SOFT COMPUTING

SEMESTER -7TH

BRANCH – EE & EEE

MODULE-I

WHAT IS INTELLIGENCE?

Artificial intelligence is a property of machines which gives it ability to mimic the human thought process. The intelligent machines are developed based on the intelligence of a subject, of a designer, of a person, of a human being.

WHAT IS AI?

Artificial Intelligence is concerned with the design of intelligence in an artificial device. The term was coined by McCarthy in 1956.

There are two ideas in the definition.

1. Intelligence
2. Artificial device

WHAT IS INTELLIGENCE?

- Is it that which characterize humans? Or is there an absolute standard of judgement?
- Accordingly there are two possibilities:
- A system with intelligence is expected to behave as intelligently as a human
- A system with intelligence is expected to behave in the best possible manner
- Secondly what type of behaviour are we talking about?
- Are we looking at the thought process or reasoning ability of the system?
- Or are we only interested in the final manifestations of the system in terms of its actions?

TYPICAL AI PROBLEMS

While studying the typical range of tasks that we might expect an “intelligent entity” to perform, we need to consider both “common-place” tasks as well as expert tasks.
Examples of common-place tasks include

- Recognizing people, objects.
- Communicating (through *natural language*).
- *Navigating* around obstacles on the streets

These tasks are done matter of fact fully and routinely by people and some other animals. Expert tasks include:

- Medical diagnosis.
- Mathematical problem solving
- Playing games like chess

These tasks cannot be done by all people, and can only be performed by skilled specialists.

Now, which of these tasks are easy and which ones are hard? Clearly tasks of the first type are easy for humans to perform, and almost all are able to master them. However, when we look at what computer systems have been able to achieve to date, we see that their achievements include performing sophisticated tasks like medical diagnosis, performing symbolic integration, proving theorems and playing chess.

INTELLIGENT BEHAVIOUR

This discussion brings us back to the question of what constitutes intelligent behaviour. Some of these tasks and applications are:

1. Perception involving image recognition and computer vision
2. Reasoning
3. Learning
4. Understanding language involving natural language processing, speech processing
5. Solving problems
6. Robotics

PRACTICAL APPLICATIONS OF AI

AI components are embedded in numerous devices e.g. in copy machines for automatic correction of operation for copy quality improvement. AI systems are in everyday use for identifying credit card fraud, for advising doctors, for recognizing speech and in helping complex planning tasks. Then there are intelligent tutoring systems that provide students with personalized attention.

Thus AI has increased understanding of the nature of intelligence and found many applications. It has helped in the understanding of human reasoning, and of the nature of intelligence. It has also helped us understand the complexity of modelling human reasoning.

APPROACHES TO AI

STRONG AI:- aims to build machines that can truly reason and solve problems. These machines should be self aware and their overall intellectual ability needs to be indistinguishable from that of a human being. Excessive optimism in the 1950s and 1960s concerning strong AI has given way to an appreciation of the extreme difficulty of the problem. Strong AI maintains that suitably programmed machines are capable of cognitive mental states.

WEAK AI:- deals with the creation of some form of computer-based artificial intelligence that cannot truly reason and solve problems, but can act as if it were intelligent. Weak AI holds that suitably programmed machines can simulate human cognition.

APPLIED AI:- aims to produce commercially viable "smart" systems such as, for example, a security system that is able to recognise the faces of people who are permitted to enter a particular building. Applied AI has already enjoyed considerable success.

COGNITIVE AI: - computers are used to test theories about how the human mind works--for example, theories about how we recognise faces and other objects, or about how we solve abstract problems.

LIMITS OF AI TODAY

AI systems have been able to achieve limited success in some of these tasks.

- In Computer vision, the systems are capable of face recognition
- In Robotics, we have been able to make vehicles that are mostly autonomous.
- In Natural language processing, we have systems that are capable of simple machine translation.
- Today's Expert systems can carry out medical diagnosis in a narrow domain
- Speech understanding systems are capable of recognizing several thousand words continuous speech
- Planning and scheduling systems had been employed in scheduling experiments with the Hubble Telescope.
- The Learning systems are capable of doing text categorization into about a 1000 topics
- In Games, AI systems can play at the Grand Master level in chess (world champion), checkers, etc.

What can AI systems NOT do yet?

- Understand natural language robustly (e.g., read and understand articles in a newspaper)
- Surf the web
- Interpret an arbitrary visual scene
- Learn a natural language
- Construct plans in dynamic real-time domains
- Exhibit true autonomy and intelligence Applications

WHAT IS SOFT COMPUTING?

An approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision. It is characterized by the use of inexact solutions to computationally hard tasks such as the solution of nonparametric complex problems for which an exact solution can't be derived in polynomial of time.

WHY SOFT COMPUTING APPROACH?

Mathematical model & analysis can be done for relatively simple systems. More complex systems arising in biology, medicine and management systems remain intractable to conventional mathematical and analytical methods. Soft computing deals with imprecision, uncertainty, partial truth and approximation to achieve tractability, robustness and low solution cost. It extends its application to various disciplines of Engg. and science. Typically human can:

1. Take decisions
2. Inference from previous situations experienced
3. Expertise in an area
4. Adapt to changing environment
5. Learn to do better
6. Social behaviour of collective intelligence

CHARACTERISTICS OF NEURO-FUZZY & SOFT COMPUTING:

1. Human Expertise
2. Biologically inspired computing models
3. New Optimization Techniques
4. Numerical Computation
5. New Application domains
6. Model-free learning
7. Intensive computation
8. Fault tolerance
9. Goal driven characteristics
10. Real world applications

Intelligent Control Strategies (Components of Soft Computing): The popular soft computing components in designing intelligent control theory are:

1. Fuzzy Logic
2. Neural Networks
3. Evolutionary Algorithms

HARD COMPUTING

- **IN 1996 , L. A. ZADE INTRODUCED THE TERM HARD COMPUTING**
- **ACCORDING TO ZADE, WE TERM A COMPUTING AS HARD COMPUTING IF**
    - **PRECISE RESULT IS GUARANTEED**
    - **CONTROL ACTION IS UNAMBIGUOUS**
    - **CONTROL ACTION IS PRE DEFINED WITH MATHEMATICAL MODEL OR ALGORITHM**
    - **TRUTHNESS VALUE IS 100% OR 1**

DIFFERENCE BETWEEN HARD COMPUTING AND SOFT COMPUTING

HARD COMPUTING

1. It is a conventional computing method that requires a precisely stated analytical model and often a lot of computational time
2. Based on binary logic, crisp system, numerical analysis, crisp software
3. It required program to be written
4. It uses two valued logic.
5. It is deterministic.
6. It require exact input data
7. It is strictly sequential.
8. It produces precise answer

SOFT COMPUTING

1. It is the tolerant of imprecision, uncertainty, partial truth and approximation.
2. Based on fuzzy logic, neural network, probabilistic reasoning.
3. It can evaluate its own program.
4. It uses multi valued logic.
5. It is stochastic.
6. It requires ambiguous, noisy, non exact data.
7. It allows parallel computation.
8. It can yield approximate answer.

HISTORY OF SOFT COMPUTING

- **INITIALIZED BY THE SCIENTIST NAMED LOTFI. A. ZADEH IN THE YEAR 1981.**
- **ZADEH DEFINED SOFT COMPUTING AS A FUSION OF MULTI DISCIPLINARY SYSTEM SUCH AS FUZZY LOGIC, NEURAL NETWORK, EVOLUTIONARY COMPUTING, GENETIC ALGORITHM, PROBABILISTIC APPROACH TO SOLVE A PROBLEM.**
- **SOFT COMPUTING WAS DEVELOPED AS FOLLOWS:-**

SC = EC + NN + FL WHERE

- **SC = SOFT COMPUTING – ZADEH (1981)**
- **EC = EVOLUTIONARY COMPUTING – RECHENBERG (1960)**
- **NN = NEURAL NETWORK – MC CULLOCH (1943)**
- **FL = FUZZY LOGIC – ZADEH (1965)**

EC = GP + ES + EP + GA

Where

- **EC = EVOLUTIONARY COMPUTING – RECHENBERG (1960)**
- **GP = GENETIC PROGRAMING – KOZA (1992)**
- **ES = EVOLUTIONARY STRATEGY – RICHENBERG(1965)**
- **EP = EVOLUTIONARY PROGRAMING – FOZEL (1962)**
- **GA = GENETIC ALGORITHM – HOLLARD (1970)**

FUZZY LOGIC:

Most of the time, people are fascinated about fuzzy logic controller. At some point of time in Japan, the scientists designed fuzzy logic controller even for household appliances like a room heater or a washing machine. Its popularity is such that it has been applied to various engineering products.

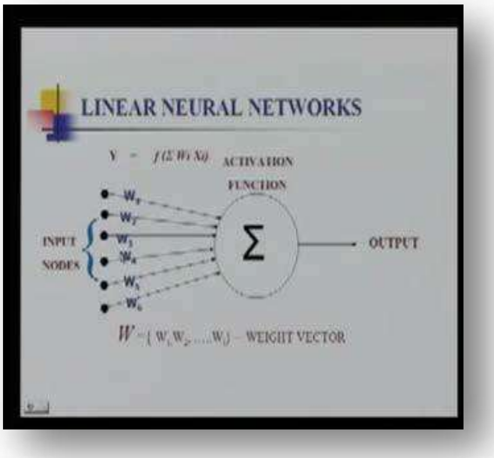
FUZZY NUMBER OR FUZZY VARIABLE:

We are discussing the concept of a fuzzy number. Let us take three statements: zero, almost zero, near zero. Zero is exactly zero with truth value assigned 1. If it is almost 0, then I can think that between minus 1 to 1, the values around 0 is 0, because this is almost 0. I am not very precise, but that is the way I use my day to day language in interpreting the real world. When I say near 0, maybe the bandwidth of the membership which represents actually the truth value. You can see that it is more, bandwidth increases near 0. This is the concept of fuzzy number. Without talking about membership now, but a notion is that I allow some small bandwidth when I say almost 0. When I say near 0 my bandwidth still further increases. In the case minus 2 to 2, when I encounter any data between minus 2 to 2, still I will consider them to be near 0. As I go away from 0 towards minus 2, the confidence level how near they are to 0 reduces; like if it is very near to 0, I am very certain. As I progressively go away from 0, the level of confidence also goes down, but still there is a tolerance limit. So when zero I am precise, I become imprecise when almost and I further become more imprecise in the third case.
When we say fuzzy logic, that is the variables that we encounter in physical devices, fuzzy numbers are used to describe these variables and using this methodology when a controller is designed, it is a fuzzy logic controller.

NEURAL NETWORKS:

Neural networks are basically inspired by various way of observing the biological organism. Most of the time, it is motivated from human way of learning. It is a learning theory. This is an artificial network that learns from example and because it is distributed in nature, fault tolerant, parallel processing of data and distributed structure.

The basic elements of artificial Neural Network are: input nodes, weights, activation function and output node. Inputs are associated with synaptic weights. They are all summed and passed through an activation function giving output y. In a way, output is summation of the signal multiplied with synaptic weight over many input channels.



LINEAR NEURAL NETWORKS

$Y = f(\Sigma W_i X_i)$

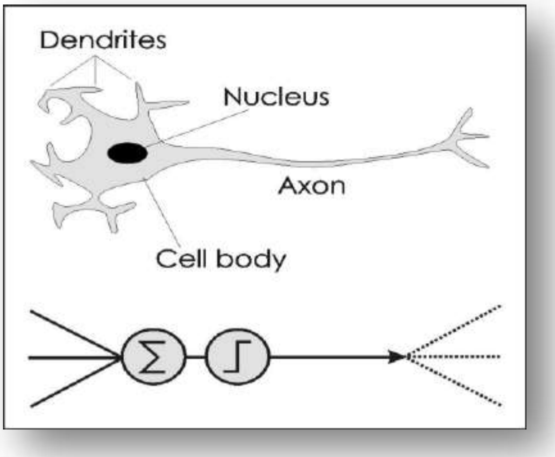


Fig. Basic elements of an artificial neuron

Biological Nuron Model

Analogy of biological neuron and artificial neuron

Through axon this neuron actuates the signal and this signal is sent out through synapses to various neurons. Similarly shown a classical artificial neuron (bottom).This is a computational unit. There are many inputs reaching this. The input excites this neuron. Similarly, there are many inputs that excite this computational unit and the output again excites many other units like here. Like that taking certain concepts in actual neural network, we develop these artificial computing models having similar structure.

There are various locations where various functions take place in the brain.
If we look at a computer and a brain, this is the central processing unit and a brain. Let us compare the connection between our high speed computers that are available in the market today and a brain. Approximately there are 10 to the power of 14 synapses in the human brain, whereas typically you will have 10 to the power of 8 transistors inside a CPU. The element size is almost comparable, both are 10 to the power minus 6 and energy use is almost like 30 Watts and comparable actually; that is energy dissipated in a brain is almost same as in a computer. But you see the processing speed. Processing speed is only 100 hertz; our brain is very slow, whereas computers nowadays, are some Giga hertz.
When you compare this, you get an idea that although computer is very fast, it is very slow to do intelligent tasks like pattern recognition, language understanding, etc. These are certain activities which humans do much better, but with such a slow speed, 100 Hz contrast
between these two, one of the very big difference between these two is the structure; one is brain, another is central processing unit is that the brain learns, we learn. Certain mapping that is found in biological brain that we have studied in neuroscience is not there in a central

processing unit and we do not know whether self awareness takes place in the brain or somewhere else, but we know that in a computer there is no self-awareness.

Neural networks are analogous to adaptive control concepts that we have in control theory and one of the most important aspects of intelligent control is to learn the control parameters, to learn the system model. Some of the learning methodologies we will be learning here is the error-back propagation algorithm, real-time learning algorithm for recurrent network, Kohonen's self organizing feature map & Hopfield network.
Features of Artificial Neural Network (ANN) models:

1. Parallel Distributed information processing
2. High degree of connectivity between basic units
3. Connections are modifiable based on experience
4. Learning is a continuous unsupervised process
5. Learns based on local information
6. Performance degrades with less units

EVOLUTIONARY ALGORITHMS:

These are mostly derivative free optimization algorithms that perform random search in a systematic manner to optimize the solution to a hard problem. In this course Genetic Algorithm being the first such algorithm developed in 1970‟s will be discussed in detail. The other algorithms are swarm based that mimic behaviour of organisms, or any systematic process.

FUZZY SETS BASIC CONCEPTS

- Characteristic Function (Membership Function)
- Notation
- Semantics and Interpretations
- Related crisp sets
- Support, Bandwidth, Core, α-level cut
- Features, Properties, and More Definitions
- Convexity, Normality
- Cardinality, Measure of Fuzziness
- MF parametric formulation
- Fuzzy Set-theoretic Operations
- Intersection, Union, Complementation
- T-norms and T-conorms
- Numerical Examples
- Fuzzy Rules and Fuzzy Reasoning
- Extension Principle and Fuzzy Relations
- Fuzzy If-Then Rules
- Fuzzy Reasoning
- Fuzzy Inference Systems
- Mamdani Fuzzy Models

- Sugeno Fuzzy Models
- Tsukamoto Fuzzy Models
- Input Space Partitioning
- Fuzzy Modeling.

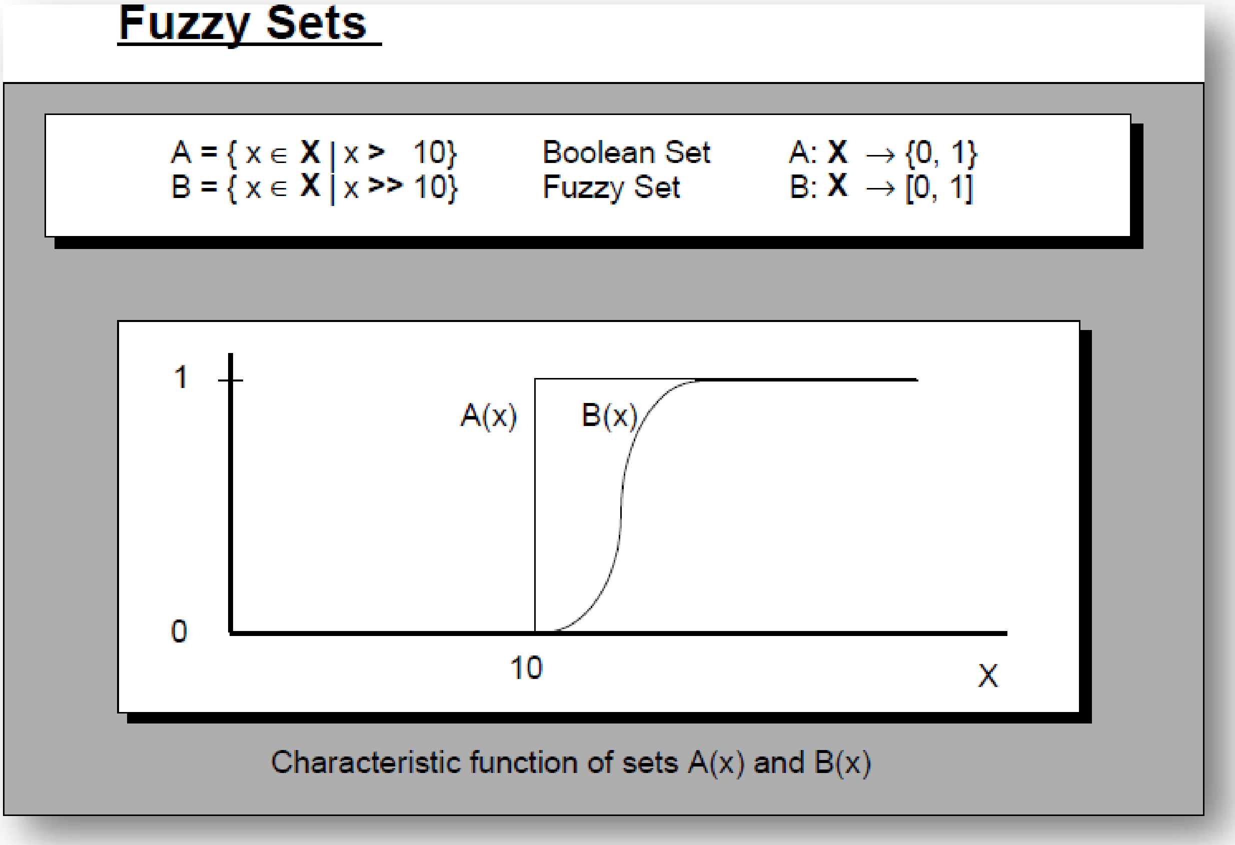
The father of fuzzy logic is **Lotfi Zadeh** who is still there, proposed in 1965. Fuzzy logic can manipulate those kinds of data which are imprecise.

BASIC DEFINITIONS & TERMINOLOGY:

Fuzzy Number:

A fuzzy number is fuzzy subset of the universe of a numerical number that satisfies condition of normality & convexity. It is the basic type of fuzzy set. Why fuzzy is used? Why we will be learning about fuzzy? The word fuzzy means that, in general sense when we talk about the real world, our expression of the real world, the way we quantify the real world, the way we describe the real world, are not very precise.

Fuzzy logic is logic which is not very precise. Since we deal with our world with this imprecise way, naturally, the computation that involves the logic of impreciseness is much more powerful than the computation that is being carried through a precise manner, or rather precision logic based computation is inferior; not always, but in many applications, they are very inferior in terms of technological application in our day to day benefits, the normal way.



Fuzzy Sets

$A = \{ x \in \mathbf{X} \mid x > 10\}$ Boolean Set $A: \mathbf{X} \to \{0, 1\}$

$B = \{ x \in \mathbf{X} \mid x >> 10\}$ Fuzzy Set $B: \mathbf{X} \to [0, 1]$

Characteristic function of sets A(x) and B(x)

Fig. Difference in Fuzzy and crisp boundary

As fuzzy means from precision to imprecision. Here, when I say 10, I have an arrow at 10, pointing that I am exactly meaning 10 means 10.00000 very precise. When I say they are all almost 10, I do not mean only 10, rather in the peripheral 10. I can tolerate a band from minus 9 to 9, whereas if I go towards 9 or 11, I am going away from 10, the notion of 10. That is what is almost 10, that is around 10, but in a small bandwidth, I still allow certain bandwidth for 10.

This concept to be imprecise is fuzzy or to deal with the day to day data that we collect or we encounter and representing them in an imprecise manner like here almost 0, near 0, or hot, cold, or tall; if I am referring to height, tall, short medium. This kind of terminology that we normally talk or exchange among ourselves in our communication actually deals with imprecise data rather than precise data. Naturally, since our communications are imprecise, the computation resulting out of such communication language, the language which is imprecise must be associated with some logic.

SET: A collection of objects having one or more common characteristics. For example, set of natural number, set of real numbers, members, or elements. Objects belonging to a set is represented as x belonging to A, where A is a set.

UNIVERSE OF DISCOURSE:

Defined as “a collection of objects all having the same characteristics”. Notation: U or X, and elements in the universe of discourse are: u or x

THE NOMENCLATURE/ NOTATION OF A FUZZY SET - how do we represent a fuzzy set there? One way is that let the elements of X be x_1, x_2, up to x_n ; then the fuzzy set A is denoted by any of the following nomenclature.
Mainly 2 types:
1. Numeric
2. Functional
Every member x of a fuzzy set A is assigned a fuzzy index. This is the membership grade μ_A (x) in the interval of 0 to 1, which is often called as the grade of membership of x in A. In a classical set, this membership grade is either 0 or 1; it either belongs to set A or does not belong. But in a fuzzy set this answer is not precise, answer is, it is possible. It is belonging to set A with a fuzzy membership 0.9 and I say it belongs to A with a fuzzy membership 0.1; that is, when I say 0.9, more likely it belongs to set A. When I say 0.1, less likely it belongs to set A. Fuzzy sets are a set of ordered pairs given by A.

MEMBERSHIP FUNCTION - a membership function µ A x is characterized by µ A that maps all the members in set x to a number between 0 to 1, where x is a real number describing an object or its attribute, X is the universe of discourse and A is a subset of X.

Mathematically, It can be expressed as:- $A = \{(x, \mu_A(x)) \mid x \in X\},$

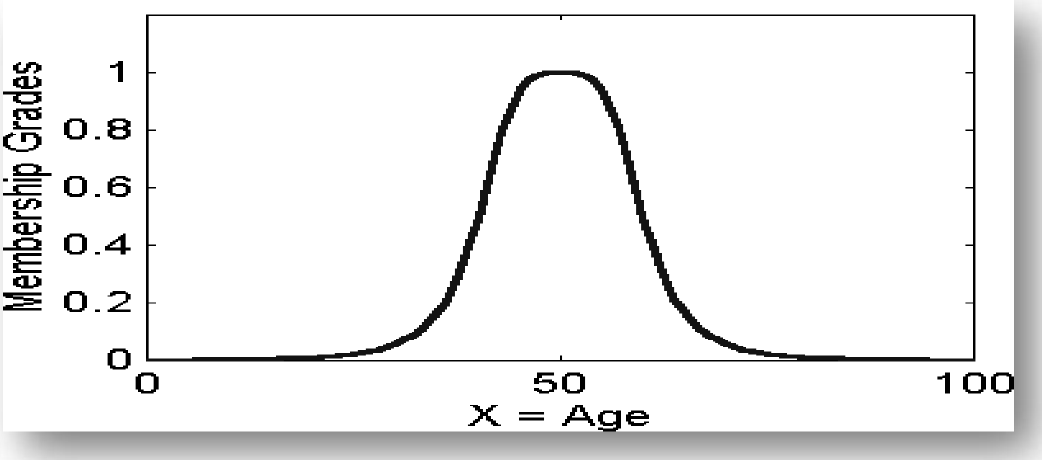


Fig. Fuzzy Set with Cont. Universe

Fuzzy set B = “about 50 years old”

X = Set of positive real numbers (continuous) B = {(x, $\mu_B(x)$) | x in X}

$\mu_B(x)=f(x)$

$$\mu_B(x) = \frac{1}{1+\left(\frac{x-50}{10}\right)^4}$$

Linguistic variable and linguistic values:

Linguistic variable is a variable expressed in linguistic terms of language e.g. “Age” that assumes various linguistic values like: middle aged, young, old. The linguistic variables are characterized by membership functions.

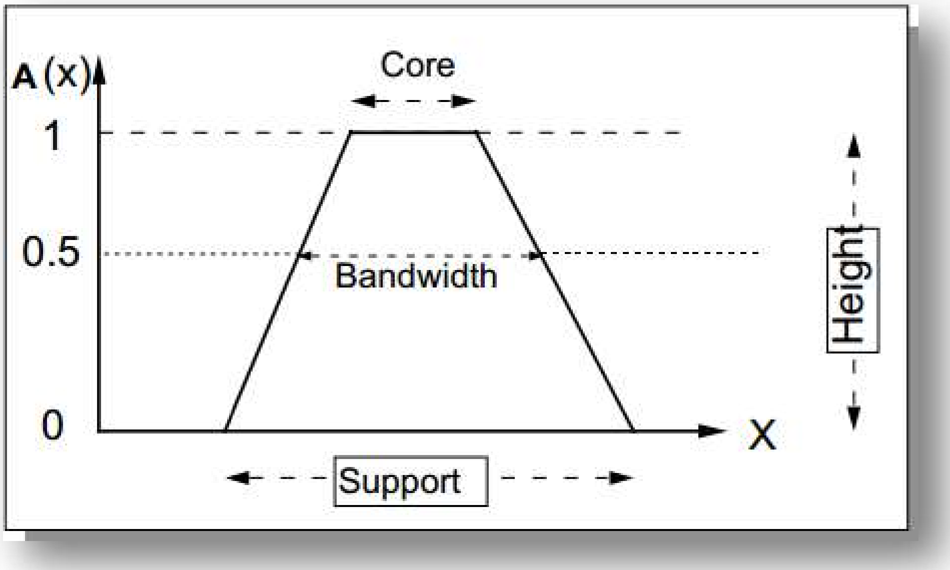


Fig. A membership function showing support, bandwidth, core, crossover points

SUPPORT:

Support of a fuzzy set A is the set of all points x in X such that $\mu_A(x)>0$. Support (A)= {x| $\mu_A(x)>0$}

CORE:

The core of a fuzzy set A is the set of all points x in X such that $\mu_A(x)=1$ core (A)= {x| $\mu_A(x)=1$}

NORMALITY:

A fuzzy set A is normal if its core is nonempty. Always there is at least one x with $\mu_A(x)=1$ then it is normal.

CROSSOVER POINT:

A cross over point in fuzzy set A is the x with $\mu_A(x)=0.5$ crossover (A)= {x| $\mu_A(x)=0.5$}

BANDWIDTH:

For a normal & convex fuzzy set

Width (A)=$|x_2-x_1|$, where x_2 & x_1 are crossover points. fuzzy singleton:

A fuzzy set whose support is a single point in X with $\mu_A(x)=1$ is called a fuzzy singleton.

The **α-cut** or **α-level set** of a fuzzy set A is a crisp set defined by

$$A_\alpha = \{x | \mu_A(x) \geq \alpha\}.$$

Strong α-cut or **strong α-level set** are defined similarly:

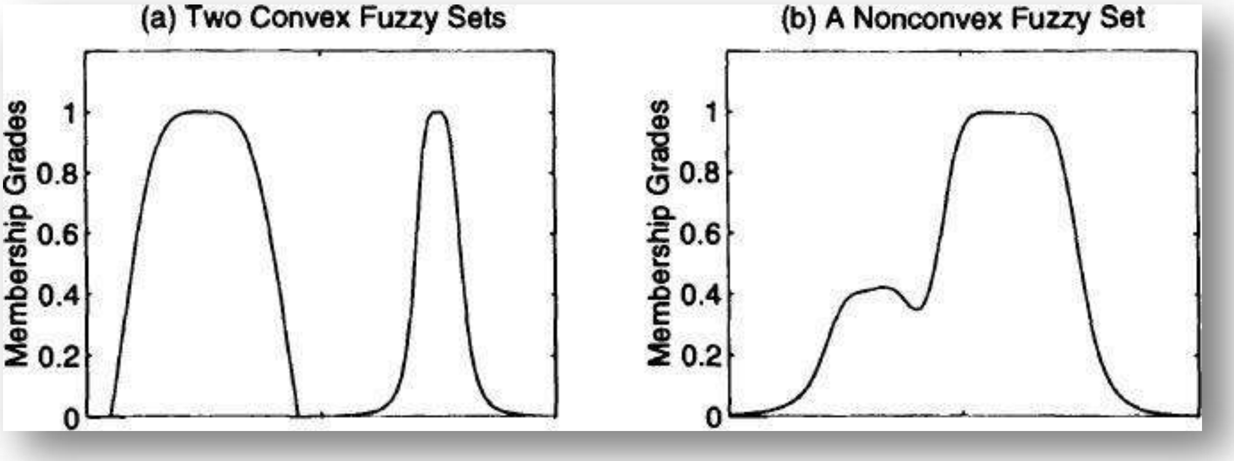
$$A'_\alpha = \{x | \mu_A(x) > \alpha\}.$$

CONVEXITY:

A fuzzy set A is **convex** if and only if for any $x_1,\ x_2 \in X$ and any $\lambda \in [0, 1]$,

$$\mu_A(\lambda x_1 + (1 - \lambda)x_2) \geq \min\{\mu_A(x_1), \mu_A(x_2)\}.$$

Alternatively, A is convex if all its α-level sets are convex.



SYMMETRY:

A fuzzy set is symmetric if its MF is symmetric about a certain point x=c such that,
$\mu_A(c+x) = \mu_A(c-x)$ for all x in X

PARAMETERIZATION OF MEMBERSHIP FUNCTION:

Once we talk about each member in a fuzzy set associated with membership function, you must know how to characterize this membership function. The parameters are adjusted to fine tune a fuzzy inference system to achieve desired I/O mapping. The membership functions given here are one- dimensional. 2 dimensional MFs can be formed by cylindrical extension from these basic MFs.

1. TRIANGULAR MF:-

$$\text{triangle}(x;a,b,c) = \begin{cases} 0, & x \le a. \\ \frac{x-a}{b-a}, & a \le x \le b. \\ \frac{c-x}{c-b}, & b \le x \le c. \\ 0, & c \le x. \end{cases}$$

Where a<b<c & that are x coordinates of the corners of triangular MF

2. TRAPEZODIAL MF

$$\text{trapezoid}(x;a,b,c,d) = \begin{cases} 0, & x \le a. \\ \frac{x-a}{b-a}, & a \le x \le b. \\ 1, & b \le x \le c. \\ \frac{d-x}{d-c}, & c \le x \le d. \\ 0, & d \le x. \end{cases}$$

Where a<b<c<d & that are x coordinates of the corners of trapezoidal MF

3. BELL MF

$$\text{bell}(x;a,b,c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}},$$

Where c is the centre & a is adjusted to vary the width of MF, b controls slope at crossover points.

Bell membership function is also termed as Cauchy MF.

4. GAUSSIAN MF

$$\text{gaussian}(x;c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}.$$

Where c is the centre & Ϭ is the width of MF.

5. LEFT-RIGHT MF:

$$\text{LR}(x;c,\alpha,\beta) = \begin{cases} F_L\left(\frac{c-x}{\alpha}\right), & x \le c. \\ F_R\left(\frac{x-c}{\beta}\right), & x \ge c, \end{cases}$$

6. SIGMOIDAL MF:

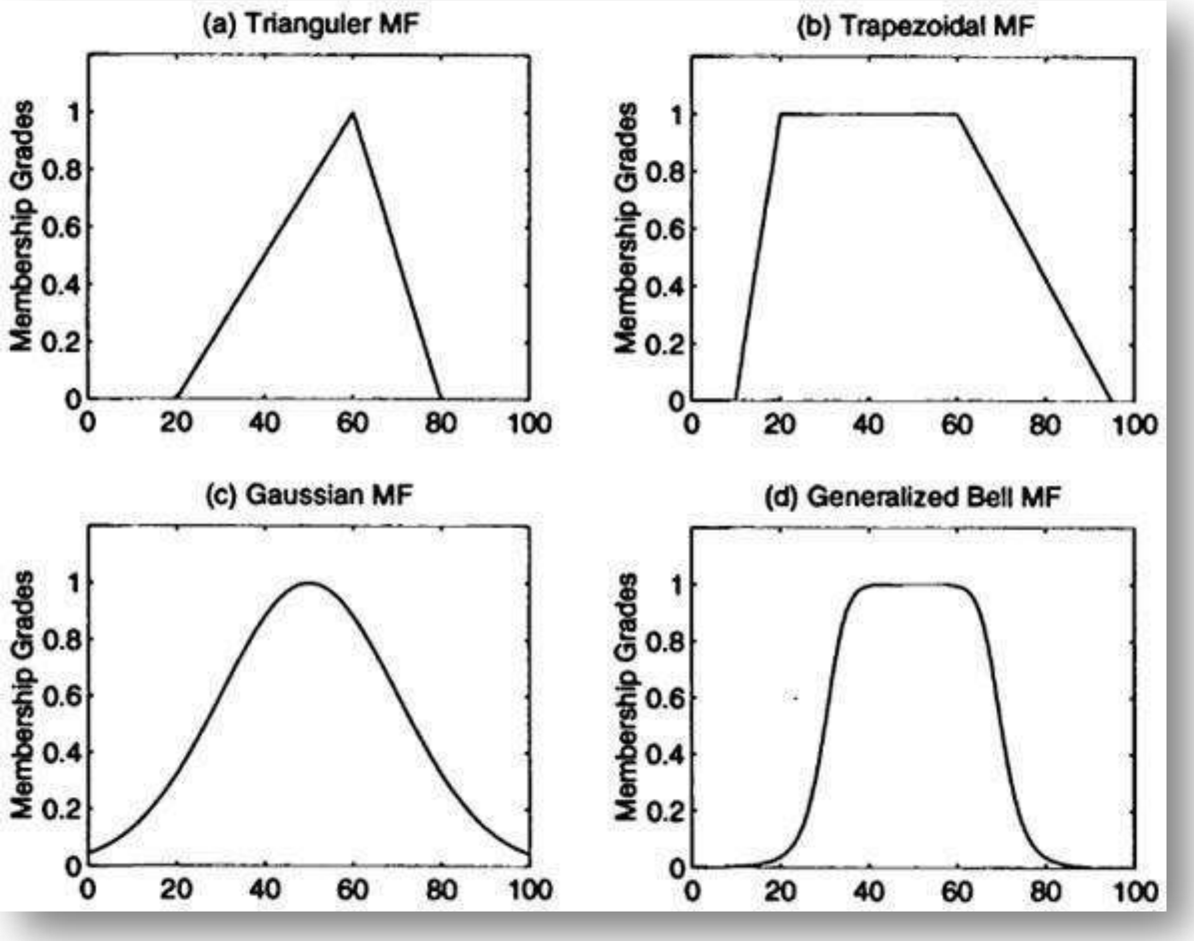
A **sigmoidal MF** is defined by

$$\mathrm{sig}(x; a, c) = \frac{1}{1 + \exp[-a(x - c)]},$$

where a controls the slope at the crossover point $x = c$.

It can be open left or open right depending on sign of a.

GRAPHICAL REPRESENTATION:-



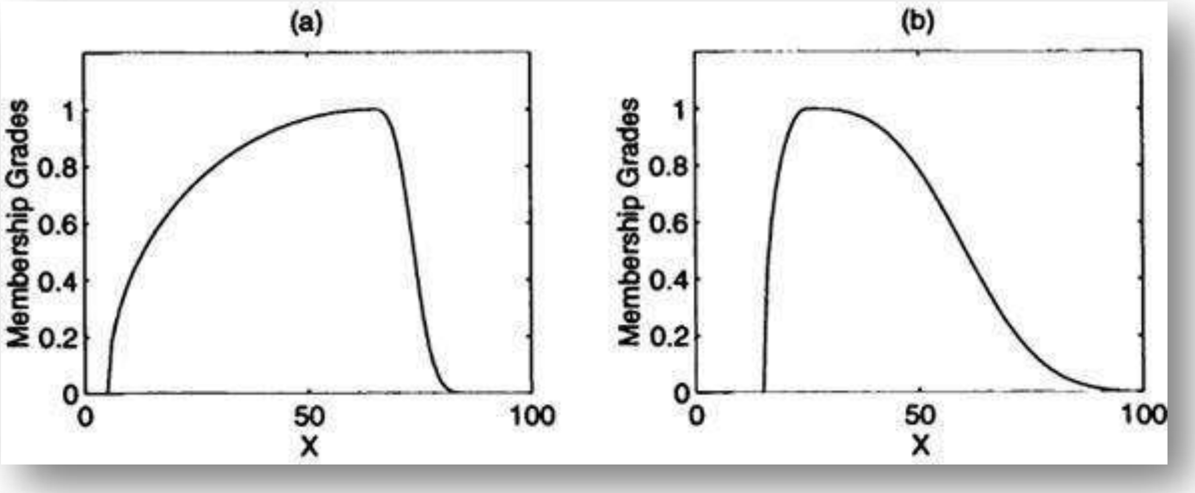


Fig. Membership functions a. Triangle b. Trapezoidal c. Gaussian d. Bell, e. Left f. Right

Fuzzy set operations:

The main features of operation on fuzzy set are that unlike conventional sets, operations on fuzzy sets are usually described with reference to membership function. When I say operation, I do not do with the member itself, but I manipulate. When I say operation, I manipulate the membership of the members in a set; members are not manipulated, rather the membership function of the member is manipulated. This is very important; that is, x and µ(x). In classical set what is manipulated is x.

If I say, x is 1 In classical set when I say x is 1 then, I would say 1 minus x is 0. In this, the manipulation concerns with the member; whereas any kind of manipulation in fuzzy set does not involve with x; rather it involves µx.

Containment or subset:

Fuzzy set A is **contained** in fuzzy set B (or, equivalently, A is a **subset** of B, or A is smaller than or equal to B) if and only if $\mu_A(x) \leq \mu_B(x)$ for all x. In symbols,

$$A \subseteq B \Longleftrightarrow \mu_A(x) \leq \mu_B(x)$$

Three common operations: intersection which we say is the minimum function, union, which we say is the maximum function and then fuzzy complementation.

Standard fuzzy operations:

INTERSECTION (CONJUNCTION) OR T-NORM:

We can easily see that, the membership of A (green) intersection B(red) in fig. is all the members that belongs to, that is common between A and B. Their membership will follow these (blue) curves. There are two things we are doing. We have 2 sets. One is set A and the other is set B. Classically, what we see is the common members between A and B. We are not only seeing the common members, here we are also seeing, what is their membership function.

$$\mu_C(x) = \min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x).$$

The membership function is computed minimum; that is, μ_A intersection μ_B is minimum of μ_Ax and μ_Bx. That is the membership function. When there is a common member between A and B, the membership function wherever is minimum that is retained and the other one is thrown away. The member is retained; what is changing is the membership function.

UNION (DISJUNCTION) OR T-CO-NORM OR S-NORM:

$$\mu_C(x) = \max(\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x).$$

The membership function is computed maximum; that is, μ_A union μ_B is maximum of μ_Ax and μ_Bx. That is the membership function.

COMPLEMENT (NEGATION):

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x).$$

The compliment of a Fuzzy Set A can be represented as

They are commutative. A union B is B union A; A intersection B is B intersection A. It is like classical sets; fuzzy sets equally hold.

DIFFERENCE BETWEEN TWO FUZZY SETS:

A-B = {(x, μ_{A-B} (x)) for all x belongs X}

μ_{A-B} (x) = A Λ B^C

MULTIPLICATION OF TWO FUZZY SETS:-

A.B = {(x, $\mu_{A.B}$ (x)) for all x belongs X}

$\mu_{A.B}$ (x) = μ_A(X) Λ μ_B(X)

POWER OF FUZZY SETS

The POWER "n" of a fuzzy set A is defined in an universe of discourse X can be represented as

A^n = {(x, $\mu_A{}^n$(x)) for all x belongs X}

$\mu_A{}^n(x) = [\mu_A(X)]^n$

The power set has two relations

1. **CONCENTRATION**
2. **DILATION**

1. **CONCENTRATION:- If the POWER "n" is taken as "2" then it is called as CONCENTRATION**

$$\mathrm{CON}(A) = A^2,$$

2. **DILATION:- If the POWER "n" is taken as "1/2" then it is called as DILATION**

$$\mathrm{DIL}(A) = A^{0.5}.$$

CARTESIAN PRODUCT & CO-PRODUCT:

Let A & B be fuzzy sets in X & Y respectively, then Cartesian product of A & B is a fuzzy set in the product space X x Y with the membership function

$$\mu_{A\times B}(x, y) = \min(\mu_A(x), \mu_B(y)).$$

Similarly, Cartesian co-product A+B is a fuzzy set

$$\mu_{A+B}(x, y) = \max(\mu_A(x), \mu_B(y)).$$

Both Product & Co-product are characterized by 2- dimensional MFs.

FUZZY EXTENSION PRINCIPLE:

Consider a function *y* = *f* (*x*).
If we known *x* it is possible to determine *y*.
Is it possible to extend this mapping when the input, *x*, is a fuzzy value.
The extension principle developed by Zadeh (1975) and later by Yager (1986) establishes how to extend the domain of a function on a fuzzy set.

Suppose that *f* is a function from *X* to *Y* and *A* is a fuzzy set on *X* defined as

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + . . . + \mu_A(x_n)/x_n.$$

The extension principle states that the image of fuzzy set *A* under the mapping *f* (.) can be expressed as a fuzzy set *B* defined as

$$B = f\ (A) = \mu_A(x_1)/y_1 + \mu_A(x_2)/y_2 + . . . + \mu_A(x_n)/y_n \text{ where } y_i = f\ (x_i\ )$$

A point to point mapping from a set A to B through a function is possible. If it is many to one for two x in A then the membership function value in set B is calculated for f(x) as max value of MF.

Let

$$A = 0.1/-2 + 0.4/-1 + 0.8/0 + 0.9/1 + 0.3/2$$

and

$$f(x) = x^2 - 3.$$

Upon applying the extension principle, we have

$$\begin{aligned} B &= 0.1/1 + 0.4/-2 + 0.8/-3 + 0.9/-2 + 0.3/1 \\ &= 0.8/-3 + (0.4 \vee 0.9)/-2 + (0.1 \vee 0.3)/1 \\ &= 0.8/-3 + 0.9/-2 + 0.3/1, \end{aligned}$$

FUZZY RELATION:

CRISP MAPPINGS:

Consider the Universe *X* = {−2, −1, 0, 1, 2} Consider the set *A* = {0, 1}

Using the Zadeh notation *A* = {0/ −2 + 0/−1 + 1/ 0 + 1/ 1 + 0/ 2} Consider the mapping *y* = |4*x*| + 2

What is the resulting set *B* on the Universe *Y* = {2, 6, 10}

It is possible to achieve the results using a relation that express the mapping *y* = |4*x*| + 2.

Lets *X* = {−2,−1, 0, 1, 2}.

Lets *Y* = {0, 1, 2 . . . 9, 10}

The relation

$$R = \begin{array}{c|ccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

B* = *A* ◦ *R

$A = \{\frac{0}{-2} + \frac{0}{-1} + \frac{1}{0} + \frac{1}{1} + \frac{0}{2}\}$ or more conveniently $A = \{0, 0, 1, 1, 0\}$

Using $\chi_B(y) = \vee_{x\in X}(\chi_A(x) \wedge \chi_R(x,y))$
we find

$$\chi_B(y) = \begin{cases} 1, & \text{for } y = 2, 6 \\ 0, & \text{otherwise} \end{cases}$$

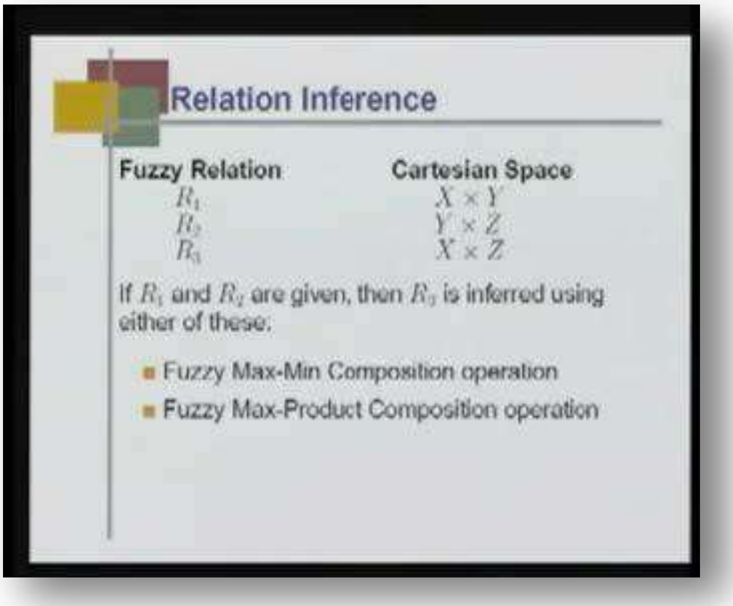
Or

$$B = \left\{ \frac{0}{0} + \frac{0}{1} + \frac{1}{2} + \frac{0}{3} + \frac{0}{4} + \frac{0}{5} + \frac{1}{6} + \frac{0}{7} + \frac{0}{8} + \frac{0}{9} + \frac{0}{10} \right\}$$

FUZZY MAPPINGS:

Let X and Y be two universes of discourse. Then

$$\mathcal{R} = \{(\,(x,y),\ \mu_{\mathcal{R}}(x,y)\,)\mid (x,y) \in X \times Y\}$$



Relation Inference

Fuzzy Relation	Cartesian Space
R_1	$X \times Y$
R_2	$Y \times Z$
R_3	$X \times Z$

If R_1 and R_2 are given, then R_3 is inferred using either of these:

- Fuzzy Max-Min Composition operation
- Fuzzy Max-Product Composition operation

Fig. Inferring Fuzzy relation

Max-min composition or Max-min product:

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two fuzzy relations defined on $X \times Y$ and $Y \times Z$, respecti
The **max-min composition** of $\mathcal{R}_1$ and $\mathcal{R}_2$ is a fuzzy set defined by

$$\mathcal{R}_1 \circ \mathcal{R}_2 = \{[(x,z),\ \max_y \ \min(\mu_{\mathcal{R}_1}(x,y), \mu_{\mathcal{R}_2}(y,z))] | x \in X, y \in Y, z \in Z\},$$

or, equivalently,

$$\begin{aligned} \mu_{\mathcal{R}_1 \circ \mathcal{R}_2}(x,z) &= \max_y \ \min[\mu_{\mathcal{R}_1}(x,y), \mu_{\mathcal{R}_2}(y,z)] \\ &= \vee_y \ [\mu_{\mathcal{R}_1}(x,y) \wedge \mu_{\mathcal{R}_2}(y,z)], \end{aligned}$$

Max-product composition:

$$\mu_{\mathcal{R}_1 \circ \mathcal{R}_2}(x, z) = \max_y [\mu_{\mathcal{R}_1}(x, y)\mu_{\mathcal{R}_2}(y, z)].$$

Fuzzy If-then rules:

A **fuzzy if-then rule** (also known as **fuzzy rule**, **fuzzy implication**, or **fuzzy conditional statement**) assumes the form

If x is A then y is B

"x is A" is antecedent or premise which tells the fact "y is B" is consequence or conclusion

The whole statement is the rule. Eg. If tomato is red then it is ripe.

These if then rules are the base of fuzzy reasoning. If then rules are of different types:

1. Single rule with single antecedent
2. Single rule with multiple antecedent
3. Multiple with multiple antecedent

Steps of Fuzzy reasoning:

Shown in fig. For 2 rules what will be the consequent MF after aggregation

1. Degree of compatibility
2. Firing strength
3. Qualified consequent MF
4. Aggregate all qualified consequent MFs to obtain an overall MF

FUZZY MODELLING:

Fuzzy Inferencing

The process of fuzzy reasoning is incorporated into what is called a Fuzzy Inferencing System. It is comprised of three steps that process the system inputs to the appropriate system outputs. These steps are

1) Fuzzification
2) Rule Evaluation
3) Defuzzification.

The system is illustrated in the following figure.

Each step of fuzzy inferencing is described in the following sections.

Fuzzification

Fuzzification is the first step in the fuzzy inferencing process. This involves a domain transformation where crisp inputs are transformed into fuzzy inputs. Crisp inputs are exact inputs measured by sensors and passed into the control system for processing, such as temperature, pressure, rpm's, etc.. Each crisp input that is to be processed by the FIU has its own group of membership functions or sets to which they are transformed. This group of membership functions exists within a universe of discourse that holds all relevant values that the crisp input can possess. The following shows the structure of membership functions within a universe of discourse for a crisp input.

Where:

Degree of membership: degree to which a crisp value is compatible to a membership function, value from 0 to 1, also known as truth value or fuzzy input.

Membership function, MF: defines a fuzzy set by mapping crisp values from its domain to the sets associated degree of membership.

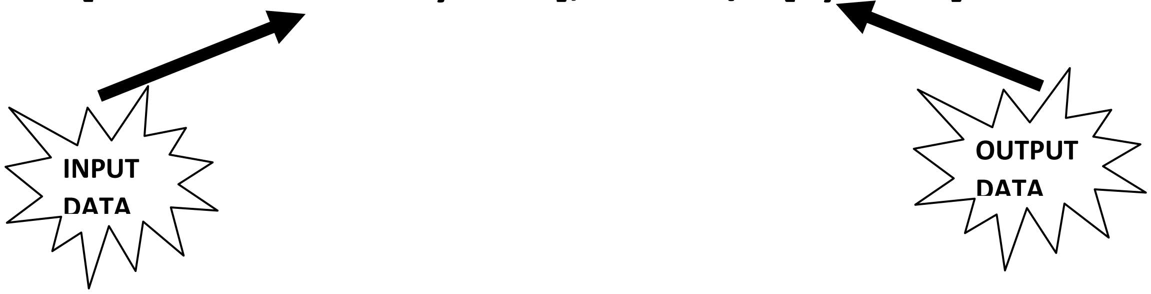
GENERALIZED FUZZY INFERENCE RULE:-

There are two types of fuzzy inference rule present

1. Generalized MODUS PONEN (GMP)
2. Generalized MODUS TOLLEN (GMT)

GMP: - It can be explained as

[If "x" is A THEN "y" is B]; x is A^{l} / ["y" is B^{l}]



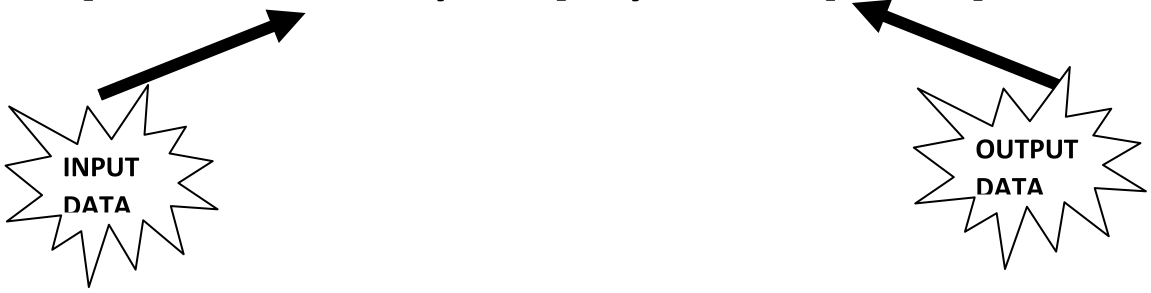
Here A, B, A^{l}, B^{l} are four fuzzy sets in which AI and BI are two special cases of A and B respectively.

Mathematically B^{l} = A^{l} Composition { (A×B) U A^{l}×Y)}

Otherwise B^{l} = A^{l} composition R, where R = { (A×B) U A^{l}×Y)}

GMT: - It can be explained as

[If “x” is A THEN “y” is B] ; “y” is B^{l} / [x is A^{l}]



Mathematically A^{l} = B^{l} Composition { (A×B) U A^{l}×Y)}

Otherwise A^{l} = B^{l} composition R, where R = { (A×B) U A^{l}×Y)}

DEFUZZIFICATION:-

The method of conversion of a fuzzy set into a defuzzified value is called as defuzzification.

3 methods are there for defuzzification

1. Centroid Method/Centre of Area/Centre of gravity Method [COM]
2. Centre of Sum Method [COS]
3. Mean of Maxima Method [MOM]

CENTROID METHOD

Mathematically it can be expressed as

$$x^* = \frac{\int \mu(x).x\,dx}{\int \mu(x).dx}$$ **for continuous values of x**

$$x^* = \frac{\sum x.\mu(x)}{\sum \mu(x)}$$

CENTRE OF SUM METHOD:-

In this method, the overlapping area is considered twice, where as in case of centroid method the overlapping area is considered once.

Therefore the centre of sum method develops a resultant membership function can be considered by taking the algebraic sum of each individual membership function of different fuzzy sets

Mathematically:-

$$\mathbf{x^*} = \frac{\sum x . \sum \square(x)}{\sum \square(x)}$$

MEAN OF MAXIMA METHOD:-

In this method the mean of membership function values are calculated for wach fuzzy sets and then we are considering the maximum value out of all mean values of different fuzzy sets.

Mathematically:-

$$\mathbf{x^*} = \frac{\sum x .}{\mathbf{M}}$$ **HERE M = { X/μ(X) IS THE HEIGHT OF THE FUZZY SET}**

FUZZY INFERENCE MODEL

MAMDANI FUZZY INFERENCE MODEL:-

RULES:-

STEP-1:-

CALCULATION OF FIRING LEVELS α1 AND α2

α1 = A1 (X0) Λ B1 (Y0)

α2 = A2 (X0) Λ B2 (Y0)

STEP-2:-

CALCULATION OF INDIVIDUAL RULE OUTPUT

C1* AND C2*

C1* = α1 Λ C1 (Z)

C2* = α2 Λ C2 (Z)

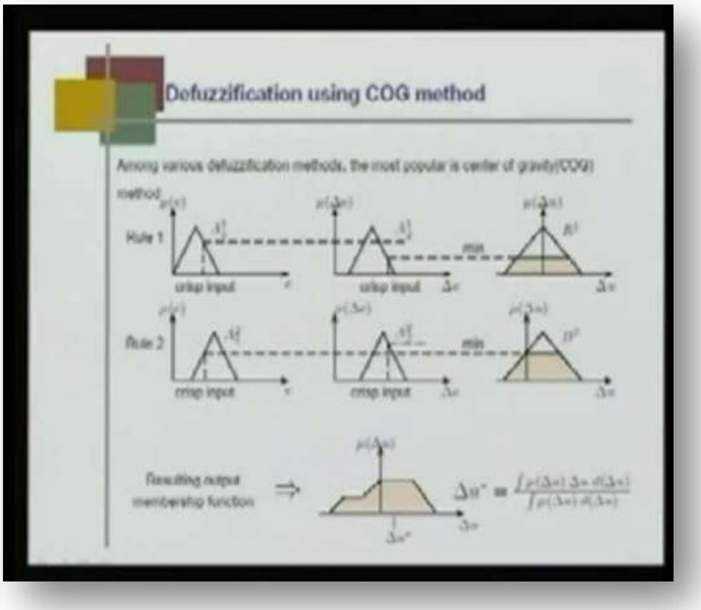
STEP-3:-

CALCULATION OF OVERALL RULE OUTPUT

C* = C1* Uʹ C2*

STEP-4:-

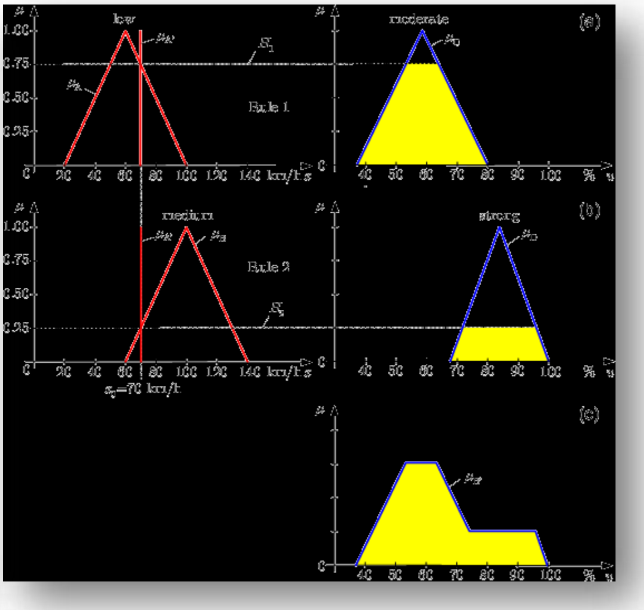
CALCULATION OF CRISP OUTPUT BY DEFUZZIFICATION.



Defuzzification using COG method

Among various defuzzification methods, the most popular is center of gravity(COG) method

Rule 1
Rule 2
crisp input
crisp input
crisp input
crisp input
min
min
Resulting output membership function
⇒
$\Delta u^* = \frac{\int \mu(\Delta u)\,\Delta u\, d(\Delta u)}{\int \mu(\Delta u)\, d(\Delta u)}$



TAKAGI–SUGENO FUZZY SYSTEMS:

The number of rules required by the Mamdani model are reduced here. They employ function of the input fuzzy linguistic variable as the consequent of the rules. What happens here is that a fuzzy dynamic model is expressed in the form of local rules. Local rule means if there are two variables x1 = a and x2 = b, then the plant dynamics can be represented either as a linear dynamical system or a nonlinear dynamical system, as a known dynamical system.

TSK Fuzzy Rule

- If x is A and y is B then z = f(x,y)

– Where A and B are fuzzy sets in the antecedent, and

– Z = f(x,y) is a crisp function in the consequence.

- Usually f(x,y) is a polynomial in the input variables x and y, but it can be any function describe the output of the model within the fuzzy region specified by the antecedence of the rule.

RULES:-

STEP-1:-

CALCULATION OF FIRING LEVELS W1 AND W2

W1 = A1 (X0) Λ B1 (Y0)

W2 = A2 (X0) Λ B2 (Y0)

STEP-2:-

CALCULATION OF INDIVIDUAL CRISP CONTROL OUTPUT

Z1 = P1 X + Q1 Y + R1

Z2 = P2 X + Q2 Y + R2

STEP-3:-

CALCULATION OF OVERALL CRISP OUTPUT

$$Z = \frac{w1x1+w2x2}{w1+w2}$$

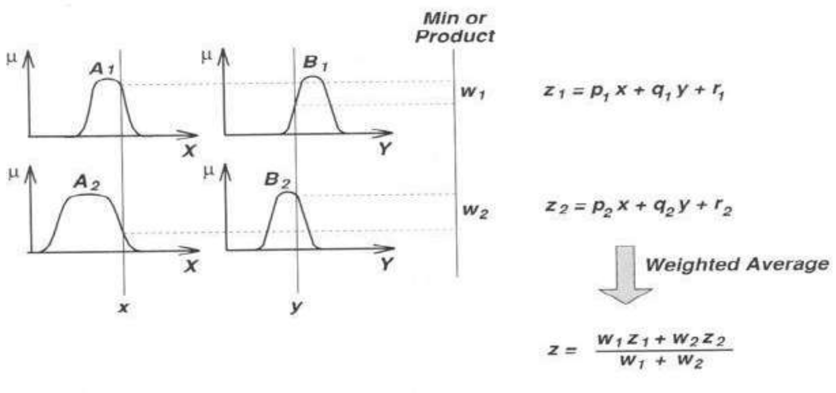

Min or Product
μ
A1
μ
B1
w1
$z_1 = p_1x + q_1y + r_1$
X
Y
μ
A2
μ
B2
w2
$z_2 = p_2x + q_2y + r_2$
X
Y
Weighted Average
x
y
$z = \frac{w_1z_1 + w_2z_2}{w_1 + w_2}$

Fig. Rules in first order TSK fuzzy model

TSUKAMOTO FUZZY INFERENCE MODEL:-

TSUKAMOTO FUZZY INFERENCE MODEL RULE

RULES:-

STEP-1:-

CALCULATION OF FIRING LEVELS α1 AND α2

α1 = A1 (X0) Λ B1 (Y0)

α2 = A2 (X0) Λ B2 (Y0)

STEP-2:-

CALCULATION OF INDIVIDUAL CRISP CONTROL OUTPUT

Z1 AND Z2

α1 = C1 (Z)

α2 = C2 (Z)

STEP-3:-

CALCULATION OF OVERALL CRISP CONTROL OUTPUT

$$Z_0 = \frac{\alpha1\ Z1+\alpha2\ Z2}{\alpha1+\alpha2}$$

MODULE-II

NEURAL NETWORK:-

INTRODUCTION:-

What is a neuron?
A neuron is the basic processing unit in a neural network sitting on our brain. It consists of

1. Nucleus

2. Axon- Output node

3. Dendrites-Input node

4. Synaptic junction

The dynamics of this synaptic junction is complex. We can see the signal inputs from the action of a neuron and through synaptic junction an output is actuated which is carried over through dendrites to another neuron. Here, these are the neurotransmitters. We learned from our experience that these synaptic junctions are either reinforced or in the sense they behave in such a way that the output of synaptic junction may excite a neuron or inhibit the neuron. This reinforcement of the synaptic weight is a concept that has been taken to artificial neural model.
The objective is to create artificial machine and this artificial neural networks are motivated by certain features that are observed in human brain, like as we said earlier, parallel distributed information processing.

ANN MODELS:-

- Parallel, distributed information processing
- High degree of connectivity among basic units
- Connections are modifiable based on experience
- Learning is a constant process and usually unsupervised
- Learning is based upon local information
- Performance degrades gracefully if some units are removed

Artificial neural networks are among the most powerful learning models. They have the versatility to approximate a wide range of complex functions representing multi-dimensional input-output maps. Neural networks also have inherent adaptability, and can perform robustly even in noisy environments.

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 simple processing elements (neurons) 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.

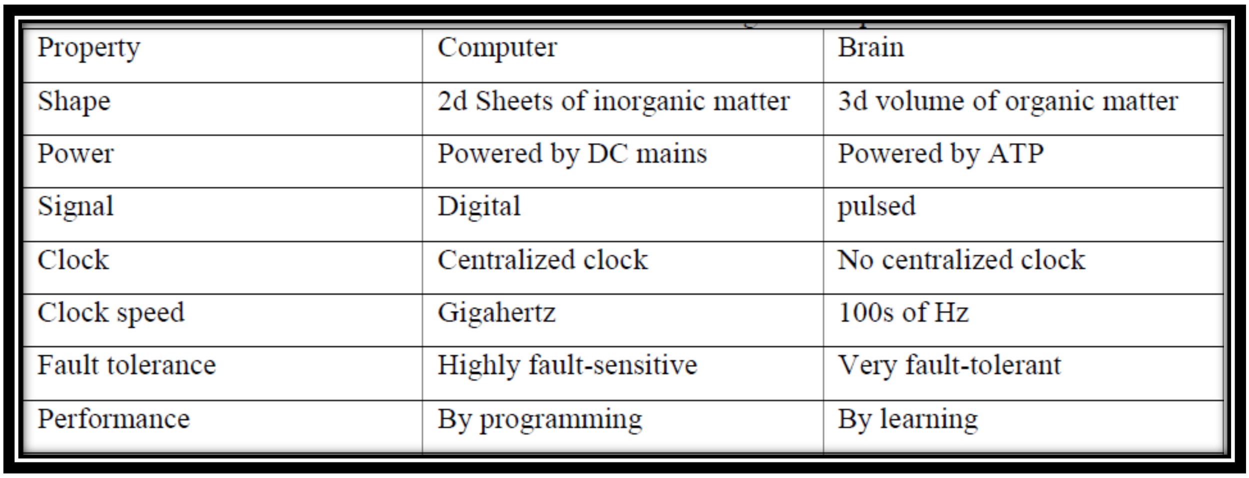
Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well. ANNs can process information at a great speed owing to their highly massive parallelism.

A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

ADVANTAGES OF ANN:

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
- Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
- 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.
- 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.

Difference between the brain and a digital Computer



Property	Computer	Brain
Shape	2d Sheets of inorganic matter	3d volume of organic matter
Power	Powered by DC mains	Powered by ATP
Signal	Digital	pulsed
Clock	Centralized clock	No centralized clock
Clock speed	Gigahertz	100s of Hz
Fault tolerance	Highly fault-sensitive	Very fault-tolerant
Performance	By programming	By learning

Differences human brain & ANN:

- Computer has such fast speed of GHz, a traditional computer, however, when it comes to certain processing like pattern recognition and language understanding, the brain is very fast.
- Intelligence and self-awareness, are absent in an artificial machine.

AN ARTIFICIAL NEURON:

Basic computational unit in an artificial neural network is neuron. Obviously, it has to be an artificial neuron.

This artificial neuron has three basic elements:

1. Nodes,
2. Weights and
3. Activation function.

Between input nodes and output nodes, there are synaptic weights w_1, w_2, w_3, w_4, w_5 and w_6. There can be as many weights and these weights are multiplied with the signal as they reach the output unit, where the output is simply sum of the signal multiplied with the weights and then this output goes to an activation function f.

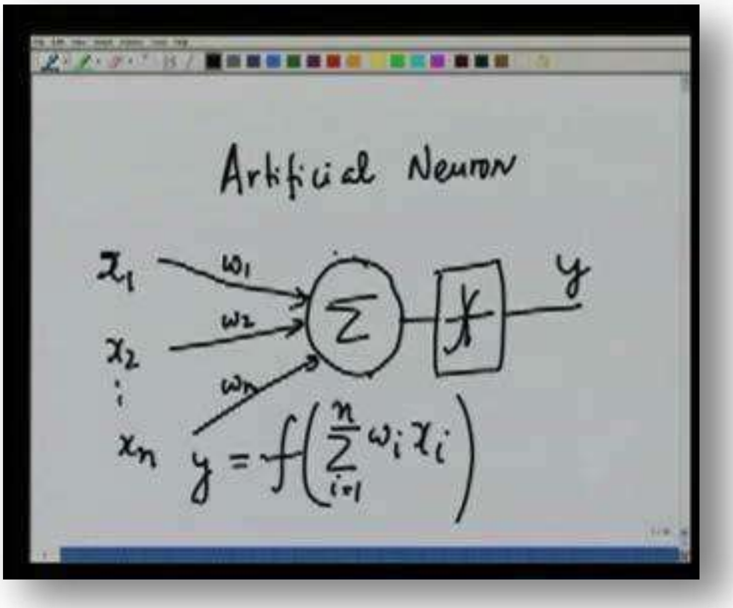


Fig. Basic processing unit- the neuron

In a simple neuron, if input signals be x 1, x2, ... xn with weights w1 , w2 and wn respectively. The weighted sum will activate this total output by an activation function f. That is your output. What you are seeing is actually a nonlinear map from input vector xi to output y. A single neuron has single output but multiple inputs. Inputs are multiple for a single neuron and the output is unique, y and this output y and the input bear a nonlinear relationship, by f. Neural networks can be built using this single neuron. We can use the single neuron and build neural networks.

ANALOGY WITH BRAIN:

Artificial Neural Network (ANN) is a system which performs information processing. An ANN resembles or it can be considered as a generalization of mathematical model of human brain assuming that

1. Information processing occurs at many simple elements called neurons.
2. Signals are passed between neurons over connection links.
3. Each connection link has an associated weight, which in a typical neural net multiplies the signal transmitted.

ANN is built with basic units called *neurons* which greatly resemble the neurons of human brain. A neural net consists of a large number of simple processing elements called neurons. Each neuron applies an activation function to its net input to determine its output signal. Every neuron is connected to other neurons by means of directed communication links, each with an associated weight. Each neuron has an internal state called its *activation level*, which is a function of the inputs it has received. As and when the neuron receives the signal, it gets added up and when the cumulative signal reaches the activation level the neuron sends an output. Till then it keeps receiving the input. So activation level can be considered as a threshold value for us to understand.

In general, a neural network is characterized by

1. Pattern of connections between the neurons called its architecture
2. Method of determining the weights on the connections called its training or learning algorithm
3. Its internal state called its Activation function.

The arrangement of neurons into layers and the connection patterns within and between layers is called the net *architecture*. A neural net in which the signals flow from the input units to the output units in a forward direction is called *feed forward nets*.
Interconnected competitive net in which there are closed loop signal paths from a unit back to it is called a *recurrent network*. In addition to architecture, the method of setting the values of the weights called *training* is an important characteristic of neural nets. Based on the training methodology used neural nets can be distinguished into *supervised* or *unsupervised* neural nets. For a neural net with supervised training, the training is accomplished by presenting a sequence of training vectors or patterns each with an associated target output vector. The weights are then adjusted according to a learning algorithm. For neural nets with unsupervised training, a sequence of input vectors is provided, but no target vectors are specified. The net modifies the weights so that the most similar input vectors are assigned to the same output unit. The neural net will produce a representative vector for each cluster formed. Unsupervised learning is also used for other tasks, in addition to clustering.

ACTIVATION FUNCTIONS:

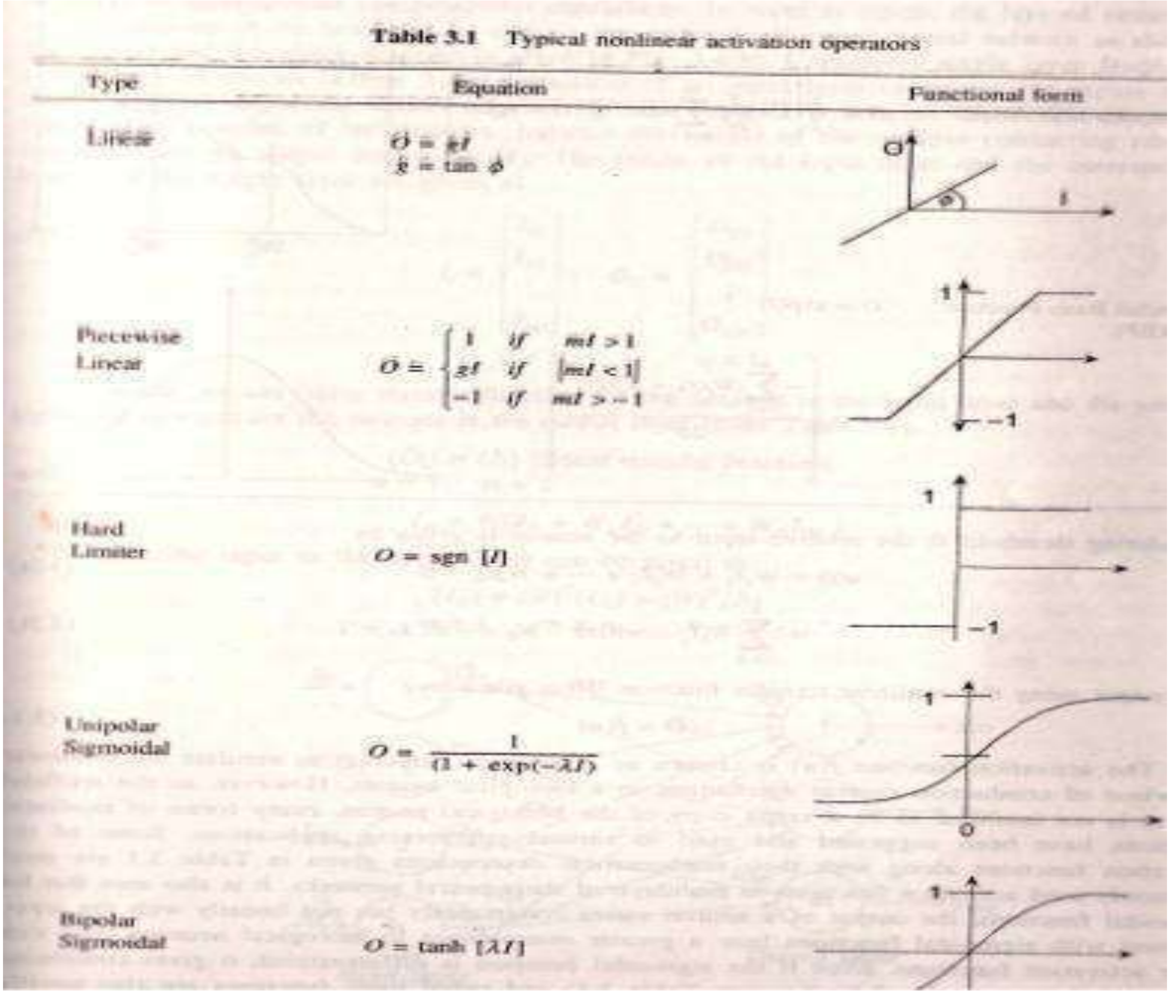

O
φ
I
1
−1
1
−1
1
0
1

Table 3.1 Typical nonlinear activation operators

Type	Equation	Functional form
Linear	$O = gI$ $g = \tan\phi$	
Piecewise Linear	$O = \begin{cases} 1 & \text{if} \quad mI > 1 \\ gI & \text{if} \quad \lvert mI < 1 \rvert \\ -1 & \text{if} \quad mI > -1 \end{cases}$	
Hard Limiter	$O = \operatorname{sgn}[I]$	
Unipolar Sigmoidal	$O = \dfrac{1}{(1 + \exp(-\lambda I))}$	
Bipolar Sigmoidal	$O = \tanh[\lambda I]$	

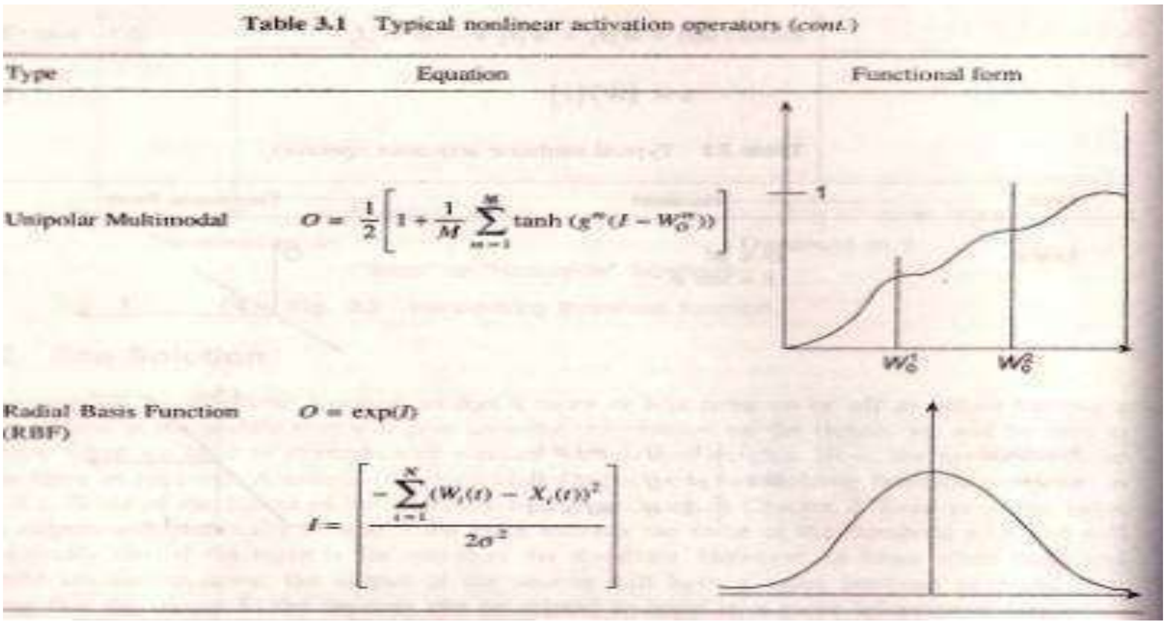


Table 3.1 Typical nonlinear activation operators *(cont.)*

Type	Equation	Functional form
Unipolar Multimodal	$O = \dfrac{1}{2}\left[1 + \dfrac{1}{M}\sum_{m=1}^{M} \tanh\left(g^m(I - W_O^m)\right)\right]$	
Radial Basis Function (RBF)	$O = \exp(I)$ $I = \left[\dfrac{-\sum_{i=1}^{N}(W_i(t) - X_i(t))^2}{2\sigma^2}\right]$	

ARCHITECTURE:

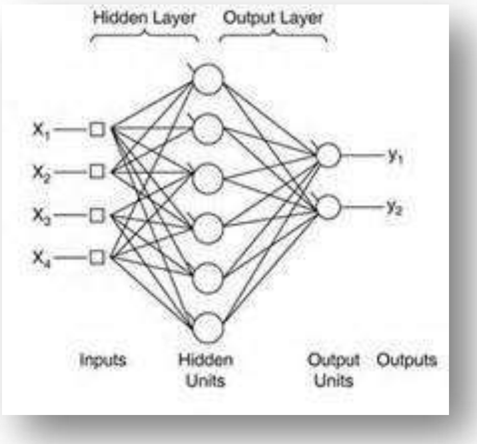


Fig. Architecture of multilayer Neural network

Artificial neural networks are represented by a set of nodes, often arranged in layers, and a set of weighted directed links connecting them. The nodes are equivalent to neurons, while the links denote synapses. The nodes are the information processing units and the links acts as communicating media.

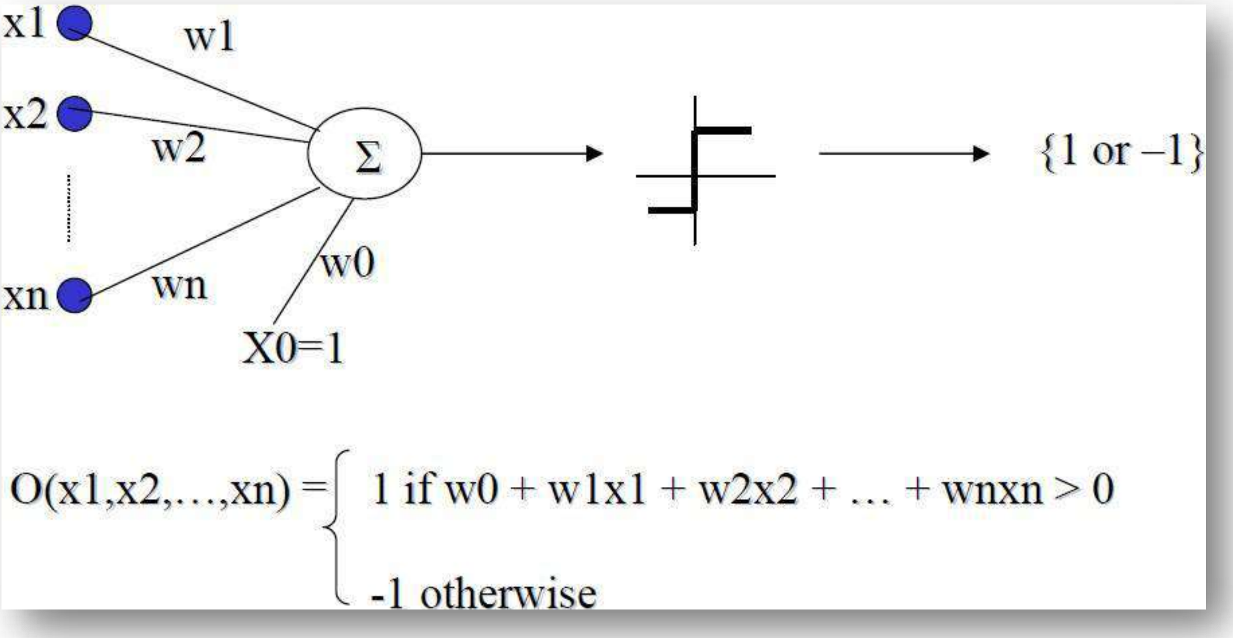
A neural network may have different layers of neurons like

1. Input layer
2. Hidden layer
3. Output layer

The input layer receives input data from the user and propagates a signal to the next layer called the hidden layer. While doing so it multiplies the weight along with the input signal. The hidden layer is a middle layer which lies between the input and the output layers. The hidden layer with non linear activation function increases the ability of the neural network to solve many problems than the case without the hidden layer. The output layer sends its calculated output to the user from which decision can be made. Neural nets can also be classified based on the above stated properties. There are a wide variety of networks depending on the nature of information processing carried out at individual nodes, the topology of the links, and the algorithm for adaptation of link weights.

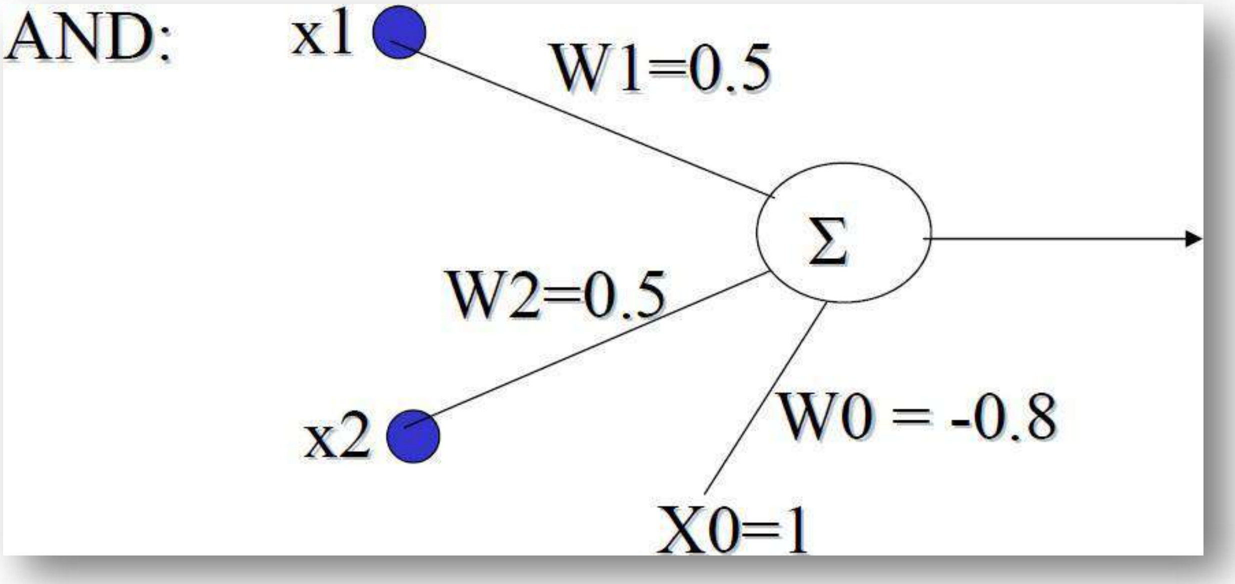
PERCEPTRON:

Definition: It's a step function based on a linear combination of real-valued inputs. If the combination is above a threshold it outputs a 1, otherwise it outputs a –1. This consists of a single neuron with multiple inputs and a single output. It has restricted information processing capability. The information processing is done through a transfer function which is either linear or non-linear.



- A perceptron can learn only examples that are called “linearly separable”. These are examples that can be perfectly separated by a hyperplane.
- Perceptrons can learn many boolean functions: AND, OR, NAND, NOR, but not XOR However, every boolean function can be represented with a perceptron network that has two levels of depth or more.

The weights of a perceptron implementing the AND function is shown below.



MULTI-LAYERED PERCEPTRON (MLP):

It has a layered architecture consisting of input, hidden and output layers. Each layer consists of a number of perceptrons. The output of each layer is transmitted to the input of nodes in other layers through weighted links. Usually, this transmission is done only to nodes of the next layer, leading to what are known as feed forward networks. MLPs were proposed to extend the limited information processing capabilities of simple perceptrons, and are highly versatile in terms of their approximation ability. Training or weight adaptation is done in MLPs using supervised backpropagation learning.

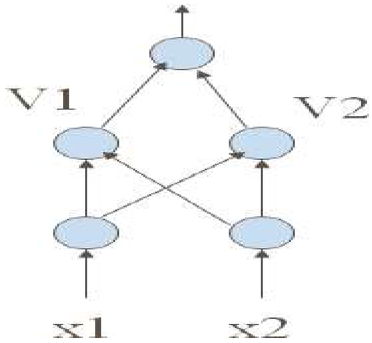
ADDING A HIDDEN LAYER:

The perceptron, which has no hidden layers, can classify only linearly separable patterns. The MLP, with at least 1 hidden layer can classify *any* linearly non-separable classes also.

An MLP can approximate any continuous multivariate function to any degree of accuracy, provided there are sufficiently many hidden neurons (Cybenko, 1988; Hornik et al, 1989). A more precise formulation is given below.

A serious limitation disappears suddenly by adding a single hidden layer.

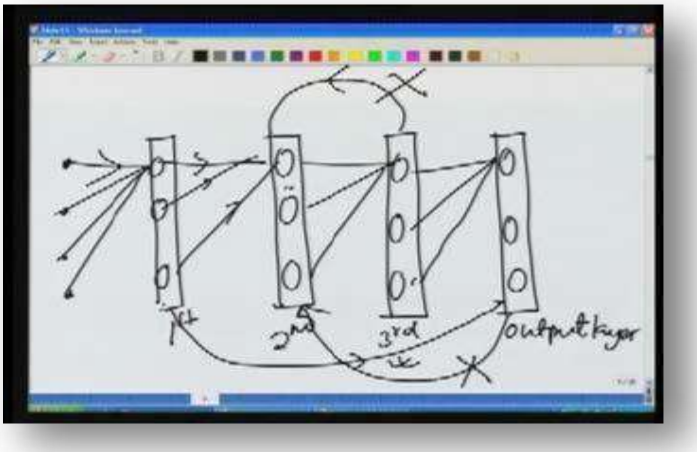
It can easily be shown that the XOR problem which was not solvable by a Perceptron can be solved by a MLP with a single hidden layer containing two neurons.



MLP for solving XOR

RECURRENT NEURAL NETWORKS:

RNN topology involves backward links from output to the input and hidden layers. The notion of time is encoded in the RNN information processing scheme. They are thus used in applications like speech processing where inputs are time sequences data.



SELF-ORGANIZING MAPS:

SOMs or Kohonen networks have a grid topology, wit unequal grid weights. The topology of the grid provides a low dimensional visualization of the data distribution. These are thus used in applications which typically involve organization and human browsing of a large volume of data. Learning is performed using a winner take all strategy in a unsupervised mode. It is described in detail later.

SINGLE LAYER NETWORK:

A neural net with only input layer and output layer is called single layer neural network. A neural network with input layer, one or more hidden layers and an output layer is called a multilayer neural network. A single layer network has limited capabilities when compared to the multilayer neural networks.

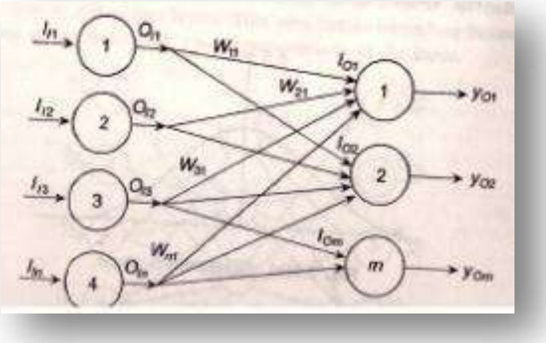


Fig. Single Layer feed forward Neural Network

PERCEPTRON LEARNING

Learning a perceptron means finding the right values for W. The hypothesis space of a perceptron is the space of all weight vectors.

The perceptron learning algorithm can be stated as below.

1. Assign random values to the weight vector
2. Apply the *weight update rule* to every training example
3. Are all training examples correctly classified?
    a. Yes. Quit
    b. No. Go back to Step 2.

There are two popular weight update rules.

a) The perceptron rule, and

b) Delta rule

THE PERCEPTRON RULE

For a new training example X = (x1, x2, …, xn), update each weight according to this rule:

wi = wi + Δwi

Where Δwi = η (t-o) xi t: target output

o: output generated by the perceptron

η: constant called the learning rate (e.g., 0.1) Comments about the perceptron training rule:

Example means training data.

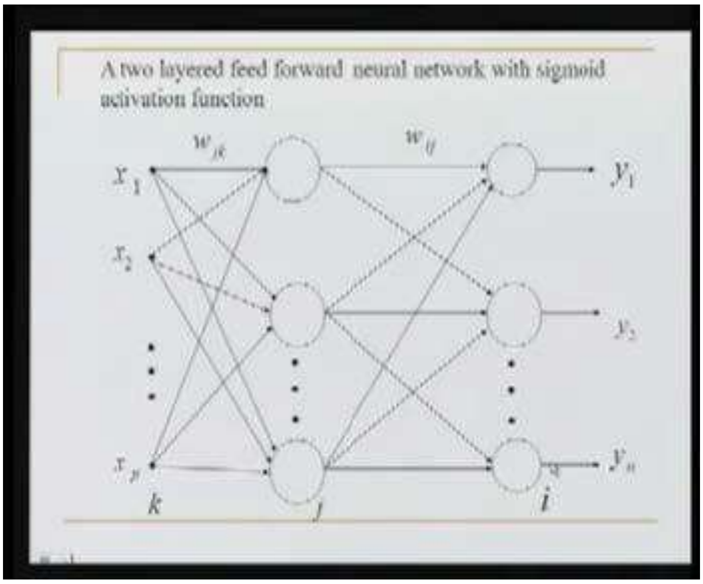
- If the example is correctly classified the term (t-o) equals zero, and no update on the weight is necessary.
- If the perceptron outputs –1 and the real answer is 1, the weight is increased.

- If the perceptron outputs a 1 and the real answer is -1, the weight is decreased.
- Provided the examples are linearly separable and a small value for η is used, the rule is proved to classify all training examples correctly (i.e, is consistent with the training data).

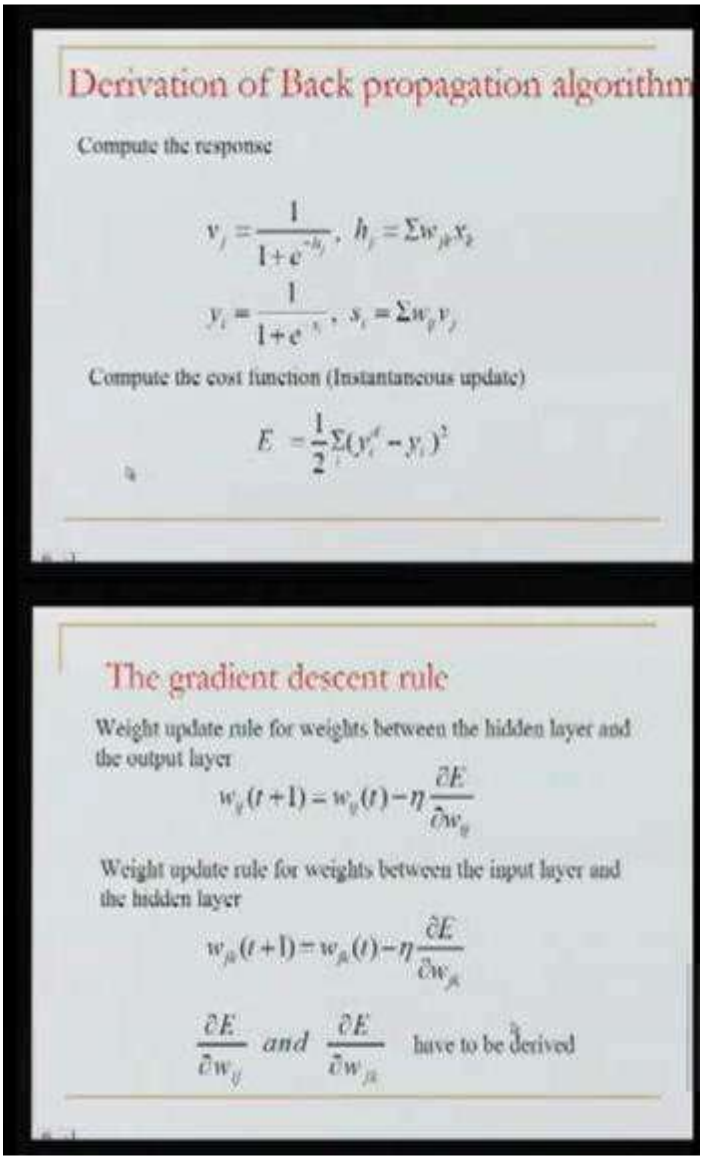
MULTI LAYER NETWORK:

Multilayer feed forward network has more hidden layers and again, when I say feed forward network, the connections are all allowed only from any layer to its succeeding layer, but the connections are not allowed from any layer to its preceding layer. The example is you see here there are four layers. These are all inputs. First hidden layer, second hidden layer, third hidden layer and this is output layer. When we say the number of layers, we do not count the input layer as one of the layers. When I say two layered network, then I have only one hidden layer and next layer becomes output layer.

The algorithm that was derived using gradient descent for nonlinear neural networks with nonlinear activation function is popularly known as back propagation learning algorithm, although the learning algorithm still is derived using gradient descent rule.



A two layered feed forward neural network with sigmoid activation function



Derivation of Back propagation algorithm

Compute the response

$$v_j = \frac{1}{1+e^{-h_j}}, \quad h_j = \Sigma w_{jk} x_k$$

$$y_i = \frac{1}{1+e^{-s_i}}, \quad s_i = \Sigma w_{ij} v_j$$

Compute the cost function (Instantaneous update)

$$E = \frac{1}{2} \sum_i (y_i^d - y_i)^2$$

The gradient descent rule

Weight update rule for weights between the hidden layer and the output layer

$$w_{ij}(t+1) = w_{ij}(t) - \eta \frac{\partial E}{\partial w_{ij}}$$

Weight update rule for weights between the input layer and the hidden layer

$$w_{jk}(t+1) = w_{jk}(t) - \eta \frac{\partial E}{\partial w_{jk}}$$

$\frac{\partial E}{\partial w_{ij}}$ *and* $\frac{\partial E}{\partial w_{jk}}$ have to be derived

After choosing the weights of the network randomly, the back propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

i) Feed-forward computation
ii) Back propagation to the output layer
iii) Back propagation to the hidden layer
iv) Weight updates

The algorithm is stopped when the value of the error function has become sufficiently small.

In the case of p > 1 an input-output pattern, an extended network is used to compute the error function for each of them separately. The weight corrections The Back propagation Algorithm are computed for each pattern and so we get, for example, for weight w(1)ij the corrections

$$\Delta_1 w_{ij}^{(1)}, \Delta_2 w_{ij}^{(1)}, \ldots, \Delta_p w_{ij}^{(1)}$$

The necessary update in the gradient direction is then

$$\Delta w_{ij}^{(1)} = \Delta_1 w_{ij}^{(1)} + \Delta_2 w_{ij}^{(1)} + \cdots + \Delta_p w_{ij}^{(1)}$$

Back Propagation Neural Network

Backpropagation is a training method used for a multi layer neural network. It is also called the generalized delta rule. It is a gradient descent method which minimizes the total squared error of the output computed by the net. Any neural network is expected to respond correctly to the input patterns that are used for training which is termed as memorization and it should respond reasonably to input that is similar to but not the same as the samples used for training which is called generalization.

The training of a neural network by back propagation takes place in three stages

1. Feed forward of the input pattern
2. Calculation and Back propagation of the associated error
3. Adjustments of the weights

After the neural network is trained, the neural network has to compute the feed forward phase only. Even if the training is slow, the trained net can produce its output immediately.

ARCHITECTURE

A multi layer neural network with one layer of hidden units is shown in the figure. The output units and the hidden units can have biases. These bias terms are like weights on connections from units whose output is always 1. During feed forward the signals flow in the forward direction i.e. from input unit to hidden unit and finally to the output unit. During back propagation phase of learning, the signals flow in the reverse direction.

ALGORITHM

The training involves three stages

1. Feed forward of the input training pattern
2. Back propagation of the associated error
3. Adjustments of the weights.

During feed forward, each input unit (Xi) receives an input signal and sends this signal to each of the hidden units Z1, Z2, …Zn. Each hidden unit computes its activation and sends its signal to each output unit. Each output unit computes its activation to compute the output or the response of the neural net for the given input pattern.

During training, each output unit compares its computed activation yk, with its target value tk to determine the associated error for the particular pattern. Based on this error the factor ∂k for all m values are computed. This computed ∂k is used to propagate the error at the output unit Yk back to all units in the hidden layer. At a later stage it is also used for updation of weights between the output and the hidden layer. In the same way ∂j for all p values are computed for each hidden unit Zj. The values of ∂j are not sent back to the input units but are used to update the weights between the hidden layer and the input layer. Once all the ∂ factrs are known, the weights for all layers are changed simultaneously. The adjustment to all weights wjk is based on the factor ∂k and the activation zj of the hidden unit Zj. The change in weight to the connection between the input layer and the hidden layer is based on ∂j and the activation xi of the input unit.

ACTIVATION FUNCTION

An activation function for a back propagation net should have important characteristics. It should be continuous, Differentiable and monotonically non- decreasing. For computational efficiency, it is better if the derivative is easy to calculate. For the commonly used activation function, the derivative can be expressed in terms of the value of the function itself. The function is expected to saturate asymptotically. The commonly used activation function is the binary sigmoidal function.

TRAINING ALGORITHM

The activation function used for a back propagation neural network can be either a bipolar sigmoid or a binary sigmoid. The form of data plays an important role in choosing the type of the activation function. Because of the relationship between the value of the function and its derivative, additional evaluations of exponential functions are not required to be computed.

ALGORITHM

- Step 0: Initialize weights
- Step 1: While stopping condition is a false, do step 2 to 9
- Step 2: For each training pair, do steps 3 - 8 *feed forward*
- Step 3: Input unit receives input signal and propagates it to all units in the hidden layer
- Step 4: Each hidden unit sums its weighted input signals
- Step 5: Each output unit sums its weighted input signals and applied its activation function to compute its output signal.

BACKPROPAGATION

- Step 6: Each output unit receives a target pattern corresponding to the input training pattern, computes its error information term δk = (tk – yk) f‟ (y_ink) Calculates its bias correction term ΔWok = αδk And sends δk to units in the layer below
- Step 7: Each hidden unit sums its delta inputs Multiplies by the derivative of its activation function to calculate its error information term Calculates its weight correction term Δvij = αδjxi And calculates its bias correction term Δvoj = αδj *Update weights and biases*
- Step 8: Each output unit updates its bias and weights Wjk(new) = wjk(old) + Δ wjk Each hidden unit updates its bias and weights Vij (new) = vij (old) + Δvij

- Step9: Test stopping condition.

Radial Basis Function Networks:

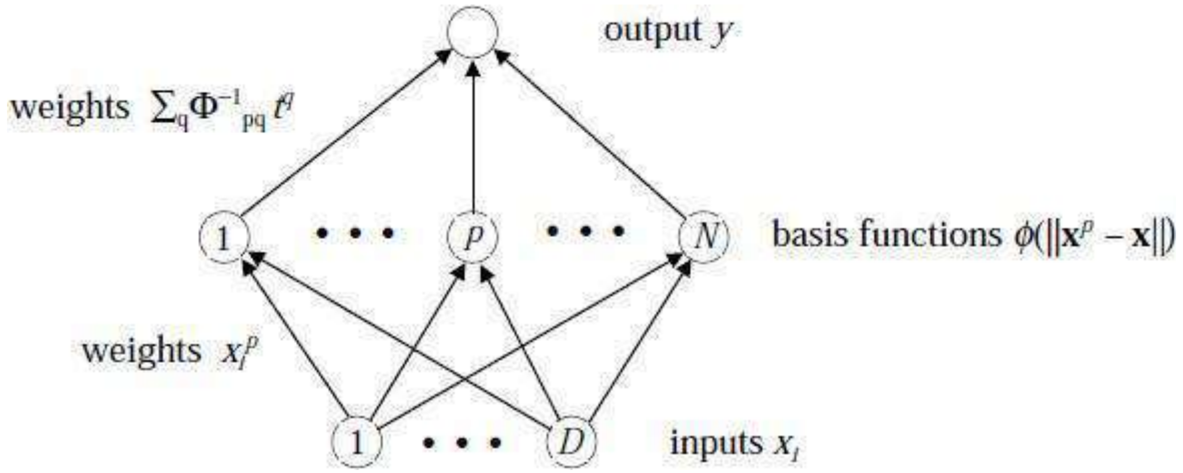

output y
weights $\sum_q \Phi^{-1}_{pq} t^q$
1
p
N
basis functions $\phi(\|\mathbf{x}^p - \mathbf{x}\|)$
weights x_i^p
1
D
inputs x_i

Fig. RBF network

These are 3-layer networks that can approximate any continuous function through a basis function expansion.

• The basis functions here (which are data dependent as earlier) exhibit some radial symmetry.

• These networks have the so called perfect interpolation property.

The function represented by an RBF network with p hidden nodes can be written as

$$y = \sum_{j=1}^{p} w_j \, \phi\left(||X - \theta_j||\right) ,$$

Training ANNs

Training or Learning of Artificial Neural Network can be broadly classified into 3 types such as

1. Supervised Learning
2. Unsupervised Learning
3. Reinforced Learning

-: LEARNING CLASSIFICATIONS:-

SUPERVISED LEARNING:-

Supervised Learning method can be classified as follows.

- Error Correction Gradient Descent Learning
  - Least Mean Square Learning
  - Back Propagation Learning
- Stochastic Learning

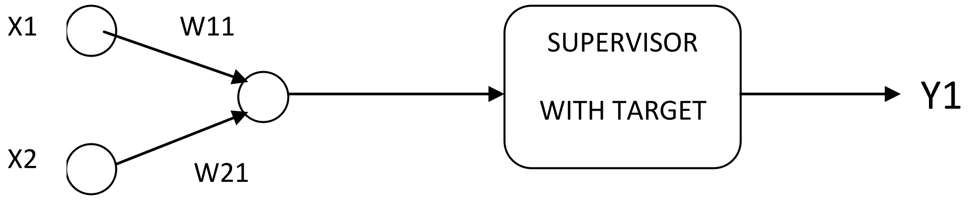
UNSUPERVISED LEARNING:-

Unsupervised Learning method can be classified as follows.

- Hebbian Learning
- Competitive Learning

DESCRIPTION ABOUT SUPERVISED LEARNINGS:-

- In supervised learning method, the inputs are executed to produce a particular or desired output in presence of a supervisor or a teacher.
- In this method of learning, a supervisor or a teacher is present to supervise the training process.
- The supervisor is having the desired output.
- After calculation of output, the calculated output is then compared with the desired output which is there with the supervisor. Then error (if any) is calculated.
- If any error is present, then the weights or information are adjusted till the error is zero or eliminated.
- In general, the mathematical expression regarding weight adjustment can be represented as $(W)^{New} = (W)^{Old} + \Delta W$
  Here ΔW is the weight adjustment required to eliminate the error.



DESCRIPTION ABOUT UNSUPERVISED LEARNINGS:-

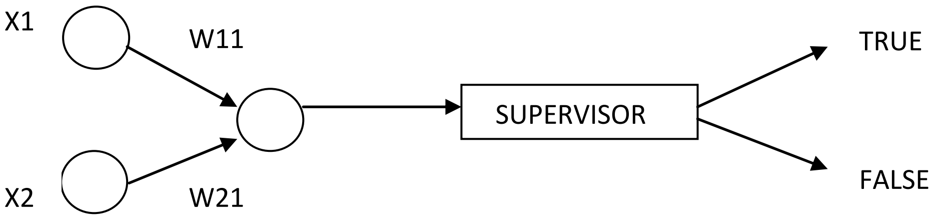
- In unsupervised learning method, there is no supervisor or teacher present to assess the output.
- It is the total network itself which executes the output by observing the surrounding and past experience.
- The weight adjustment can be done by forming a correlation matrix by considering respective inputs and transpose of output.
- This can be mathematically explained as follows

$$\mathbf{W = SUMMATION\ (X_I \times Y_J^T)}$$

J = JTH OUTPUT NEURON
I = ITH INPUT NEURON

DESCRIPTION ABOUT REINFORCED LEARNINGS:-

- In reinforced learning, one supervisor is present only to indicate whether the computed output is correct or wrong.
- For the correct output, the reward is present in terms of allowing the respective input neuron to execute its corresponding output first.
- For wrong output, the punishment is there to give penalty to update its weight first randomly until and unless the actual output is computed.



DESCRIPTION ABOUT HEBBIAN LEARNING:-

- This type unsupervised learning was proposed by the scientist named Hebb in the year 1949.
- As per this learning rule, the weight adjustment can be done by the formation of the correlation weight matrix by considering respective inputs and transpose of output.
- This can be mathematically explained as follows

$$\mathbf{W = SUMMATION\ (X_I \times Y_J^T)}$$

J = JTH OUTPUT NEURON
I = ITH INPUT NEURON

DESCRIPTION ABOUT COMPETITIVE LEARNING:-

- In this method, those neurons which respond strongly to input stimuli have their weights updated.
- When an input pattern is presented, all neurons in the layer compete and the winning neuron undergoes weight adjustment. Hence it is also called as “winner-takes-all” strategy.

DESCRIPTION ABOUT STOCHASTIC LEARNING:-

- In this method, weights are adjusted in a probabilistic fashion. An example is evident in simulated annealing that describes the learning mechanism employed by Bolzmann and Cauchy machines, which are a kind of neural network systems.

DESCRIPTION ABOUT GRADIENT DESCENT LEARNING:-

- This method is based upon the minimization of error E defined in terms of weights and the activation function of the network. Also, it is required that the activation function employed by the network is differentiable, as the weight update is dependent on the gradient of the error E.
- Thus if ΔW_{ij} is the weight update of the link connecting the ith and jth neuron of the two neighbouring layers, then $\Delta W_{ij} = \eta \frac{dE}{dWij}$
- Here η is called as learning rate parameter and $\frac{dE}{dWij}$ is called as error gradient with reference to the weight W_{ij}

EARLY NEURAL NETWORK ARCHITECTURES

ALGORITHM FOR ROSENBLATT’S PERCEPTRON MODEL

Fixed increment perceptron learning algorithm for classification problem with n input features (x1, x2, ..., xn) and two output classes (0 or 1)

Algorithm fixed increment perceptron learning ($\overline{Xj}, \overline{Yj}, \overline{W}$)

```
Step 1: Create a perceptron with (n+1) input neurons $X_0, X_1, X_2, \ldots . X_n$ where $X_0 = 1$ is the bias input. Let O be the output neuron.

Step 2: Initialize $\overline{W}$ = ( W0, W1, W2,.....Wn) to random weights.

Step 3: Iterate through the input patterns $\overline{Xj}$ of the training set using the weight set, i.e. compute the weighted sum of inputs $net_j = \sum_{i=0}^{n} Xi\, Wi$ for each input pattern j.
```

Step 4: Compute the output Yj by using the step function.

$Yj = F(net_j) = 1,\ net_j > 0$

$= 0$, Otherwise.

Step 5: Compare the computed output Yj with the target output for each input pattern j. If all the input patterns have been classified correctly, output the weights and exit.

Step 6: Otherwise, update the weights as given below:

If the computed output Yj is 1 but should have been 0,

Then $W_N = W_O - \alpha X_i$ Where i= 0,1,2,...n

If the computed output Yj is 0 but should have been 1,

Then $W_N = W_O + \alpha X_i$ Where i= 0,1,2,...n

Here α is the learning rate parameter and is a constant.

Step 7: Go to step 3.

End of the algorithm.

ADALINE Network

- The Adaptive Linear Neural Element Network framed by Bernard Widrow.
- It is a single layer supervised neural network model.
- Here, there is only one output neuron.
- The output values are bipolar (-1 or +1).
- The input values Xi may be binary, bipolar or real valued.
- The bias weight is W0 with an input link X0 = +1
- If the weighted sum of the inputs is greater than or equal to 0 then the output is 1 otherwise it is -1.
- The supervised learning algorithm adopted by the network is similar to perceptron learning algorithm and it is based upon LMS algorithm or Delta Rule.
- The rule is given by $W_{NEW} = W_{OLD} + \alpha\ (T-O)X_i$
  Here α is called as the learning coefficient, T is called as Target Output, O is called as computed output and X_i is the input
- ADALINE network is applied virtually in high speed modems, telephonic switching systems to cancel the echo in long distance communication circuits.

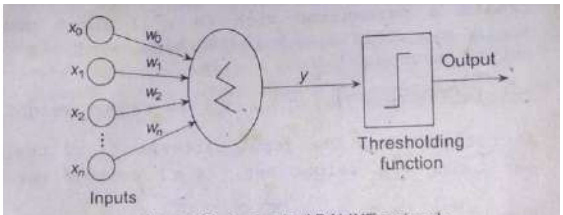


Fig. 2.19 A simple ADALINE network.

MADALINE Network

- A "MADALINE" network is called as Many ADALINE network.
- It is a combination of a number of ADALINES. The network of ADALINE can span many layers as shown in the figure below.

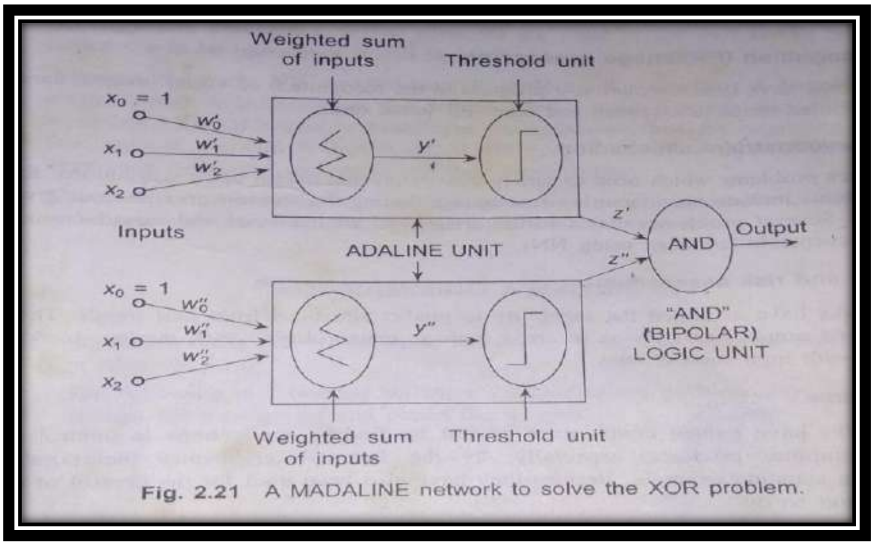


Fig. 2.21 A MADALINE network to solve the XOR problem.

- The use of multiple ADALINE helps counter the problem of non linear reparability. For example, the MADALINE network with two units exhibits the capability to solve the XOR problem.
- In this, each ADALINE unit receives the input bits X_1, X_2 and the bias input X_0 =1 as its inputs.
- The weighted sum of the inputs is calculated and passed onto the bipolar threshold units.
- The logical "and" ing (bipolar) of two threshold outputs are computed to obtain the final output.
- Here, if the threshold outputs are both +1 and -1 then the final output is +1.
- If the threshold outputs are different, (i.e.) (+ 1 ,-1) then the final output is -1.
- Inputs which are of even parity produce positive output.
- Inputs which are of odd parity produce negative output.

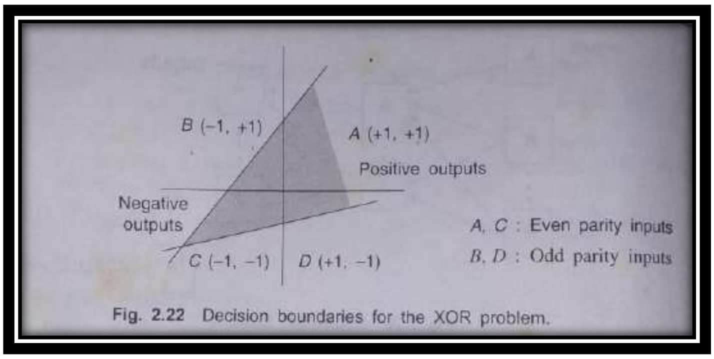


Fig. 2.22 Decision boundaries for the XOR problem.

APPLICATION OF NEURAL NETWORK

ANN can be applied in the following application domains

- Pattern recognition (PR)/ Image Processing :- Neural Network have shown remarkable process in the recognition of visual images, handwritten characters, printed characters, speech and other PR based tasks.
- Optimization/Constraint satisfaction: - This comprises problems which need to satisfy constraints and obtain optimal solutions.
- Forecasting and risk assessment: - Neural Networks have exhibited the capability to predict situations from the past trends. They have therefore more applications in areas such as meteorology, stock market, banking, econometrics with high success rates.
- Control systems:- Neural Network have gained commercial ground by finding applications in control systems in chemical plants, robots etc.

GENERALIZED DELTA RULE

- In error correction gradient descent supervised learning method; the error is calculated by considering the targeted output and the computed output.
- By considering the error the weights are adjusted and it is done by means of Generalized Delta Rule.
- The main objective of this rule is to minimize the error by expressing the difference of the computed output and the target output in terms of input and weight matrix.

Let T = Target Output

O = Computed Output

ϵ = Error

W_i = Weight of the i_{th} input neuron

Least square error = E = ½ [Target output – Computed output]^2.....................(1)

$E = ½ [T-O]^2 = ½ \ [T – F(W_i.X_i)]^2$...(2)

But; $O = F (X_i.W_i)$

By taking gradient descent or partial derivative of equation (2) with respect to W_i, we

get $\nabla E = ½ \times 2 \ [T – F(W_i.X_i)] \ . \ [– F'(W_i.X_i)X_i]$

$\nabla E = - [T – F(W_i.X_i)] \ . \ [F'(W_i.X_i) \ X_i]$...(3)

For minimization of error, the updated weight is calculated as;

$\Delta W_i = - \mu \nabla E$...(4)

Where µ is a positive constant and –ve sign indicate the reduction of error.

By putting the value of equation (3) in equation (4), we get

$\Delta W_i = \mu\ [T – F(W_i.X_i)]\ .\ [\ F'(W_i.X_i)\ X_i]$..(5)

By using discrete mathematics the new weight can be calculated as

$W_{NEW} = W_{OLD} + \Delta W$..(6)

Putting the value of ΔW from equation (5) in equation (6)

$W_{NEW} = W_{OLD} + \mu\ [T – F(W_i.X_i)]\ .\ [\ F'(W_i.X_i)\ X_i]$..................................(7)

For a sigmoidal non linear function

$F'(W_i.X_i) = ½\ [\ T – O^2]$

So; $W_{NEW} = W_{OLD} + \mu\ [T – F(W_i.X_i)].\ ½\ [\ T – O^2]\ X_i$

$W_{NEW} = W_{OLD} + \mu/2\ \{[T – O].\ [\ T – O^2]\ X_i\}$

So the Generalised Delta Rule can be written as

$W_{NEW} = W_{OLD} + \alpha\ [T – O].\ X_i$

Here α is called as learning rate parameter

T = Target Output

O = Computed Output

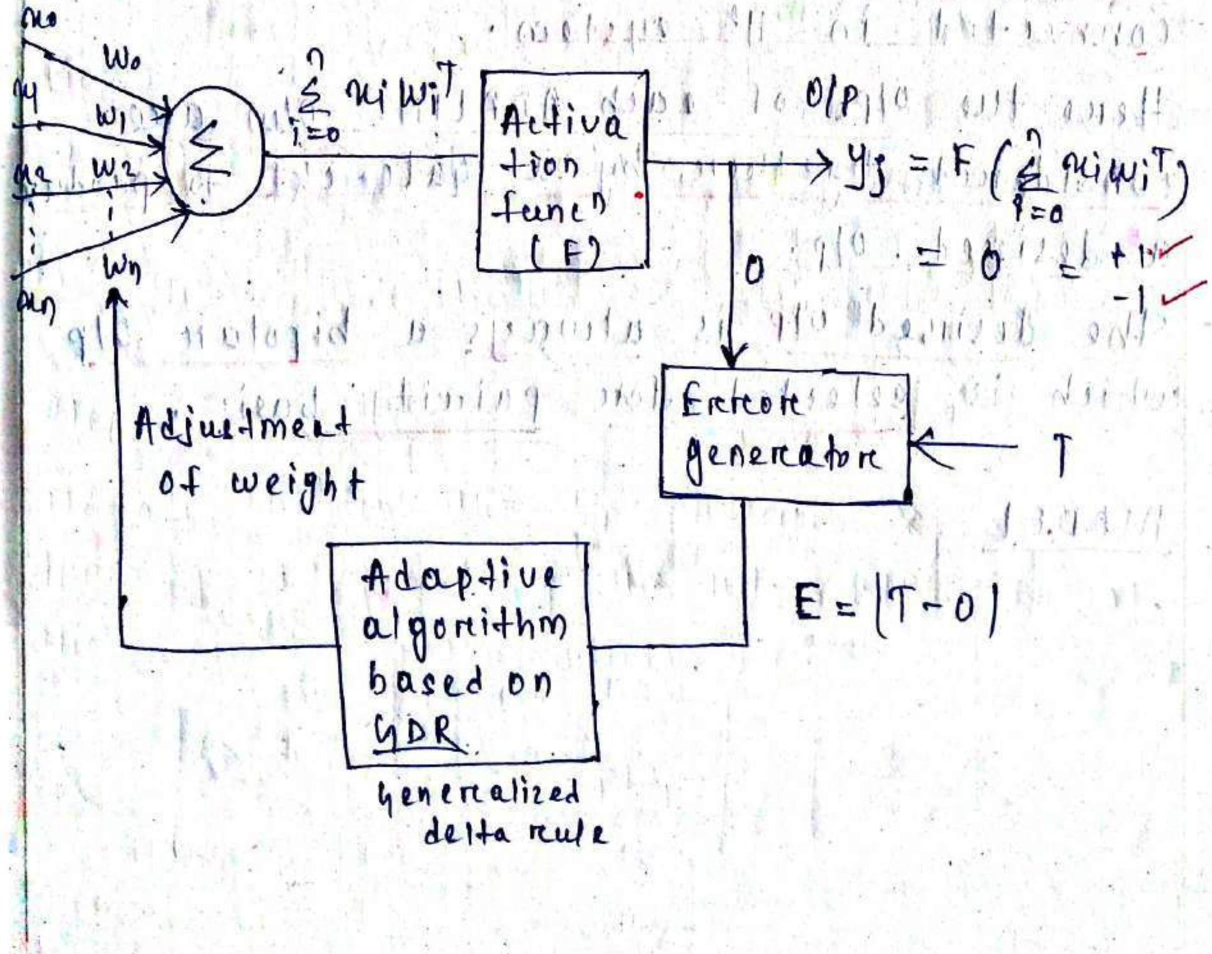
X_i = Input Neuron whose weight is to be adjusted.

Dt - 21/09/19

Adaptive linear Neural Network Element. (ADALINE)

- It was developed by the scientist name Bernand Widraw & it was based on supervised learning methods.
- This supervised learning method adopt a generalized delta rule.
- This algorithm is usedful to update the weight or informations & then to produce the desired o/p.
- In this n/w, there is one o/p neuron present. & its value is always bi-polar in nature (+1/-1)
- In this n/w, the I/p x_i can be either bi-polar or binary or any numerical valuep. but o/p is always bi-polar.

MODEL



O/p Calculation

- The o/p can be calculated as

$$O = O/p = y_j = \begin{cases} +1 , & \sum_{i=1}^{n} x_i w_i^T > 0 \\ -1 , & \text{Otherwise} \end{cases}$$

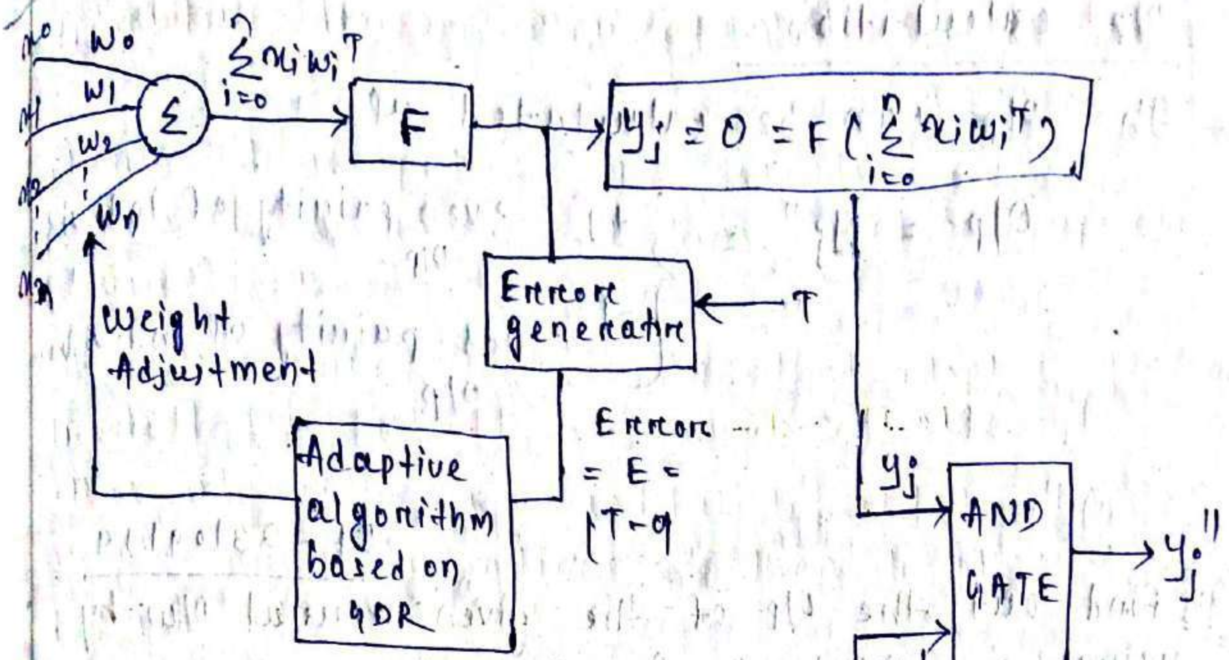
Application

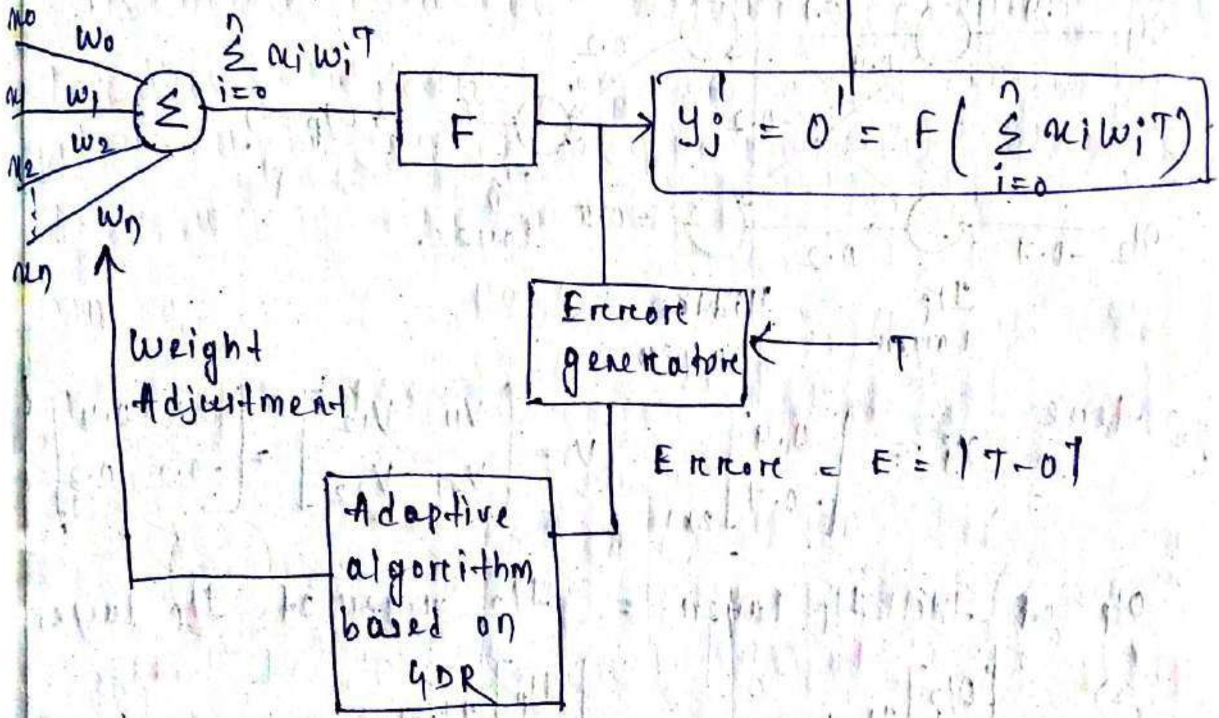
- The ADALINE n/w is having an adaptive algorithm therefore it is very much useful in high speed MODEMS & $||^{le}$ telephonic switching system.

MADALINE (Many Adaptive Lineare neural n/w element)

- In this complecated n/w more than one ADALINE n/ws are present which are connected in $||^{le}$ system.
- Here the o/p's of each ADALINE n/w are considered together by a gate-ckt to produce a desired o/p.
- The desired o/p is always a bipolar o/p which is selected for pairity basis.

MODEL





$y_j'' =$

y_j	y_j'	y_j''	Remark
+1	+1	+1	Even Pairity
+1	-1	-1	Odd Pairity
-1	+1	-1	Odd pairity
-1	-1	+1	Even pairity

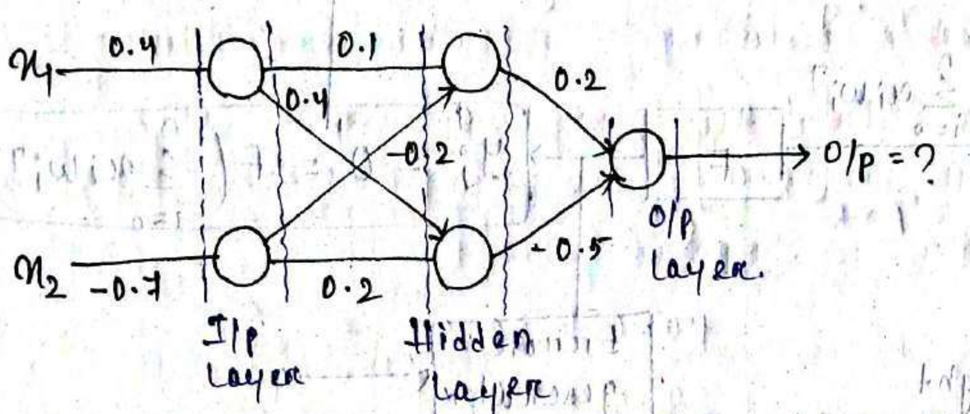
O/P calculation

- The O/p can be calulated as

$$O/p = y_j'' = \begin{cases} +1, & \text{even pairity of Individual O/p} \\ -1, & \text{Odd pairity of Individual O/p} \end{cases}$$

Dt - 23/09/19

Q
Find out the O/p of the given neural n/w by using sigmoidal func^n, at $\lambda = 1$



Sol^n Here $x_i = \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}_{2\times1}$ $V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix}_{2\times2}$

O/p of the I/p layer = I/p ~~layer~~ of I/p layer

$$\Rightarrow [O/p]_{I/p\ Layer} = [I/p]_{I/p\ Layer}$$

$$\Rightarrow [O/p]_{I/p\ layer} = X = \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}_{2\times1}$$

$$[I/p]_{\substack{Hidden \\ Layer}} = [V]^T [O/p]_{I/p\ Layer}$$

$$= \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix}^T_{2\times2} \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix}_{2\times2} \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}_{2\times1} = \begin{bmatrix} 0.04 + 0.14 \\ 0.16 - 0.14 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.18 \\ 0.02 \end{bmatrix}_{2\times1}$$

$$O/P = \frac{1}{1+e^{-\lambda(I/P)}}$$ (Bcz here sigmoidal func^n is given)

In hidden layer we have a sigmoidal func^n

Therefore

$$[O/P]_{\text{Hidden layer}} = \begin{bmatrix} \frac{1}{1+e^{-\lambda(0.18)}} \\ \frac{1}{1+e^{-\lambda(0.02)}} \end{bmatrix}_{2\times1} = \begin{bmatrix} 0.544 \\ 0.504 \end{bmatrix}_{2\times1}$$

Here

$$W = \begin{bmatrix} 0.2 \\ -0.5 \end{bmatrix}_{2\times1}$$

$$[I/P]_{O/P\ \text{layer}} = W^T [O/P]_{\text{hidden layer}}$$

$$= \begin{bmatrix} 0.2 & -0.5 \end{bmatrix}_{1\times2} \begin{bmatrix} 0.544 \\ 0.504 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} -0.143 \end{bmatrix}_{1\times1}$$

The $[O/P]_{O/P\ \text{layer}}$ can be calculated by using sigmoidal func^n as

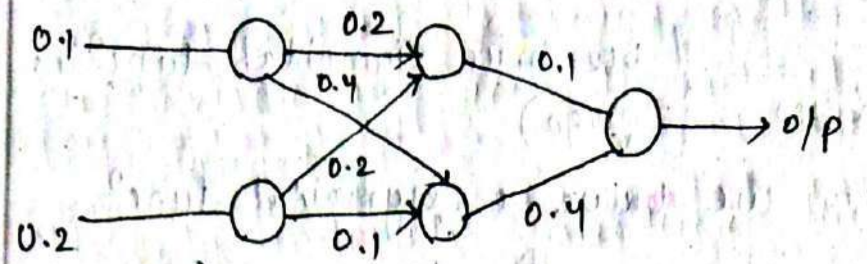
$$[O/P]_{I/P\ \text{layer}} = \frac{1}{1+e^{-\lambda(-0.143)}}$$

$$= \frac{1}{1+e^{0.143}}$$

$$= 0.464$$

cont.

2) Find out the O/P by using sigmoidal function.



$x_i = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$

$V = \begin{bmatrix} 0.2 & 0.4 \\ 0.2 & 0.1 \end{bmatrix}$

$W = \begin{bmatrix} 0.1 \\ 0.4 \end{bmatrix}$

Sol O/P of I/P layer = I/P of I/P layer

$$[O/P]_{I/p} = [I/P]_{I/p} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$$

$$[I/P]_{Hidden} = [V^T] \cdot [O/P]_{I/p}$$

$$= \begin{bmatrix} 0.2 & 0.2 \\ 0.4 & 0.1 \end{bmatrix}_{2\times2} \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.02 + 0.04 \\ 0.04 + 0.02 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.06 \\ 0.06 \end{bmatrix}_{2\times1}$$

[O/P of Hidden ~~o/p~~ layer] = W^T [I/P of hidden layer]

[O/P]

$$= \begin{bmatrix} 0.1 & 0.4 \end{bmatrix}_{1\times2} \begin{bmatrix} 0.06 \\ 0.06 \end{bmatrix}_{2\times1}$$

$$= [0.006 + 0.024]$$

$$= [0.03]$$

$$[O/P]_{Hidden} = \begin{bmatrix} \frac{1}{1+e^{-\lambda(0.06)}} \\ \frac{1}{1+e^{-\lambda(0.06}} \end{bmatrix} = \begin{bmatrix} 0.514 \\ 0.514 \end{bmatrix}$$

$$[I/P]_{O/P} = W^T \times [O/P]_{Hidden}.$$

$$= \begin{bmatrix} 0.1 & 0.4 \end{bmatrix}_{1\times2} \times \begin{bmatrix} 0.514 \\ 0.514 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.0514 + 0.2056 \end{bmatrix}$$

$$= \begin{bmatrix} 0.257 \end{bmatrix}_{1\times1}$$

$$[O/P]_{O/P} = \frac{1}{1+e^{-\lambda(0.257)}}$$

$$= \frac{1}{1+e^{-0.257}}$$

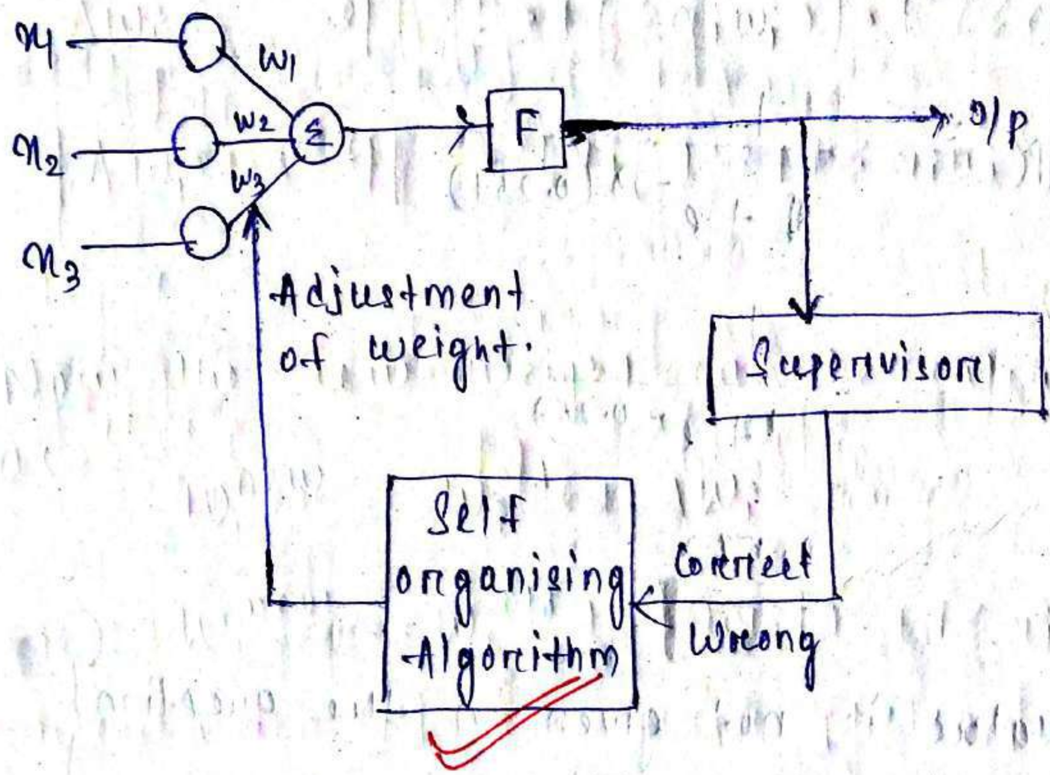
$$= \underline{0.563}$$

[If λ value is not given in the question take it 1]

cont. →

Reinforced learning method.

- In reinforced learning method, though a supervisor is present but it does not have any target.
- It only indicate whether the computed o/p is right or wrong.
- For right o/p, there is a reward & for wrong o/p, there is a penalty to again rearrange the weight till we get the desired o/p.
- In this case a self organising algorithm.



Dt. 24/09/19

Generalized delta function

In an error connection gradient descent method, the error is calculated by considering the target o/p & calculated o/p.

By considering the error, the wt. are adjusted & it is done by means of generalized delta rule (GDR)

Delta rule

The main objective of this rule is to minimize the error by expressing the difference of computed o/p & the target o/p in terms of I/p & weights.

let T ⟷ Target o/p
O ⟷ Computed o/p
E ⟶ Error
w_i ⟷ weight of ith neuron

Now the least mean square error can be calculated as

$$E = \frac{1}{2}(\text{Target} - \text{computed o/p})^2$$

$$\boxed{E = \frac{1}{2}(T-O)^2}$$

$$\Rightarrow \boxed{E = \frac{1}{2}\left[T - F\left(\sum x_i w_i\right)\right]^2} \longrightarrow ①$$

Here $o/p = O = F\left(\sum x_i w_i\right)$

- By considering the derivative of error w.r.to weight (w_i) we get,

$$\frac{dE}{dw_i} = \nabla E = \frac{1}{2} \times 2\left[T - F\left(\sum x_i w_i\right)\right].$$

$$O - F'\left(\sum x_i w_i\right).x_i$$

$$\boxed{\nabla E = -\left[T - F\left(\textstyle\sum x_i w_i\right)\right] \cdot F'\left(\textstyle\sum x_i w_i\right) \cdot x_i} \quad \text{(2)}$$

- For minimization of error the updated weight can be represented as

$$\boxed{\Delta W_i = -\eta . \nabla E} \longrightarrow \text{(3)}$$

- Here η is a +ve constant & -ve sign indicates the red^n of weight

- Now putting the value of ∇E from eq^n(2) to eq^n(3) we have,

$$\Delta W_i = -\eta\left[-\left[T - F\left(\textstyle\sum x_i w_i\right)\right] \cdot F'\left(\textstyle\sum x_i w_i\right) . x_i\right]$$

$$\boxed{\Delta W_i = \eta x_i \left[T - F\left(\textstyle\sum x_i w_i\right)\right] . F'\left(\textstyle\sum x_i w_i\right.} \quad \text{(4)}$$

- Now the new weight can be calculated as

$$W_i^{new} = W_i^{old} + \Delta W_i$$

$$\Rightarrow W_i^{new} = W_i^{old} + \left[\eta x_i \left[T - F\left(\textstyle\sum x_i w_i\right)\right] . F'\left(\textstyle\sum x_i w_i\right)\right]$$

$$\Rightarrow \boxed{W_i^{new} = W_i^{old} + \eta x_i \left[(T - O) \cdot \frac{dO}{dw_i}\right]} \rightarrow \text{GDR}$$

* If the function considered is a sigmoidal func^n, then the resultant generalized delta rule can be calculated as

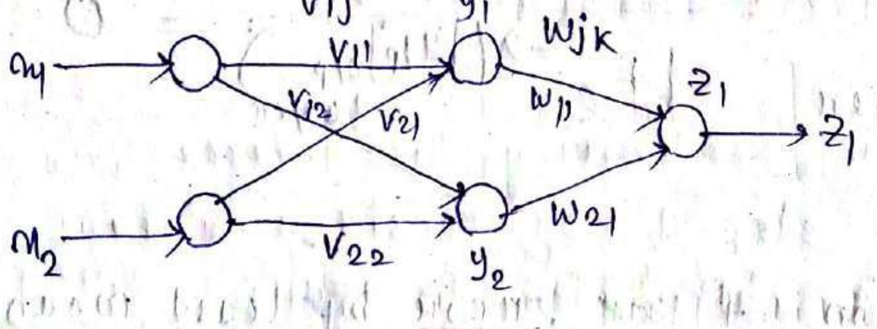
$$\boxed{w_i^{new} = w_i^{old} + \frac{\eta}{2}[T-O][T-O^2]\cdot x_i}$$

where $F'(\Sigma x_i w_i) = \frac{1}{2}[T-O^2]$

imp

Back Propagation learning Algorithm (BPLA)

- For a single layer feed forward n/w, it is very much easier to train the total n/w by means of different supervised learning methods.
- But if the n/w is a multi-layer neural n/w, then it is very much difficult to learn the n/w by means of calculation of error & wt. adjustment
- Therefore, a new algorithm is developed called as Back Propagation learning algorithm, which is used for error correction & wt. adjustment by using generalized delta rule.



Algorithm

Step-1

Initialization of I/p & wt. matrix as $[I]^0$, $[V]^0$ & $[W]^0$.

Step-2

Calculation of O/p of I/p layer. $[O/p]_{I/p\ layer}$

$$[O/p]_{I/p\ layer} = [I/p]_{I/p\ layer}.$$

Step-3

$$[I/P]_{\text{Hidden Layer}} = [V]^T \cdot [O/P]_{\text{I/P Layer}}.$$

Step-4

$$[O/P]_{\text{Hidden Layer}} = \frac{1}{1+e^{-\lambda\left([I/P]_{\text{Hidden Layer}}\right)}}$$

$\lambda \rightarrow 0$ to 1

→ λ is called as sigmoidal gain or learning rate parameter.

Step-5

$$[I/P]_{\text{O/P Layer}} = [W]^T \cdot [O/P]_{\text{Hidden Layer}}.$$

Step-6

$$[O/P]_{\text{O/P Layer}} = \frac{1}{1+e^{-\lambda\left([I/P]_{\text{O/P Layer}}\right)}} = O$$

Step-7

calculation of ~~err~~ error by least mean square method.

$$\text{error} = E = \sqrt{\frac{\Sigma(T-O)^2}{n}}$$ more than one value

T → Target

O → computed O/P

n → no. of O/P neuron

Step-8

calculation of gradient descent or derivative of error $d = [(T-O)\,O\,(1-O)]$

$$d = \left[\left(T - [O]_{O/P}\right) [O]_{\substack{O/P \\ layer}} \left(1 - [O]_{\substack{O/P \\ layer}}\right)\right]$$

Step-9

calculation of new o/p of Hidden Layer by using old o/p Hidden layer & derivative error (d)

$$\boxed{Y = [O/P]_{\substack{Hidden \\ Layer}} \cdot d}$$

Dt- 27/09/19.

Step-10

Calculation of updated weight from Hidden to o/p Layer

$$\boxed{[\Delta W_{JK}] = \alpha \cdot [W_{JK}]^{0} + \eta Y}$$

where α is called as Momentum co-efficient.

η is called as learning rate co-efficient

Step-11

Calculation of new error at Hidden Layer o/p i.e.

$$\boxed{e = [W_{JK}]^{0} \cdot d}$$

Step-12

Partial derivative of error calculation at hidden Layer i.e.

$$d^{*} = \left[e\ [O/P]_{\substack{Hidden \\ Layer}}\ 1 - [O]_{\substack{Hidden \\ Layer}}\right]_{2\times 1}$$

Step-13

Calculation of new O/p of I/p layer

$$\boxed{X = [O/P]_{\text{I/p Layer}} \cdot d^{*}}$$

Step-14

calculation of updated weight from I/p to hidden layer i.e.

$$\boxed{\Delta V_{ij} = \alpha [V_{ij}]^{0} + \eta X}$$

Step-15

Now calculation of new weights

$$\boxed{\begin{aligned} [V_{ij}]^{1} &= [V_{ij}]^{0} + [\Delta V_{ij}] \\ [W_{jk}]^{1} &= [W_{jk}]^{0} + [\Delta W_{jk}] \end{aligned}}$$

Step-16

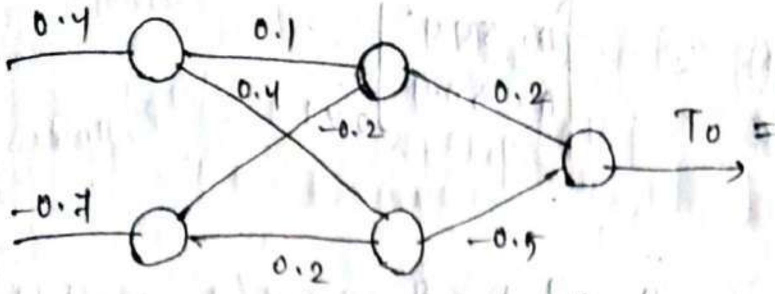
Now again continuing the process from step-3 to step-7 by considering new weights to calculated the error.

Step-17

The iterative process will continue to till the stopping cond^n arrise.

Stopping cond^n → { Target = O/p
Error ~~is less~~ ≤ tolerance value

6Q



$\eta = 0.6$

$\alpha = 0$

find out the updated o/p by using by using back-propagating algorithm.

Sol^n **Step-1**

$$[I]^0 = \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix} \quad [V]^0 = \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix}$$

$$[W]^0 = \begin{bmatrix} 0.2 \\ -0.5 \end{bmatrix}$$

Step-2

$$[O/P]_{\substack{I/P \\ \text{Layer}}} = [I/P]_{\substack{O/P \\ \text{Layer}}} = \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}$$

Step-3

$$[I/P]_{\substack{\text{Hidden} \\ \text{Layer}}} = [V]^T \; [O/P]_{\substack{I/P \\ \text{Layer}}}.$$

$$= \underset{2\times2}{\begin{bmatrix} 0.1 & -0.2 \\ 0.4 & 0.2 \end{bmatrix}} \underset{2\times1}{\begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}}$$

$$= \begin{bmatrix} 0.04 + 0.14 \\ 0.16 - 0.14 \end{bmatrix}_{2\times1} = \begin{bmatrix} 0.18 \\ 0.02 \end{bmatrix}_{2\times1}$$

Step-4

$$[O/P]_{\substack{\text{Hidden} \\ \text{Layer}}} = \frac{1}{1 + e^{-\lambda[I/P]_{\text{Hidden}}}}$$

$$[O/P]_{\substack{\text{Hidden}\\\text{Layer}}} = \begin{bmatrix} 0.544 \\ 0.504 \end{bmatrix}_{2\times 1}$$

Step-5

$$[I/P]_{\substack{O/P\\\text{Layer}}} = [W]^T [O/P]_{\substack{\text{Hidden}\\\text{Layer}}}$$

$$= \begin{bmatrix} 0.2 & -0.5 \end{bmatrix}_{1\times 2} \begin{bmatrix} 0.544 \\ 0.504 \end{bmatrix}_{2\times 1}$$

$$= \begin{bmatrix} 0.1088 - 0.252 \end{bmatrix}_{1\times 1}$$

$$= \begin{bmatrix} -0.1432 \end{bmatrix}$$

Step-6

$$[O/P]_{\substack{O/P\\\text{Layer}}} = \frac{1}{1 + e^{-(-0.1432)}}$$

$$= \begin{bmatrix} 0.464 \end{bmatrix}$$

Step-7

$$E = \frac{\Sigma (T-O)^2}{n} = \frac{(0.1 - 0.464)^2}{1}$$

$$= 0.132496$$

Step-8 =

$$d = (0.1 - 0.464)\, [O/P]_{O/P} \, \big[1 - [O/P]_{O/P}\big]$$

$$= \begin{bmatrix} -0.364 \end{bmatrix} \begin{bmatrix} 0.464 \end{bmatrix} \begin{bmatrix} 0.536 \end{bmatrix}$$

$$= -0.09058$$

step-9

$$\text{new o/p} = Y = [\text{o/p}]_{\text{Hidden}} \times d.$$

$$= \begin{bmatrix} 0.544 \\ 0.504 \end{bmatrix} \times (-0.09058)$$

$$= \begin{bmatrix} -0.04924 \\ -0.04565 \end{bmatrix}_{2\times 1}$$

step-10

$$[\Delta W_{jk}] = \alpha \cdot [W_{jk}]^{0} + \eta Y$$

$$= \eta Y \quad [\because \alpha = 0]$$

$$= 0.6 \begin{bmatrix} -0.04924 \\ -0.04565 \end{bmatrix} \times$$

$$= \begin{bmatrix} -0.029544 \\ -0.027\underset{4}{3}9 \end{bmatrix}_{2\times 1}$$

step-11.

$$e = [W_{jk}]^{0} \cdot d$$

$$= \begin{bmatrix} 0.2 \\ -0.5 \end{bmatrix} (-0.09058)$$

$$= \begin{bmatrix} -0.018116 \\ 0.04529 \end{bmatrix}$$

step-12

$$d^{*} = \begin{bmatrix} -0.018116 \\ 0.04529 \end{bmatrix} \left[\begin{matrix} 0.544 \\ 0. \end{matrix} \right.$$

Step-12

$$d^{*} = \left[e \quad [O/P]_{\substack{\text{Hidden}\\\text{Layer}}} \quad 1 - [O/P]_{\text{Hidden}} \right]$$

$$= \begin{bmatrix} -0.018116 \times 0.544 \times (1-0.544) \\ 0.04529 \times 0.504 \times (1-0.504) \end{bmatrix}^{2\times1}_{2\times1}$$

$$= \begin{bmatrix} -4.49392 \times 10^{-3} \\ 0.011321 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} -0.00449 \\ 0.011321 \end{bmatrix}_{2\times1}$$

Step-13

$$X = [O/P]_{\substack{I/P\\\text{Layer}}} \cdot d^{*}$$

As matrix multiplication is not possible so we take the transpose of weight term.

$$= \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}^{T}_{2\times1} \begin{bmatrix} -0.00449 \\ 0.011321 \end{bmatrix}^{T}_{2\times1}$$

$$= \begin{bmatrix} 0.4 & -0.7 \end{bmatrix}_{1\times2} \begin{bmatrix} -0.00449 \\ 0.011321 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} -0.001796 - 0.007924 \end{bmatrix}_{1\times1}$$

$$= \begin{bmatrix} -0.00972 \end{bmatrix}_{1\times1}$$

step-14

$$\Delta V_{ij} = \cancel{\alpha [V_{ij}]^0 +} \eta X$$

$$\cancel{= \eta X}$$

$$=$$

$$X = \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}_{2\times 1} \begin{bmatrix} -0.00449 & 0.011321 \end{bmatrix}_{1\times 2}$$

$$= \cancel{-0.001796}$$

$$= \begin{bmatrix} -0.001796 & 0.0045284 \\ 0.003143 & -0.007924 \end{bmatrix}_{2\times 2}$$

Step-14

$$\Delta V_{ij} = \alpha [V_{ij}]^0 + \eta X$$

$$= \eta X \qquad [\text{bcz } \alpha = 0]$$

$$= 0.6 \begin{bmatrix} -0.001796 & 0.0045284 \\ 0.003143 & -0.007924 \end{bmatrix}_{2\times 2}$$

$$= \begin{bmatrix} -0.0010776 & 0.00271704 \\ 0.0018858 & -0.0047544 \end{bmatrix}_{2\times 2}$$

Step-15

$$[V_{ij}]' = [V_{ij}]^0 + [\Delta V_{ij}]$$

$$= \begin{bmatrix} 0.1 & 0.4 \\ -0.2 & 0.2 \end{bmatrix}_{2\times 2} + \begin{bmatrix} -0.0010776 & 0.00271704 \\ 0.0018858 & -0.0047544 \end{bmatrix}_{2\times 2}$$

$$= \begin{bmatrix} 0.0989224 & 0.40271704 \\ -0.1981142 & 0.1952456 \end{bmatrix}$$

$$[W_{ij}] = [W_{ij}]^{0} + [\Delta W_{ij}]^{0}$$

$$= \begin{bmatrix} 0.2 \\ -0.5 \end{bmatrix} + \begin{pmatrix} -0.029544 \\ -0.0274 \end{pmatrix}$$

$$= \begin{bmatrix} 0.170456 \\ -0.5274 \end{bmatrix}$$

Step-16

$$[I/P]_{Hidden} = [V]^{T} \cdot [O/P]_{I/P\ layer}.$$

$$= \begin{bmatrix} 0.0989224 & 0.40271704 \\ -0.1981142 & 0.1952456 \end{bmatrix}^{T} \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}$$

$$= \begin{bmatrix} 0.0989224 & -0.1981142 \\ 0.40271704 & 0.1952456 \end{bmatrix}_{2\times2} \begin{bmatrix} 0.4 \\ -0.7 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.03956896 + 0.13867994 \\ 0.161086881 - 0.13667192 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.1782489 \\ 0.02441489 \end{bmatrix}_{2\times1}$$

$$[O/P]_{Hidden} = \begin{bmatrix} 0.544446 \\ 0.50610341 \end{bmatrix}_{2\times1}$$

$$[I/P]_{O/P} = \begin{bmatrix} 0.2 & -0.5 \end{bmatrix}_{1\times2} \begin{bmatrix} 0.544446 \\ 0.50610341 \end{bmatrix}_{2\times1}$$

$$= \begin{bmatrix} 0.10888892 - 0.253057 \end{bmatrix}_{1\times1}$$

$$= \begin{bmatrix} -0.144162 \end{bmatrix}_{1\times1}$$

$$[O/P]_{O/P} = \frac{1}{1+e^{-(-0.144162)}}$$

$$= 0.464021$$

$$E = \frac{(0.1 - 0.464021)^2}{1} = 0.1325$$

Step-17

Here $T \simeq O = 0.1$

Dt - 30/09/19

Kohonen Self Organizing Map

- This ~~self~~ organising map is a clustring algorithm developed by the scientist Teuvo Kohonen. So, it is known as Kohonen self organising map.
- It creates a map of relationship among the I/p pattern.
- This map is reduced representation of the original data & it is equivalant to the learning vector quantization method. (LVQM)

Principle

- In self organizing map (SOM), the weight vector of the cluster unit, which matches the I/p pattern very closedly is considered first for upgradation of weight.
- The matching & similarity betn the wt. & I/p vector is calculated by Eucledian Difference & the minm difference value will suggest about the winner cluster unit.
- The Eucledian difference can be represented as

$$\boxed{D_j = \sum_{i=1}^{n} |W_{ij} - x_i|^2}$$

where j = no. of o/p clusters.

n = no. of I/p pattern.

[O/P] O/p

E =

Step-17

Here

- In LVQ the Euclidean distance betn the I/p pattern (x) & the wt (W) is calculated for every I/p pattern

- The minimum Euclidean difference gives rise the selection of the perfect I/p pattern to get the target o/p.

Dt - 30/09/19

Kohonen Self Organizing Map

- This self organising map is a clustring algorith developed by the scientist Teuvo Kohonen. So, it i known as Kohonen self organising map.

- It creates a map of relationship among the I/p pattern.

- This map is reduced representation of the origina data & it is equivalant to the learning vector quantization method. (LVQM)

Principle

- In self organizing map (SOM), the weight vector of the cluster unit, which matches the I/p pattern very closedly is considered first for upgradation of weight.

- The matching & similarity betn the wt. & I/p vector is calculated by Eucledian Difference & the minm difference value will suggest about the winner cluster unit.

- The Eucledian difference can be represented as

$$\boxed{D_j = \sum_{i=1}^{n} |w_{ij} - x_i|^2}$$

where j = no. of o/p cluster.

n = no. of I/p pattern.

Algorithm

Step-1

- Initialization of I/p vector, O/p cluster & wt. vector

Step-2

- Calculation of Eucledian distance

$$\boxed{D_j = \sum_{i=1}^{n} |W_{ij} - x_i|^2}$$

Step-3

- Determination of min^m value of D_j & cal. of Winner O/p unit.

Step-4

- Update the weights associated with winner O/p

Step-5

- The updated weight can be calculated as

$$\boxed{W_{ij}^{new} = W_{ij}^{old} + \alpha \left(x_i - W_{ij}^{old} \right)}$$

Here α = learning rate $(0 < \alpha < 1)$

Step-6

- Update the α-value as

$$\boxed{\alpha^{new} = 0.5\,\alpha^{old}}$$

Step-7

- Repeat from step-2 to step-6 till the stopping condn arrise

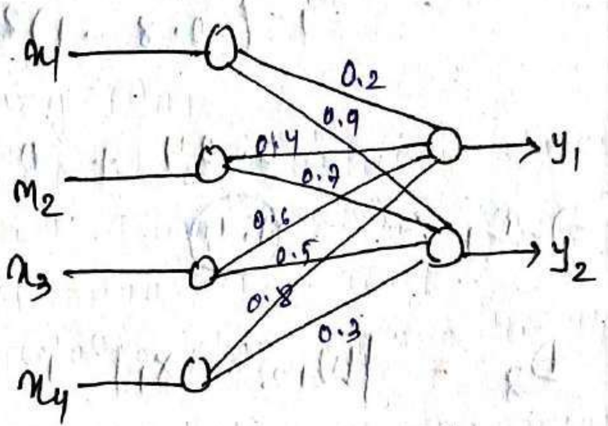
* Stopping condn → 1) α value is very very small or less than the Thresold value. $\alpha <<<<$ Thres old valu

ii) o/p will be equal to target.

Q. Construct a K-SOM to clustere the foure given vectors [0 0 1 1], [1 0 0 0], [0 1 1 0], [0 0 0 1]

The no. of clustere to be formed is 2. Assume an initial learning reate $\alpha = 0.5$ & the weight matrix is given as

$$W_{ij} = \begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 0.2 & 0.9 \\ 2 & 0.4 & 0.7 \\ 3 & 0.6 & 0.5 \\ 4 & 0.8 & 0.3 \end{array}$$



Sol^n ~~Step 1~~

Considere the 1^st I/p pattern, [0 0 1 1.]

~~Initi~~

Step-1

Initialization of I/p vector & wt. vector.

$$W_{ij} = \begin{array}{c|cc} & 1 & 2 \\ \hline 1 & 0.2 & 0.9 \\ 2 & 0.4 & 0.7 \\ 3 & 0.6 & 0.5 \\ 4 & 0.8 & 0.3 \end{array} \qquad I/p = [0 \; 0 \; 1 \; 1]$$

Step-2

Cal. of Eucledian difference (D_j)

$$D_j = \sum_{i=1}^{n} |W_{ij} - x_i|^2$$

$$D_j = \sum_{i=1}^{4} |w_{ij} - x_i|^2$$

$j = 1, 2.$

$$D_1 = \sum_{i=1}^{4} |w_{11} - x_1|^2 + |w_{21} - x_2|^2 + |w_{31} - x_3|^2 + |w_{41} - x_4|^2$$

$$= (0.2 - 0)^2 + (0.4 - 0)^2 + (0.6 - 1)^2 + (0.8 - 1)^2$$

$$= 0.04 + 0.16 + 0.16 + 0.04$$

$$\Rightarrow D_1 = \boxed{0.4}$$

$$D_2 = |w_{12} - x_1|^2 + |w_{22} - x_2|^2 + |w_{32} - x_3|^2 + |w_{42} - x_4|^2$$

$$= |0.9 - 0|^2 + |0.7 - 0|^2 + |0.5 - 1|^2 + |0.3 - 1|^2$$

$$= 0.81 + 0.49 + 0.25 + 0.49$$

$$D_2 = \boxed{2.04}$$

Step-3

The min^m D_j is determined as $D_1 = 0.4$ & it is calculated by considering the o/p y_1. Therefore the winner o/p unit is y_1. ~~&~~

Step-4

Update the wt. associated with the winner o/p y_1.

The wts. which are to be updated first are w_{11}, w_{21}, w_{31} & w_{41}

step-5

$$W_{ij}^{new} = W_{ij}^{old} + \alpha(x_i - w_{ij}^{old})$$

$$W_{11}^{new} = W_{11}^{old} + \alpha(x_1 - W_{11}^{old})$$
$$= 0.2 + 0.5(0 - 0.2)$$
$$= 0.1$$

$$W_{21}^{new} = W_{21}^{old} + \alpha(x_2 - W_{21}^{old})$$
$$= 0.4 + 0.5(0 - 0.4)$$
$$= 0.2$$

$$W_{31}^{new} = W_{31}^{old} + \alpha(x_3 - W_{31}^{old})$$
$$= 0.6 + 0.5(1 - 0.6)$$
$$= 0.6 + 0.5 \times 0.4$$
$$= 0.8$$

$$W_{41}^{new} = W_{41}^{old} + \alpha(x_4 - W_{41}^{old})$$
$$= 0.8 + 0.5(1 - 0.8)$$
$$= 0.9$$

Now the new weight matrix can be represented as,

$$W_{ij}^{new} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 2 \end{array} \\ \begin{bmatrix} 0.1 & 0.9 \\ 0.2 & 0.7 \\ 0.8 & 0.5 \\ 0.9 & 0.3 \end{bmatrix} \end{array}$$

Step - 6

$$\alpha^{new} = 0.5 \alpha^{old}$$

$$= 0.5 \times 0.5$$

$$= 0.25$$

Step - 7

Repeat from step-2 to step-6 till the stopping condn arrise.

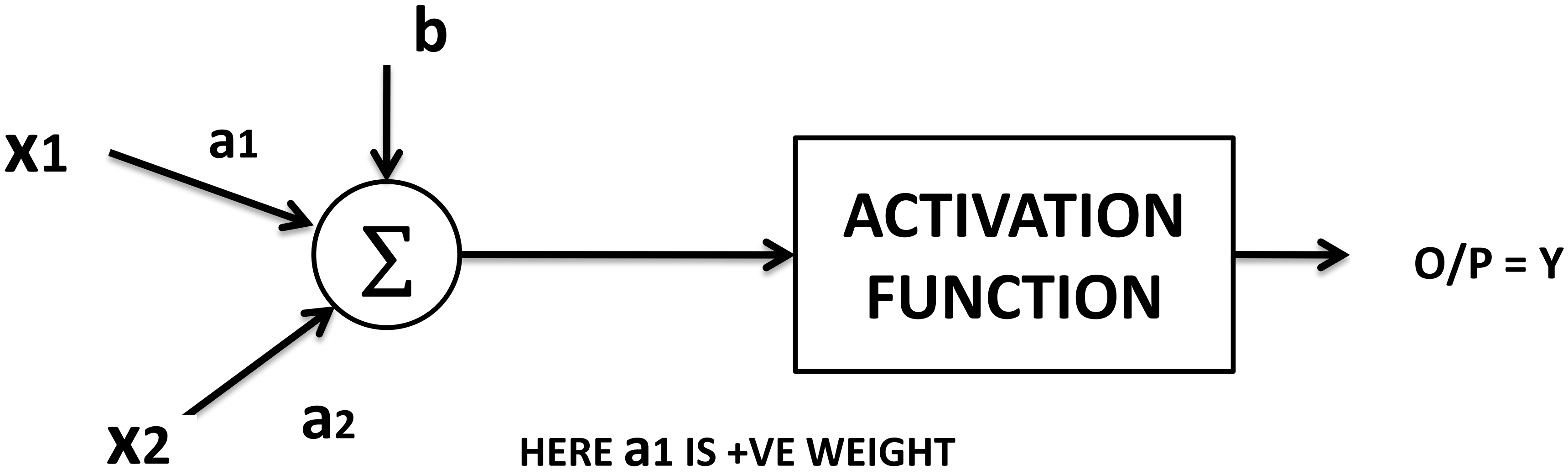
LINEAR SEPARABILITY
IN
ANN

LINEAR SEPARABILITY

- In ANN, it does not give any exact solution for a non linear problem. However it will provide an approximate solution to the non linear problems.
- Therefore , the linear separability is the concept where the input spaced region is separated by the network responses as positive section and negative section.

LINEAR SEPARABILITY

- The positive response which is present due to the inputs are placed separately than that of negative response.
- Consider the following model



HERE a1 IS +VE WEIGHT

a2 IS –VE WEIGHT

LINEAR SEPARABILITY

- In this case the output

$$Y = \begin{cases} 1 , Yin > 0 \\ 0 , Yin <= 0 \end{cases}$$

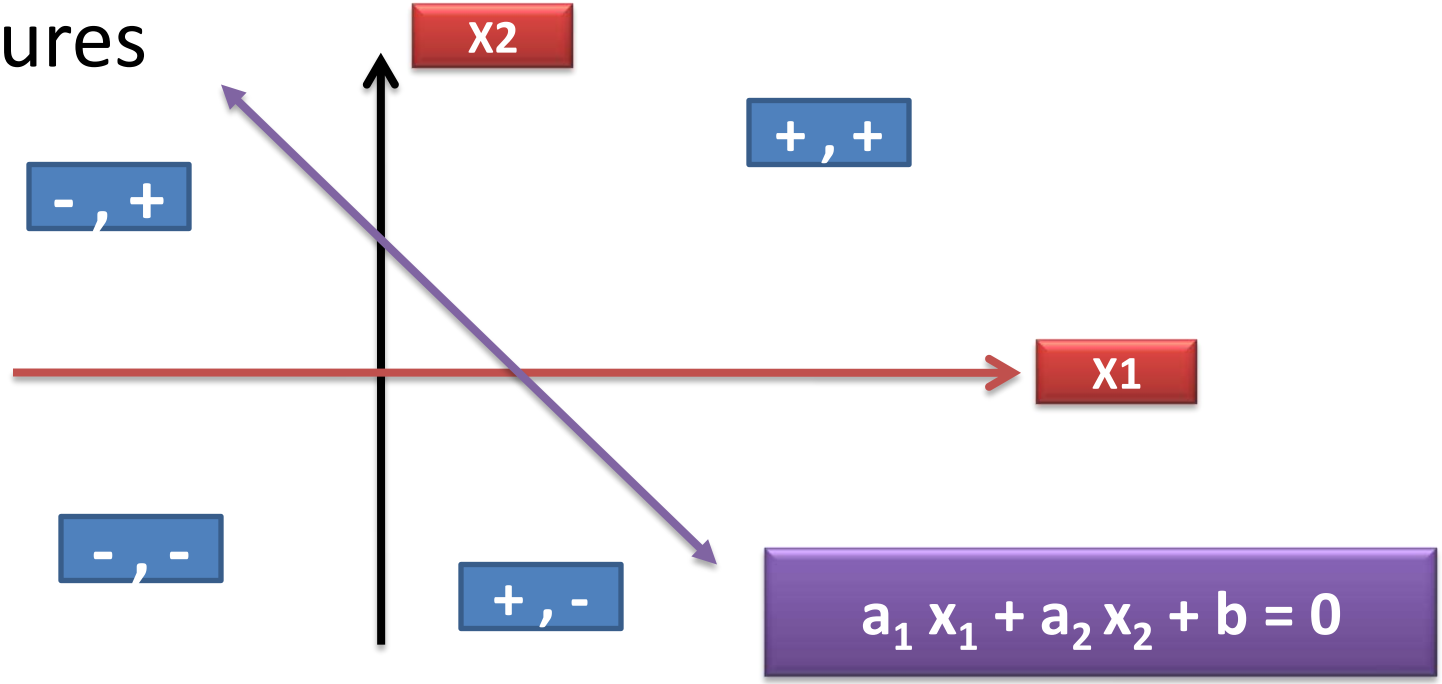
By considering Yin = 0, we get

$\Rightarrow a_1\ x_1 + a_2\ x_2 + b = 0$

$\Rightarrow$ It will represent a straight line or a linear 2D plane.

LINEAR SEPARABILITY

- This line or linear plane is called as decision boundary line or plane which will separate the positive response inputs from negative response inputs as shown in the following figures



X2

+ , +

- , +

X1

- , -

+ , -

$$a_1 x_1 + a_2 x_2 + b = 0$$

LINEAR SEPARABILITY

- In the above diagram, the **decision boundary is linearly separating** the inputs which will produce the positive response or negative response.
- The inputs which are **present above the line** of separability **produce positive response** and the inputs which are **present below the line** of separability will **produce negative response**.
- The linear separability problem can be observed by considering different logic gates such as (XOR, AND, OR etc.) to produce the desired output.

SIMULATED ANNEALING

SIMULATED ANNEALING

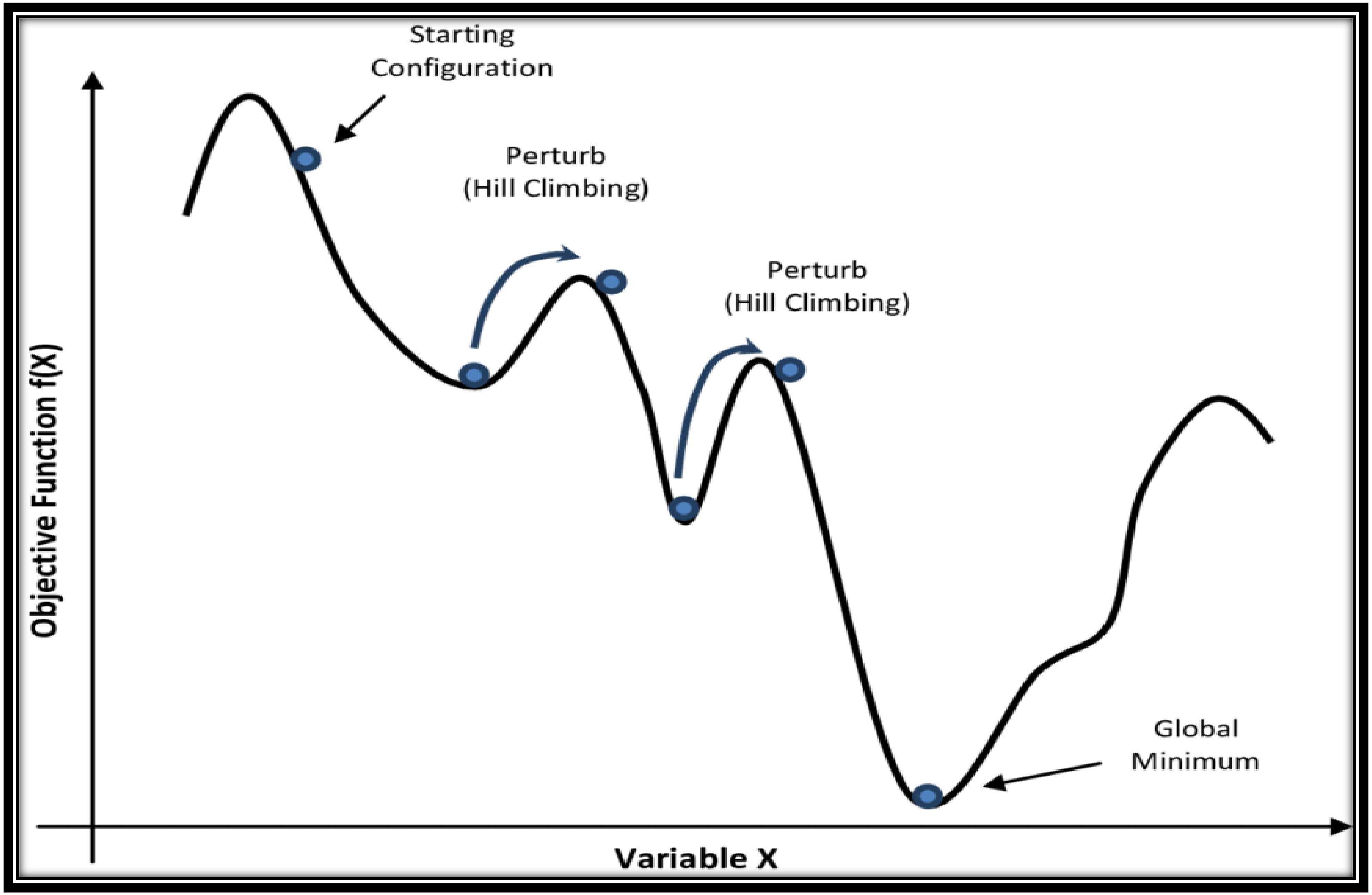
- Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem.

- It is often used when the search space is discrete (e.g., the traveling salesman problem). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to exact algorithms such as gradient descent, Branch and Bound.

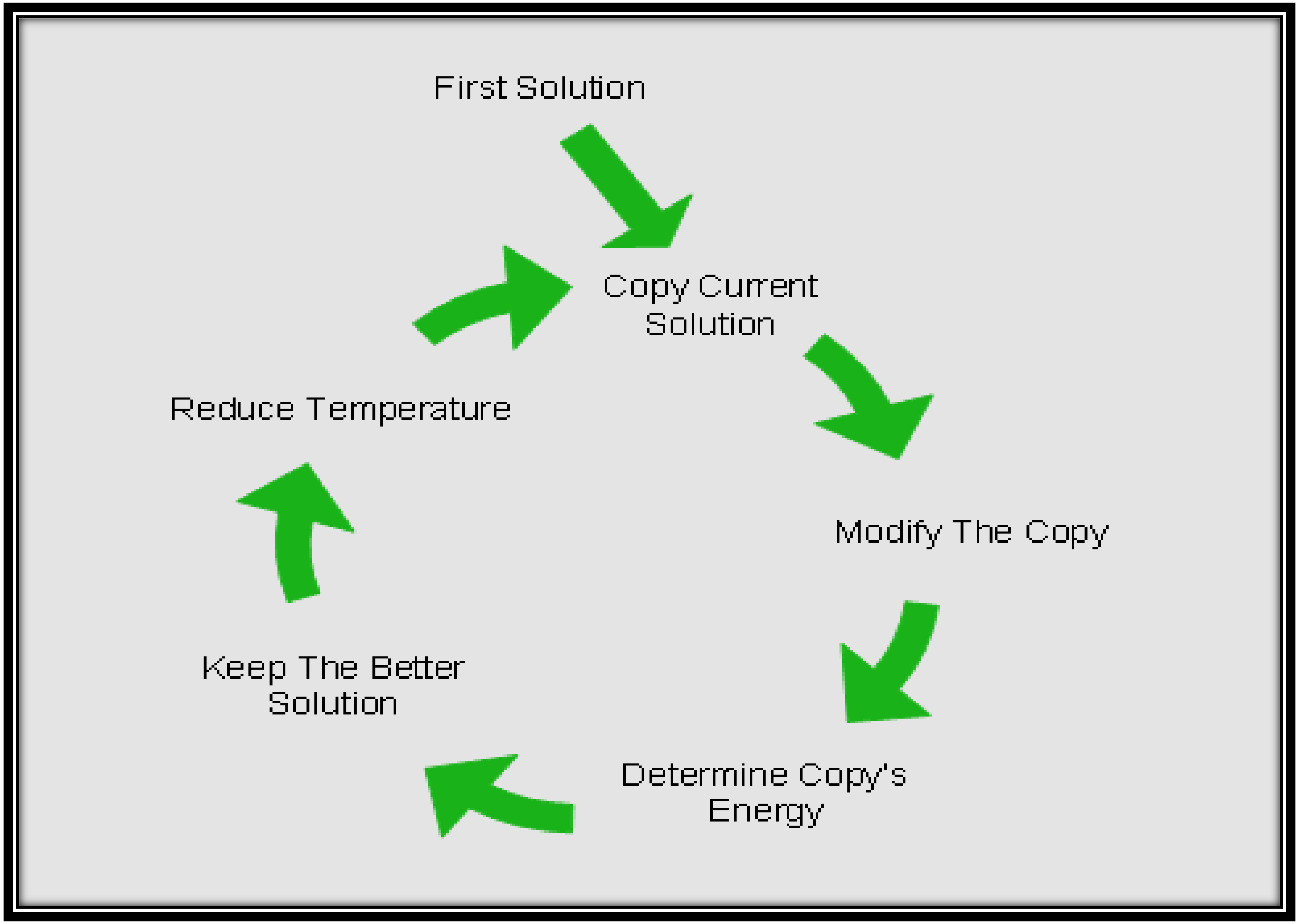
ANNEALING

- In metallurgy and materials science, annealing is a heat treatment that alters the physical and sometimes chemical properties of a material to increase its ductility and reduce its hardness, making it more workable. It involves heating a material above its recrystallization temperature, maintaining a suitable temperature for an appropriate amount of time and then cooling.

ANNEALING


Starting Configuration
Perturb (Hill Climbing)
Perturb (Hill Climbing)
Objective Function f(X)
Global Minimum
Variable X

Simulated Annealing



First Solution

Copy Current Solution

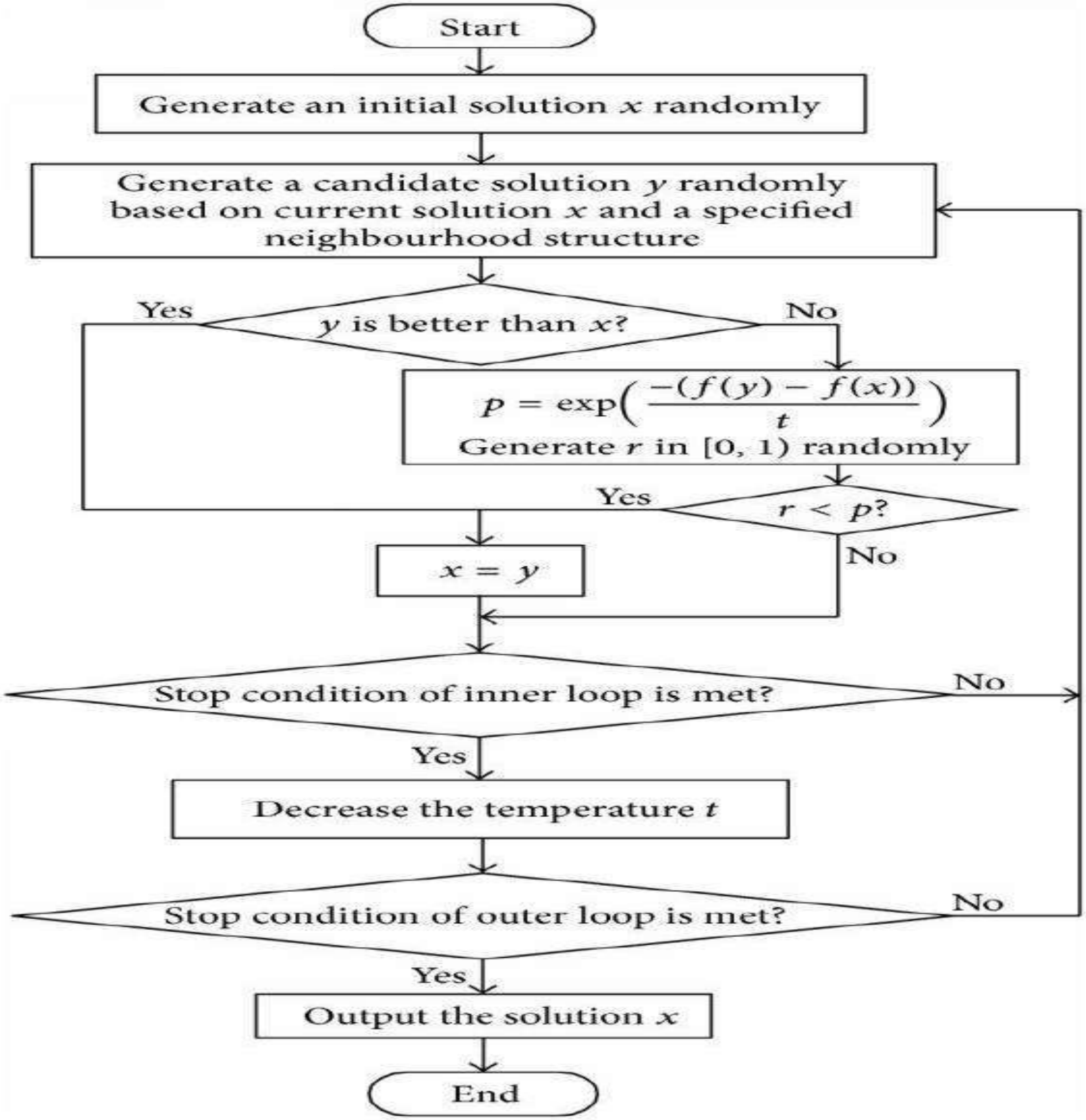
Modify The Copy

Determine Copy's Energy

Keep The Better Solution

Reduce Temperature

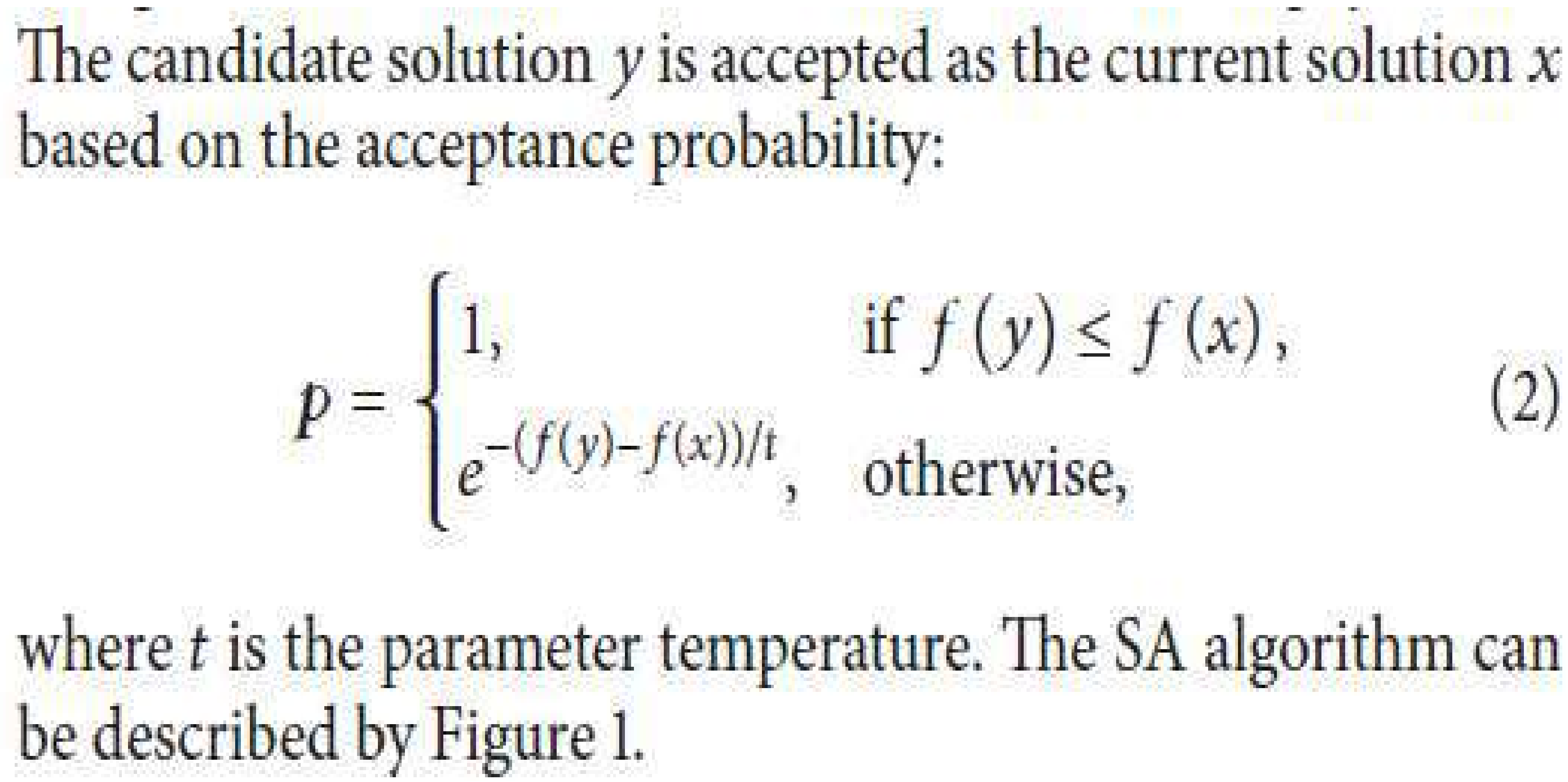
ALGORITHM


Start
Generate an initial solution x randomly
Generate a candidate solution y randomly based on current solution x and a specified neighbourhood structure
y is better than x?
Yes
No
$p = \exp\left(\frac{-(f(y) - f(x))}{t}\right)$
Generate r in [0, 1) randomly
$r < p$?
Yes
No
$x = y$
Stop condition of inner loop is met?
No
Yes
Decrease the temperature t
Stop condition of outer loop is met?
No
Yes
Output the solution x
End

Simulated Annealing

- **Simulated Annealing Algorithm.** SA algorithm is commonly said to be the oldest among the metaheuristics and surely one of the few algorithms that have explicit strategies to avoid local minima.
- The origins of SA are in statistical mechanics and it was first presented for combinatorial optimization problems.
- The fundamental idea is to accept moves resulting in solutions of worse quality than the current solution in order to escape from local minima.
- The probability of accepting such a move is decreased during the search through parameter temperature. SA algorithm starts with an initial solution x, and candidate solution y is then generated (either randomly or using some pre specified rule) from the neighbourhood of x

Simulated Annealing



The candidate solution y is accepted as the current solution x based on the acceptance probability:

$$p = \begin{cases} 1, & \text{if } f(y) \leq f(x), \\ e^{-(f(y)-f(x))/t}, & \text{otherwise,} \end{cases} \qquad (2)$$

where t is the parameter temperature. The SA algorithm can be described by Figure 1.

Question

Consider a four input network with a training input vector and initial weights are given as

$$X_i = \begin{matrix} +1 \\ -1 \\ +1 \\ -1 \end{matrix} \qquad W_i = \begin{matrix} +1 \\ -1 \\ +0 \\ -1 \end{matrix}$$

In addition , the desired or target output is -1 and the training rate μ = 0.1 and sigmoidal function is used as the non linear function. Then update the weight by Generalized Delta Rule.

RADIAL BASIS FUNCTION

Radial Basis Function Networks:

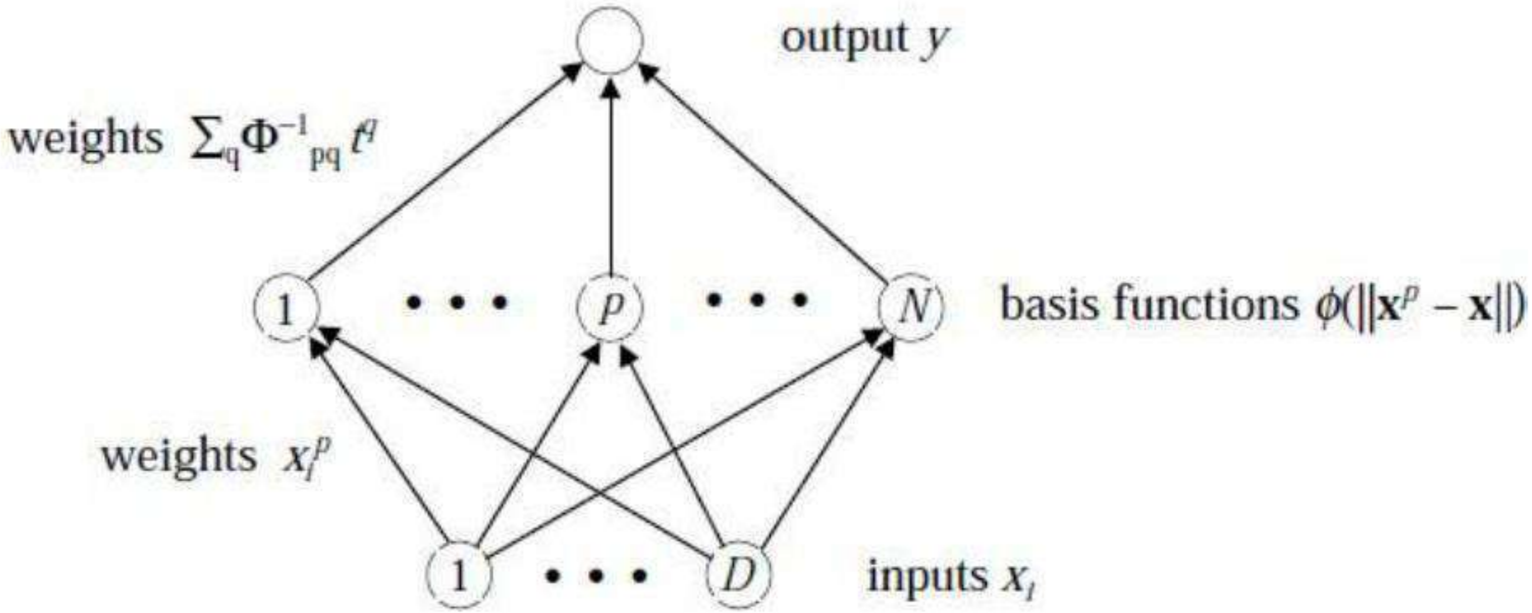

output y
weights $\sum_q \Phi^{-1}{}_{pq}\, t^q$
1
p
N
basis functions $\phi(\|\mathbf{x}^p - \mathbf{x}\|)$
weights x_i^p
1
D
inputs x_i

Fig. RBF network

1. These are 3-layer networks that can approximate any continuous function through a basis function expansion.
2. The basis functions here (which are data dependent as earlier) exhibit some radial symmetry.
3. These networks have the so called perfect interpolation property.
4. The function represented by an RBF network with p hidden nodes can be written as

$$y = \sum_{j=1}^{p} w_j \, \phi \left(\| X - \theta_j \| \right) ,$$

- X is the input to the network.
- Wj is weight from jth hidden node to the output.
- Ø(||X – θ *j* ||*) is the output of the jth hidden node* and θ *j* is the parameter vector associated with jth hidden node, j = 1, ▪ ▪ ▪ , p.

Contd…

Here the output can be written as

$$y = \sum_{j=1}^{p} w_j \, \exp\left(- \frac{||X - \theta_j||^2}{2\sigma^2} \right)$$

- **The θ *j* is called the centre of the jth hidden or RBF node and δ is called the width.**
- **We can have different δ for different hidden nodes.**
- **We next consider learning the parameters of a RBF network from training samples.**
- **Let {(Xi, di), i = 1, ▪ ▪ ▪ ,N} be the training set.**
- **Suppose we are using the Gaussian RBF.**
- **Then we need to learn the centers (θ *j) and widths (δ) of the hidden nodes and the weights into the output node (wj).***

ANFIS ARCHITECTURE

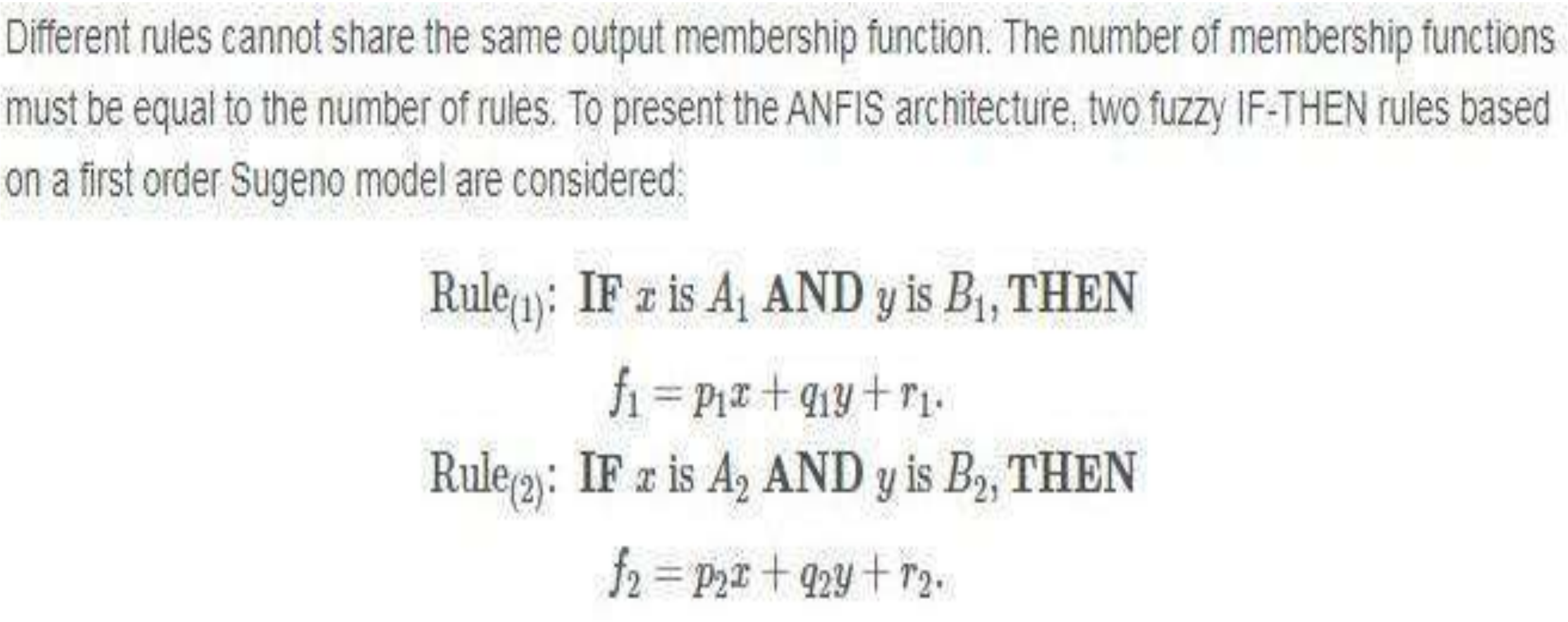
Adaptive Neuro-Fuzzy Inference System

- The Adaptive Neuro-Fuzzy Inference System technique was originally presented by Jang in 1993
- ANFIS is a simple data learning technique that uses Fuzzy Logic to transform given inputs into a desired output through highly interconnected Neural Network processing elements and information connections, which are weighted to map the numerical inputs into an output.
- ANFIS combines the benefits of the two machine learning techniques (Fuzzy Logic and Neural Network) into a single technique.
- An ANFIS works by applying Neural Network learning methods to tune the parameters of a Fuzzy Inference System (FIS).

FEATURES THAT ENABLE ANFIS

- ❑ It refines fuzzy IF-THEN rules to describe the behavior of a complex system;
- ❑ It does not require prior human expertise
- ❑ It is easy to implement
- ❑ It enables fast and accurate learning
- ❑ It offers desired data set; greater choice of membership functions to use; strong generalization abilities; excellent explanation facilities through fuzzy rules; and
- ❑ It is easy to incorporate both linguistic and numeric knowledge for problem solving.

ANFIS RULE



Different rules cannot share the same output membership function. The number of membership functions must be equal to the number of rules. To present the ANFIS architecture, two fuzzy IF-THEN rules based on a first order Sugeno model are considered:

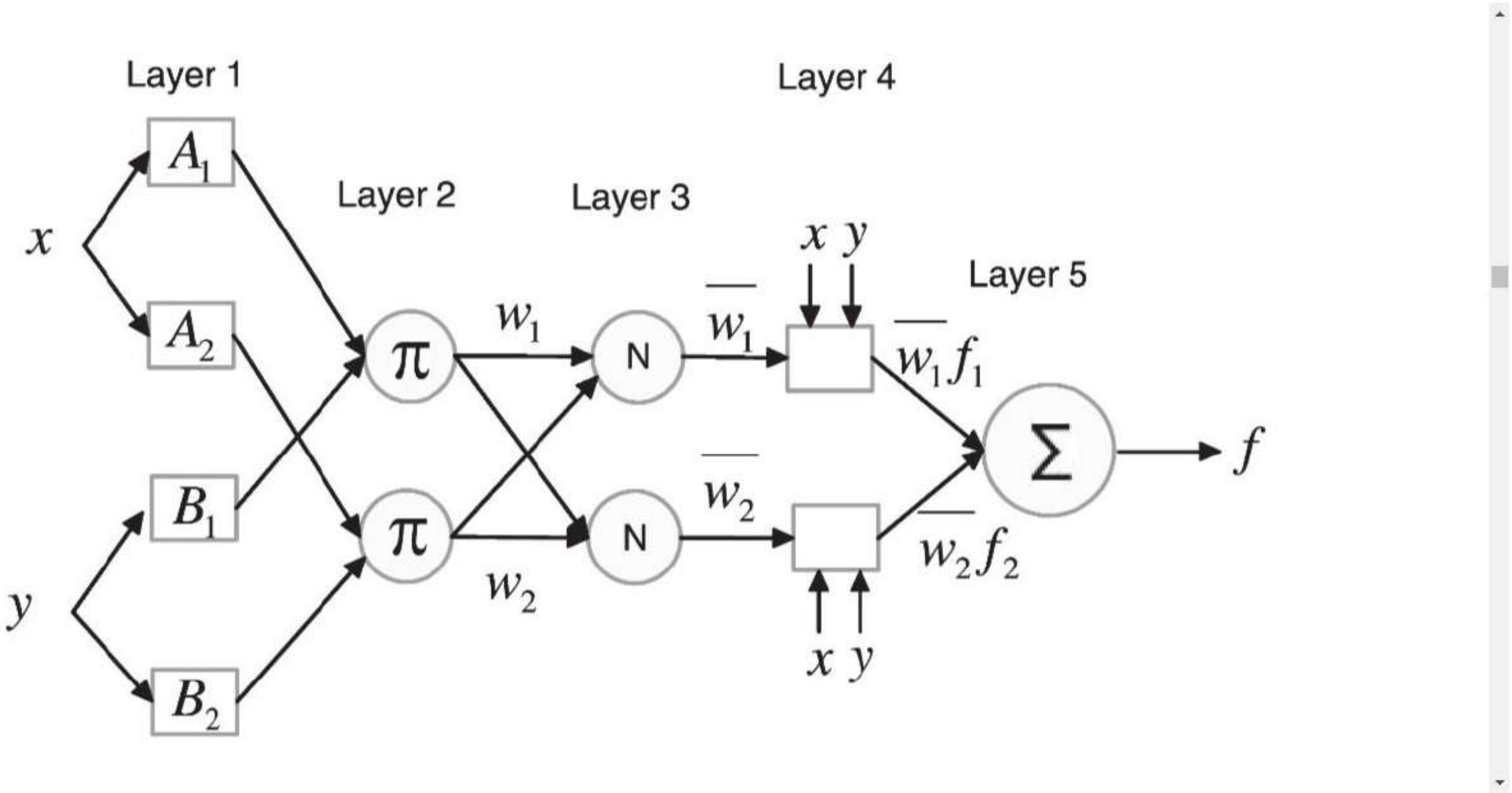
$$\text{Rule}_{(1)}\text{:}\ \ \textbf{IF}\ x\ \text{is}\ A_1\ \textbf{AND}\ y\ \text{is}\ B_1,\ \textbf{THEN}$$

$$f_1 = p_1 x + q_1 y + r_1.$$

$$\text{Rule}_{(2)}\text{:}\ \ \textbf{IF}\ x\ \text{is}\ A_2\ \textbf{AND}\ y\ \text{is}\ B_2,\ \textbf{THEN}$$

$$f_2 = p_2 x + q_2 y + r_2.$$

ARCHITECTURE



LAYER - 1

Let $O_{L,i}$ represent the output of “L” layer for ith node , then the output of layer 1 can be represented as

$O\ 1,i = \mu_{Ai}(x)$, where i = 1,2

$O\ 1,i = \mu_{Bi}(y)$, where i = 1,2

A_1X , A_2X, $B_1\ Y$, $B_2\ Y$

Where “μ” stands for the membership grade function , which is called as adaptive function.

LAYER-2

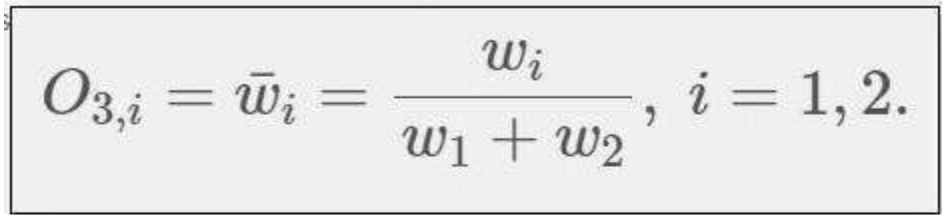
- Every node in Layer-2 is a fixed node labelled with product and it will execute the multiplication of the input to the layer.

- The output of the layer-2 can be written as

$O2,i = W_i = \mu_{Ai}(x) . \mu_{Bi}(y)$, where $i = 1,2$

LAYER-3

- Every node in Layer-3 is a fixed node labelled with **NORM.**
- These nodes represent the ratio of weight coming out from the Layer -2 .
- The output of this layer-3 of ith row can be written as

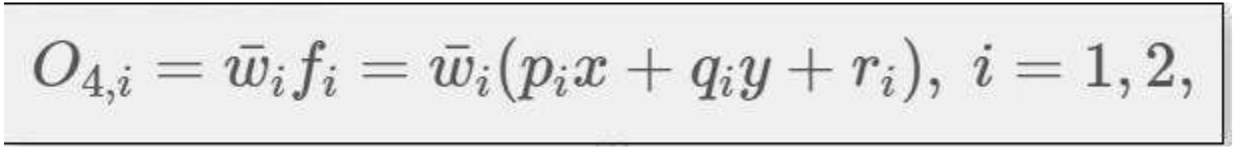


$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2.$$

LAYER - 4

- Every node “i” in layer-4 is an adaptive node with a particular node function.

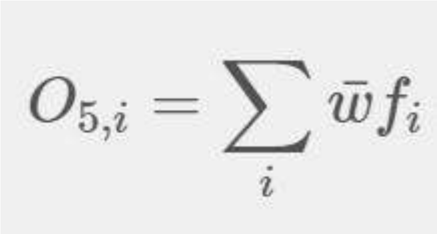
- The output of layer – 4 of the i th row can be written as



$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i), \ i = 1, 2,$$

LAYER -5

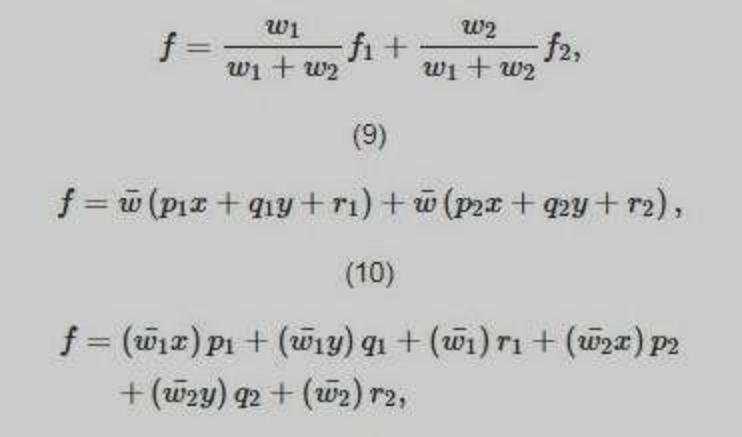
- The node of this layer – 5 labelled with "SUM" that execute the summation of all the input signal from layer – 4 .
- The output of layer – 5 can be written as



$$O_{5,i} = \sum_{i} \bar{w} f_i$$

Hybrid Learning Algorithm

- The learning algorithm for ANFIS is a hybrid algorithm that is a combination of gradient descent and least squares methods.
- The overall output can be given by



$$f = \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2,$$

(9)

$$f = \bar{w}\left(p_1 x + q_1 y + r_1\right) + \bar{w}\left(p_2 x + q_2 y + r_2\right),$$

(10)

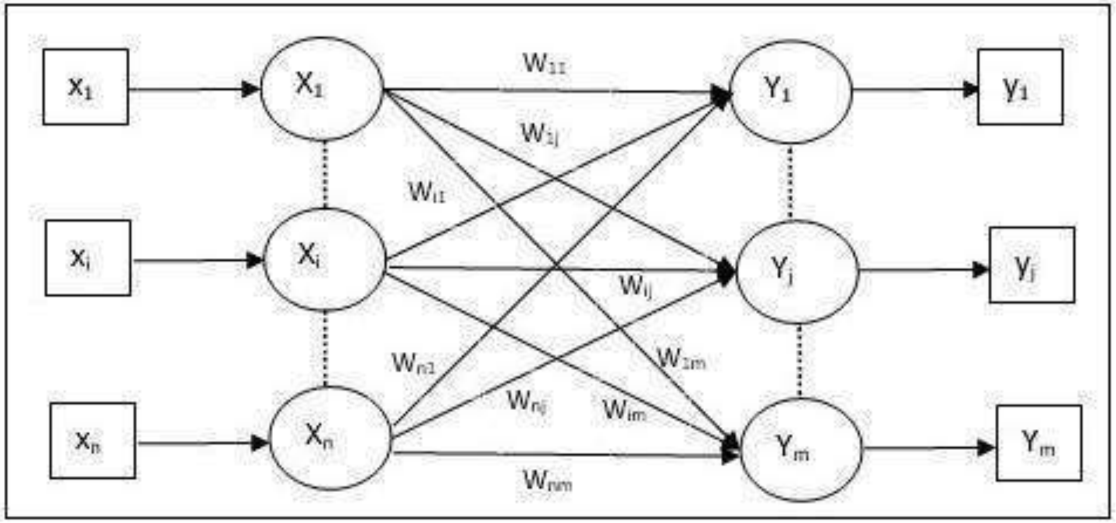
$$\begin{aligned} f = {} & \left(\bar{w}_1 x\right) p_1 + \left(\bar{w}_1 y\right) q_1 + \left(\bar{w}_1\right) r_1 + \left(\bar{w}_2 x\right) p_2 \\ & + \left(\bar{w}_2 y\right) q_2 + \left(\bar{w}_2\right) r_2, \end{aligned}$$

LEARNING VECTOR QUANTIZATION

- Learning Vector Quantization LVQ, different from Vector quantization VQ and Kohonen Self-Organizing Maps KSOM, basically is a competitive network which uses supervised learning.
- We may define it as a process of classifying the patterns where each output unit represents a class.
- As it uses supervised learning, the network will be given a set of training patterns with known classification along with an initial distribution of the output class.
- After completing the training process, LVQ will classify an input vector by assigning it to the same class as that of the output unit.

Architecture

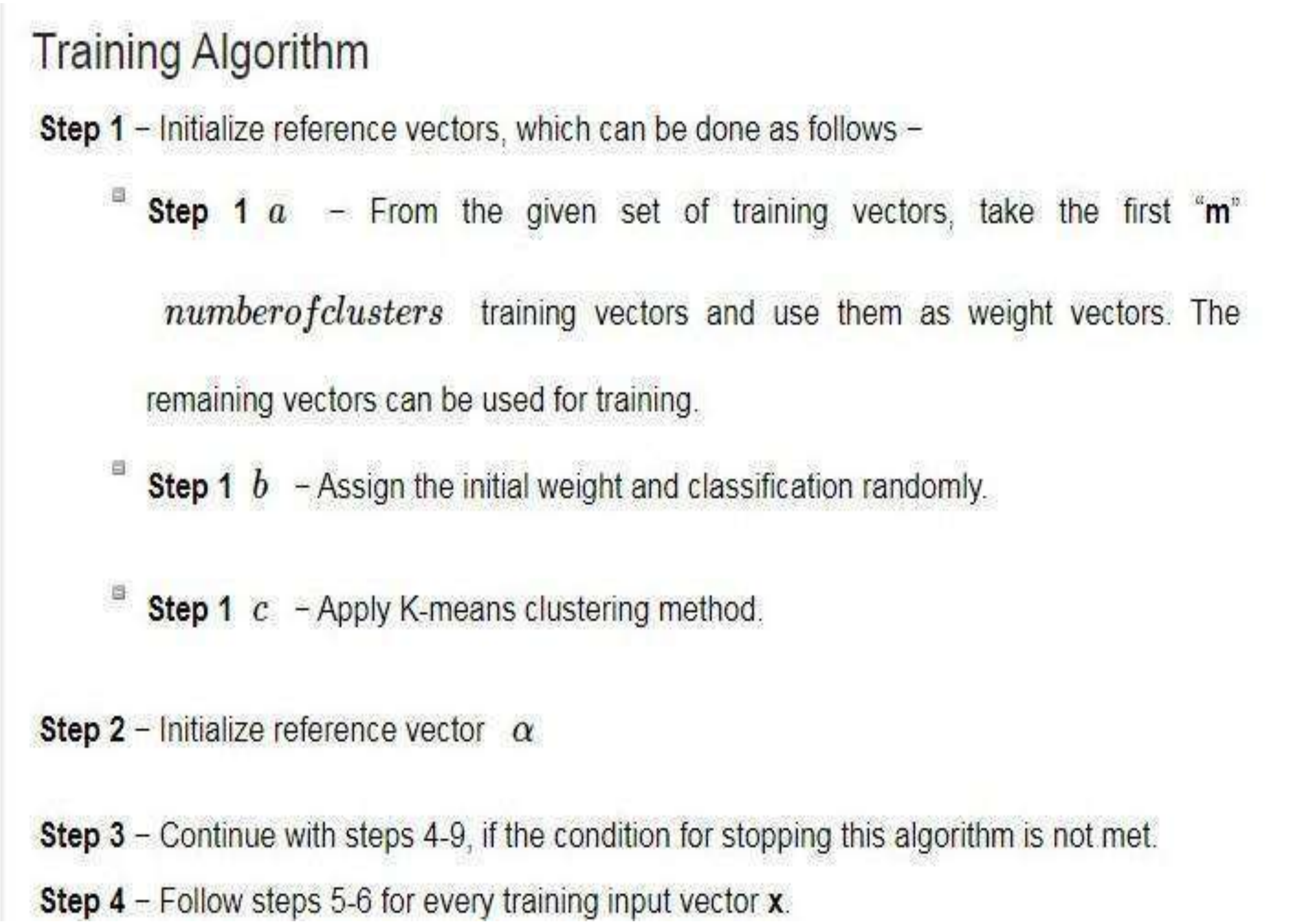
Following figure shows the architecture of LVQ which is quite similar to the architecture of KSOM. As we can see, there are **"n"** number of input units and **"m"** number of output units. The layers are fully interconnected with having weights on them.



Parameters Used

Following are the parameters used in LVQ training process as well as in the flowchart

- **x** = training vector $(x_1,...,x_i,...,x_n)$
- **T** = class for training vector **x**
- $\mathbf{w_j}$ = weight vector for $\mathbf{j^{th}}$ output unit
- $\mathbf{C_j}$ = class associated with the $\mathbf{j^{th}}$ output unit



Training Algorithm

Step 1 – Initialize reference vectors, which can be done as follows –

- **Step 1** a – From the given set of training vectors, take the first “**m**” $numberofclusters$ training vectors and use them as weight vectors. The remaining vectors can be used for training.
- **Step 1** b – Assign the initial weight and classification randomly.
- **Step 1** c – Apply K-means clustering method.

Step 2 – Initialize reference vector α

Step 3 – Continue with steps 4-9, if the condition for stopping this algorithm is not met.

Step 4 – Follow steps 5-6 for every training input vector **x**.

Step 5 – Calculate Square of Euclidean Distance for **j = 1 to m** and **i = 1 to n**

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

Step 6 – Obtain the winning unit **J** where **D** j is minimum.

Step 7 – Calculate the new weight of the winning unit by the following relation –

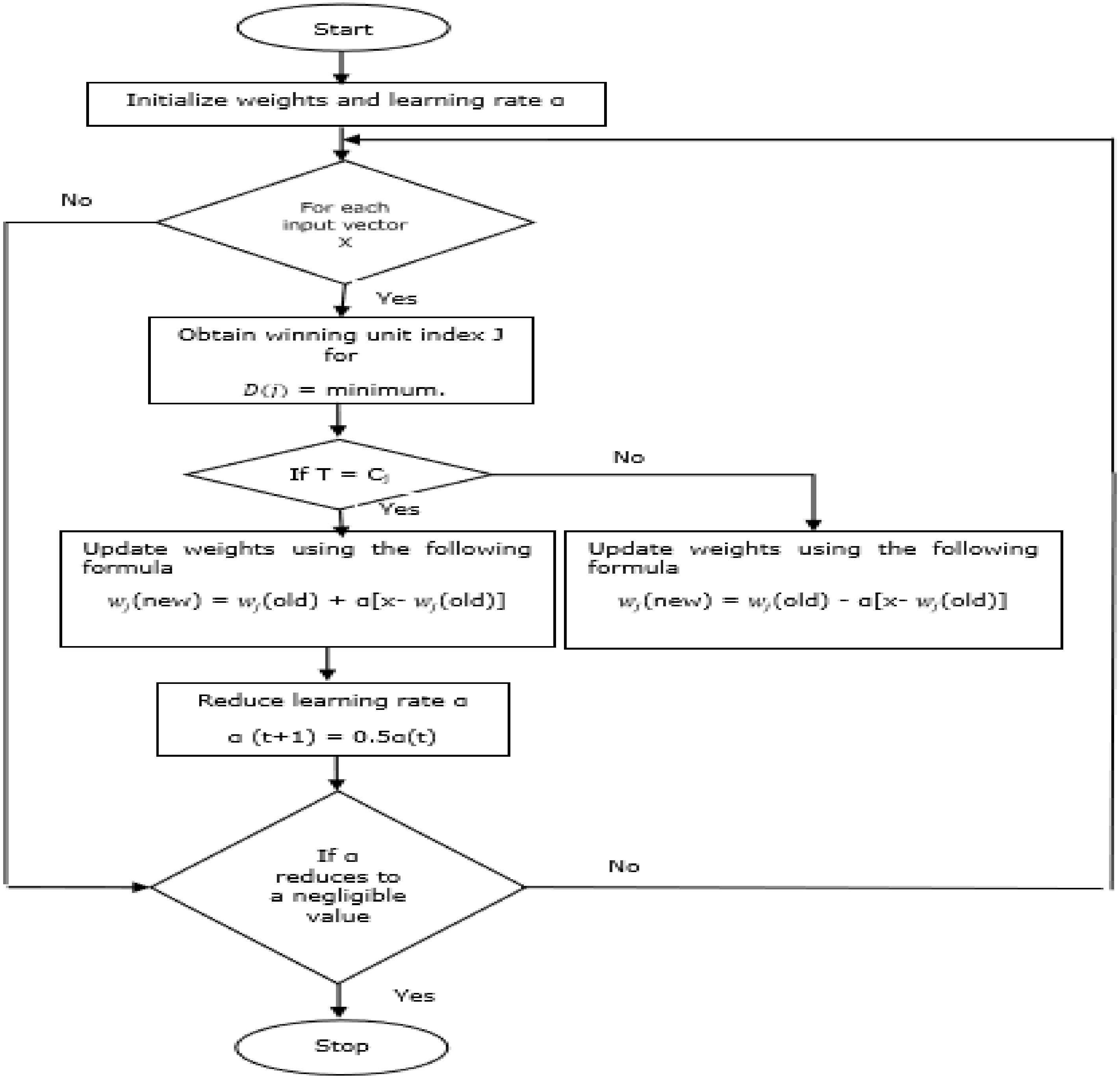
if **T = C$_j$** then $w_j(new) = w_j(old) + \alpha[x - w_j(old)]$

if **T ≠ C$_j$** then $w_j(new) = w_j(old) - \alpha[x - w_j(old)]$

Step 8 – Reduce the learning rate α .

Step 9 – Test for the stopping condition. It may be as follows –

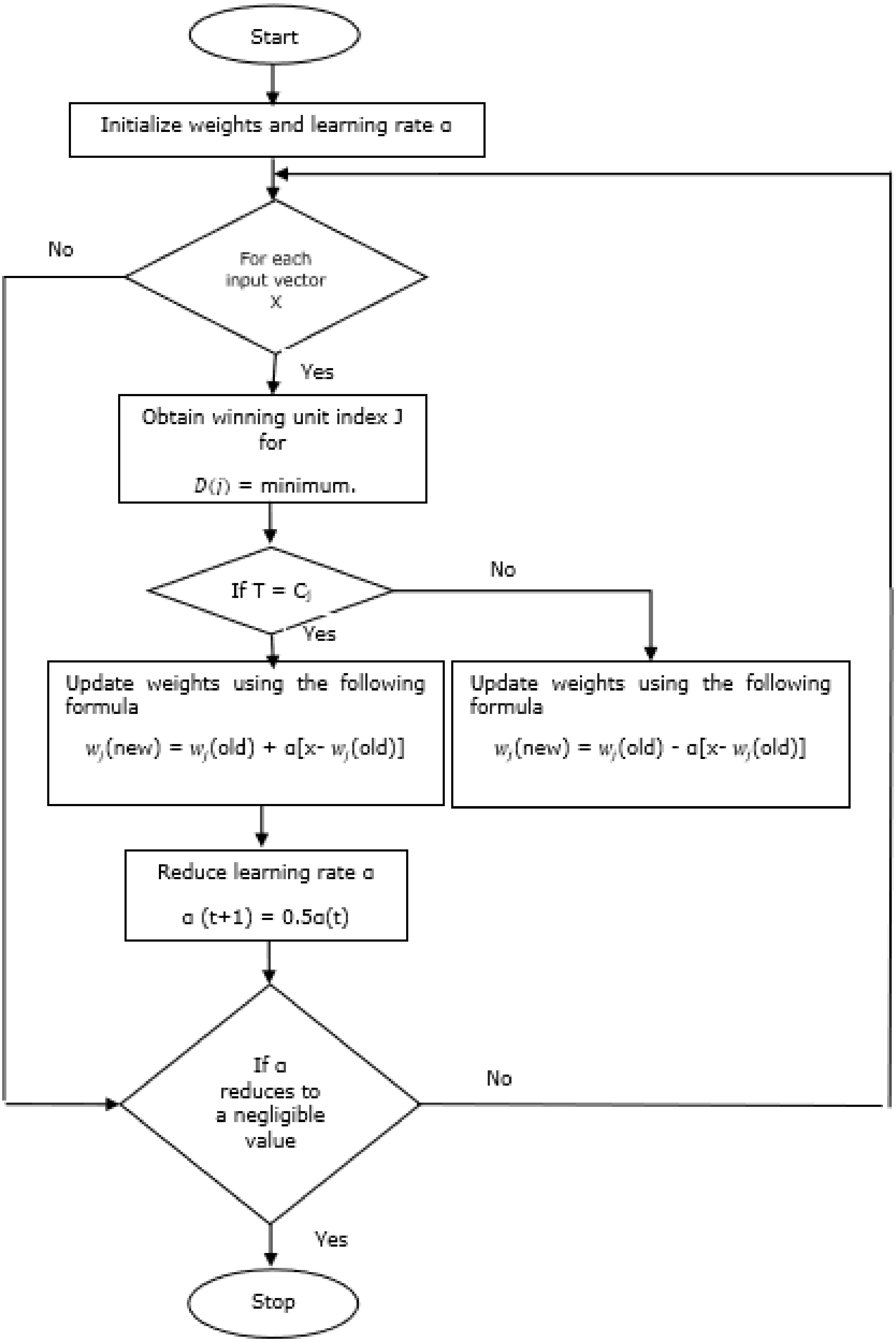
- Maximum number of epochs reached.
- Learning rate reduced to a negligible value.


Start
Initialize weights and learning rate α
For each input vector X
No
Yes
Obtain winning unit index J for $D(j)$ = minimum.
If T = C_j
No
Yes
Update weights using the following formula $w_j(new) = w_j(old) + \alpha[x- w_j(old)]$
Update weights using the following formula $w_j(new) = w_j(old) - \alpha[x- w_j(old)]$
Reduce learning rate α
α (t+1) = 0.5α(t)
If α reduces to a negligible value
No
Yes
Stop

- The LVQ was developed by the scientist Kohonen, which adjust the boundary between the cluster to minimize the miss classification.
- An LVQ has a single layer of nodes which have a particular class or subclass or pattern.
- During the training process for each input pattern , the LVQ finds the output nodes with best matching to the training pattern.

CONTD..

- By this repeated process the LVQ finds that output nodes which has the similar pattern with training unit and after that optimum situation arise.
- Applications of LVQ:-
    1. Optical Character Reorganization
    2. Convert Speech Into Phonemes


Start
Initialize weights and learning rate α
For each input vector X
No
Yes
Obtain winning unit index J for
$D(j)$ = minimum.
If T = C_j
No
Yes
Update weights using the following formula
w_j(new) = w_j(old) + α[x- w_j(old)]
Update weights using the following formula
w_j(new) = w_j(old) - α[x- w_j(old)]
Reduce learning rate α
α (t+1) = 0.5α(t)
If α reduces to a negligible value
No
Yes
Stop

Associate Memory Network

These kinds of neural networks work on the basis of pattern association, which means they can store different patterns and at the time of giving an output they can produce one of the stored patterns by matching them with the given input pattern. These types of memories are also called **Content-Addressable Memory** CAM . Associative memory makes a parallel search with the stored patterns as data files.

Following are the two types of associative memories we can observe −

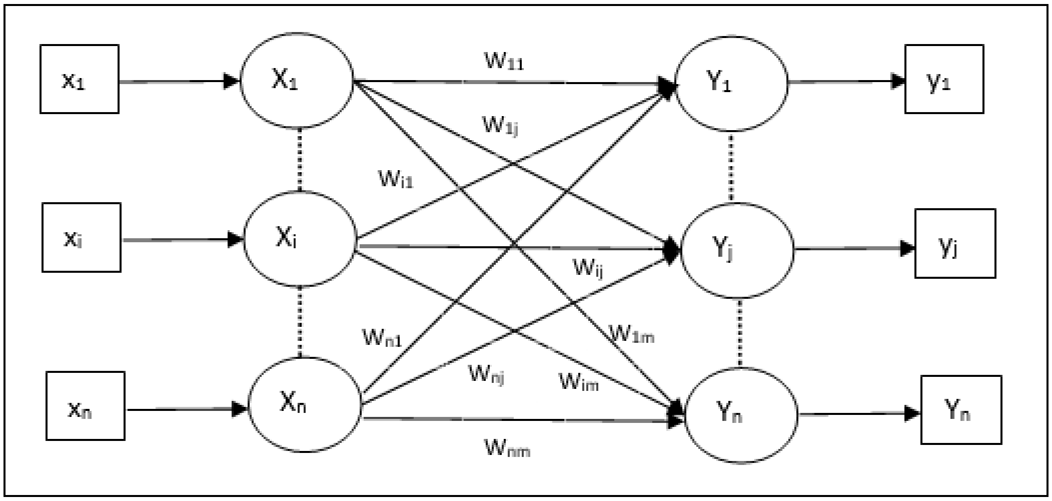
- Auto Associative Memory
- Hetero Associative memory

Auto Associative Memory

This is a single layer neural network in which the input training vector and the output target vectors are the same. The weights are determined so that the network stores a set of patterns.

Architecture

As shown in the following figure, the architecture of Auto Associative memory network has **'n'** number of input training vectors and similar **'n'** number of output target vectors.



Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as **w$_{ij}$ = 0** $i = 1 to n, j = 1 to n$

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows −

$$x_i \; = \; s_i \, (i \; = \; 1 \, to \, n)$$

Step 4 − Activate each output unit as follows −

$$y_j \; = \; s_j \, (j \; = \; 1 \, to \, n)$$

Step 5 − Adjust the weights as follows −

$$w_{ij}(new) \; = \; w_{ij}(old) \; + \; x_i y_j$$

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb's rule.

Step 2 − Perform steps 3-5 for each input vector.

Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit **j = 1 to n**

$$y_{inj} \; = \; \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 − Apply the following activation function to calculate the output

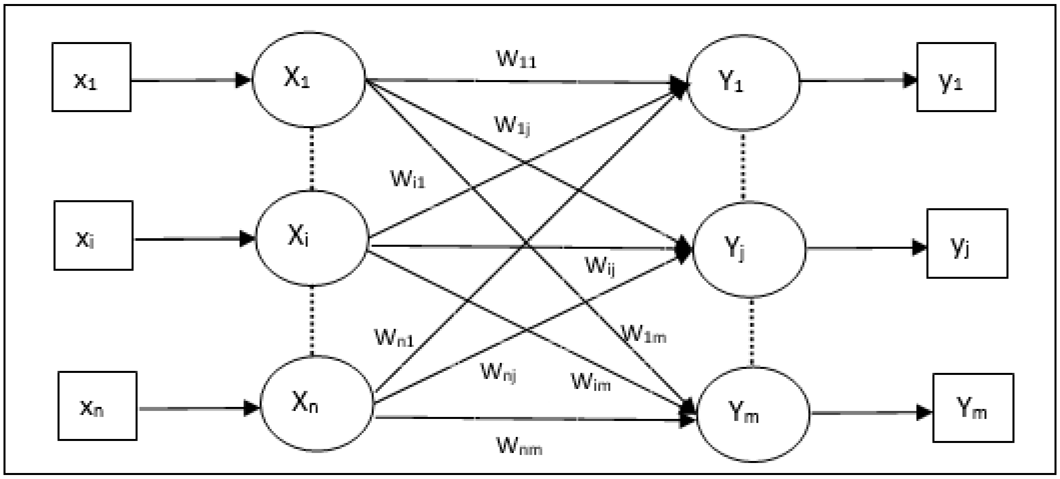
$$y_j \; = \; f(y_{inj}) \; = \; \begin{cases} +1 & if \, y_{inj} \; > \; 0 \\ -1 & if \, y_{inj} \; \leqslant \; 0 \end{cases}$$

Hetero Associative memory

Similar to Auto Associative Memory network, this is also a single layer neural network. However, in this network the input training vector and the output target vectors are not the same. The weights are determined so that the network stores a set of patterns. Hetero associative network is static in nature, hence, there would be no non-linear and delay operations.

Architecture

As shown in the following figure, the architecture of Hetero Associative Memory network has **‘n’** number of input training vectors and **‘m’** number of output target vectors.



Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as **w_{ij} = 0** $i = 1ton, j = 1tom$

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows −

$$x_i \;=\; s_i \;(i \;=\; 1\, to\, n)$$

Step 4 − Activate each output unit as follows −

$$y_j \;=\; s_j \;(j \;=\; 1\, to\, m)$$

Step 5 − Adjust the weights as follows −

$$w_{ij}(new) \;=\; w_{ij}(old) \;+\; x_i y_j$$

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb’s rule.

Step 2 − Perform steps 3-5 for each input vector.

Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit **j = 1 to m;**

$$y_{inj}\ =\ \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 − Apply the following activation function to calculate the output

$$y_j\ =\ f(y_{inj})\ =\ \begin{cases}+1 & if\, y_{inj}\ >\ 0\\0 & if\, y_{inj}\ =\ 0\\-1 & if\, y_{inj}\ <\ 0\end{cases}$$

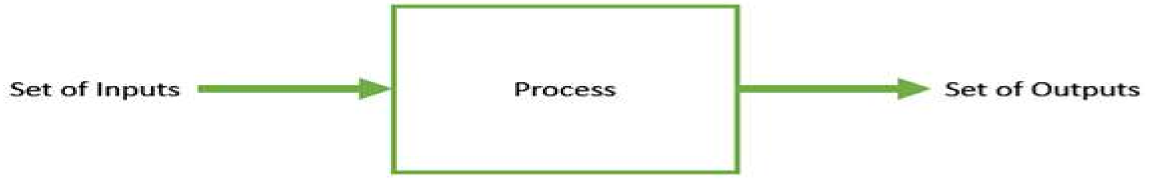
MODULE-IV
Genetic Algorithm

INTRODUCTION

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of **Genetics and Natural Selection**. It is frequently used to find optimal or near-optimal solutions to difficult problems which otherwise would take a lifetime to solve. It is frequently used to solve optimization problems, in research, and in machine learning.

INTRODUCTION TO OPTIMIZATION

Optimization is the process of making something better. In any process, we have a set of inputs and a set of outputs as shown in the following figure.



Optimization refers to finding the values of inputs in such a way that we get the “best” output values. The definition of “best” varies from problem to problem, but in mathematical terms, it refers to maximizing or minimizing one or more objective functions, by varying the input parameters.

The set of all possible solutions or values which the inputs can take make up the search space. In this search space, lies a point or a set of points which gives the optimal solution. The aim of optimization is to find that point or set of points in the search space.

WHAT ARE GENETIC ALGORITHMS?

Nature has always been a great source of inspiration to all mankind. Genetic Algorithms (GAs) are search based algorithms based on the concepts of natural selection and genetics. GAs are a subset of a much larger branch of computation known as Evolutionary Computation.

GAs were developed by John Holland and his students and colleagues at the University of Michigan, most notably David E. Goldberg and has since been tried on various optimization problems with a high degree of success.

In GAs, we have a pool or a population of possible solutions to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more “fitter” individuals. This is in line with the Darwinian Theory of “Survival of the Fittest”.

In this way we keep “evolving” better individuals or solutions over generations, till we reach a stopping criterion.

Genetic Algorithms are sufficiently randomized in nature, but they perform much better than random local search (in which we just try various random solutions, keeping track of the best so far), as they exploit historical information as well.

ADVANTAGES OF GA

GA has various advantages which have made them immensely popular. These include –

- Does not require any derivative information (which may not be available for many real-world problems).
- Is faster and more efficient as compared to the traditional methods.
- Has a very good parallel capability.
- Optimizes both continuous and discrete functions and also multi-objective problems.
- Provides a list of “good” solutions and not just a single solution.
- Always gets an answer to the problem, which gets better over the time.
- Useful when the search space is very large and there are a large number of parameters involved.

LIMITATIONS OF GA

Like any technique, GA also suffers from a few limitations. These include –

- GA is not suited for all problems, especially problems which are simple and for which derivative information is available.
- Fitness value is calculated repeatedly which might be computationally expensive for some problems.
- Being stochastic, there are no guarantees on the optimality or the quality of the solution.
- If not implemented properly, the GA may not converge to the optimal solution.

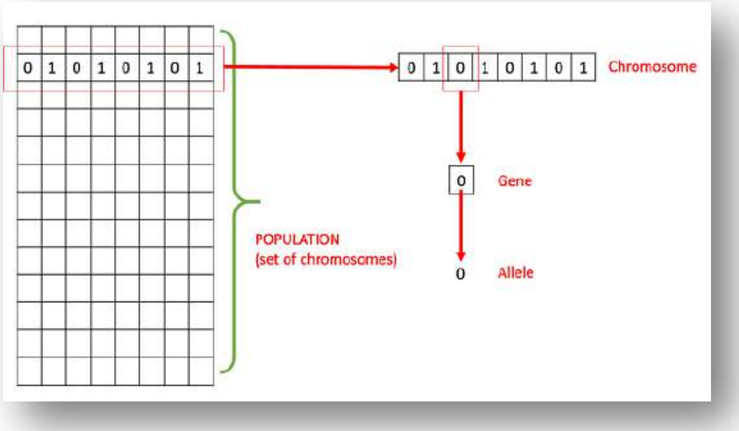
GENETIC ALGORITHMS – FUNDAMENTALS

This section introduces the basic terminology required to understand GAs. Also, a generic structure of GAs is presented in both **pseudo-code and graphical forms**. The reader is advised to properly understand all the concepts introduced in this section and keep them in mind when reading other sections of this tutorial as well.

BASIC TERMINOLOGY

Before beginning a discussion on Genetic Algorithms, it is essential to be familiar with some basic terminology which will be used throughout this tutorial.

- **Population** − It is a subset of all the possible (encoded) solutions to the given problem. The population for a GA is analogous to the population for human beings except that instead of human beings, we have Candidate Solutions representing human beings.
- **Chromosomes** − A chromosome is one such solution to the given problem.
- **Gene** − A gene is one element position of a chromosome.
- **Allele** − It is the value a gene takes for a particular chromosome.



- **Genotype** − Genotype is the population in the computation space. In the computation space, the solutions are represented in a way which can be easily understood and manipulated using a computing system.

- **Phenotype** − Phenotype is the population in the actual real world solution space in which solutions are represented in a way they are represented in real world situations.

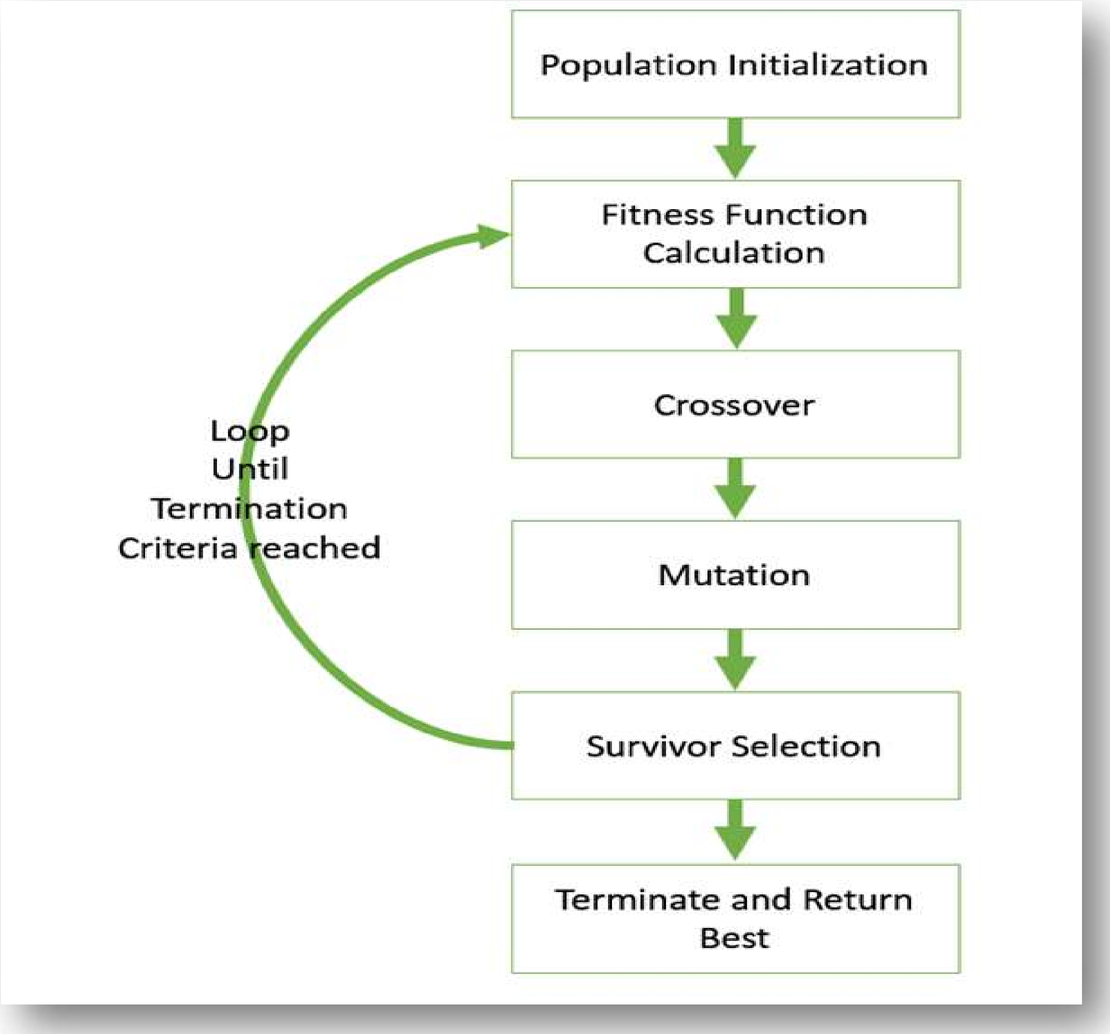
- **Decoding and Encoding** − For simple problems, the **phenotype and genotype** spaces are the same. However, in most of the cases, the phenotype and genotype spaces are different. Decoding is a process of transforming a solution from the genotype to the phenotype space, while encoding is a process of transforming from the phenotype to genotype space. Decoding should be fast as it is carried out repeatedly in a GA during the fitness value calculation.

- **Fitness Function** − A fitness function simply defined is a function which takes the solution as input and produces the suitability of the solution as the output. In some cases, the fitness function and the objective function may be the same, while in others it might be different based on the problem.

- **Genetic Operators** − These alter the genetic composition of the offspring. These include crossover, mutation, selection, etc.

BASIC STRUCTURE

The basic structure of a GA is as follows –


Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

GENETIC ALGORITHMS - FITNESS FUNCTION

The fitness function simply defined is a function which takes a **candidate solution to the problem as input and produces as output** how "fit" our how "good" the solution is with respect to the problem in consideration.

Calculation of fitness value is done repeatedly in a GA and therefore it should be sufficiently fast. A slow computation of the fitness value can adversely affect a GA and make it exceptionally slow.

In most cases the fitness function and the objective function are the same as the objective is to either maximize or minimize the given objective function. However, for more complex problems with multiple objectives and constraints, an **Algorithm Designer** might choose to have a different fitness function.

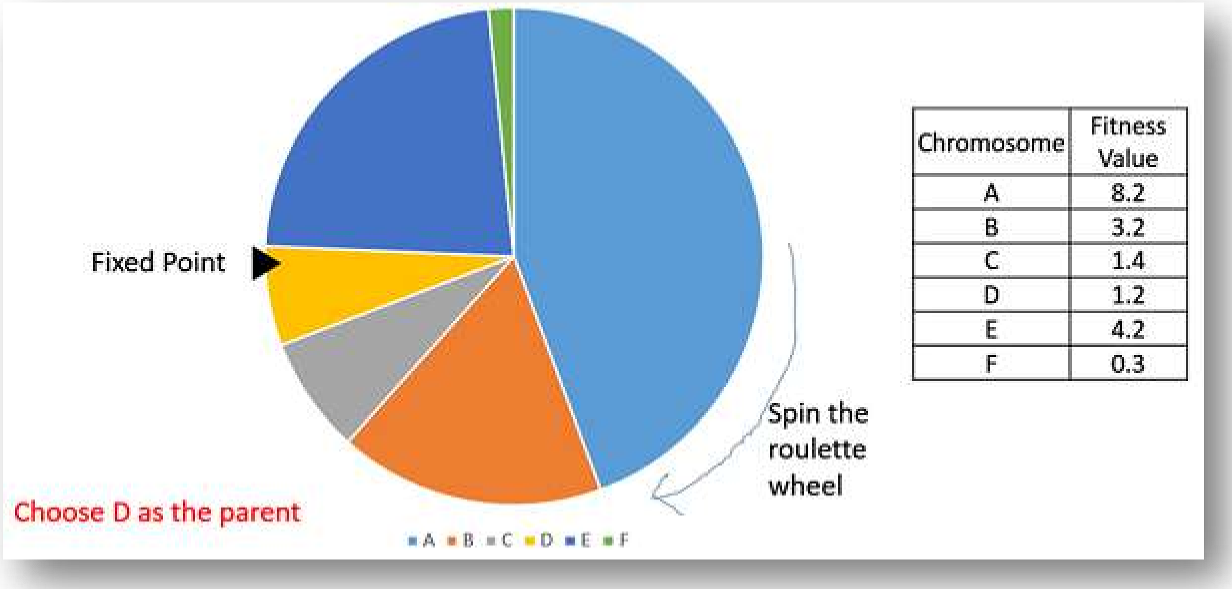
A fitness function should possess the following characteristics –

- The fitness function should be sufficiently fast to compute.
- It must quantitatively measure how fit a given solution is or how fit individuals can be produced from the given solution.

GENETIC ALGORITHMS - PARENT SELECTION:-

ROULETTE WHEEL SELECTION

In a roulette wheel selection, the circular wheel is divided as described before. A fixed point is chosen on the wheel circumference as shown and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent. For the second parent, the same process is repeated.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

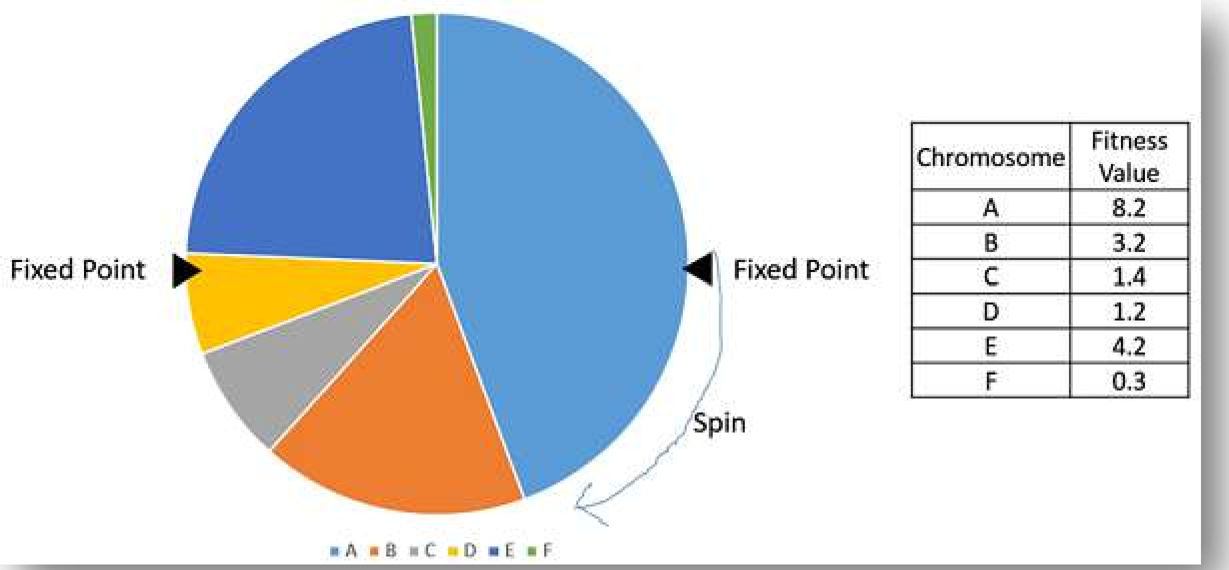
It is clear that a fitter individual has a greater pie on the wheel and therefore a greater chance of landing in front of the fixed point when the wheel is rotated. Therefore, the probability of choosing an individual depends directly on its fitness.

Implementation wise, we use the following steps –

- Calculate S = the sum of a finesses.
- Generate a random number between 0 and S.
- Starting from the top of the population, keep adding the finesses to the partial sum P, till P<S.
- The individual for which P exceeds S is the chosen individual.
- **Stochastic Universal Sampling (SUS)**
- Stochastic Universal Sampling is quite similar to Roulette wheel selection; however instead of having just one fixed point, we have multiple fixed points as shown in the

following image. Therefore, all the parents are chosen in just one spin of the wheel. Also, such a setup encourages the highly fit individuals to be chosen at least once.

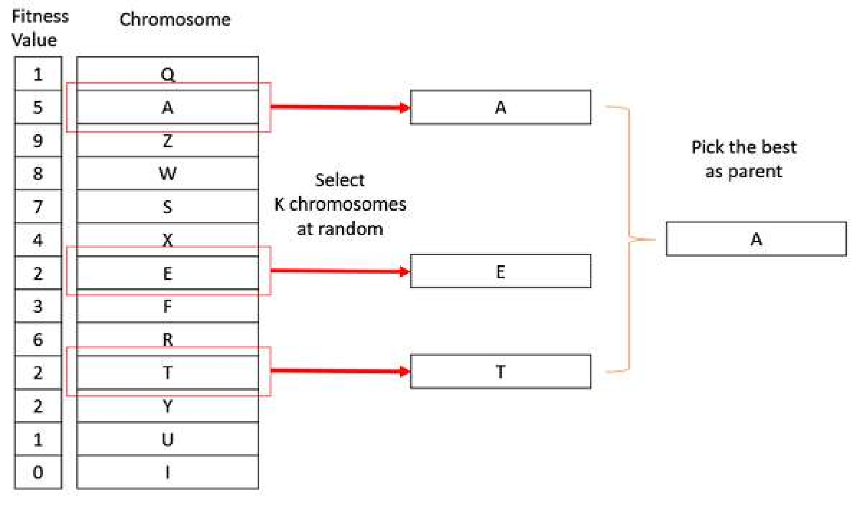
- It is to be noted that fitness proportionate selection methods don't work for cases where the fitness can take a negative value.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

TOURNAMENT SELECTION

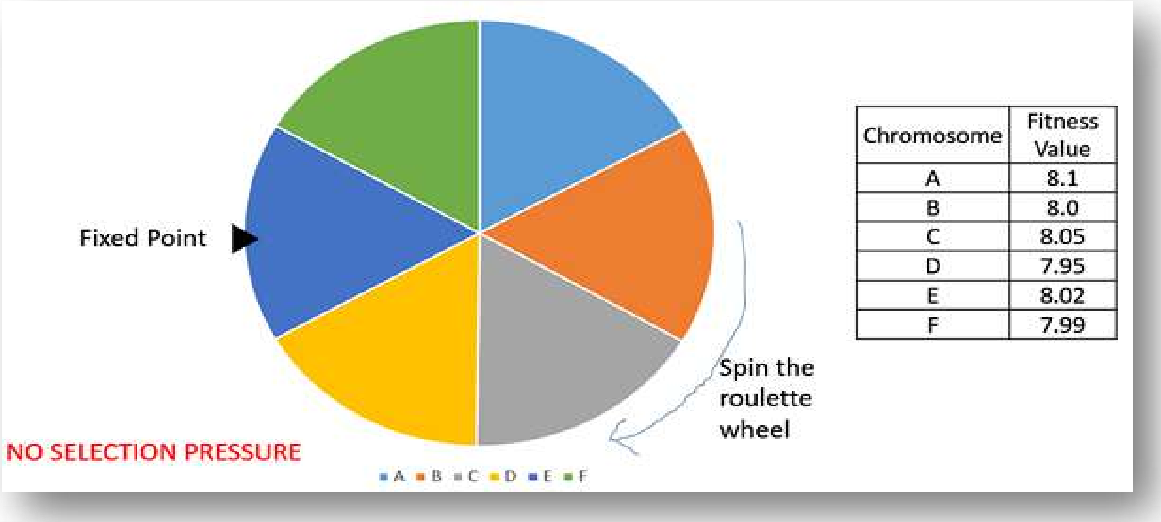
- In K-Way tournament selection, we select K individuals from the population at random and select the best out of these to become a parent. The same process is repeated for selecting the next parent. Tournament Selection is also extremely popular in literature as it can even work with negative fitness values.



Fitness Value	Chromosome
1	Q
5	A
9	Z
8	W
7	S
4	X
2	E
3	F
6	R
2	T
2	Y
1	U
0	I

RANK SELECTION

- Rank Selection also works with negative fitness values and is mostly used when the individuals in the population have very close fitness values (this happens usually at the end of the run). This leads to each individual having an almost equal share of the pie (like in case of fitness proportionate selection) as shown in the following image and hence each individual no matter how fit relative to each other has an approximately same probability of getting selected as a parent. This in turn leads to a loss in the selection pressure towards fitter individuals, making the GA to make poor parent selections in such situations.



Chromosome	Fitness Value
A	8.1
B	8.0
C	8.05
D	7.95
E	8.02
F	7.99

- In this, we remove the concept of a fitness value while selecting a parent. However, every individual in the population is ranked according to their fitness. The selection of the parents depends on the rank of each individual and not the fitness. The higher ranked individuals are preferred more than the lower ranked ones.

Chromosome	Fitness Value	Rank
A	8.1	1
B	8.0	4
C	8.05	2
D	7.95	6
E	8.02	3
F	7.99	5

RANDOM SELECTION

- In this strategy we randomly select parents from the existing population. There is no selection pressure towards fitter individuals and therefore this strategy is usually avoided.

GENETIC ALGORITHMS – CROSSOVER

INTRODUCTION TO CROSSOVER

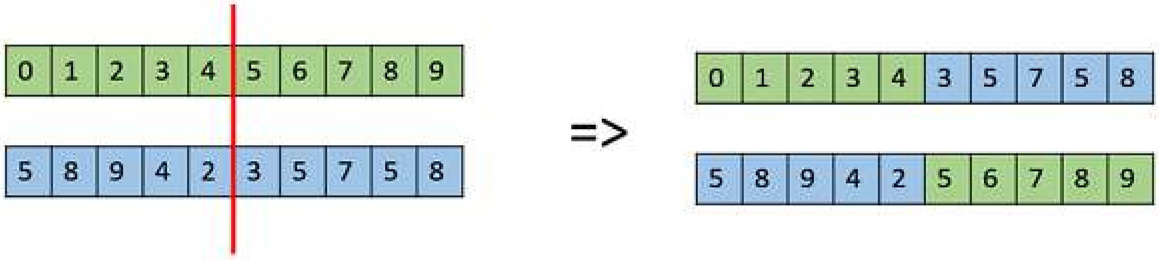
The crossover operator is analogous to reproduction and biological crossover. In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents. Crossover is usually applied in a GA with a high probability – $\boldsymbol{p_c}$.

CROSSOVER OPERATORS

In this section we will discuss some of the most popularly used crossover operators. It is to be noted that these crossover operators are very generic and the GA Designer might choose to implement a problem-specific crossover operator as well.

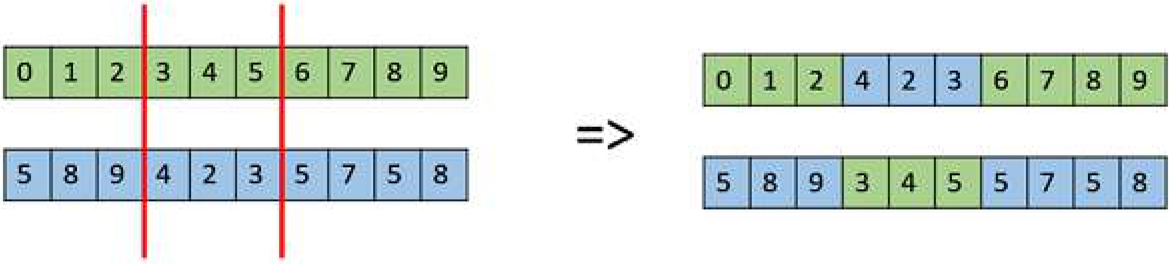
ONE POINT CROSSOVER

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.



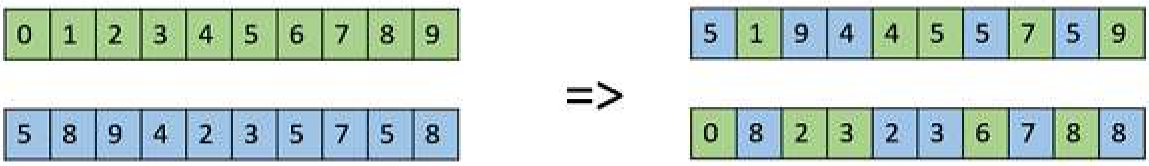
MULTI POINT CROSSOVER

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.



UNIFORM CROSSOVER

In a uniform crossover, we don’t divide the chromosome into segments; rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it’ll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parent.

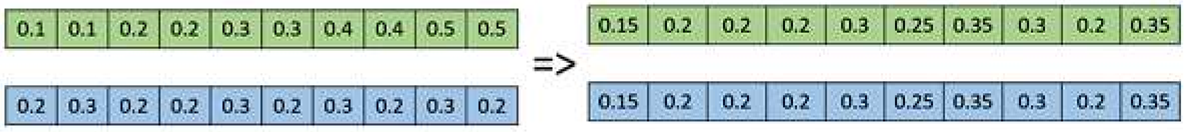


WHOLE ARITHMETIC RECOMBINATION

This is commonly used for integer representations and works by taking the weighted average of the two parents by using the following formulae –

- Child1 = α.x + (1-α).y
- Child2 = α.x + (1-α).y

Obviously, if α = 0.5, then both the children will be identical as shown in the following image.



GENETIC ALGORITHMS – MUTATION

INTRODUCTION TO MUTATION

In simple terms, mutation may be defined as a small random tweak in the chromosome, to get a new solution. It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – p_m. If the probability is very high, the GA gets reduced to a random search.

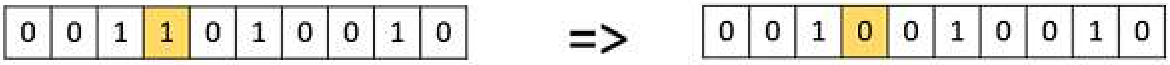
Mutation is the part of the GA which is related to the “exploration” of the search space. It has been observed that mutation is essential to the convergence of the GA while crossover is not.

MUTATION OPERATORS

In this section, we describe some of the most commonly used mutation operators. Like the crossover operators, this is not an exhaustive list and the GA designer might find a combination of these approaches or a problem-specific mutation operator more useful.

BIT FLIP MUTATION

In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.



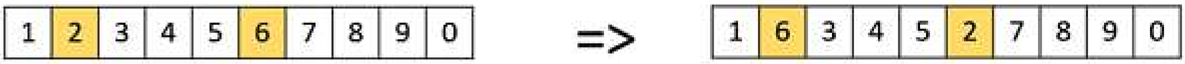
0 0 1 1 0 1 0 0 1 0 => 0 0 1 0 0 1 0 0 1 0

RANDOM RESETTING

Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

SWAP MUTATION

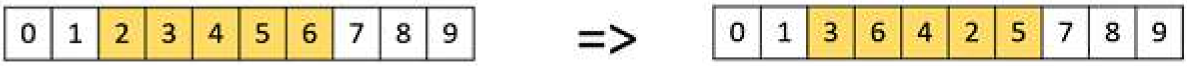
In swap mutation, we select two positions on the chromosome at random, and interchange the values. This is common in permutation based encodings.



1 2 3 4 5 6 7 8 9 0 => 1 6 3 4 5 2 7 8 9 0

SCRAMBLE MUTATION

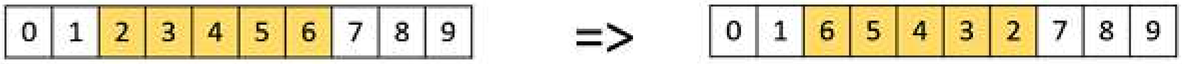
Scramble mutation is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.



0 1 2 3 4 5 6 7 8 9 => 0 1 3 6 4 2 5 7 8 9

INVERSION MUTATION

In inversion mutation, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.



0 1 2 3 4 5 6 7 8 9 => 0 1 6 5 4 3 2 7 8 9

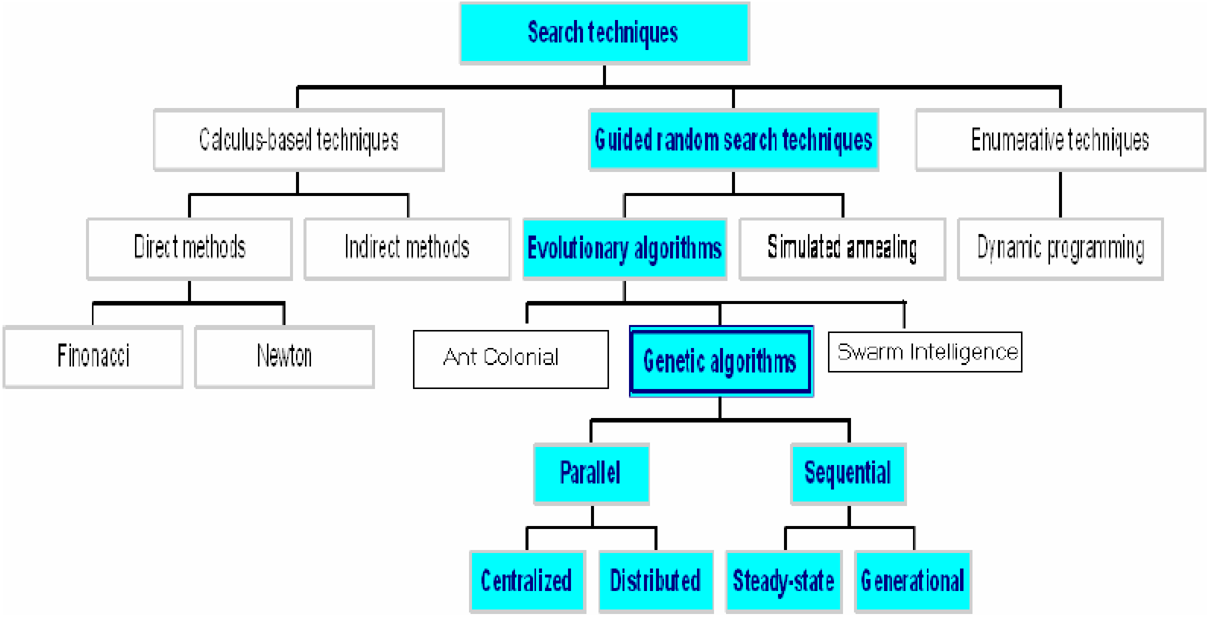
GENETIC ALGORITHMS - APPLICATION AREAS

Genetic Algorithms are primarily used in optimization problems of various kinds, but they are frequently used in other application areas as well.

In this section, we list some of the areas in which Genetic Algorithms are frequently used. These are −

- **Optimization** − Genetic Algorithms are most commonly used in optimization problems wherein we have to maximize or minimize a given objective function value under a given set of constraints. The approach to solve Optimization problems has been highlighted throughout the tutorial.
- **Economics** − GAs are also used to characterize various economic models like the cobweb model, game theory equilibrium resolution, asset pricing, etc.
- **Neural Networks** − GAs are also used to train neural networks, particularly recurrent neural networks.
- **Parallelization** − GAs also have very good parallel capabilities, and prove to be very effective means in solving certain problems, and also provide a good area for research.
- **Image Processing** − GAs are used for various digital image processing (DIP) tasks as well like dense pixel matching.
- **Vehicle routing problems** − With multiple soft time windows, multiple depots and a heterogeneous fleet.
- **Scheduling applications** − GAs are used to solve various scheduling problems as well, particularly the time tabling problem.
- **Machine Learning** − as already discussed, genetics based machine learning (GBML) is a niche area in machine learning.
- **Robot Trajectory Generation** − GAs have been used to plan the path which a robot arm takes by moving from one point to another.
- **Parametric Design of Aircraft** − GAs have been used to design aircrafts by varying the parameters and evolving better solutions.
- **DNA Analysis** − GAs have been used to determine the structure of DNA using spectrometric data about the sample.
- **Multimodal Optimization** − GAs are obviously very good approaches for multimodal optimization in which we have to find multiple optimum solutions.
- **Traveling salesman problem and its applications** − GAs have been used to solve the TSP, which is a well-known combinatorial problem using novel crossover and packing strategies.

Classes of Search Techniques



The principle of Darwinian evolution theory i.e., *survival of the fittest is evaluated by a fitness function* derived from objective function. Every individual in a population searches to be the best according to a fitness function in its own way (randomly).

Basic Concepts:

Optimization means to make the objective function max or min. That means in evolutionary computing where the individuals/ elements represent possible solutions, an element exists such that the fitness of that element is the maximum or minimum among all others‟ fitness depending on it is maximization or minimization problem.

Optimization can be classified as:

1. Deterministic-Uses derivative or gradient to reach final solution

2. Stochastic- Derivative free optimization, a type of random search, suitable for non- linearity, discontinuity escape from local optima and non-convex region

Components of Genetic Algorithm:

The individuals are genes which encode a trait or a parameter. The design space is to be converted to genetic space. It is parallel processing by a population used when single point approach of traditional methods cannot find a possible solution with in the required time frame.

Important common aspects of evolutionary/swarm optimization algorithms:

It is an iterative process where best solution is searched by a population in search space evaluating a fitness function.

1. Search space-Space for all feasible solutions is called search space.
2. Solution- It is the point with maximum or minimum value of fitness function.
3. Fitness function- A function derived from objective function
4. Population size- A number of points in a search space used in parallel for computing is called population, generally ranging from 30 to 200.
5. Constraints- Lower and upper bounds
6. Stopping criteria- it can be no. of iterations, or minimum value of error in fitness or minimum improvement from previous iteration

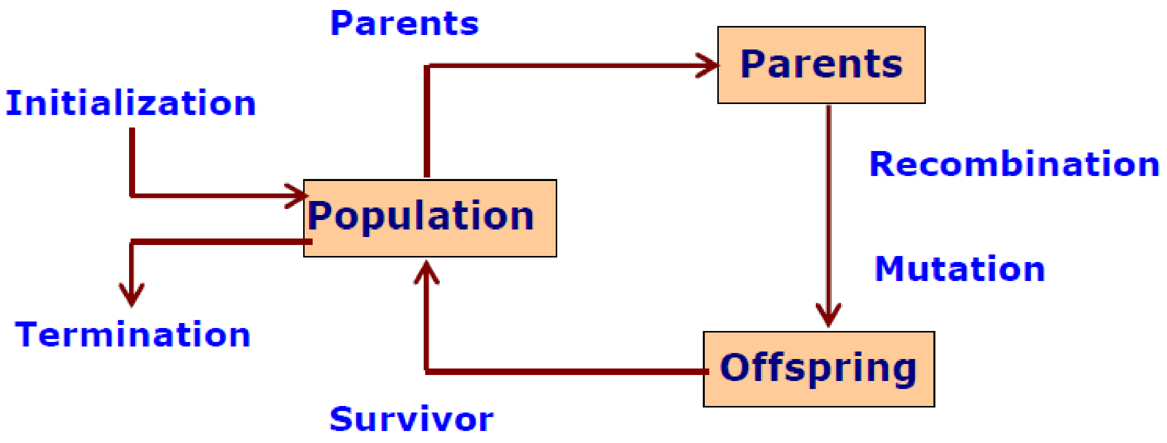
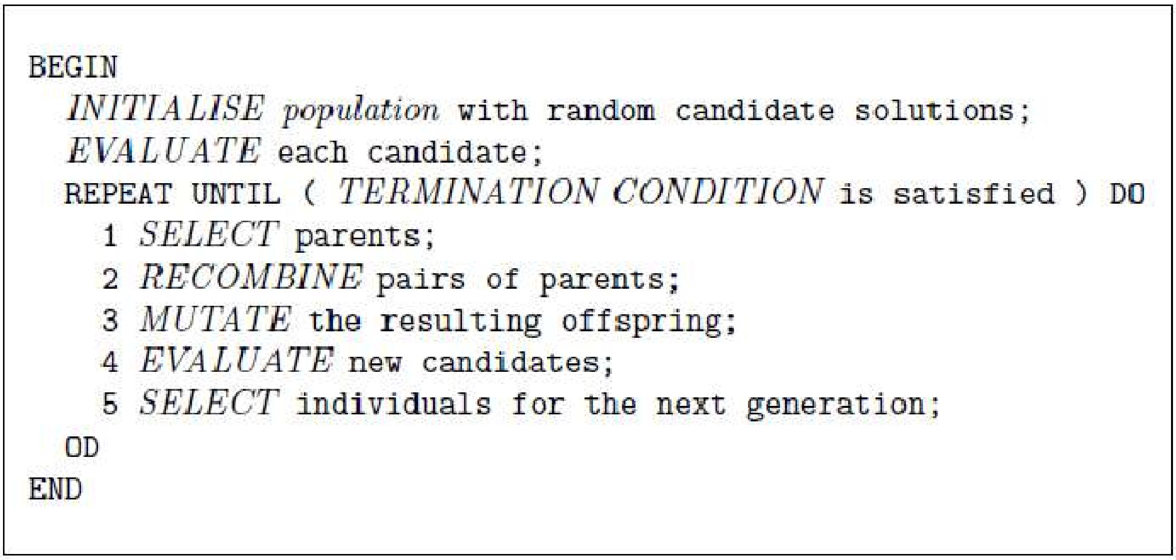


FIG. Basic cycle of EA

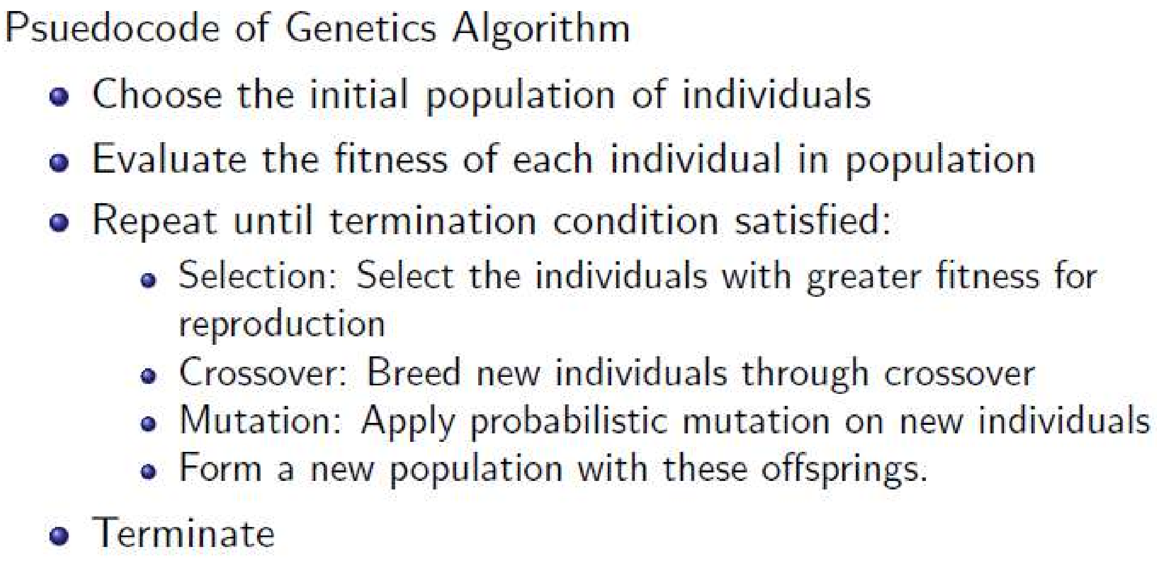
Basic flow chart of EA:

The initial population is usually generated randomly in all EAs. The termination condition may be a desired fitness function, maximum number of generations etc. In selection, individuals with better fitness functions from generation „*i*' are taken to generate individuals of „*i+1*"th generation. New population (*offspring*) is created by applying *recombination* and *mutation* to the selected individuals (*parents*). Recombination creates one or two new individuals by swaping (crossing over) the genome of a *parent* with another. Recombined individual is then *mutated* by changing a single element (genome) to create a new individual.

Finally, the new population is evaluated and the process is repeated. Each step is described in more detail below with special reference to GA. GA was proposed by Davis E. Goldberg.



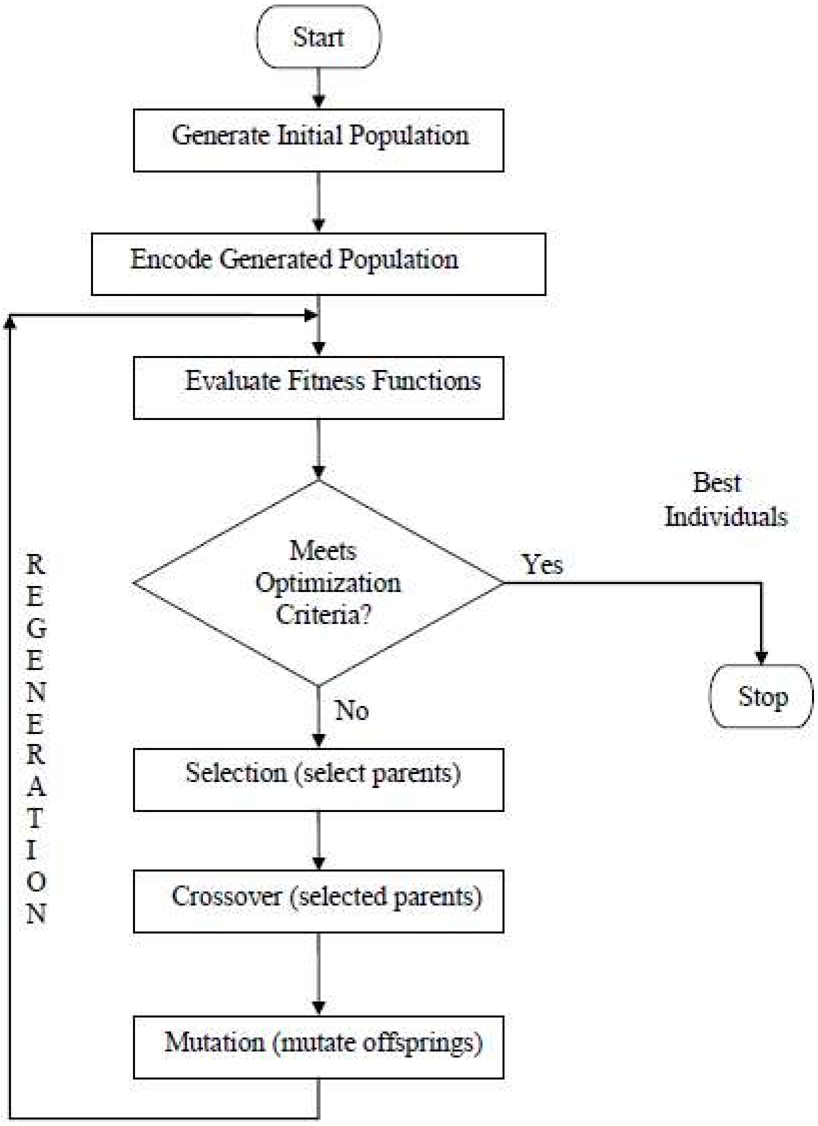
```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```



Psuedocode of Genetics Algorithm

- Choose the initial population of individuals
- Evaluate the fitness of each individual in population
- Repeat until termination condition satisfied:
  - Selection: Select the individuals with greater fitness for reproduction
  - Crossover: Breed new individuals through crossover
  - Mutation: Apply probabilistic mutation on new individuals
  - Form a new population with these offsprings.
- Terminate

FLOW CHART OF GENETIC ALGORITHM:-


Start
Generate Initial Population
Encode Generated Population
Evaluate Fitness Functions
Meets Optimization Criteria?
Yes
Best Individuals
Stop
No
Selection (select parents)
Crossover (selected parents)
Mutation (mutate offsprings)
REGENERATION

ADVANTAGES AND DISADVANTAGES OF EA:

EA can be efficiently used for highly complex problems with multi-objectivity, non-linearity etc. It provides not only a single best solution, but the 2nd best, 3rd best and so on as required. It gives quick approximate solutions. EA methods can very well incorporate with other local search algorithms. There are some drawbacks also in using EA techniques. An optimal solution cannot be ensured on using EA methods, which are usually known as heuristic search methods. Convergence of EA techniques are problem oriented. Sensitivity analysis should be carried out to find out the range in which the model is efficient. Also, the implementation of these techniques requires good programming skill.

Differences and similarities between GA and other traditional methods:

Differences:

1. GA uses coding which discretizes search space even though function is continuous
2. A discrete function can be handled with no extra cost
3. Works with a population of points instead of single, so multiple optimal solutions can be captured at a time, reducing no. of run of algorithm

Similarities:

1. Search direction in traditional algorithms is used to find new point, where as 2 points are used to define search direction a in crossover in GA
2. Search direction is not fixed for all points, as mutation works in GA

New Variants of GA:

According to the encoding, selection, crossover or mutation methods and adaptive changing of the probabilities with convergence many variants like continuous GA, binary GA, RCGA, NSGA etc have been developed. In addition the new ones are

1. Messy GA
2. Parallel GA
3. Multiobjective GA

Issues for GA Parameter settings:

Choosing basic implementation issues:

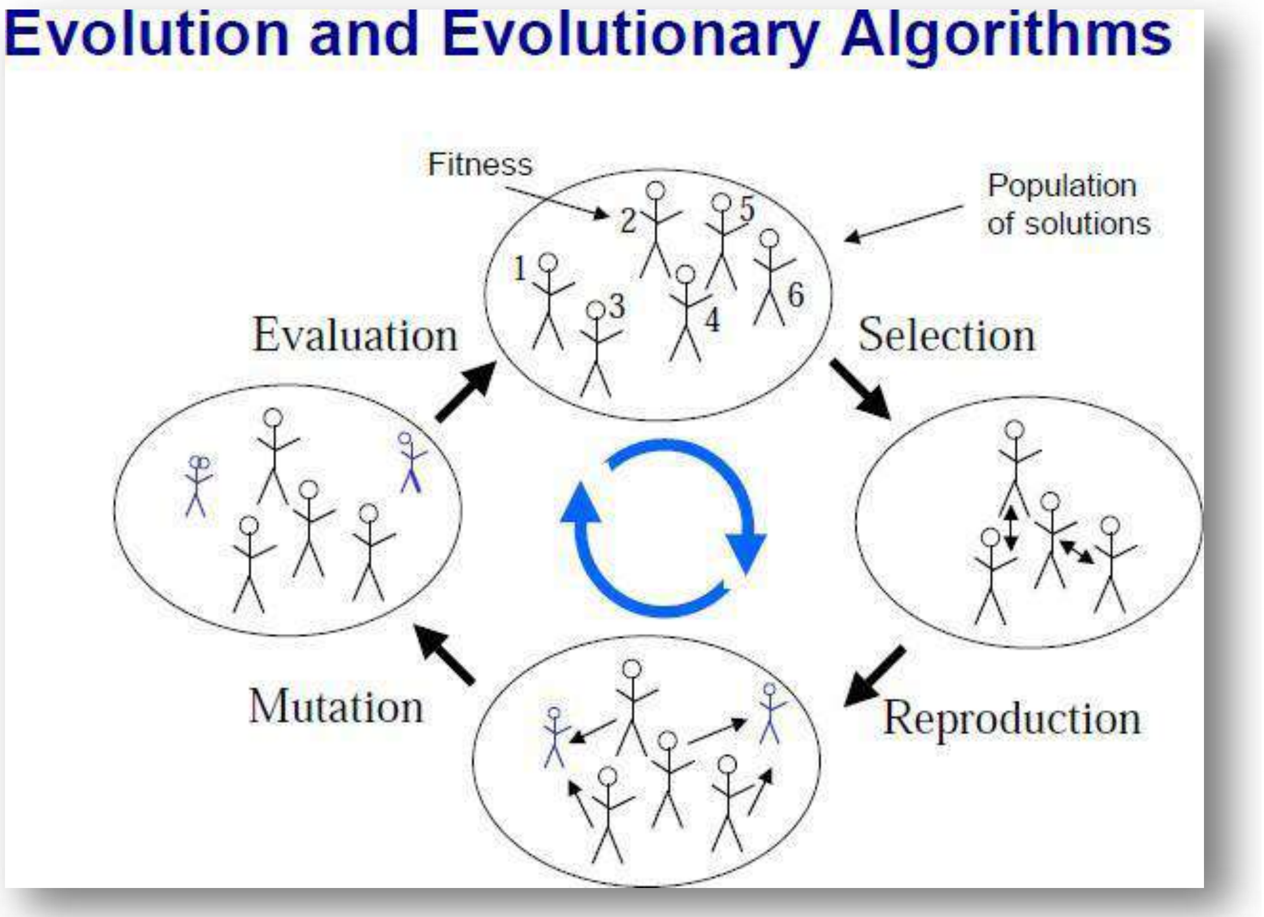
1. Representation
2. Population size, mutation rate, ...
3. Selection, deletion policies
4. Crossover, mutation operators
5. Termination Criteria
6. Performance, scalability
7. Solution is only as good as the evaluation function (often hardest part)

Benefits of Genetic Agorithms

1. Concept is easy to understand
2. Modular, separate from application
3. Supports multi-objective optimization
4. Good for “noisy”environments
5. Always gives answer; answer gets better with time
6. Inherently parallel; easily distributed
7. Multiple ways to speed up and improve a GA-based application as knowledge about problem domain is gained
8. Easy to exploit previous or alternate solutions
9. Flexible building blocks for hybrid applications
10. Substantial history and range of use

MODULE-IV

Evolutionary Computing, Simulated Annealing, Random Search, Downhill Simplex Search, Swarm optimization



Evolution and Evolutionary Algorithms

DRAWBACK OF TRADITIONAL TECHNIQUES

Computing tasks have to be

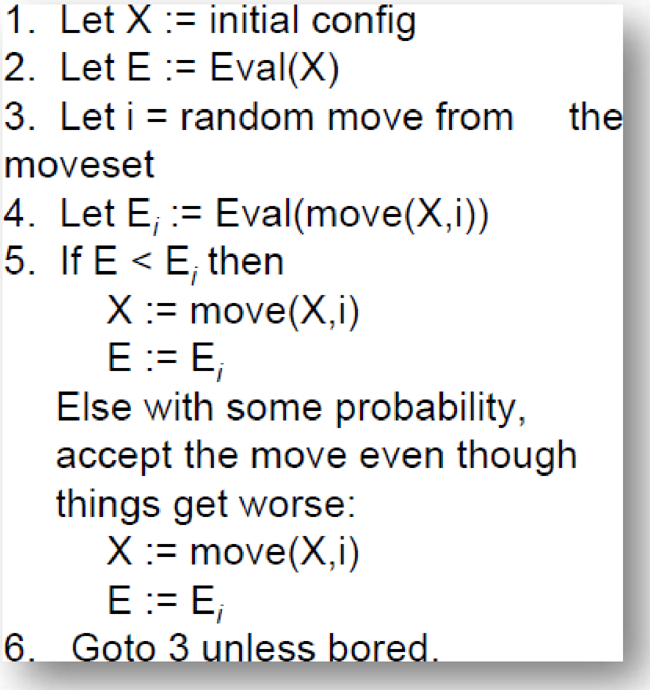
1. well-defined
2. fairly predictable
3. computable in reasonable time with serial computers

ANALOGY-BASED ALGORITHMS

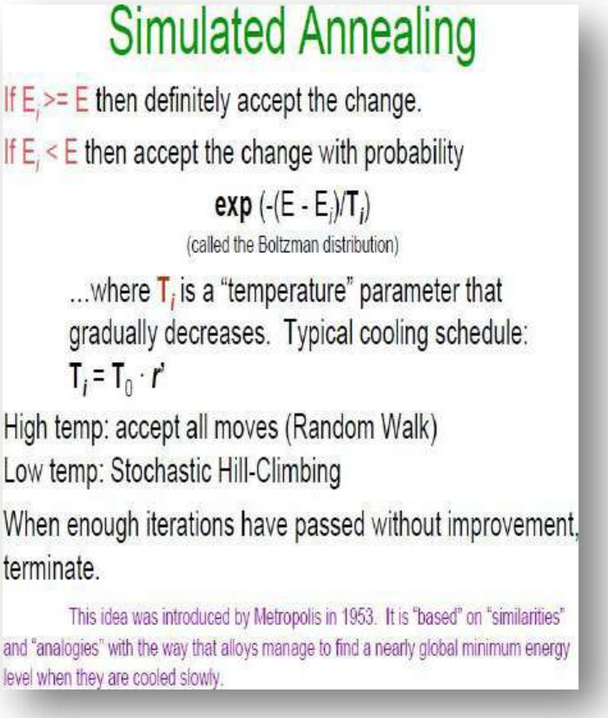
For any natural phenomenon you can think of, there will be at least one AI research group that will have a combinatorial optimization algorithm "based" on "analogies" and "similarities" with the phenomenon. Here's the beginning of the list…

1. Metal cooling annealing
2. Evolution / Co-evolution / Sexual Reproduction
3. Thermodynamics
4. Societal Markets
5. Management Hierarchies
6. Ant/Insect Colonies
7. Immune System
8. Animal Behaviour Conditioning
9. Neuron / Brain Models
10. Hill-climbing (okay, that's a stretch…)
11. Particle Physics

SIMULATED ANNEALING



```
1. Let X := initial config
2. Let E := Eval(X)
3. Let i = random move from the moveset
4. Let $E_i$ := Eval(move(X,i))
5. If E < $E_i$ then
        X := move(X,i)
        E := $E_i$
   Else with some probability,
   accept the move even though
   things get worse:
        X := move(X,i)
        E := $E_i$
6. Goto 3 unless bored.
```



Simulated Annealing

If E_i >= E then definitely accept the change.

If E_i < E then accept the change with probability

$$\exp(-(E - E_i)/T_i)$$

(called the Boltzman distribution)

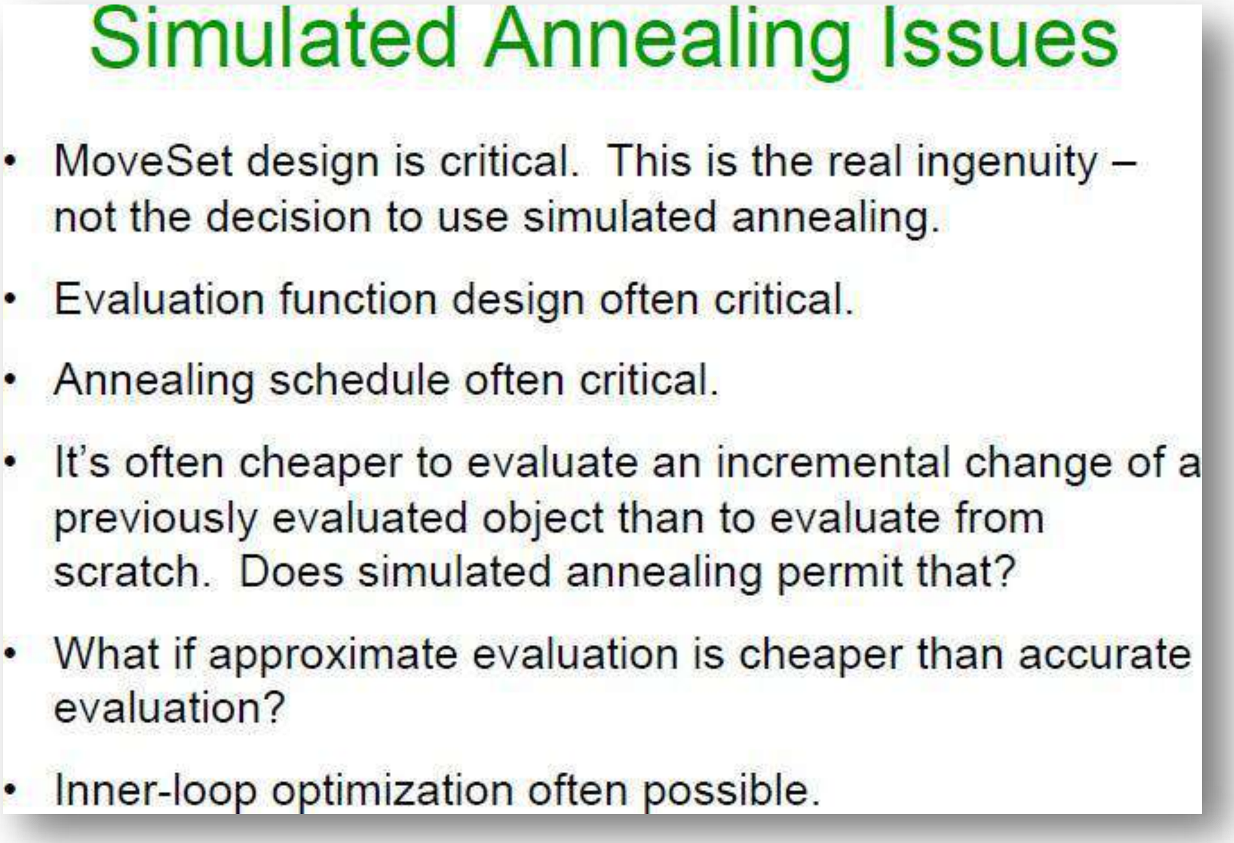
…where T_i is a "temperature" parameter that gradually decreases. Typical cooling schedule: $T_i = T_0 \cdot r^i$

High temp: accept all moves (Random Walk)

Low temp: Stochastic Hill-Climbing

When enough iterations have passed without improvement, terminate.

This idea was introduced by Metropolis in 1953. It is "based" on "similarities" and "analogies" with the way that alloys manage to find a nearly global minimum energy level when they are cooled slowly.



Simulated Annealing Issues

- MoveSet design is critical. This is the real ingenuity – not the decision to use simulated annealing.
- Evaluation function design often critical.
- Annealing schedule often critical.
- It's often cheaper to evaluate an incremental change of a previously evaluated object than to evaluate from scratch. Does simulated annealing permit that?
- What if approximate evaluation is cheaper than accurate evaluation?
- Inner-loop optimization often possible.

ADVANTAGES OF SIMULATED ANNEALING

1. Simulated annealing is sometimes empirically much better at avoiding local minima than hill-climbing. It is a successful, frequently-used, algorithm. Basic hill climbing algorithm is so prone to getting caught in local optimums.
2. This is because a hill climber algorithm will simply accept neighbour solutions that are better than the current solution. When the hill climber can't find any better neighbours, it stops.
3. Not much opportunity to say anything formal about it (though there is a proof that with an infinitely slow cooling rate, you'll find the global optimum).

RANDOM SEARCH ALGORITHMS

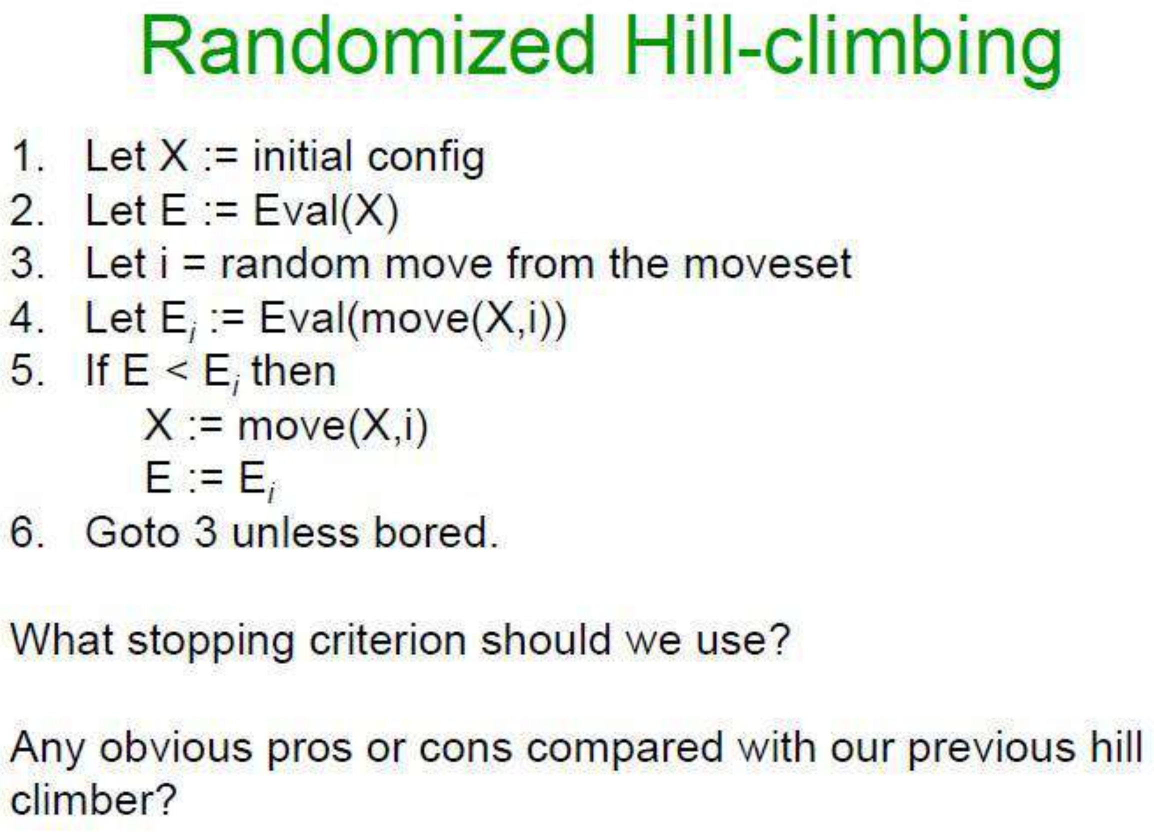
Random search algorithms are useful for many ill-structured global optimization problems with continuous and/or discrete variables. Typically random search algorithms sacrifice a guarantee of optimality for finding a good solution quickly with convergence results in probability. Random search algorithms include simulated annealing, tabu search, genetic algorithms, evolutionary programming, particle swarm optimization, ant colony optimization, cross-entropy, stochastic approximation, multistart and clustering algorithms, to name a few. They may be categorized as global (exploration) versus local (exploitation) search, or instance- based versus model-based.

However, one feature these methods share is the use of probability in determining their iterative procedures. This article provides an overview of these random search algorithms, with a probabilistic view that ties them together. A random search algorithm refers to an algorithm that uses some kind of randomness or probability (typically in the form of a pseudo-random number generator) in the definition of the method, and in the literature, may be called a Monte Carlo method or a stochastic algorithm. The term met heuristic is also commonly associated with random search algorithms.

GENERIC RANDOM SEARCH ALGORITHM

- **STEP 0.** Initialize algorithm parameters ⍰ 0, initial points X0 ⍰ S and iteration index k = 0.
- **STEP 1.** Generate a collection of candidate points Vk+1 ⍰ S according to a specific generator and associated sampling distribution.
- **STEP 2.** Update Xk+1 based on the candidate points Vk+1, previous iterates and algorithmic parameters. Also update algorithm parameters ⍰ k+1.
- **STEP 3.** If a stopping criterion is met, stop. Otherwise increment k and return to Step 1.
- This generic random search algorithm depends on two basic procedures, the generator in Step 1 that produces candidate points, and the update procedure in Step 2.

HILL CLIMBING



Randomized Hill-climbing

```
1. Let X := initial config
2. Let E := Eval(X)
3. Let i = random move from the moveset
4. Let $E_i$ := Eval(move(X,i))
5. If E < $E_i$ then
       X := move(X,i)
       E := $E_i$
6. Goto 3 unless bored.
```

What stopping criterion should we use?

Any obvious pros or cons compared with our previous hill climber?

SWARM OPTIMIZATION

Swarm intelligence (SI) is the collective behaviour of decentralized, self- organized systems, natural or artificial. SI systems consist typically of a population of simple agents or boids interacting locally with one another and with their environment. The inspiration often comes from nature, especially biological systems. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local, and to a certain degree random, interactions between such agents lead to the emergence of "intelligent" global behaviour, unknown to the individual agents. Examples in natural systems of SI include ant colonies, bird flocking, animal herding, bacterial growth, fish schooling and Microbial intelligence. It is infact a multi-agent system that has self-organized behaviour that shows some intelligent behaviour.

Two principles in swarm intelligence

Self-Organization is based on:

- activity amplification by positive feedback activity balancing by negative feedback amplification of random fluctuations multiple interactions

Stigmergy - stimulation by work - is based on:

- work as behavioural response to the environmental state an environment that serves as a work state memory work that does not depend on specific agents

ROOTS IN MODELS OF SOCIAL INSECTS BEHAVIOR:

- Foraging Behaviour
- Division Of Labour And Task Allocation
- Cemetery Organization
- Nest Building
- Collaborative Transport

PROPERTIES OF COLLECTIVE INTELLIGENCE SYSTEMS:

- Distributed computation
- Direct and indirect interactions
- Agents equipped with simple computational capabilities
- Robustness
- Adaptiveness

Multiple interactions among agents Simple agents (e.g., rule based)
Systems composed of many agents
Positive feedback

- Amplification of random fluctuations and structure formation Reinforcement of most common behaviour patterns

Negative feedback

- Saturation Competition Resource exhaustion

PARTICLE SWARM OPTMIZATION

- Population initialized by assigning random positions and velocities; potential solutions are then flown through hyperspace.
- Each particle keeps track of its "best" (highest fitness) position in hyperspace.
- This is called "pbest" for an individual particle
- It is called "gbest" for the best in the population
- It is called "lbest" for the best in a defined neighbourhood
- At each time step, each particle stochastically accelerates toward its pbest and gbest (or lbest).

ANT ALGORITHM:

Algorithms inspired by the behaviour of real ants Examples:

- Foraging
- Corpse clustering
- Division of labour

While walking ants deposit a substance called ***pheromone*** on the ground. They choose with higher probability paths that are marked by stronger pheromone concentrations.
Cooperative interaction which leads to the emergence of short(est) paths.

BEES ALGORITHM:

- The queen moves randomly over the combs eggs are more likely to be put down in the neighbourhood of brood
- honey and pollen are deposited randomly in empty cells
- four times more honey is brought to the hive than pollen
- removal ratios for honey: 0.95; pollen: 0.6
- removal of honey and pollen is proportional to the number of surrounding cells containing brood

SWARM OPTIMIZATION APPLICATIONS:

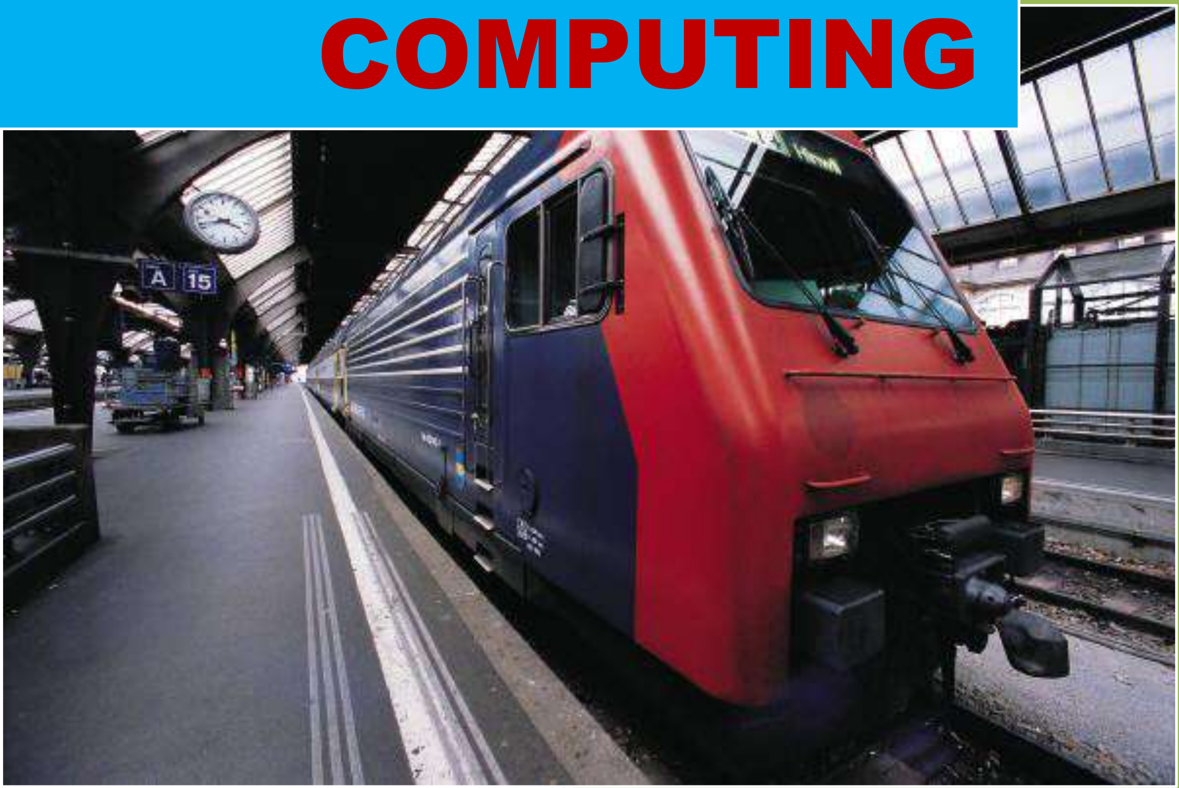
1. Combinatorial optimization
2. Mixed integer-continuous optimization
3. Networks: AntNet
4. Data clustering and exploratory data analysis
5. Coordinated motion
6. Self-assembling

NO FREE LUNCH THEOREM:

A number of "no free lunch" (NFL) theorems are establish that for any algorithm, any elevated performance over one class of problems is offset by performance over another class. These theorems result in a geometric interpretation of what it means for an algorithm to be well suited to an optimization problem. Applications of the NFL theorems to information-theoretic aspects of optimization and benchmark measures of performance are also presented.

2020

QUESTION BANK – SOFT COMPUTING



ER. SATYARANJAN DAS

DEPARTMENT OF ELECTRICAL ENGINEERING, ABIT, CUTTACK

1. What is Soft Computing? Explain it is different from hard computing, and its application Or Discuss problem solving techniques?
2. Discuss different techniques used in Soft Computing, its applications?
3. Difference between BNN and ANN?
4. Explain Biological Neural network in terms of Axom, Synapse, dendrites, Synaptic Gap?
5. What is Percetron Model, and explain activation function, NET, Target output, Actual output, Error?
6. Explain Delta Rule in brief?
7. Explain McCulloch–Pitts Neuron in brief?
8. Define and explain the meaning of term “Artificial Intelligence”?
9. Differentiate AI problems Vs Conventional problems?
10. Explain Production System its applications and types?
11. Explain Depth First Search, Best First Search, Breadth First Search?
12. Explain Hill Climbing Procedure and its applications?
13. Explain A* and AO* algorithm?
14. What are the important issues in knowledge representation?
15. Explain Linear Seprability using an example? Or Is XOR Gate Linear Separable?
16. Explain Hebb Learning in brief?
17. Explain Learning in Neural Network? Or Explain Supervised Learning, Unsupervised Learning, Reinforcement Learning?
18. Explain ADALINE and MADALINE Network?
19. Explain Gradient Descent method?
20. Explain Backpropagation Training Algorithm in brief, and discuss applications of it?
21. Explain Radial Basis Function Network (RBFN) in brief?
22. Discuss Associative Memory in brief?
23. Discuss Unsupervised Learning Networks? Short note on KSOM, CPN, ART?
24. What is Competitive Learning?
25. Differentiate between ART1 and ART2?
26. Discuss Boltzmann Machine in brief?
27. What are Applications of Neural network?
28. Explain Feature map? Discuss Kohonen Self Organization Map(KSOM) in brief?
29. Explain Counter Propagation in Brief? Explain full counter and feed counter propagation network?
30. Explain Adaptive Resonance Theory (ART) , and also explain ART1 and ART2?
31. Explain Fuzzy Logic, fuzzy set?
32. Explain Fuzzy Set Vs Crisp Set?
33. Explain Membership function, fuzzy set, fuzzy if-then Rules?
34. Explain Fuzzy Inference System (FIS)?
35. Explain Mamdani model and Takagi model?
36. Explain neuro fuzzy hybrid, neuro genetic hybrid and fuzzy genetic hybrid system?

37. Explain Genetic Algorithm in term of individual, gene, fitness, population, encoding, selection, crossover, mutation?
38. Explain different type of cross over operation?
39. Explain Generalized Modus Ponen and Generalized Modus Tonen?
40. Explain Modus ponen, modus tonen, if-then rule?
41. Explain Max-min Composition and Max-product composition?
42. Explain Union, Intersection, complement operations of fuzzy set?
43. Explain Classical set Vs Fuzzy Set?
44. Comparison between Mamdani and Takagi Model?
45. How Genetic Algorithm is different from traditional algorithms?
46. Explain Genetic algorithm in terms of Reproduction, Selection, Evaluation and Replacement?
47. Write short notes on Roulette wheel Selection, Random selection, Tournament Selection, Boltzman Selection?
48. Discuss Crossover operation in GA and its types?
49. Discuss Mamdani Model for FIS?
50. Discuss Takagi Model for FIS?
51. Discuss different Defuzzification methods?
52. Difference between fuzzification and defuzzfication?
53. Explain following terms Core, Boundary, Support in term of Fuzzy Logic?
54. Explain Fuzzy Expert System?
55. Discuss different encoding techniques used for GA?

1. Explain different parts of human brain.
2. Explain model of an artificial neuron.
3. Explain Adaline.
4. Explain Neural Network architecture.
5. Explain Unsupervised Learning Neural Networks.
6. Explain Supervised Learning Neural Networks.
7. Explain Competitive Learning Networks .
8. Explain Kohonen Self-Organizing Networks.
9. Explain Hebbian's Learning.
10. Explain backpropagation algorithm.
11. Explain learning in neural network.
12. Write application of neural network.
13. Explain basic Fuzzy Set Operations.
14. Explain Membership Function.
15. Explain Fuzzy Rule based system.
16. Differentiate fuzzy and crisp sets.
17. Explain Fuzzy Inference Systems
18. Write application of Fuzzy logic.
19. Explain Fuzzy Relations.
20. Explain Fuzzy Reasoning.
21. Explain Adaptive Neuro-Fuzzy Inference Systems
22. Difference between Traditional Algorithms and Genetic Algorithm.
23. Explain creation of offspring.
24. Explain binary encoding.
25. Explain octal encoding.
26. Explain permutation encoding.
27. Explain fitness function.
28. Explain roulette wheel selection.
29. Explain Boltzman selection.
30. Explain reproduction process.
31. Explain Cross Over process.
32. Explain Neuro-Fuzzy Systems.
33. Write application of genetic algorithm.
34. Explain ANFIS.
35. Explain RBFN.
36. Explain Evolving Connectionist model.
37. Write applications for Adaptive Systems
38. Explain fuzzy associative memory.
39. Explain Neuro-Genetic hybrid Systems.
40. Explain genetic algorithm based backpropagation network.

Core of soft Computing is

A. Fuzzy Computing, Neural Computing, Genetic Algorithms
B. Fuzzy Networks and Artificial Intelligence
C. Artificial Intelligence and Neural Science
D. Neural Science and Genetic Science
ANSWER: A

Who initiated the idea of Soft Computing

A. Charles Darwin
B. Lofti A Zadeh
C. Rechenberg
D. Mc_Culloch
ANSWER: B

Fuzzy Computing

A. mimics human behaviour
B. doesn’t deal with 2 valued logic
C. deals with information which is vague, imprecise, uncertain, ambiguous, inexact, or probabilistic
D. All of the above
ANSWER: D

Neural Computing

A. mimics human brain
B. information processing paradigm
C. Both (a) and (b)
D. None of the above
ANSWER: C

Genetic Algorithm is a part of

A. Evolutionary Computing
B. inspired by Darwin's theory about evolution - "survival of the fittest"
C. are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics
D. All of the above
ANSWER: D

Supervised Learning is

A. learning with the help of examples
B. learning without teacher
C. learning with the help of teacher
D. learning with computers as supervisor
ANSWER: C

Unsupervised learning is

A. learning without computers
B. problem based learning
C. learning from environment
D. learning from teachers
ANSWER: C

Conventional Artificial Intelligence is different from soft computing in the sense

A. Conventional Artificial Intelligence deal with prdicate logic where as soft computing deal with fuzzy logic
B. Conventional Artificial Intelligence methods are limited by symbols where as soft computing is based on empirical data
C. Both (a) and (b)
D. None of the above
ANSWER: C

In supervised learning

A. classes are not predefined
B. classes are predefined
C. classes are not required
D. classification is not done
ANSWER: B

ANN is composed of large number of highly interconnected processing elements(neurons) working in unison to solve problems.

A. True
B. False
ANSWER: A

Artificial neural network used for

A. Pattern Recognition
B. Classification
C. Clustering
D. All of these
ANSWER: D

A Neural Network can answer

A. For Loop questions
B. what-if questions
C. IF-The-Else Analysis Questions
D. None of these
ANSWER: B

Ability to learn how to do tasks based on the data given for training or initial experience

A. Self Organization
B. Adaptive Learning
C. Fault tolerance
D. Robustness
ANSWER: B

Feature of ANN in which ANN creates its own organization or representation of information it receives during learning time is

A. Adaptive Learning
B. Self Organization
C. What-If Analysis
D. Supervised Learning
ANSWER: B

In artificial Neural Network interconnected processing elements are called

A. nodes or neurons
B. weights
C. axons
D. Soma
ANSWER: A

Each connection link in ANN is associated with ________ which has information about the input signal.

A. neurons
B. weights
C. bias
D. activation function
ANSWER: B

Neurons or artificial neurons have the capability to model networks of original neurons as found in brain

A. True
B. False
ANSWER: A

Internal state of neuron is called __________, is the function of the inputs the neurons receives

A. Weight
B. activation or activity level of neuron
C. Bias
D. None of these
ANSWER: B

Neuron can send ________ signal at a time.

A. multiple
B. one
C. none
D. any number of
ANSWER: B

Artificial intelligence is

A. It uses machine-learning techniques. Here program can learn From past experience and adapt themselves to new situations
B. Computational procedure that takes some value as input and produces some value as output.
C. Science of making machines performs tasks that would require intelligence when performed by humans

D. None of these

ANSWER: C

Expert systems

A. Combining different types of method or information
B. Approach to the design of learning algorithms that is structured along the lines of the theory of evolution
C. an information base filled with the knowledge of an expert formulated in terms of if-then rules
D. None of these

ANSWER: C

Falsification is

A. Modular design of a software application that facilitates the integration of new modules
B. Showing a universal law or rule to be invalid by providing a counter example
C. A set of attributes in a database table that refers to data in another table
D. None of these

ANSWER: B

Evolutionary computation is

A. Combining different types of method or information
B. Approach to the design of learning algorithms that is structured along the lines of the theory of evolution.
C. Decision support systems that contain an information base filled with the knowledge of an expert formulated in terms of if-then rules.
D. None of these

ANSWER: B

Extendible architecture is

A. Modular design of a software application that facilitates the integration of new modules
B. Showing a universal law or rule to be invalid by providing a counter example
C. A set of attributes in a database table that refers to data in another table
D. None of these

ANSWER: A

Massively parallel machine is

A. A programming language based on logic
B. A computer where each processor has its own operating system, its own memory, and its own hard disk
C. Describes the structure of the contents of a database.
D. None of these
ANSWER: B

Search space

A. The large set of candidate solutions possible for a problem
B. The information stored in a database that can be, retrieved with a single query.
C. Worth of the output of a machine learning program that makes it understandable for humans
D. None of these
ANSWER: A

n(log n) is referred to

A. A measure of the desired maximal complexity of data mining algorithms
B. A database containing volatile data used for the daily operation of an organization
C. Relational database management system
D. None of these
ANSWER: A

Perceptron is

A. General class of approaches to a problem.
B. Performing several computations simultaneously
C. Structures in a database those are statistically relevant
D. Simple forerunner of modern neural networks, without hidden layers
ANSWER: D

Prolog is
A. A programming language based on logic
B. A computer where each processor has its own operating system, its own memory, and its own hard disk
C. Describes the structure of the contents of a database
D. None of these
ANSWER: A

Shallow knowledge

A. The large set of candidate solutions possible for a problem
B. The information stored in a database that can be, retrieved with a single query
C. Worth of the output of a machine learning program that makes it understandable for humans
D. None of these
ANSWER: B

Quantitative attributes are

A. A reference to the speed of an algorithm, which is quadratically dependent on the size of the data
B. Attributes of a database table that can take only numerical values
C. Tools designed to query a database
D. None of these
ANSWER: B

Subject orientation

A. The science of collecting, organizing, and applying numerical facts
B. Measure of the probability that a certain hypothesis is incorrect given certain observations.
C. One of the defining aspects of a data warehouse, which is specially built around all the existing applications of the operational data
D. None of these
ANSWER: C

Vector

A. It do not need the control of the human operator during their execution
B. An arrow in a multi-dimensional space. It is a quantity usually characterized by an ordered set of scalars
C. The validation of a theory on the basis of a finite number of examples
D. None of these
ANSWER: B

Transparency

A. The large set of candidate solutions possible for a problem
B. The information stored in a database that can be retrieved with a single query
C. Worth of the output of a machine learning program that makes it understandable for humans
D. None of these
ANSWER: C

Membership function defines the fuzziness in a fuzzy set irrespective of the elements in the set, which are discrete or continuous.

A. True
B. False
ANSWER: A

The membership functions are generally represented in

A. Tabular Form
B. Graphical Form
C. Mathematical Form
D. Logical Form
ANSWER: B

Membership function can be thought of as a technique to solve empirical problems on the basis of

A. knowledge
B. examples
C. learning
D. experience
ANSWER: D

Three main basic features involved in characterizing membership function are
A. Intution, Inference, Rank Ordering
B. Fuzzy Algorithm, Neural network, Genetic Algorithm
C. Core, Support , Boundary
D. Weighted Average, centre of Sums, Median
ANSWER: C

The region of universe that is characterized by complete membership in the set is called

A. Core
B. Support
C. Boundary
D. Fuzzy
ANSWER: A

A fuzzy set whose membership function has at least one element x in the universe whose membership value is unity is called

A. sub normal fuzzy sets
B. normal fuzzy set
C. convex fuzzy set
D. concave fuzzy set
ANSWER: B

In a Fuzzy set a prototypical element has a value

A. 1
B. 0
C. infinite
D. Not defined
ANSWER: A

A fuzzy set wherein no membership function has its value equal to 1 is called

A. normal fuzzy set
B. Subnormal fuzzy set.
C. convex fuzzy set
D. concave fuzzy set
ANSWER: B

A fuzzy set has a membership function whose membership values are strictly monotonically increasing or strictly monotonically decreasing or strictly monotonically increasing than strictly monotonically decreasing with increasing values for elements in the universe

A. convex fuzzy set
B. concave fuzzy set
C. Non concave Fuzzy set
D. Non Convex Fuzzy set
ANSWER: A

The membership values of the membership function are not strictly monotonically increasing or decreasing or strictly monotonically increasing than decreasing.

A. Convex Fuzzy Set
B. Non convex fuzzy set
C. Normal Fuzzy set
D. Sub normal fuzzy set
ANSWER: B

The crossover points of a membership function are defined as the elements in the universe for which a particular fuzzy set has values equal to

A. infinite
B. 1
C. 0
D. 0.5
ANSWER: D

ASSIGNMENT 1

1. Describe working biological neuron
2. Classify the working operation difference between Biological and Artificial Neuron with a neat diagram
3. Differentiate between Auto Associative and Hetero Associative Memory.
4. Classify the types of Learning Methods
5. Describe any two factors affecting the Back-Propagation Training

ASSIGNMENT 2

1. Differentiate between types of learning rules
2. Describe the Fuzzy Logic Controller in detail.
3. Illustrate the Fuzzy inference procedures involved in designing Fuzzy Logic Controllers
4. Describe any two parameters that are used for selection in training Back-Propagation Neural Network.
5. What are the important aspects of using Genetic Algorithms

ASSIGNMENT 3

1. Describe mutation operation used in genetic modelling
2. Consider A = {(x1,0.2), (x2,0.7), (x3,0.4)} and B = {(y1,0.5),(y2,0.6)} are two fuzzy sets defined in the universe of discourse X = {x1,x2,x3} and Y = {y1,y2} respectively. Find cartesian product of A and B.
3. Elaborate any one application based on Fuzzy Logic Controller in detail.
4. Describe any two methods of Defuzzification.
5. Describe max-min composition in Fuzzy Logic Control.

ASSIGNMENT 4

1. Illustrate the Fuzzy inference procedures involved in designing Fuzzy Logic Controllers
2. Consider A = {(x1,0.2), (x2,0.7), (x3,0.4)} and B = {(y1,0.5),(y2,0.6)} are two fuzzy sets defined in the universe of discourse X = {x1,x2,x3} and Y = {y1,y2} respectively. Find cartesian product of A and B.
3. Elaborate any one application based on Fuzzy Logic Controller in detail.
4. Describe any two methods of Defuzzification.
5. Describe max-min composition in Fuzzy Logic Control.

ASSIGNMENT 5

1. Describe any two parameters that are used for selection in training Back-Propagation Neural Network.
2. What are the important aspects of using Genetic Algorithms?
3. Describe mutation operation used in genetic modelling
4. Explain the selection procedure for Genetic algorithm
5. Describe any one application of Genetic Algorithm in detail

TUTORIAL 1

1. List the factors affecting the Back-Propagation Training
2. Describe the properties of classical sets.
3. Define Cardinality of Classical Set and Fuzzy Set with an example
4. What is the use of Fuzzification and Defuzzification?
5. Describe the Fuzzy Logic Controller in detail.
6. Mention the Fuzzy inference procedures involved in designing Fuzzy Logic Controllers
7. What are the important aspects of using Genetic Algorithms
8. List the three simple Genetic Algorithm Operators which are largely used.
9. Consider A = {(x1,0.2), (x2,0.7), (x3,0.4)} and B = {(y1,0.5),(y2,0.6)} are two fuzzy sets defined in the universe of discourse X = {x1,x2,x3} and Y = {y1,y2} respectively. Find cartesian product of A and B.
10. Describe the properties of Fuzzy Sets.
11. Describe any two methods of Defuzzification
12. Elaborate any one application of Fuzzy Logic Controller
13. Describe the operation of Mutation in Genetic Algorithm in detail.
14. Write the differences between Fuzzy Sets and Classical Sets
15. What are the steps involved in designing a Fuzzy Logic Controller?
16. Describe any one application of Genetic Algorithms

TUTORIAL 2

1. Describe any two factors factors affecting the Back-Propagation Training
2. Describe the Fuzzy Logic Controller in detail.
3. Illustrate the Fuzzy inference procedures involved in designing Fuzzy Logic Controllers
4. Describe any two parameters that are used for selection in training Back-Propagation Neural Network.
5. What are the important aspects of using Genetic Algorithms
6. Describe mutation operation used in genetic modelling
7. Consider A = {(x1,0.2), (x2,0.7), (x3,0.4)} and B = {(y1,0.5),(y2,0.6)} are two fuzzy sets defined in the universe of discourse X = {x1,x2,x3} and Y = {y1,y2} respectively. Find cartesian product of A and B.

8. Elaborate any one application based on Fuzzy Logic Controller in detail.
9. Describe any two methods of Defuzzification.
10. Describe max-min composition in Fuzzy Logic Control.

SHORT ANSWER TYPE QUESTION ON ANN

1. What is Artificial Intelligence? Give an example of where AI is used on a daily basis.

ANSWER:- "Artificial Intelligence (AI) is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans." "The capability of a machine to imitate the intelligent human behavior."

2. What are the different types of AI?
   - Reactive Machines AI: Based on present actions, it cannot use previous experiences to form current decisions and simultaneously update their memory. Example: Deep Blue
   - Limited Memory AI: Used in self-driving cars. They detect the movement of vehicles around them constantly and add it to their memory.
   - Theory of Mind AI: Advanced AI that has the ability to understand emotions, people and other things in the real world.
   - Self Aware AI: AIs those posses human-like consciousness and reactions. Such machines have the ability to form self-driven actions.
   - Artificial Narrow Intelligence (ANI): General purpose AI, used in building virtual assistants like Siri.
   - Artificial General Intelligence (AGI): Also known as strong AI. An example is the Pillo robot that answers questions related to health.
   - Artificial Superhuman Intelligence (ASI): AI that possesses the ability to do everything that a human can do and more. An example is the Alpha 2 which is the first humanoid ASI robot.

3. Explain the different domains of Artificial Intelligence.
   - **Machine Learning:** It's the science of getting computers to act by feeding them data so that they can learn a few tricks on their own, without being explicitly programmed to do so.
   - **Neural Networks:** They are a set of algorithms and techniques, modeled in accordance with the human brain. Neural Networks are designed to solve complex and advanced machine learning problems.
   - **Robotics:** Robotics is a subset of AI, which includes different branches and application of robots. These Robots are artificial agents acting in a real-world environment. An AI Robot works by manipulating the objects in it's surrounding, by perceiving, moving and taking relevant actions.
   - **Expert Systems:** An expert system is a computer system that mimics the decision-making ability of a human. It is a computer program that uses artificial intelligence (AI) technologies to simulate the judgment and behavior of a human

or an organization that has expert knowledge and experience in a particular field.

- **Fuzzy Logic Systems:** Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) boolean logic on which the modern computer is based. Fuzzy logic Systems can take imprecise, distorted, noisy input information.
- **Natural Language Processing:** Natural Language Processing (NLP) refers to the Artificial Intelligence method that analyses natural human language to derive useful insights in order to solve problems.

Introduction to Soft Computing

Solutions to Practice Sheet : FL-1

Topics:

- *Introduction to Soft Computing*
- *Fuzzy logic*
- *Fuzzy membership functions*
- *Operations on Fuzzy sets*

1) Any soft-computing methodology is characterized with

(a) precise solutions
(b) control actions are unambiguous and accurate
(c) Control action is formally defined
(d) algorithm which can easily adapt with the change of dynamic environment

2) A fuzzy set A is closed if:

(a) $\lim x \rightarrow -\infty\, \mu_A(x) = 1$and $\lim x \rightarrow +\infty\, \mu_A(x) = 0$
(b) If $\lim x \rightarrow -\infty\, \mu_A(x) = \lim x \rightarrow +\infty\, \mu_A(x) = 0$
(c) If $\lim x \rightarrow -\infty\, \mu_A(x) = 0$ and $\lim x \rightarrow +\infty\, \mu_A(x) = 1$
(d) If $\lim x \rightarrow -\infty\, \mu_A(x) = \lim x \rightarrow +\infty\, \mu_A(x) = 1$

3) The support of Fuzzy Set A is the set of all points x in X (is the universe of discourse) such that

(a) $\mu_A(x) > 0$
(b) $\mu_A(x) = 1$
(c) $\mu_A(x) = 0.5$
(d) $\mu_A(x) \neq 1$

4) An equivalence between *Fuzzy vs. Probability* to that of *Prediction vs. Forecasting* is

(a) $Fuzzy \approx Prediction$
(b) $Fuzzy \approx Forecasting$
(c) $Probability \approx Forecasting$
(d) None of the above

5) Both fuzzy logic and artificial neural network are soft computing techniques because

(a) Both gives precise and accurate results.
(b) Artificial neural network gives accurate result, but fuzzy logic does not.
(c) In each, no precise mathematical model of the problem is required.
(d) Fuzzy gives exact result but artificial neural network does not.

6) Which of the following cannot be stated using fuzzy logic?

(a) Color of an apple
(b) Height of a person

(c) Date of birth of a student
(d) Speed of a car

7) Following which one is the example of ***Sigmoid Membership*** function?

(a) $\mu(x: c, \sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2}$
(b) $\mu(x: a, c) = \frac{1}{1+e^{-[a(x-c)]}}$
(c) $\mu(x: a, b, c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$
(d) $\mu(x: a, b, c) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{c-x}{c-b} & b \leq x \leq c \\ 0 & c \leq x \end{cases}$

8) How is Fuzzy Logic different from conventional control methods?

(a) IF and THEN Approach
(b) FOR Approach
(c) WHILE Approach
(d) DO Approach

9) The height h(A) of a fuzzy set A is defined as h(A) = support A(x), where x belongs to A. Then the fuzzy set A is called normal when

(a) h(A)=0
(b) h(A)<0
(c) h(A)=1
(d) h(A)>1

10) Fuzzy logic is a form of

(a) Two-valued logic
(b) Crisp set logic
(c) Many-valued logic
(d) Binary set logic

11) For $k > 1$, which of the following concept can be used to generate other linguistic hedge

(e) Concentration and Dilation
(f) Dilation
(g) Concentration
(h) None of the above

12) Given two fuzzy set A and B

$$A = \{(x1,\ 0.5),\ (x2,\ 0.1),\ (x3,\ 0.4)\} \text{ and } B = \{(x1,\ 0.2),\ (x2,\ 0.3),\ (x3,\ 0.5)\}$$

Union of the two set, that is, $A \cup B$ is given by

(a) $\{(x1,\ 0.5),\ (x2,\ 0.1),\ (x3,\ 0.4)\}$
(b) $\{(x1,\ 0.5),\ (x2,\ 0.3),\ (x3,\ 0.5)\}$
(c) $\{(x1,\ 0.2),\ (x2,\ 0.3),\ (x3,\ 0.5)\}$
(d) $\{(x1,\ 0.2),\ (x2,\ 0.1),\ (x3,\ 0.4)\}$

13) Given two Fuzzy Sets *A* and *B* with MFs μ_A and μ_B, respectively. Algebraic product or Vector product is given by:

(a) $\mathrm{\mu_A(x) \cdot \mu_B(x)}$
(b) $\mathrm{\mu_A(x) + \mu_B(x) - \mu_A(x). \mu_B(x)}$
(c) $\mathrm{min\{1, \mu_A(x) + \mu_B(x)\}}$
(d) $\mathrm{max\{0, \mu_A(x) + \mu_B(x) - 1\}}$

14) Two fuzzy sets A and B with membership functions $\mu_A(x)$ and $\mu_B(x)$, respectively defined as below.

A = **Hot Climate** with $\mu_A(x)$ as the MF.
B = **Cold Climate** with $\mu_B(x)$ as the M.F.

Pleasant climate is given by:

(a) $1 - \mu_B(x)$
(b) $\max(\mu_A(x), \mu_B(x))$
(c) $\min(\mu_A(x), \mu_B(x))$
(d) $1 - \mu_A(x)$

15) What is the **Bandwidth** of fuzzy set A which is given as follow?

$$A = (10{,}0.1), (15{,}0.2), (20{,}0.5), (25{,}0.4), (30{,}0.4), (35{,}0.5), (40{,}0.2), (45{,}0.1)$$

(e) 15
(f) -15
(g) 35
(h) 20

Introduction to Soft Computing

Practice Sheet: NN-1

Introduction to ANN

1) An ANN learn quickly if ŋ, the learning rate assumes the following value(s).

(a) $ŋ = 1$
(b) $ŋ < 1$
(c) $ŋ > 1$
(d) $ŋ = 0$

2) Which of the following is true for neural networks?

i. The training time depends on the size of the network.
ii. Neural networks can be simulated on a conventional computer.
iii. Artificial neurons are identical in operation to biological ones.

(a) i and ii are true
(b) i and iii are true
(c) ii is true.
(d) all of them are true

3) Which of the following is true for neural networks?

i. The error calculation which is followed in “Back-propagation algorithm” is the steepest descent method.
ii. Simulated annealing approach is followed in unsupervised learning.
iii. A problem whose output is linearly separable can also be solved with MLFFNN.
iv. The output of the perceptron with hard limit transfer function is more accurate than it is defined with any sigmoid transfer function.

(a) i and iii are true
(b) i and ii are true
(c) ii and iv are true
(d) all are true

4) A possible neuron specification to solve the AND problem requires a minimum of

(a) Single neuron
(b) Two neurons
(c) Three neurons
(d) Four neurons

5) What is back propagation?

(a) It is another name given to the curvy function in the perceptron
(b) It is the transmission of error back through the network to adjust the inputs
(c) It is the transmission of error back through the network to allow weights to be adjusted so that the network can learn
(d) None of the above

6) Neural Networks are complex ______________ with many parameters

(a) Linear Functions
(b) Nonlinear Functions
(c) Discrete Functions
(d) Exponential Functions

7) For problems, with error calculation, we solve using

(a) Recurrent neural networks
(b) Single layer feed forward neural network
(c) Multilayer feed forward neural network
(d) All of the above

8) Application of Neural Network includes

(a) Pattern Recognition
(b) Classification
(c) Clustering
(d) All of the above

9) If an individual x_i is dominated by p_i individuals in the current generation, then $rank(x_i)$ is

(a) 0
(b) p_i
(c) 1
(d) $1 + p_i$

10) What is perceptron in Neural network

(a) It is an auto-associative neural network
(b) It is a double layer auto-associative neural network
(c) It is a single layer feed-forward neural network with pre-processing
(d) It is a neural network that contains feedback

Introduction to Soft Computing

Practice Sheet: NN-2

Training of ANNs and Applications of ANNs

1. For the same size of training data as input, the fastest learning techniques is

   (a) Supervised training with error correction.
   (b) Supervised training with stochastic method.
   (c) Supervised training without error calculation.
   (d) Supervised training with Hebbian method.

2. Hebbian learning is a form of

   (a) Supervised Learning
   (b) Unsupervised learning
   (c) Reinforced learning
   (d) Stochastic learning

3. In case of layer calculation, the maximum time involved in

   (a) Output layer computation.
   (b) Hidden layer computation.
   (c) Equal effort in each layer.
   (d) Input layer computation.

4. The **Back Propagation Learning** algorithm is used to train

   (a) a single layer feed forward neural network only
   (b) a multiple layer feed forward neural network only
   (c) a recurrent neural network only
   (d) any artificial neural network

5. In this learning method, those neurons which responds strongly to input stimuli have their weights updated

   (a) Competitive learning

(b) Stochastic Learning
(c) Hebbian Leaning
(d) Gradient Descent Learning

6. Training Perceptron is based on

(a) Supervised learning technique.
(b) Unsupervised learning
(c) Reinforced learning
(d) Stochastic learning

7. A batch mode of training is generally implemented through the ________________________ in error calculation

(a) Minimization of median square error
(b) Maximization of median square error
(c) Maximization of mean square error
(d) Minimization of mean square error

8. Command to start matlab fuzzy toolbox is

(a) fis
(b) fuzzy
(c) fuzzybox
(d) fuzzytool

9. Which of the following is not a hybrid system?

(a) Embedded hybrid system
(b) Sequential hybrid system
(c) Auxiliary hybrid system
(d) Parallel hybrid system

10. In which of the following, one technology calls the other technology as subroutine to process or manipulate information needed

(a) Embedded hybrid system

(b) Sequential hybrid system
(c) Auxiliary hybrid system
(d) Parallel hybrid system

11. Fuzzy – Genetic Hybrid system is a

(a) Fuzzy logic in parallel with the Genetic algorithm
(b) Fuzzy logic controlled Genetic algorithm
(c) Genetic algorithm controlled Fuzzy logic
(d) None of the above

12. In Supervised learning $-ve$ sign is used to signify the fact that if $\frac{\partial E}{\partial V} > 0$, then we have to

(a) Increase V
(b) Decrease V
(c) Increase E
(d) Decrease E

13. Both fuzzy logic and artificial neural network are soft computing techniques because,

(a) Both gives precise and accurate results.
(b) Artificial neural network gives accurate result but fuzzy logic does not.
(c) In each, no precise mathematical model of the problem is required.
(d) Fuzzy gives exact result but artificial neural network does not.

14. In supervised learning, training set of data includes

(a) Input
(b) Output
(c) Both input and output
(d) None

Introduction to Soft Computing

Solution to GA-01

Optimization problems, GA operators - Encoding and Selection

1) An optimization problem is stated as follows:

$maximize\ f(x,y) = \frac{x^2}{2} + \frac{125}{y^2}$ where $x, y \in R^+$

The above optimization problem comes under the category of

(a) Unconstrained optimization problem.
(b) Linear optimization problem.
(c) Integer value optimization problem.
(d) Real value optimization problem.

2) Which of the following(s) is/are the pre-requisite(s) when Genetic Algorithms are applied to solve problems?

(i) Encoding of solutions.
(ii) Well-understood search space.
(iii) Method of evaluating the suitability of the solutions.
(iv) Contain only one optimal solution.

(a) i & ii only.
(b) ii & iii only.
(c) i & iii only.
(d) iii & iv only.

3) Which of the following(s) is/are found in Genetic Algorithms?

(i) Evolution.
(ii) Selection.
(iii) Reproduction.
(iv) Mutation.

(a) i & ii only.
(b) i, ii & iii only.
(c) ii, iii & iv only.

(d) All of the above.

4) Suppose, all steps in both SGA and SSGA remain same, except instead of selecting two individuals from the current population of size N, N_p ($N_p \ll N$) individuals as in SGA are selected. Then,

(a) Generation gap of SGA will be more than that of SSGA.
(b) Generation gap of SSGA will be more than that of SGA.
(c) Generation gap in both algorithms remains same.
(d) Nothing can be said precisely.

5) Which GA operation is computationally most expensive?

(a) Initial population creation.
(b) Selection of sub-population for mating.
(c) Reproduction to produce next generation.
(d) Convergence testing.

6) Which of the following is not true for Genetic algorithms?

(a) It is a probabilistic search algorithm.
(b) It is guaranteed to give global optimum solutions.
(c) If an optimization problem has more than one solution, then it will return all the solutions.
(d) It is an iterative process suitable for parallel programming.

7) The purpose of the fitness evaluation operation is

(a) To check whether all individual satisfies the constraints given in the problem.
(b) To decide the termination point.
(c) To select the best individuals.
(d) To identify the individual with worst cost function.

8) Which one of the following is not necessarily be considered as GA parameters?

(a) N, the population size.
(b) ∈, the obtainable accuracy.
(c) μ_p, the mutation probability.
(d) $\bar{f}$, the average fitness score.

9) Which of the following optimization problem(s) can be better solved with Order GA?

(a) 0-1 Knapsack problem.
(b) Travelling salesman problem.
(c) Job shop scheduling problem.
(d) Optimal binary search tree construction problem.

10) Which of the following is not a valid chromosome in Order GA?

(a)

1	3	5	7	2	4	6	8

(b)

A	B	D	E	A	F	H	G

(c)

1	0	0	1	1	0	0	1

(d)

14.6	-23.4	177.23

11) Roulette wheel selection scheme is preferable when

(a) Fitness values are uniformly distributed.
(b) Fitness values are non-uniformly distributed.
(c) Needs low selection pressure.
(d) Needs high population diversity.

12) What GA encoding scheme suffers from Hamming cliff problem?

(a) Binary coded GA.
(b) Real coded GA.
(c) Order GA.
(d) Tree coded GA.

Introduction to Soft Computing

Solution to GA-2

GA operators – Crossover and Mutation

1) Which GA operation is computationally most expensive?

(a) Initial population creation.
(b) Selection of sub-population for mating.
(c) Reproduction to produce next generation.
(d) Convergence testing.

2) Which selection strategy is susceptible to a high selection pressure and low population diversity?

(a) Roulette-wheel selection.
(b) Rank based selection.
(c) Tournament selection.
(d) All the above.

3) Which of the following is not a mutation operation in real coded GA?

(a) Flipping.
(b) Random mutation.
(c) Polynomial mutation.
(d) All are mutation operation in real coded GA

4) Two parent chromosomes in Order GA encoding scheme is given as follows:

			*						
1	2	3	4	5	6	7	8	9	10

			*						
10	9	8	7	6	5	4	3	2	1

A $K - point$ is selected at 4^{th} location according to single point crossover technique. Which of the following off-spring is not possible?

(a)

1	2	3	4	10	9	8	7	6	5

(b)

7	8	9	10	6	5	4	3	2	1

(c)

10	9	8	7	1	2	3	4	5	6

(d)

5	6	7	8	9	10	1	2	3	4

5) Which one of the following is not necessarily be considered as GA parameters?

(a) N, the population sizes.
(b) $\in$, the obtainable accuracy.
(c) μ_p, the mutation probability.
(d) $\bar{f}$, the average fitness scores.

6) Which GA encoding scheme gives faster execution?

(a) Binary coded GA.
(b) Real coded GA.
(c) Order GA.
(d) Tree encoded GA.

7) The purpose of the fitness evaluation operation is

(a) To check whether all individual satisfies the constraints given in the problem.
(b) To decide the termination point.
(c) To select the best individuals.
(d) To identify the individual with worst cost function.

8) If crossover between chromosomes in search space does not produce significantly different offspring, what does it imply? (if offspring consist of one half of each parent)

(i) The crossover operation is not successful.
(ii) Solution is about to be reached.
(iii) Diversity is so poor that the parents involved in the crossover operation are similar.
(iv) The search space of the problem is not ideal for GAs to operate.

(a) ii, iii & iv only.
(b) ii & iii only.
(c) i, iii & iv only.
(d) All of the above.

9) Which of the following is not a valid chromosome in Order GA?

(a)

1	3	5	7	2	4	6	8

(b)

A	B	D	E	A	F	H	G

(c)

1	0	0	1	1	0	0	1

(d)

14.6	-23.4	177.23

10) In Rank-based selection scheme, which of the following is not correct

(a) The % area to be occupied by an individual i, is given by $\frac{r_i}{\sum_{i=1}^{N} r_i}$

(b) Two or more individuals with the same fitness values should have the same rank.
(c) Individuals are arranged in a descending order of their fitness values.
(d) The proportionate based selection scheme is followed based on the assigned rank.

11) Which of the following(s) is/are the pre-requisite(s) when Genetic Algorithms are applied to solve problems?

(i) Encoding of solutions.
(ii) Well-understood search space.
(iii) Method of evaluating the suitability of the solutions.
(iv) Contain only one optimal solution.

(a) i & ii only.
(b) ii & iii only.
(c) i & iii only.
(d) iii & iv only.

12) Tournament Selection has
(a) Low population diversity and moderate selection pressure
(b) High population diversity and Moderate selection pressure
(c) Moderate population diversity and high selection pressure
(d) High population diversity and low selection pressure

13) Which of the following is a fitness scaling approach?
(a) Linear scaling
(b) Sigma scaling
(c) Power law scaling
(d) All of the above

14) If selection pressure is **HIGH,** which one is **FALSE**
(a) The search focuses only on good individuals (in terms of fitness) at the moment.
(b) It loses the population diversity.
(c) Lower rate of convergence.
(d) Leads to pre-mature convergence of the solution to a sub-optimal solution.

15) Which of the following comparison is true?

(a) In the event of restricted access to information, GAs win out in that they require much fewer information to operate than other search.
(b) Under any circumstances, GAs always outperform other algorithms.
(c) The qualities of solutions offered by GAs for any problems are always better than those provided by other search.
(d) GAs could be applied to any problem, whereas certain algorithms are applicable to limited domains.