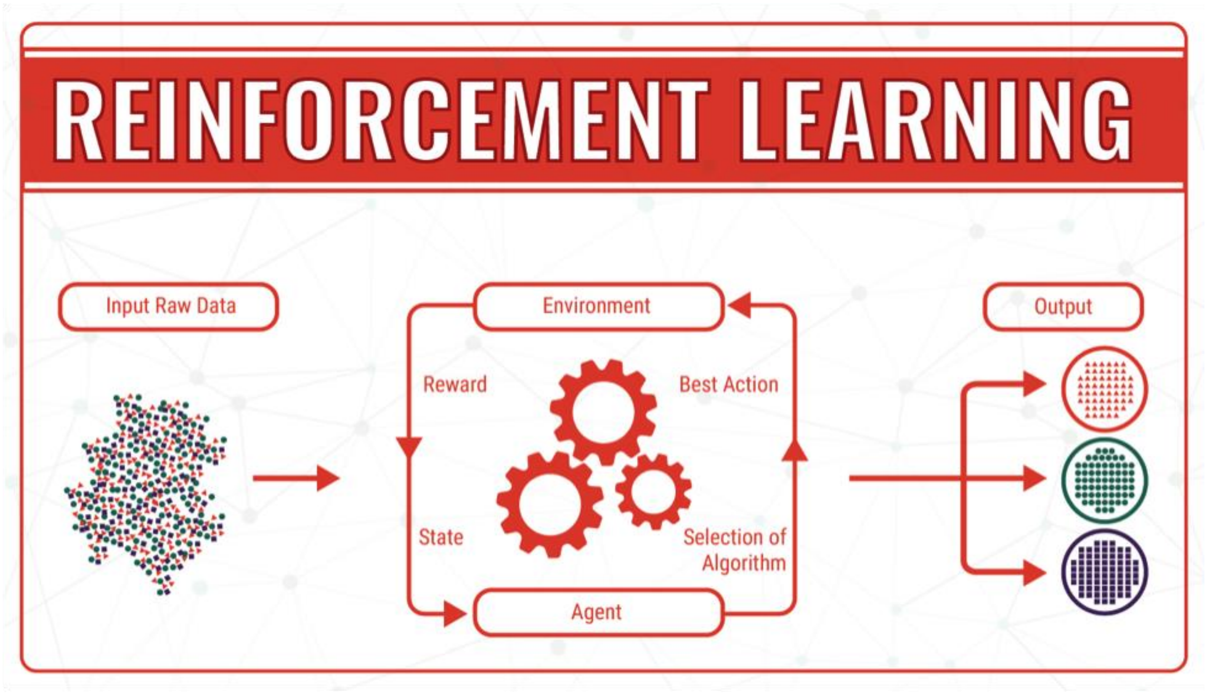
b) Unsupervised Learning Unsupervised learning is the training of an artificial intelligence (AI) algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. In unsupervised learning, an AI system may group unsorted information according to similarities and differences even though there are no categories provided. AI systems capable of unsupervised learning are often associated with generative learning models, although they may also use a retrieval-based approach (which is most often associated with supervisedlearning). Chatbots, self-drivingcars, facial cognition programs, expert systems and robots are among the systems that may use either supervised or unsupervised learning approaches.



c) Reinforcement learning

Reinforcement learning (**RL**) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize some notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

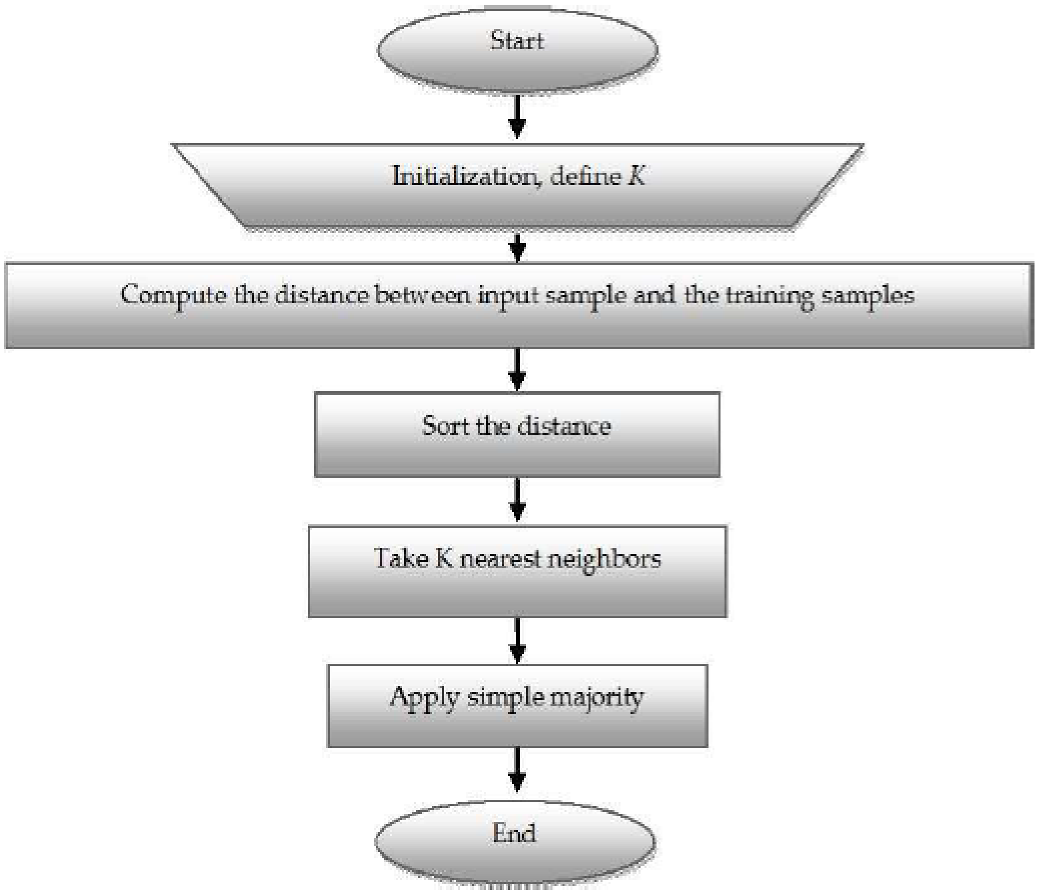
Reinforcement learning differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge). The environment is typically stated in the form of a Markov decision process (MDP), because many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible.



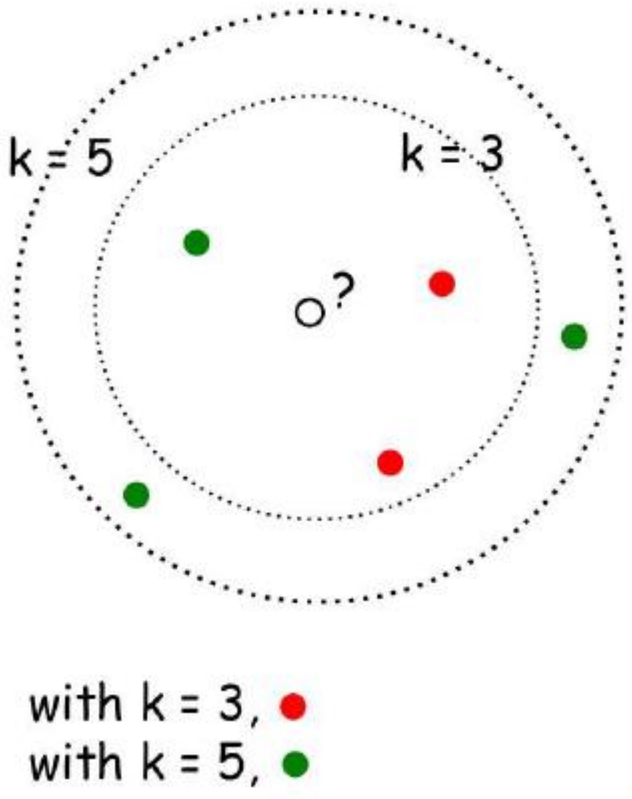
KNN(K- Nearest neighbour):

What is k- NN??

- Nearest-neighbour classifiers are based on learning by analogy, that is, by comparing a given test tuple with training tuples that are similar to it.
- The training tuples are described by n attributes.
- When k = 1, the unknown tuple is assigned the class of the training tuple that is closest to it in pattern space.


Start
Initialization, define K
Compute the distance between input sample and the training samples
Sort the distance
Take K nearest neighbors
Apply simple majority
End

When k=3 or k=5??


k = 5
k = 3
?
with k = 3, ●
with k = 5, ●

Fuzzy logic

Fuzzy Logic Systems *FLS* produce acceptable but definite output in response to incomplete, ambiguous, distorted, or inaccurate *fuzzy* input.

What is Fuzzy Logic?

Fuzzy Logic *FL* is a method of reasoning that resembles human reasoning. The approach of FL imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO.

The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO

The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as –

CERTAINLY YES
POSSIBLY YES
CANNOT SAY
CERTAINLY
POSSIBLY NO

The fuzzy logic works on the levels of possibilities of input to achieve the definite output.

Implementation

It can be implemented in systems with various sizes and capabilities ranging from small micro -controllers to large, networked, workstation-based control systems.

It can be implemented in hardware, software, or a combination of both.

Why Fuzzy Logic?

Fuzzy logic is useful for commercial and practical purposes.

It can control machines and consumer products.

It may not give accurate reasoning, but acceptable reasoning.

Fuzzy logic helps to deal with the uncertainty in engineering.

Fuzzy Logic Systems Architecture

It has four main parts as shown –

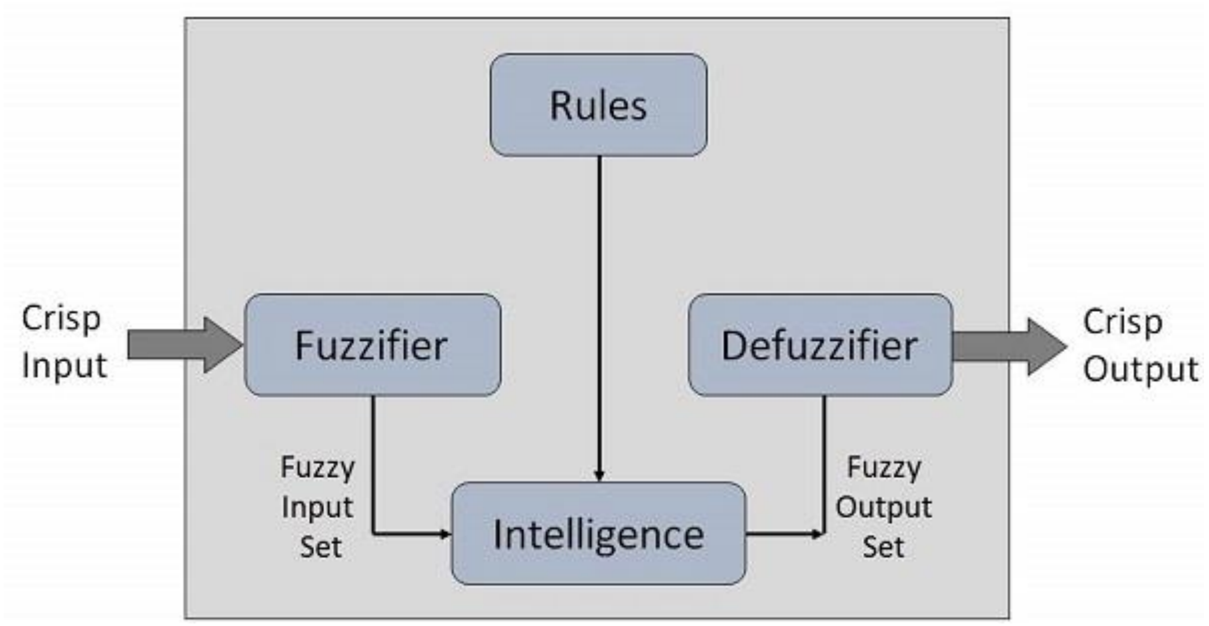
Fuzzification Module − It transforms the system inputs, which are crisp numbers, into fuzzy sets. It splits the input signal into five steps such as –

LP	x is Large Positive
MP	x is Medium Positive
S	x is small
MN	x is Medium Negative
LN	x is Large Negative

Knowledge Base − It stores IF-THEN rules provided by experts.

Inference Engine − It simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN rules.

Defuzzification Module − It transforms the fuzzy set obtained by the inference engine into a crisp value



The **membership functions work on** fuzzy sets of variables.

Membership Function

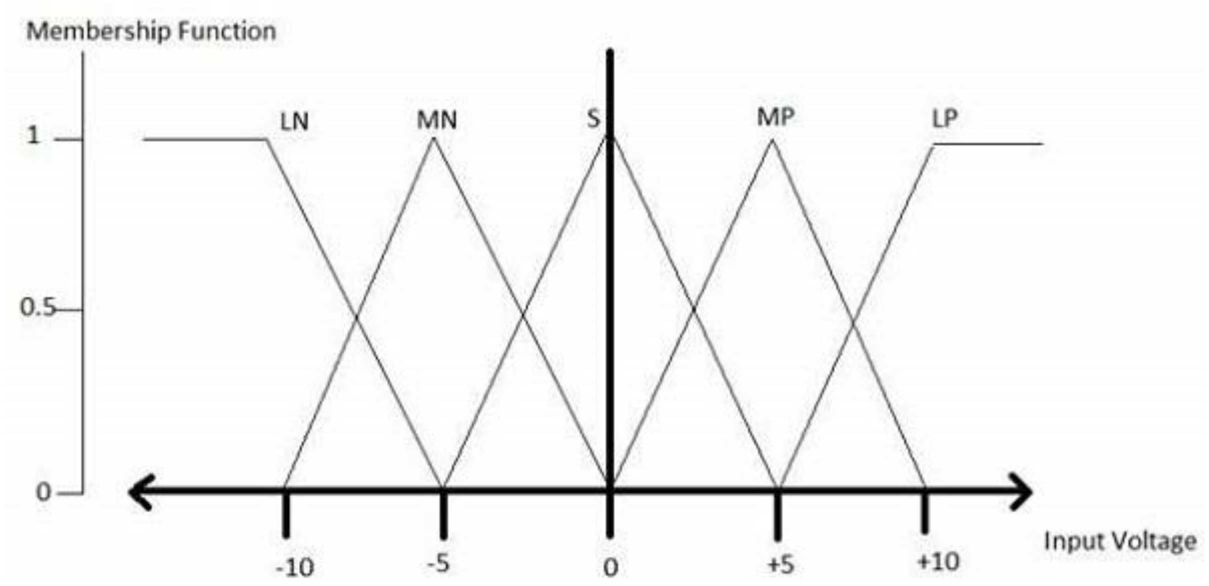
Membership functions allow you to quantify linguistic term and represent a fuzzy set graphically. A **membership function** for a fuzzy *set A* on the universe of discourse X is defined as µA:X → [0,1].

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*.

x axis represents the universe of discourse.

y axis represents the degrees of membership in the [0, 1] interval.

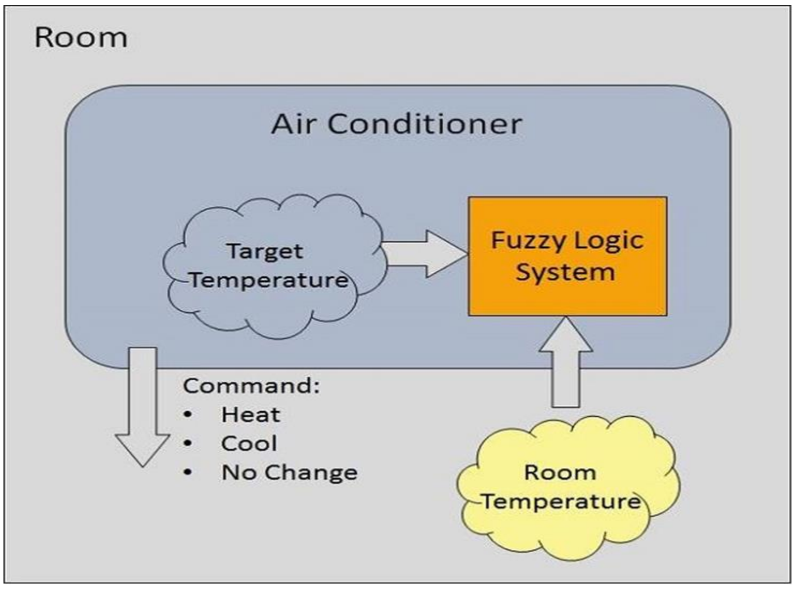
There can be multiple membership functions applicable to fuzzify a numerical value. Simple membership functions are used as use of complex functions does not add more precision in the output. All membership functions for **LP, MP, S, MN,** and **LN** are shown as below –



The triangular membership function shapes are most common among various other membership function shapes such as trapezoidal, singleton, and Gaussian. Here, the input to 5-level fuzzifier varies from -10 volts to +10 volts. Hence the corresponding output also changes.

Example of a Fuzzy Logic System

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 (start)
- Construct membership functions for them. (start)

- Construct knowledge base of rules (start)
- 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)

Development

Step 1 − Define linguistic variables and terms

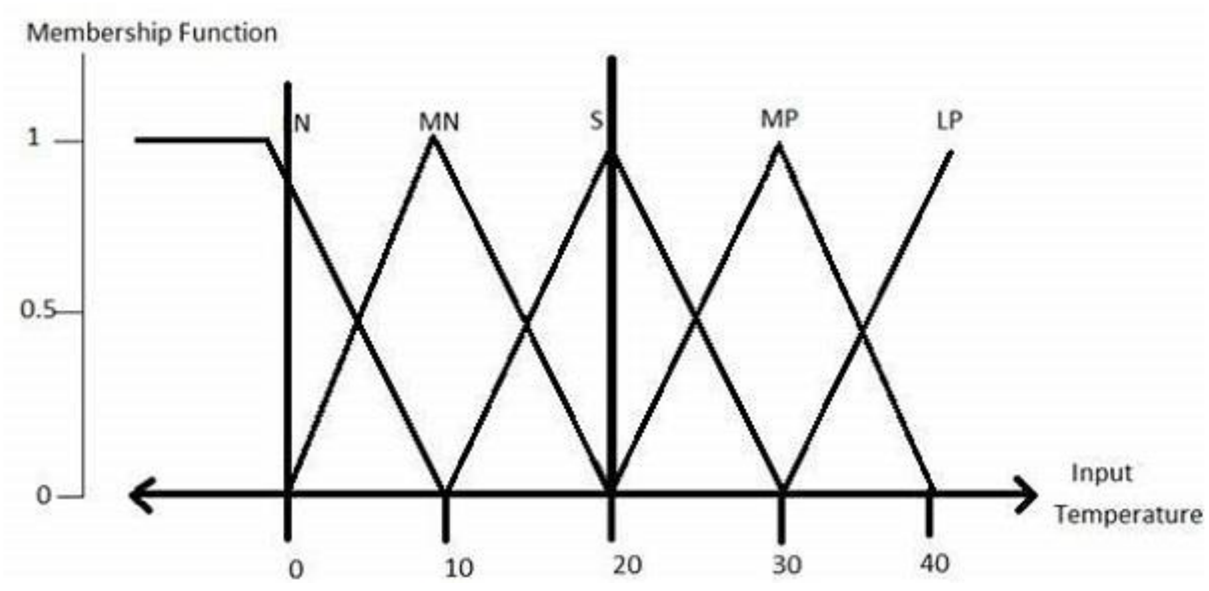
Linguistic variables are input and output variables in the form of simple words or sentences. For room temperature, cold, warm, hot, etc., are linguistic terms.

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

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.

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

Build a set of rules into the knowledge base in the form of IF-THEN-ELSE structures.

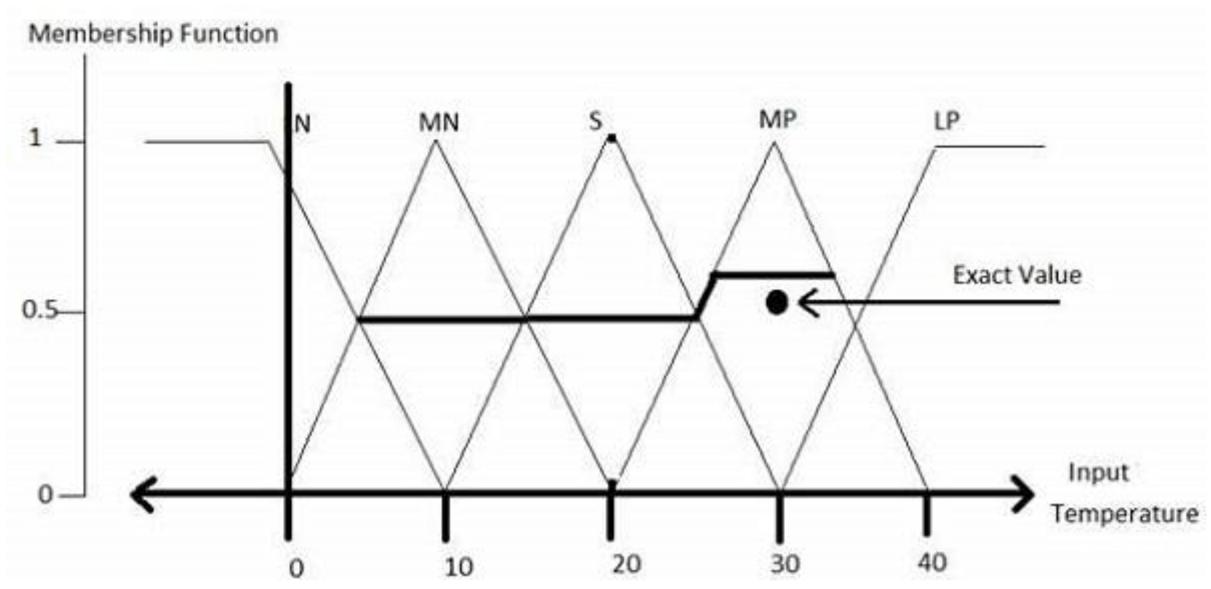
Sr. No.	**Condition**	**Action**
1	IF temperature=(Cold OR Very_Cold) AND target=Warm THEN	Heat
2	IF temperature=(Hot OR Very_Hot) AND target=Warm THEN	Cool
3	IF (temperature=Warm) AND (target=Warm) THEN	No_Change

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

The key application areas of fuzzy logic are as given −

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 of FLSs

- Mathematical concepts within fuzzy reasoning are very simple.
- You can modify a FLS by just adding or deleting rules due to flexibility of fuzzy logic.
- Fuzzy logic Systems can take imprecise, distorted, noisy input information.
- FLSs 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 of FLSs

- 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.

Naïve Bayes and Bayesian classification:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- **It is a probabilistic classifier, which means it predicts on the basis of the probability of an object**.
- Some popular examples of Naïve Bayes Algorithm are **spam filtration, Sentimental analysis, and classifying articles**.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(H \mid E) = \frac{P(E \mid H)}{P(E)} P(H).$$

Given a hypothesis *H* and evidence *E*, Bayes' theorem states that the relationship between the probability of the hypothesis before getting the evidence P(H and the probability of the hypothesis after getting the evidence $P(H|E)$

Boltzman Machine:

These are stochastic learning processes having recurrent structure and are the basis of the early optimization techniques used in ANN. Boltzmann Machine was invented by Geoffrey Hinton and Terry Sejnowski in 1985. More clarity can be observed in the words of Hinton on Boltzmann Machine.

“A surprising feature of this network is that it uses only locally available information. The change of weight depends only on the behavior of the two units it connects, even though the change optimizes a global measure” - Ackley, Hinton 1985.

Some important points about Boltzmann Machine –

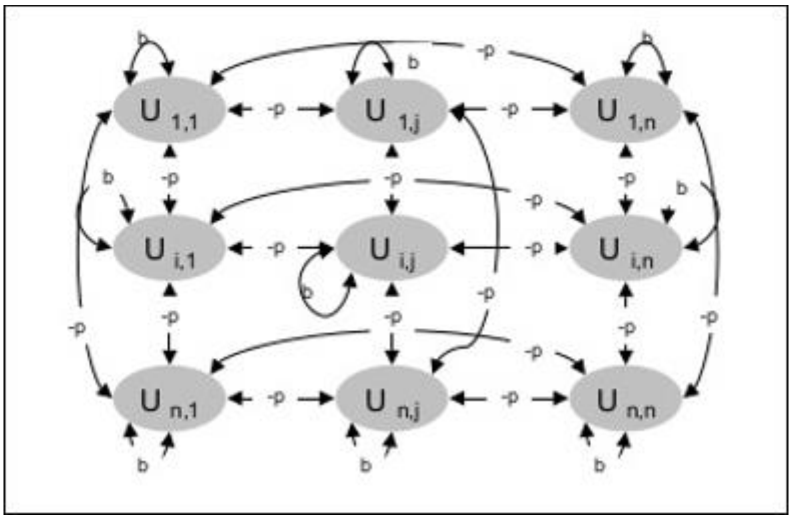
- They use recurrent structure.
- They consist of stochastic neurons, which have one of the two possible states, either 1 or 0.
- Some of the neurons in this are adaptive freestate and some are clamped frozenstate.
- If we apply simulated annealing on discrete Hopfield network, then it would become Boltzmann Machine.

Objective of Boltzmann Machine

The main purpose of Boltzmann Machine is to optimize the solution of a problem. It is the work of Boltzmann Machine to optimize the weights and quantity related to that particular problem.

Architecture

The following diagram shows the architecture of Boltzmann machine. It is clear from the diagram, that it is a two-dimensional array of units. Here, weights on interconnections between units are **–p** where **p** > **0**. The weights of self-connections are given by **b** where **b** > **0**.



Training Algorithm

As we know that Boltzmann machines have fixed weights, hence there will be no training algorithm as we do not need to update the weights in the network. However, to test the network we have to set the weights as well as to find the consensus function CFCF.

Boltzmann machine has a set of units $\mathbf{U_i}$ and $\mathbf{U_j}$ and has bi-directional connections on them.

- We are considering the fixed weight say $\mathbf{w_{ij}}$.
- $\mathbf{w_{ij} \neq 0}$ if $\mathbf{U_i}$ and $\mathbf{U_j}$ are connected.
- There also exists a symmetry in weighted interconnection, i.e. $\mathbf{w_{ij} = w_{ji}}$.
- $\mathbf{w_{ii}}$ also exists, i.e. there would be the self-connection between units.
- For any unit $\mathbf{U_i}$, its state $\mathbf{u_i}$ would be either 1 or 0.

The main objective of Boltzmann Machine is to maximize the Consensus Function CF which can be given by the following relation.

$$CF = \sum_{i} \sum_{j \leqslant i} w_{ij} u_i u_j$$

Now, when the state changes from either 1 to 0 or from 0 to 1, then the change in consensus can be given by the following relation –

$$\Delta CF = (1 - 2u_i)(w_{ij} + \sum_{j \neq i} u_i w_{ij})$$

Here **u_i** is the current state of **U_i**.

The variation in coefficient (**$1 - 2u_i$**) is given by the following relation –

$$(1 - 2u_i) = \begin{cases} +1, & U_i \; is \; currently \; off \\ -1, & U_i \; is \; currently \; on \end{cases}$$

Generally, unit **U_i** does not change its state, but if it does then the information would be residing local to the unit. With that change, there would also be an increase in the consensus of the network.

Probability of the network to accept the change in the state of the unit is given by the following relation –

$$AF(i,T) = \frac{1}{1 + exp[-\frac{\Delta CF(i)}{T}]}$$

Here, **T** is the controlling parameter. It will decrease as CF reaches the maximum value.

Testing Algorithm

Step 1 – Initialize the following to start the training –

- Weights representing the constraint of the problem
- Control Parameter T

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

Step 3 – Perform steps 4-7.

Step 4 – Assume that one of the state has changed the weight and choose the integer **I, J** as random values between **1** and **n**.

Step 5 – Calculate the change in consensus as follows –

$$\Delta CF = (1 - 2u_i)(w_{ij} + \sum_{j \neq i} u_i w_{ij})$$

Step 6 – Calculate the probability that this network would accept the change in state

$$AF(i,T) = \frac{1}{1 + exp[-\frac{\Delta CF(i)}{T}]}$$

Step 7 – Accept or reject this change as follows –

Case I – if **R < AF**, accept the change.

Case II − if **R** ≥ **AF**, reject the change.

Here, **R** is the random number between 0 and 1.

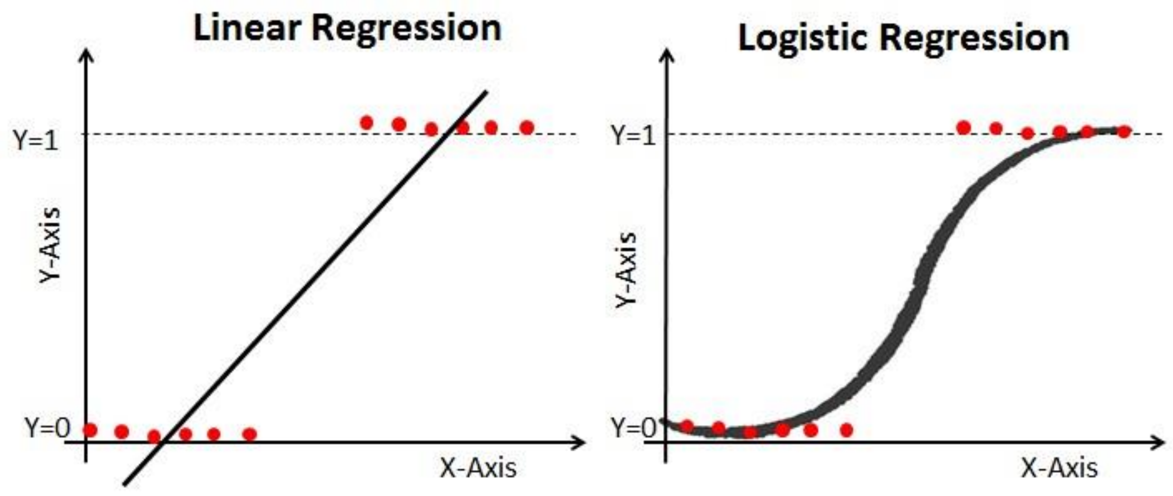
Step 8 − Reduce the control parameter temperature as follows –

$T_{new} = 0.95 T_{old}$

Step 9 − Test for the stopping conditions which may be as follows −

- Temperature reaches a specified value
- There is no change in state for a specified number of iterations

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Sometimes logistic regressions are difficult to interpret; the Intellects Statistics tool easily allows you to conduct the analysis, then in plain English interprets the output.



Clustering: **Clustering** is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait – often according to some defined distance measure.

Types of clustering:

i) Hierarchical clustering

a) Agglomerative(bottom up)

b) Divisive (Top down)

ii) Partitioning clustering

a) **K-mean**

b) Fuzzy c-mean clustering

c) QT clustering algorithm

K-mean algorithms:

- The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions,where $k < n$.
- It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.

- It assumes that the object attributes form a vector space.
- An algorithm for partitioning (or clustering) N data points into K disjoint subsets Sj containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left| x_n - \mu_j \right|^2 ,$$

where xn is a vector representing the the nth data point and uj is the geometric centroid of the data points in Sj.

- Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.
- K is positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid

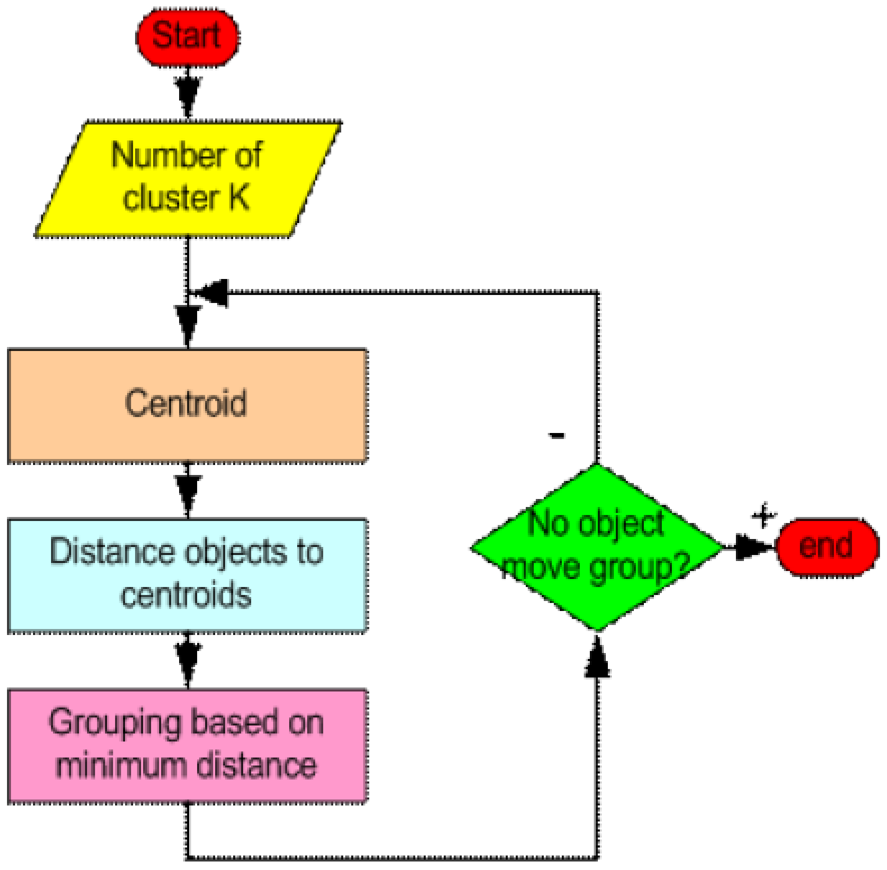

Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Figure flowchart of k-mean algorithm

<u>Support Vector Machine</u>:

Support vector machine is the new method for the classification of both linear and nonlinear data. SVM is an algorithm as follows

It use a nonlinear mapping to transform to the original training data into higher dimension. Within this new dimension, it searches for the linear optimal separating hyper planes(that is a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always separate by hyper planes. The SVM find the hyper planes using support vectors(essential training tuples) and margins(defined by the support vectors)

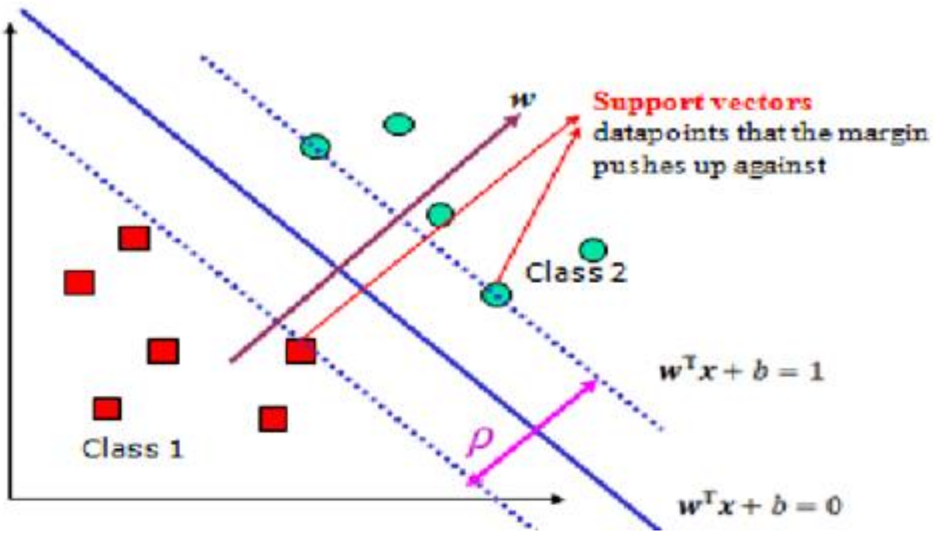
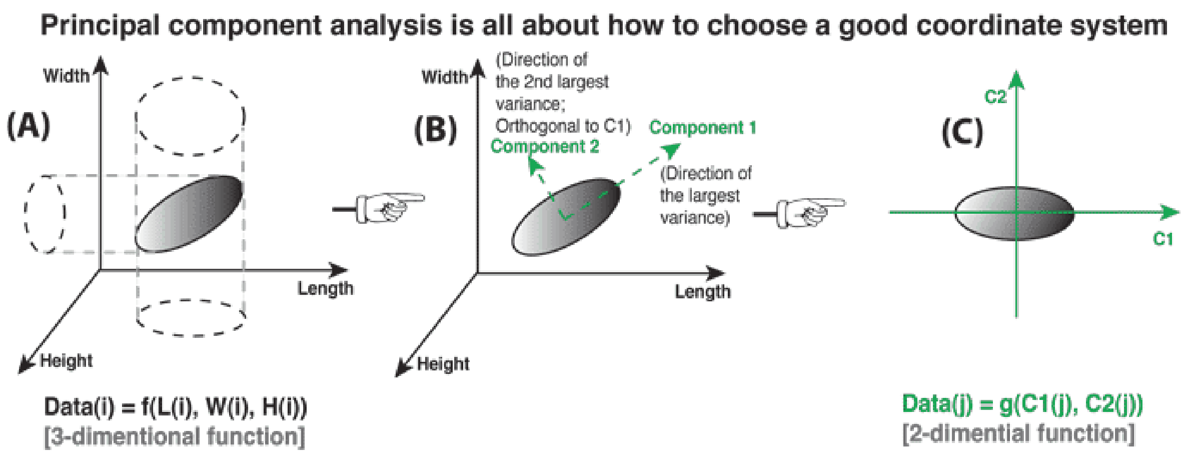


Figure. Support Vector Machine

Dimensionality Reduction (Principle Component Analysis)

- Given N data vectors from n-dimensions, find k ≤ n orthogonalvectors (principal components) that can be best used to represent data
- Steps
  - Normalize input data: Each attribute falls within the same range
  - Compute k orthonormal (unit) vectors, i.e., principal components
  - Each input data (vector) is a linear combination of the k principal component vectors
  - The principal components are sorted in order of decreasing "significance" or strength
  - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance. (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data
  - Works for numeric data only
  - Used when the number of dimensions is large



Principal component analysis is all about how to choose a good coordinate system
Width
(A)
Length
Height
Data(i) = f(L(i), W(i), H(i))
[3-dimentional function]
Width
(B)
(Direction of the 2nd largest variance; Orthogonal to C1)
Component 2
Component 1
(Direction of the largest variance)
Length
Height
(C)
C2
C1
Data(j) = g(C1(j), C2(j))
[2-dimential function]

Neural Network

A neuron is a cell in brain whose principle function is the collection, Processing, and dissemination of electrical signals. Brains Information processing capacity comes from networks of such neurons. Due to this reason some earliest AI work aimed to create such artificial networks. (Other Names are Connectionism; Parallel distributed processing and neural computing).

What is a Neural Network?

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.

Why use neural networks?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Other advantages include:

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 organization 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. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage

Neural networks versus conventional computers

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements(neurones) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Units of Neural Network

Nodes(units):
Nodes represent a cell of neural network.

Links:
Links are directed arrows that show propagation of information from one node to another node.

Activation:
Activations are inputs to or outputs from a unit.

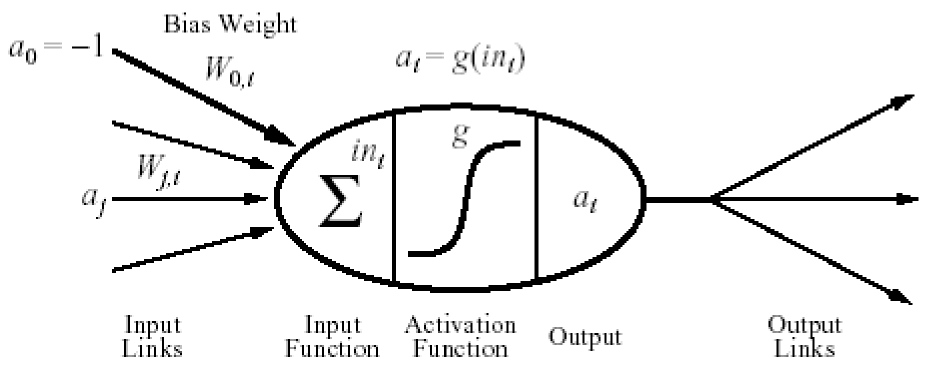
Wight:
Each link has weight associated with it which determines strength and sign of the connection.

Activation function:
A function which is used to derive output activation from the input activations to a given node is called activation function.

Bias Weight:
Bias weight is used to set the threshold for a unit. Unit is activated when the weighted sum of real inputs exceeds the bias weight.

Simple Model of Neural Network
A simple mathematical model of neuron is devised by McCulloch and Pit is given in the figure given below:

$$a_i \leftarrow g(in_i) = g\left(\Sigma_j W_{j,i} a_j\right)$$



It fires when a linear combination of its inputs exceeds some threshold.

A neural network is composed of nodes (units) connected by directed links A link from unit j to i serve to propagate the activation a_j from j to i. Each link has some numeric weight $W_{j,i}$ associated with it, which determines strength and sign of connection.

Each unit first computes a weighted sum of it's inputs:

$$in_i = \sum_{J=0}^{n} W_{j,i}\, a_j$$

Then it applies activation function g to this sum to derive the output:

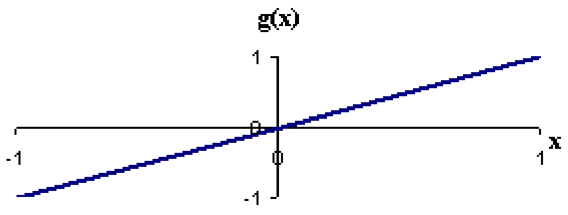
$$a_i = g(in_i) = g\left(\sum_{J=0}^{n} W_{j,i}\, a_j\right)$$

Here, a_j output activation from unit j and $W_{j,i}$ is the weight on the link j to this node. Activation function typically falls into one of three categories:

- Linear
- Threshold (*Heaviside function)*
- Sigmoid
- Sign

For **linear activation functions**, the output activity is proportional to the total weighted output.

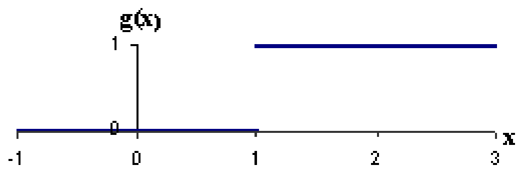
$g(x) = k\,x + c$, where k and x are constant



For **threshold activation functions**, the output are set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

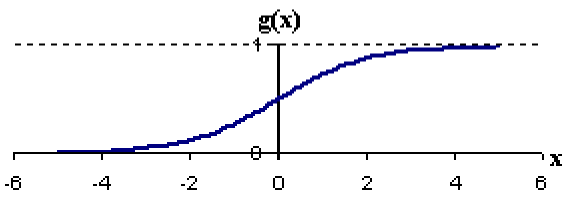
$g(x) = 1$ if $x >= k$

$= 0$ if $x < k$

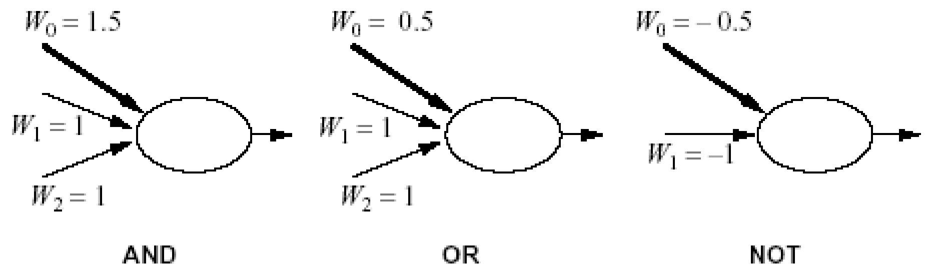


For **sigmoid activation functions**, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units. It has the advantage of differentiable.

$g(x) = 1/(1 + e^{-x})$



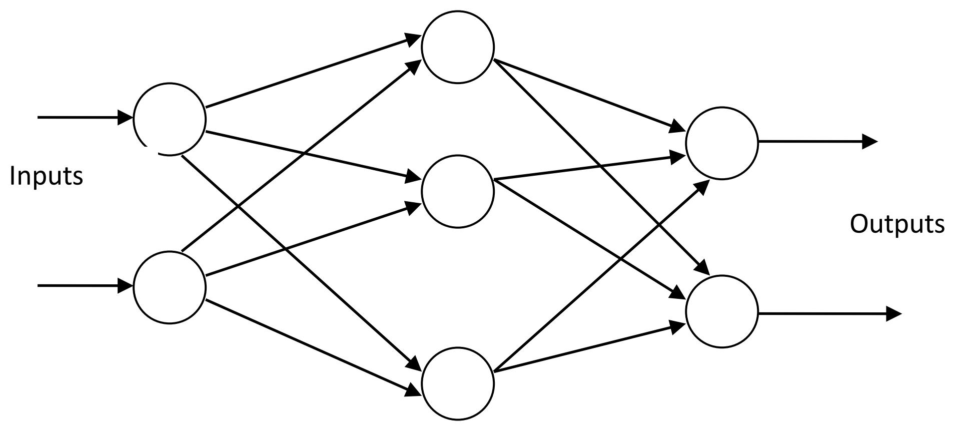
Realizing logic gates by using Neurons:



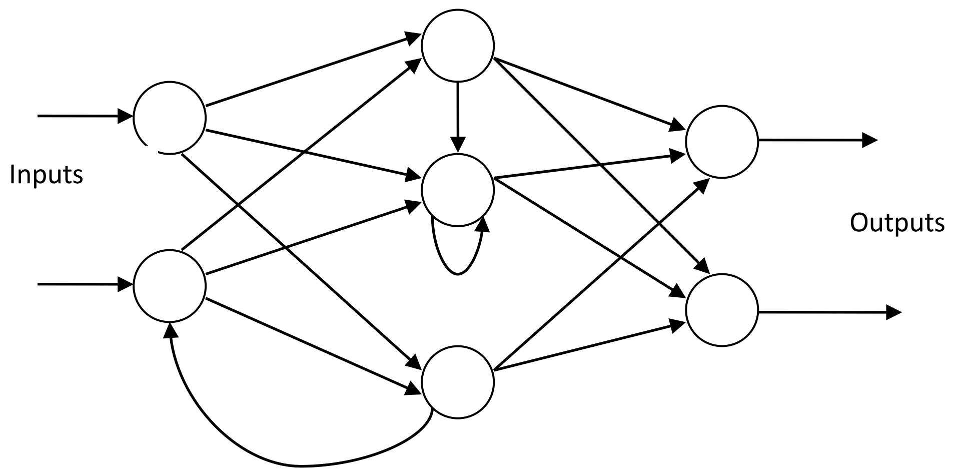
Network structures

Feed-forward networks:

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

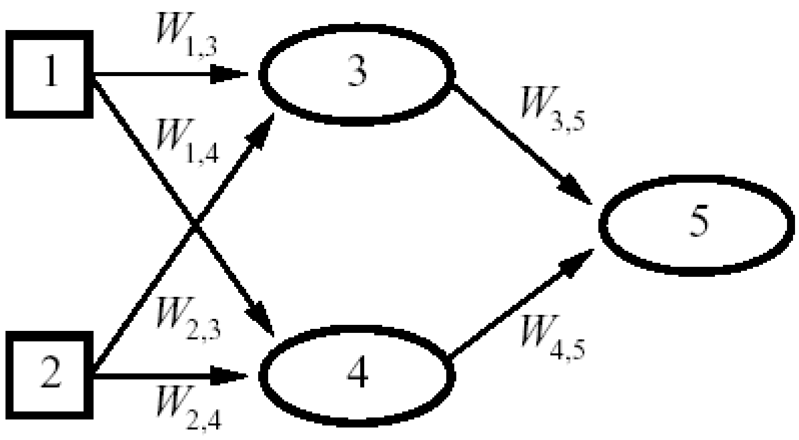


Feedback networks (Recurrent networks:)



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

Feed-forward example

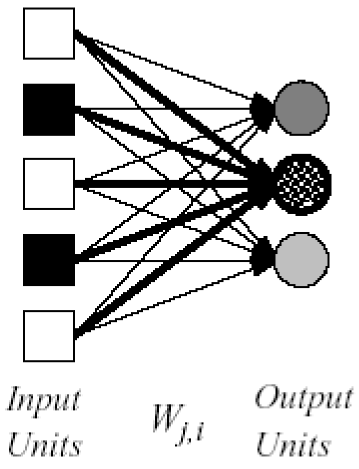


Here;
a5 = $g(W_{3;5}\ a_3 + W_{4;5}\ a_4)$
= $g(W_{3;5}\ g(W_{1;3}\ a_1 + W_{2;3}\ a_2) + W4;5\ g(W_{1;4}\ a_1 + W_{2;4}\ a_2)$

Types of Feed Forward Neural Network:

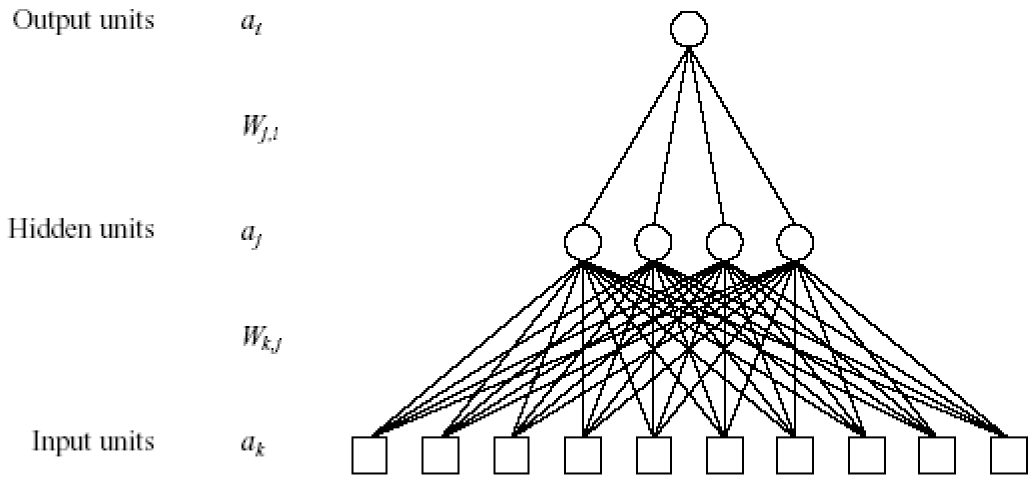
Single-layer neural networks (perceptrons)

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



Multilayer neural networks (perceptrons)

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



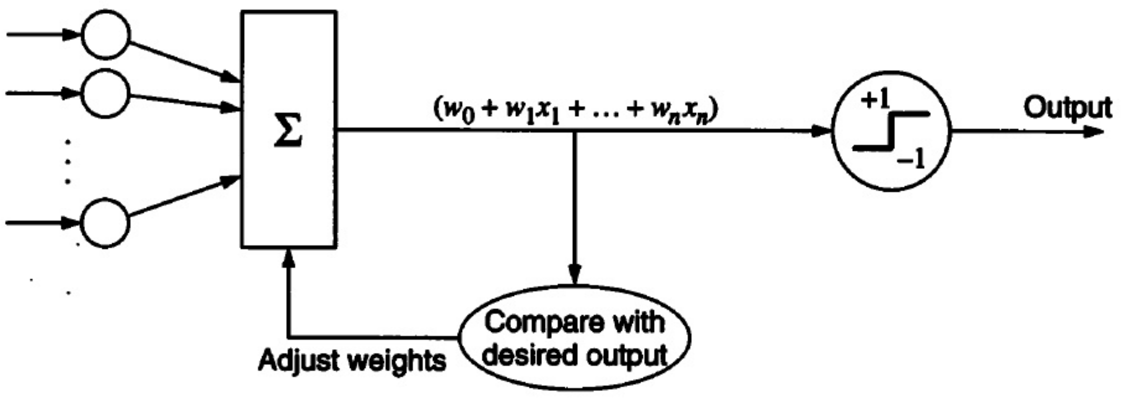
Once the number of layers, and number of units in each layer, has been selected, training is used to set the network's weights and thresholds so as to minimize the prediction error made by the network

Training is the process of adjusting weights and threshold to produce the desired result for different set of data.

Adaline Network

Adaline stands for Adaptive Linear Network. It is a simple perceptron-Like system that accomplishes classification by modifying weight in such a way as to diminish the MSE (Mean Square Error) at every iteration. This can be accomplished using gradient Adaptive Line element

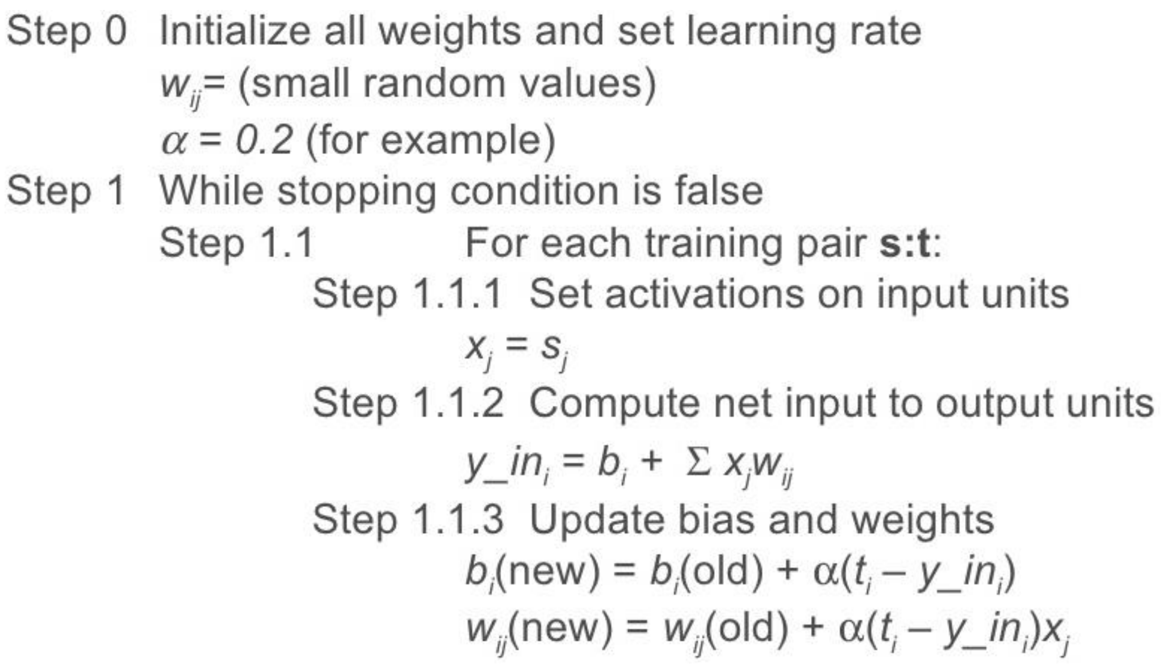
Architecture of Adaline Network:



$(w_0 + w_1x_1 + \ldots + w_nx_n)$
Σ
+1
−1
Output
Compare with desired output
Adjust weights

Figure Adaline Architecture

Algorithm for Adaline Architecture:

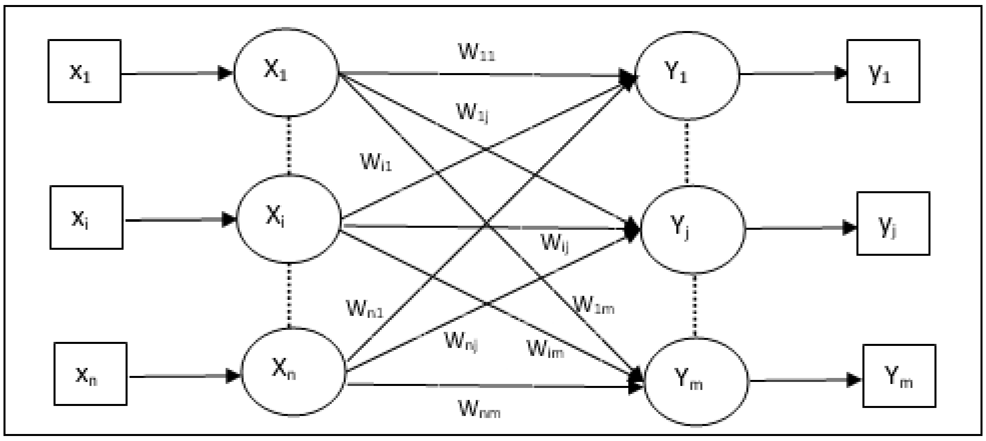


```
Step 0  Initialize all weights and set learning rate
        $w_{ij}$= (small random values)
        $\alpha = 0.2$ (for example)
Step 1  While stopping condition is false
        Step 1.1        For each training pair **s**:**t**:
            Step 1.1.1  Set activations on input units
                        $x_j = s_j$
            Step 1.1.2  Compute net input to output units
                        $y\_in_i = b_i + \Sigma x_j w_{ij}$
            Step 1.1.3  Update bias and weights
                        $b_i(\text{new}) = b_i(\text{old}) + \alpha(t_i - y\_in_i)$
                        $w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(t_i - y\_in_i)x_j$
```

Kohonen Network:

The objective of a Kohonen network is to map input vectors (patterns) of arbitrary dimension N onto a discrete map with 1 or 2 dimensions. Patterns close to one another in the input space should be close to one another in the map: they should be topologically ordered. A Kohonen network is composed of a grid of output units and N input units. The input pattern is fed to each output unit. The input lines to each output unit are weighted. These weights are initialised to small random number.

Simple Architecture of kohonen Network



Algorithm for training

Step 1 Initialize the weights, the learning rate **α** and the neighbourhood topological scheme.

Step 2 Continue step 3-9, when the stopping condition is not true.

Step 3 Continue step 4-6 for every input vector **x**.

Step 4 Calculate Square of Euclidean Distance for **j = 1 to m**

$$D(j) = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_i - w_{ij})^2$$

Step 5 Obtain the winning unit **J** where ***D***jj is minimum.

Step 6 − Calculate the new weight of the winning unit by the following relation −

$$w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)]$$

Step 7 Update the learning i7rate **α** by the following relation –

$$\alpha(t + 1) = 0.5\alpha t$$

Step 8 Reduce the radius of topological scheme.

Step 9 Check for the stopping condition for the network.

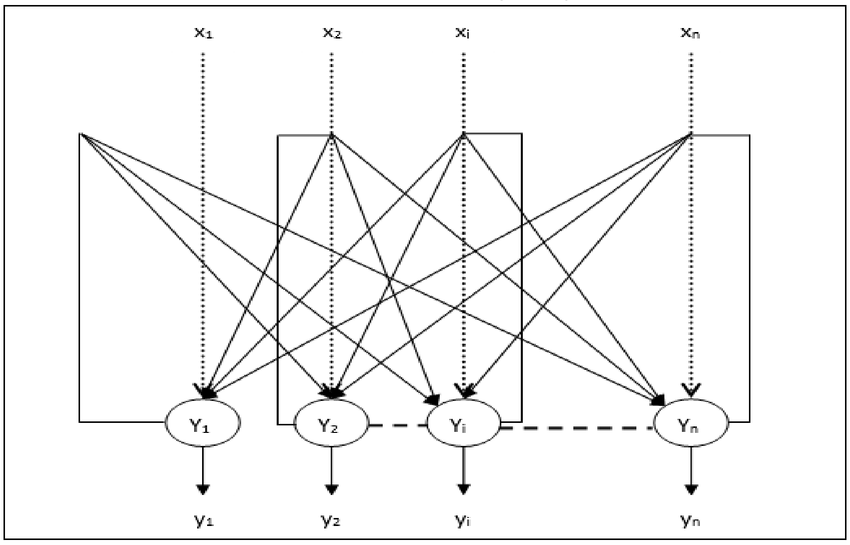
Hopfield network:

A Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be either binary 0,1 or bipolar +1,−1 in nature. The network has symmetrical weights with no self-connections i.e., $\mathbf{w_{ij}} = \mathbf{w_{ji}}$ and $\mathbf{w_{ii}} = \mathbf{0}$.

Following are some important points to keep in mind about discrete Hopfield network

- This model consists of neurons with one inverting and one non-inverting output

- The output of each neuron should be the input of other neurons but not the input of self.
- Weight/connection strength is represented by $\mathbf{w_{ij}}$.
- Connections can be excitatory as well as inhibitory. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory
- Weights should be symmetrical, i.e. $\mathbf{w_{ij} = w_{ji}}$



The output from $\mathbf{Y_1}$ going to $\mathbf{Y_2}$, $\mathbf{Y_i}$ and $\mathbf{Y_n}$ has the weights $\mathbf{w_{12}}$, $\mathbf{w_{1i}}$ and $\mathbf{w_{1n}}$ respectively. Similarly, other arcs have the weights on them.

Training Algorithm:

During training of discrete Hopfield network, weights will be updated. As we know that we can have the **binary input vectors** as well as **bipolar input vectors**. Hence, in **both the cases**, weight updates can be done with the following relation.

Case 1 − Binary input patterns

For a set of binary patterns **s** p **, p = 1 to P**

Here, **s** p **= s₁** p **, s₂** p **,...., sᵢ** p **,...., sₙ** p

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^{P}[2s_i(p) - 1][2s_j(p) - 1] \quad for\, i \neq j$$

Case 2 − Bipolar input patterns

For a set of binary patterns **s** p **, p = 1 to P**

Here, **s** p **= s$_1$** p **, s$_2$** p **,...., s$_i$** p **,...., s$_n$** p

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^{P}[s_i(p)][s_j(p)] \quad for\, i \neq j$$

Testing Algorithm

Step 1 − Initialize the weights, which are obtained from training algorithm by using Hebbian principle.

Step 2 − Perform steps 3-9, if the activations of the network is not consolidated.

Step 3 − For each input vector **X**, perform steps 4-8.

Step 4 − Make initial activation of the network equal to the external input vector **X** as follows –

$$y_i = x_i \quad for\, i = 1\, to\, n$$

Step 5 − For each unit **Y$_i$**, perform steps 6-9.

Step 6 − Calculate the net input of the network as follows −

$$y_{ini} = x_i + \sum_j y_j w_{ji}$$

Step 7 − Apply the activation as follows over the net input to calculate the output –

$$y_i = \begin{cases} 1 & if\, y_{ini} > \theta_i \\ y_i & if\, y_{ini} = \theta_i \\ 0 & if\, y_{ini} < \theta_i \end{cases}$$

Here θi is the threshold.

Step 8 − Broadcast this output **y$_i$** to all other units.

Step 9 − Test the network for conjunction.

Further for checking the energy function for discrete Hopfield network is characterised as

$$E_f = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} \theta_i y_i$$

Backpropogation algorithm:

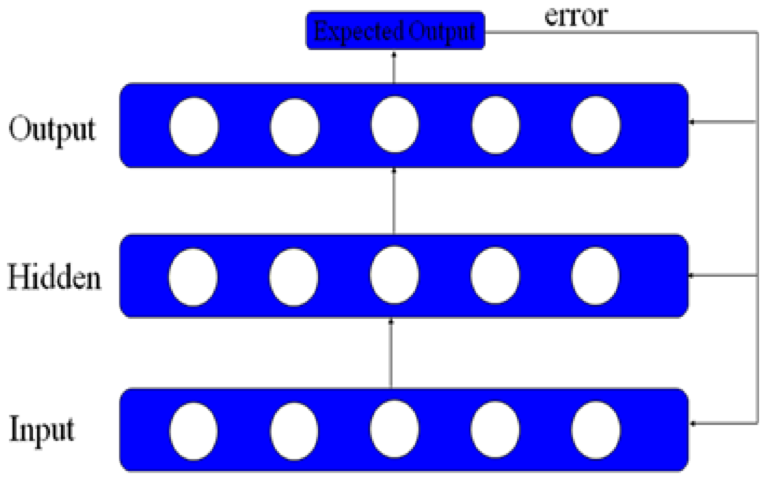
Backpropogation networks are necessarily multilayer perceptron's (usually with one input, one hidden, and one output layer). In order for the hidden layer to serve any useful function, multilayer networks must have non-linear activation functions for the multiple layers.

Characteristics:

- A multi-layered perceptron has three distinctive characteristics
  - The network contains one or more layers of hidden neurons
  - The network exhibits a high degree of connectivity
  - Each neuron has a smooth (differentiable everywhere) nonlinear activation function, the most common is the sigmoidal nonlinearity:

$$y_j = \frac{1}{1 + e^{s_j}}$$

- The Backpropogation algorithm provides a computational efficient method for training multi-layer networks.



Algorithm:

Step 0: Initialize the weights to small random values

Step 1: Feed the training sample through the network and determine the final output

Step 2: Compute the error for each output unit, for unit k it is:

$$\delta_k = (\underset{\text{Required output}}{t_k} - \overset{\text{Actual output}}{y_k})\underset{\text{Derivative of f}}{f'}(y\_in_k)$$

Step 3: Calculate the weight correction term for each output unit, for unit k it is:

$$\Delta w_{jk} = \underset{\text{A small constant}}{\alpha}\delta_k \overset{\text{Hidden layer signal}}{z_j}$$

Step 4: Propagate the delta terms (errors) back through the weights of the hidden units where the delta input for the jth hidden unit is: $\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$

The delta term for jth hidden unit is:

$$\delta_j = \delta\_in_j f'(z\_in_j)$$

Step 5: Calculate the weight correction term for the hidden units:

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

Step 6: Update the weights:

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

Step 7: Test for stopping (maximum cylces, small changes, etc)

Natural Language Processing:

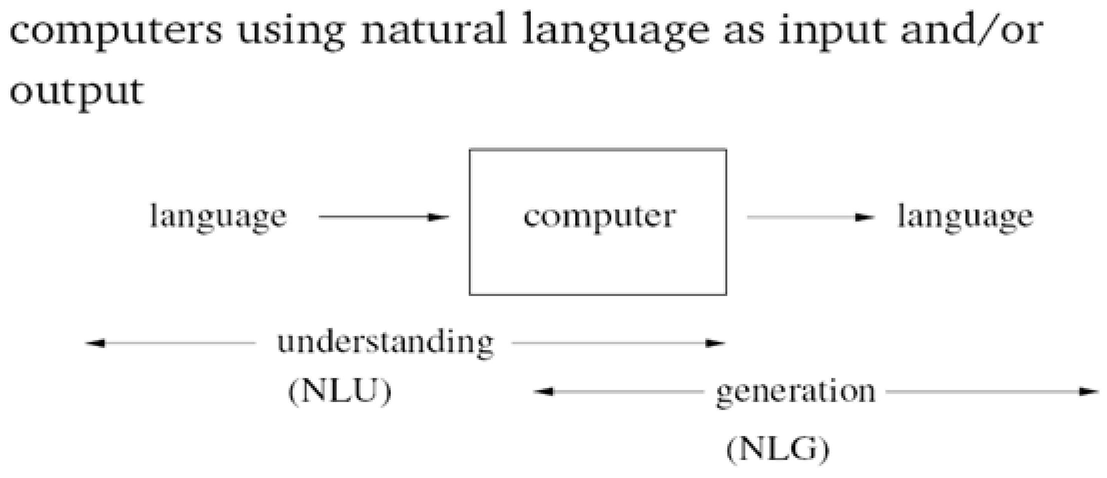
Perception and communication are essential components of intelligent behavior. They provide the ability to effectively interact with our environment. Humans perceive and communicate through their five basic senses of sight, hearing, touch, smell and taste, and their ability to generate meaningful utterances. Developing programs that understand a natural language is a difficult problem. Natural languages are large. They contain infinity of different sentences. No matter how many sentences a person has heard or seen, new ones can always be produced. Also, there is much ambiguity in a natural language. Many words have several meanings and sentences can have different meanings in different contexts. This makes the creation of programs that understand a natural language, one of the most challenging tasks in AI.

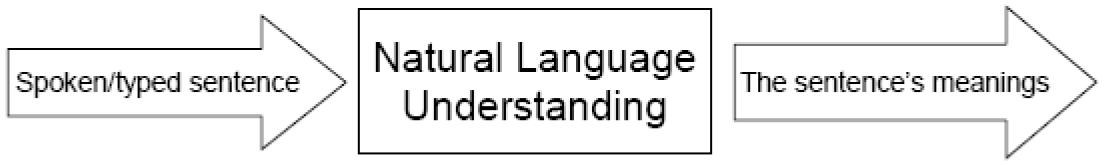
Developing programs to understand natural language is important in AI because a natural form of communication with systems is essential for user acceptance. AI programs must be able to communicate with their human counterparts in a natural way, and natural language is one of the most important mediums for that purpose. So, Natural Language Processing (NLP) is the field that deals with the computer processing of natural languages, mainly evolved by people working in the field of Artificial Intelligence.

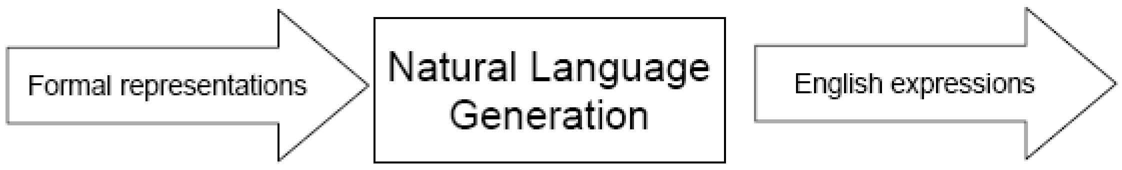
Natural Language Processing (NLP), is the attempt to extract the fuller meaning representation from the free text. Natural language processing is a technology which involves converting spoken or written human language into a form which can be processed by computers, and vice versa. Some of the better-known applications of NLP include:

- **Voice recognition software** which translates speech into input for word processors or other applications;
- **Text-to-speech synthesizers** which read text aloud for users such as the hearing-impaired;
- **Grammar and style checkers** which analyze text in an attempt to highlight errors of grammar or usage;

- **Machine translation systems** which automatically render a document such as a web page in another language.







Natural Language Generation:

"Natural Language Generation (NLG), also referred to as text generation, is a subfield of natural language processing (NLP; which includes computational linguistics)

Natural Language Generation (NLG) is the natural language processing task of generating natural language from a machine representation system such as a knowledge base or a logical form.

In a sense, one can say that an NLG system is like a translator that converts a computer based representation into a natural language representation. However, the methods to produce the final language are very different from those of a compiler due to the inherent expressivity of natural languages.

NLG may be viewed as the opposite of natural language understanding. The difference can be put this way: whereas in natural language understanding the system needs to disambiguate the input sentence to produce the machine representation language, in NLG the system needs to make decisions about how to put a concept into words.

The different types of generation techniques can be classified into four main categories:

- Scanned text systems constitute the simplest approach for single-sentence and multi-sentence text generation. They are trivial to create, but very inflexible.
- Template systems, the next level of sophistication, rely on the application of pre-defined templates or schemas and are able to support flexible alterations. The template approach is used mainly for multi-sentence generation, particularly in applications whose texts are fairly regular in structure.
- Phrase-based systems employ what can be seen as generalized templates. In such systems, a phrasal pattern is first selected to match the top level of the input, and then each part of the pattern is recursively expanded into a more specific phrasal pattern that matches some subportion of the input. At the sentence level, the phrases resemble phrase structure grammar rules and at the discourse level they play the role of text plans.
- Feature-based systems, which are as yet restricted to single-sentence generation, represent each possible minimal alternative of expression by a single feature. Accordingly, each sentence is specified by a unique set of features. In this framework, generation consists in the incremental collection of features appropriate for each portion of the input. Feature collection itself can either be based on unification or on the traversal of a feature selection network. The expressive power of the approach is very high since any distinction in language can be added to the system as a feature. Sophisticated feature-based generators, however, require very complex input and make it difficult to maintain feature interrelationships and control feature selection.

Many natural language generation systems follow a hybrid approach by combining components that utilize different techniques.

Natural Language Understanding:

Developing programs that understand a natural language is a difficult problem. Natural languages are large. They contain infinity of different sentences. No matter how many sentences a person has heard or seen, new ones can always be produced. Also, there is much ambiguity in a natural language. Many words have several meaning such as can, bear, fly, bank etc, and sentences have different meanings in different contexts.

Example :- a *can* of juice. I *can* do it.

This makes the creation of programs that understand a natural language, one of the most challenging tasks in AI. Understanding the language is not only the transmission of words. It also requires inference about the speakers' goal, knowledge as well as the context of the interaction. We say a program understand natural language if it behaves by taking the correct or acceptable action in response to the input. A word functions in a sentence as a part of speech. Parts of the speech for the English language are nouns, pronouns, verbs, adjectives, adverbs, prepositions, conjunctions and interjections. Three major issues involved in understanding language.

- A large amount of human knowledge is assumed.
- Language is pattern based, phonemes are components of the words and words make phrases and sentences. Phonemes, words and sentences order are not random.
- Language acts are the product of agents (human or machine).

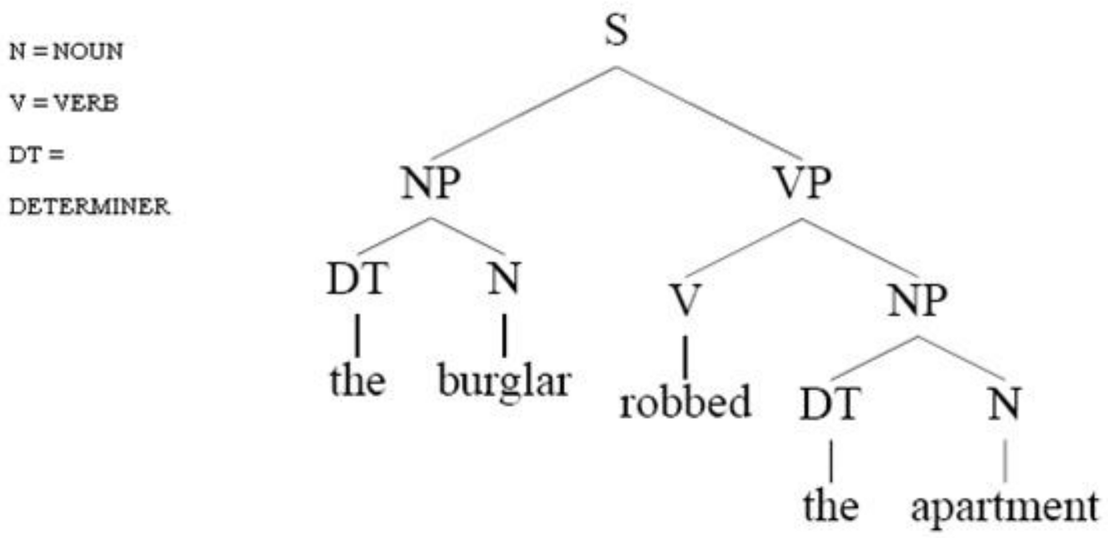
Levels of knowledge used in Language Understanding

A language understanding knowledge must have considerable knowledge about the structures of the language including what the words are and how they combine into phrases and sentences. It must also know the meanings of the words and how they contribute to the meanings of the sentence and to the context within which they are being used. The component forms of knowledge needed for an understanding of natural languages are sometimes classified according to the following levels.

- **Phonological**
  - Relates sound to the words we recognize. A phoneme is the smallest unit of the sound. Phones are aggregated to the words.
- **Morphological**
  - This is lexical knowledge which relates to the word construction from basic units called morphemes. A morpheme is the smallest unit of meaning. Eg:- *friend* + ly = friendly
- **Syntactic**
  - This knowledge relates to how words are put together or structure red together to form grammatically correct sentences in the language.
- **Semantic**
  - This knowledge is concerned with the meanings of words and phrases and how they combine to form sentence meaning.
- **Pragmatic**
  - This is high – level knowledge which relates to the use of sentences in different contexts and how the context affects the meaning of the sentence.
- **World**
  - Includes the knowledge of the physical world, the world of human social interaction, and the roles of goals and intentions in communication.

<u>Basic Parsing Techniques</u>

Before the meaning of a sentence can be determined, the meanings of its constituent parts must be established. This requires knowledge of the structure of the sentence, the meaning of the individual words and how the words modify each other. The process of determining the syntactical structure of a sentence is known as parsing. Parsing is the process of analyzing a sentence by taking it apart word – by – word and determining its structure from its constituent parts and sub parts. The structure of a sentence can be represented with a syntactic tree. When given an input string, the lexical parts or terms (root words), must first be identified by type and then the role they play in a sentence must be determined. These parts can be combined successively into larger units until a complete tree has been computed.



Noun Phrases (NP): “the burglar”, “the apartment”

Verb Phrases (VP): “robbed the apartment”

Sentences (S): “the burglar robbed the apartment”

To determine the meaning of a word, a parser must have access to a lexicon. When the parser selects the word from the input stream, it locates the world in the lexicon and obtains the word’s possible functions and features, including the semantic information.

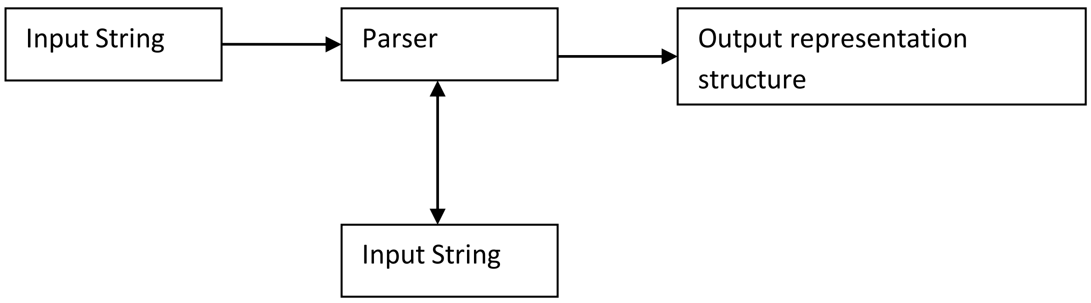


Figure :- Parsing an input to create an output structure

<u>Lexeme (Lexicon) & word forms:</u>

The distinction between these two senses of "word" is arguably the most important one in morphology. The first sense of "word", the one in which *dog* and *dogs* are "the same word", is called a lexeme. The second sense is called *word form*. We thus say that *dog* and *dogs* are different forms of the same lexeme. *Dog* and *dog catcher*, on the other hand, are different lexemes, as they refer to two different kinds of entities. The form of a word that is chosen conventionally to represent the canonical form of a word is called a lemma, or citation form.

A lexicon defines the words of a language that a system knows about. This is includes common words and words that are specific to the domain of the application. Entries include meanings for each word and its syntactic and morphological behavior.

<u>Morphology:</u>

Morphology is the identification, analysis and description of the structure of words (words as units in the lexicon are the subject matter of lexicology). While words are generally accepted as being (with clitics) the smallest units of syntax, it is clear that in most (if not all) languages, words can be related to other words by rules. For example, English speakers recognize that the words *dog*, *dogs*, and *dog catcher* are closely related. English speakers recognize these relations from their tacit knowledge of the rules of word formation in English. They infer intuitively that *dog* is to *dogs* as *cat* is to *cats*; similarly, *dog* is to *dog catcher* as *dish* is to *dishwasher* (in one sense). The rules understood by the speaker reflect specific patterns (or regularities) in the way words are formed from smaller units and how those smaller units interact in speech. In this way, morphology is the branch of linguistics that studies patterns of word formation within and across languages, and attempts to formulate rules that model the knowledge of the speakers of those languages.

Morphological analysis is the process of recognizing the suffixes and prefixes that have been attached to a word.

We do this by having a table of affixes and trying to match the input as:

prefixes +root + suffixes.

- For example: adjective + ly -> adverb. E.g.: [Friend + ly]=friendly
- We may not get a unique result.

    - "-s, -es" can be either a plural noun or a 3ps verb
    - "-d, -ed" can be either a past tense or a perfect participle

Morphological Information:

- Transform part of speech
    - *green, greenness (adjective, noun)*
    - *walk, walker (verb, noun)*
- Change features of nouns
    - *boat, boats (singular, plural)*
- Bill slept , Bill's bed
    - (subjective case, possessive case)
- Change features of verbs
    - Aspect
        - *I walk. I am walking.* (present, progressive)
    - Tense
        - *I walked. I will walk. I had been walking. (past, future, past progressive)*
    - Number and person
        - *I walk. They walk. (first person singular, third person plural)*

<u>Syntactic Analysis:</u>

Syntactic analysis takes an input sentence and produces a representation of its grammatical structure. A grammar describes the valid parts of speech of a language and how to combine them into phrases. The grammar of English is nearly context free.

A computer grammar specifies which sentences are in a language and their parse trees. A parse tree is a hierarchical structure that shows how the grammar applies to the input. Each level of the tree corresponds to the application of one grammar rule.

It is the starting point for working out the meaning of the whole sentence. Consider the following two sentences.

1. "The dog ate the bone."
2. "The bone was eaten by the dog."

Understanding the structure (via the syntax rules) of the sentences help us work out that it's the bone that gets eaten and not the dog. Syntactic analysis determines possible grouping of words in a sentence. In other cases there may be many possible groupings of words. Consider the sentence "John saw Mary with a telescope". Two different readings based on the groupings.

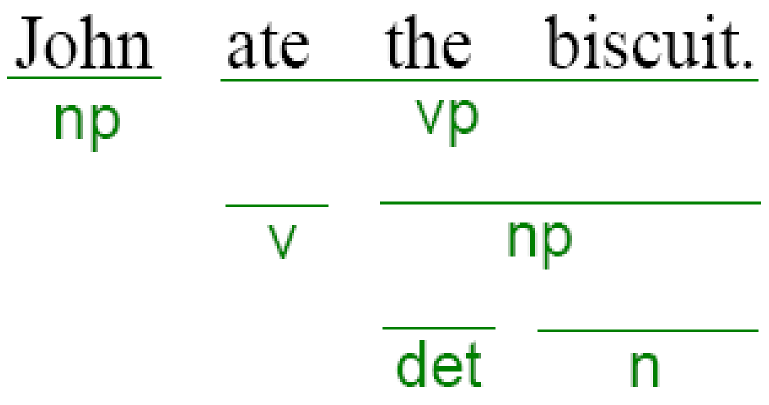
1. John saw (Mary with a telescope).
2. John (saw Mary with a telescope).

A sentence is syntactically ambiguous if there are two or more possible groupings. Syntactic analysis helps determining the meaning of a sentence by working out possible word structure. Rules of syntax are specified by writing a *grammar* for the language. A parser will check if a sentence is correct according to the grammar. It returns a representation (parse tree) of the sentence's structure. A grammar specifies allowable sentence structures in terms of basic categories such as noun and verbs. A given grammar, however, is unlikely to cover all possible grammatical sentences. Parsing sentences is to help determining their meanings, not just to check that they are correct. Suppose we want a grammar that recognizes sentences like the following.

John ate the biscuit.
The lion ate the zebra.
The lion kissed John

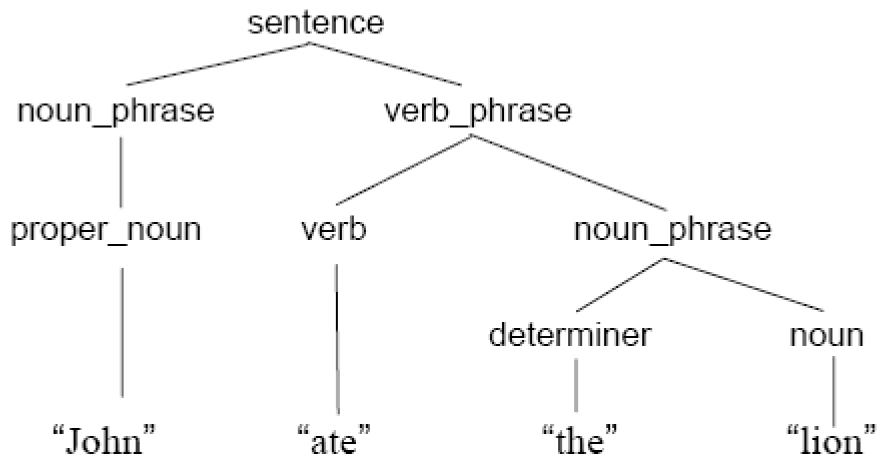
But reject incorrect sentences such as
Ate John biscuit the.
Zebra the lion the ate.
Biscuit lion kissed.

A simple grammar that deals with this is given below
sentence --> noun_phase, verb phrase.
noun_phrase --> proper_noun.
noun_phrase --> determiner, noun.
verb_phrase --> verb, noun_phrase.
proper_noun --> [mary].
proper_noun --> [john].
noun --> [zebra].
noun --> [biscuit].
verb --> [ate].
verb --> [kissed].
determiner --> [the].



Incorrect sentences like “biscuit lion kissed” will be excluded by the grammar.

- A *parse trees* illustrates the syntactic structure of the sentence.



<u>Semantic Analysis:</u>

Semantic analysis is a process of converting the syntactic representations into a meaning representation.

This involves the following tasks:

- Word sense determination
- Sentence level analysis
- Knowledge representation

- Word sense

Words have different meanings in different contexts.

Mary had a bat in her office.

- bat = `a baseball thing’
- bat = `a flying mammal’

- Sentence level analysis

Once the words are understood, the sentence must be assigned some meaning

I saw an astronomer with a telescope.

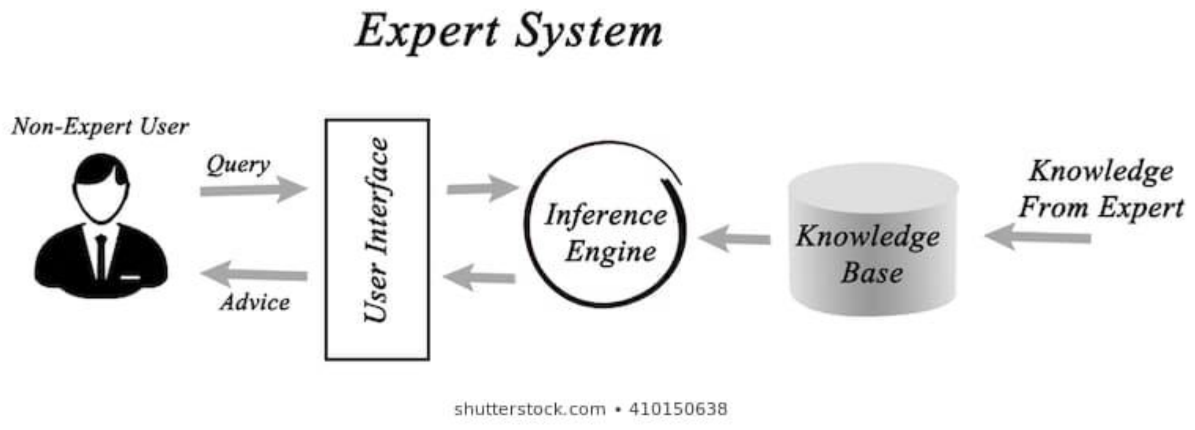
Expert Systems:

An Expert system is a set of program that manipulates encoded knowledge to solve problem in a specialized domain that normally requires human expertise. A computer system that simulates the decision- making process of a human expert in a specific domain. An expert system’s knowledge is obtained from expert sources and coded in a form suitable for the system to use in its inference or reasoning processes. The expert knowledge must be obtained from specialists or other sources of expertise, such as texts, journals, articles and data bases.

An expert system is an “intelligent” program that solves problems in a narrow problem area by using high-quality, specific knowledge rather than an algorithm.

Block Diagram

There is currently no such thing as “standard” expert system. Because a variety of techniques are used to create expert systems, they differ as widely as the programmers who develop them and the problems they are designed to solve. However, the principal components of most expert systems are knowledge base, an inference engine, and a user interface, as illustrated in the figure.



1. **Knowledge Base** :The component of an expert system that contains the system's knowledge is called its knowledge base. This element of the system is so critical to the way most expert systems are constructed that they are also popularly known as knowledge-based systems A knowledge base contains both declarative knowledge (facts about objects, events and situations) and procedural knowledge (information about courses of action). Depending on the form of knowledge representation chosen, the two types of knowledge may be separate or integrated. Although many knowledge representation techniques have been used in expert systems, the most prevalent form of knowledge representation currently used in expert systems is the rule-based production system approach. To improve the performance of an expert system, we should supply the system with some knowledge about the knowledge it possess, or in other words, metaknowledge.

2. **Inference Engine**

Simply having access to a great deal of knowledge does not make you an expert; you also must know how and when to apply the appropriate knowledge. Similarly, just having a knowledge base does not make an expert system intelligent. The system must have another component that directs the implementation of the knowledge. That element of the system is known variously as the control structure, the rule interpreter, or the inference engine. The inference engine decides which heuristic search techniques are used to determine how the rules in the knowledge base are to be applied to the problem. In effect, an inference engine "runs" an expert system, determining which rules are to be invoked, accessing the appropriate rules in the knowledge base, executing the rules , and determining when an acceptable solution has been found.

3. User Interface

The component of an expert system that communicates with the user is known as the user interface. The communication performed by a user interface is bidirectional. At the simplest level, we must be able to describe our problem to the expert system, and the system must be able to respond with its recommendations. We may want to ask the system to explain its "reasoning", or the system may request additional information about the problem from us.

Beside these three components, there is a Working Memory - a data structure which stores information about a specific run. It holds current facts and knowledge.

Stages of Expert Development System.

Although great strides have been made in expediting the process of developing an expert system, it often remains an extremely time consuming task. It may be possible for one or two people to develop a small expert system in a few months; however the development of a sophisticated system may require a team of several people working together for more than a year. An expert system typically is developed and refined over a period of several years. We can divide the process of expert system development into five distinct stages. In practice, it may not be possible to break down the expert system development cycle precisely. However, an examination of these five stages may serve to provide us with some insight into the ways in which expert systems are developed.

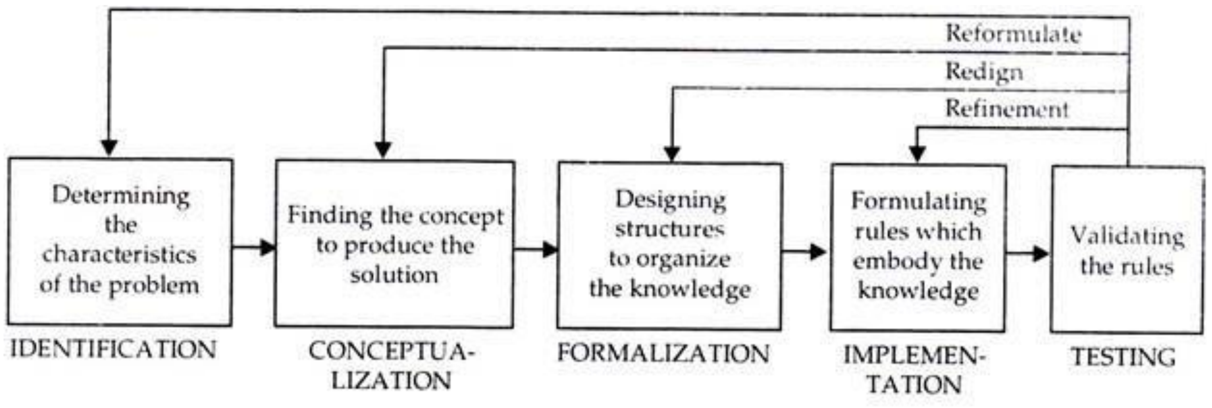

Reformulate
Redign
Refinement
Determining the characteristics of the problem
Finding the concept to produce the solution
Designing structures to organize the knowledge
Formulating rules which embody the knowledge
Validating the rules
IDENTIFICATION
CONCEPTUA-LIZATION
FORMALIZATION
IMPLEMEN-TATION
TESTING

Fig. 12.8. *The five stages of expert system development.*

Identification:

Beside we can begin to develop an expert system, it is important that we describe, with as much precision as possible, the problem that the system is intended to solve. It is not enough simply to feel that the system would be helpful in certain situation; we must determine the exact nature of the problem and state the precise goals that indicate exactly how we expect the expert system to contribute to the solution.

Conceptualization:

Once we have formally identified the problem that an expert system is to solve, the next stage involves analysing the problem further to ensure that its specifics, as well as it generalities, are understood. In the conceptualization stage the knowledge engineer frequently creates a diagram of the problem to depict graphically the relationships between the objects and processes in the problem domain. It is often helpful at this stage to divide the problem into a series of sub-problems and to diagram both the relationships among the pieces of each sub-problem and the relationships among the various sub-problems.

Formalization:

In the preceding stages, no effort has been made to relate the domain problem to the artificial intelligence technology that may solve it. During the identification and the conceptualization stages, the focus is entirely on understanding the problem. Now, during the formalization stage, the problem is connected to its proposed solution, an expert system, by analyzing the relationships depicted in the conceptualization stage. During formalization, it is important that the knowledge engineer be familiar with the following:

• The various techniques of knowledge representation and heuristic search used in expert systems.

• The expert system “tools” that can greatly expedite the development process. And

• Other expert systems that may solve similar problems and thus may be adequate to the problem at hand.

Implementation:

During the implementation stage, the formalized concepts are programmed onto the computer that has been chosen for system development, using the predetermined techniques and tools to implement a “first pass” prototype of the expert system. Theoretically, if the methods of the previous stage have been followed with diligence and care, the implementation of the prototype should be as much an art as it is a science, because following all rules does not guarantee that the system will work the first time it is implemented. Many scientists actually consider the first prototype to be a “throw-away’ system, useful for evaluating progress but hardly a usable expert system.

Testing:

Testing provides opportunities to identify the weakness in the structure and implementation of the system and to make the appropriate corrections. Depending on the types of problems encountered, the testing procedure may indicate that the system was.