IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Soft Computing

Pareto-based approaches to solve MOOPs

Prof. Debasis Samanta
Department of Computer Science & Engineering
IIT Kharagpur

Non-dominated Sorting Genetic Algorithm (NSGA)

Non-dominated Sorting Genetic Algorithm (NSGA)

N. Srinivas and K. Deb, 1994

Reference: *Multi-objective Optimization using Non-dominated Sorting in Genetic Algorithm* by N. Srinivas and K. Deb, IEEE Transaction on Evolutionary Computing, Vol. 2, No. 3, Pages 221-248, 1994.

The algorithm is based on the concept of

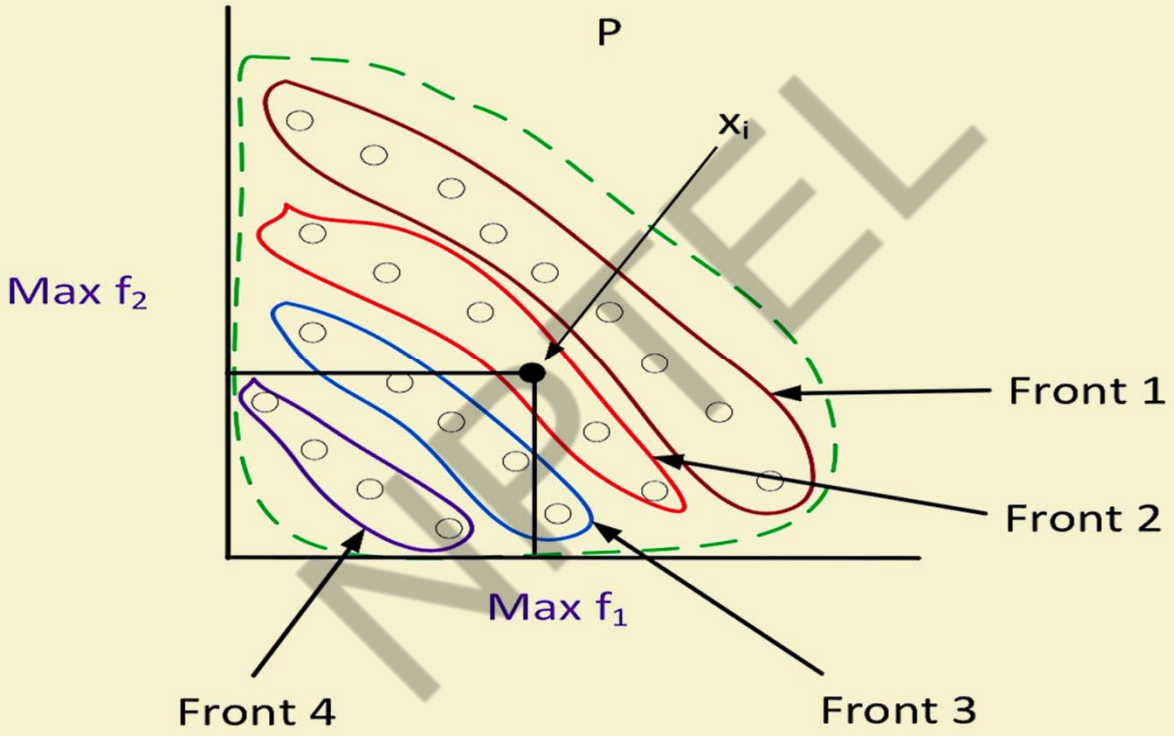
- Non-dominated sorting procedure (a ranking selection method to select good non dominated points.
- Niche method (is used to maintain stable subpopulation of good points)

Non-dominated Sorting Procedure

Notations :

- P - denotes a set of solutions (input solutions)
- x_i - denotes i-th solution
- S_i - denotes a set of solutions which dominate the solution x_i
- n_i denotes the domination count, i.e., the number of solutions which dominates x_i
- P_k - denotes a non-domination front at the k-th level (output as sorting based on the non-domination rank of solution)

Non-dominated Sorting Procedure



Non-dominated Sorting Procedure

Steps :

1) For each $x_i \in P$ do

   1. For all $x_j \neq x_i$ and $x_j \in P$
      If $x_i \leq x_j$ (i.e. if x_i) dominates x_j
      then $S_i = S_i \cup x_j$
      else if $x_j \leq x_i$ then $n_i = n_i + 1$

   2. If $n_i = 0$, keep x_i in P_1 (the first front). Set a front counter $k = 1$

2) [*INITIALIZATION*]
   For each $x_i \in P$, $n_i = 0$ and $S_i = \phi$

< –– This completes the finding of first front –– >

Non-dominated Sorting Procedure

< –– To find other fronts –– >

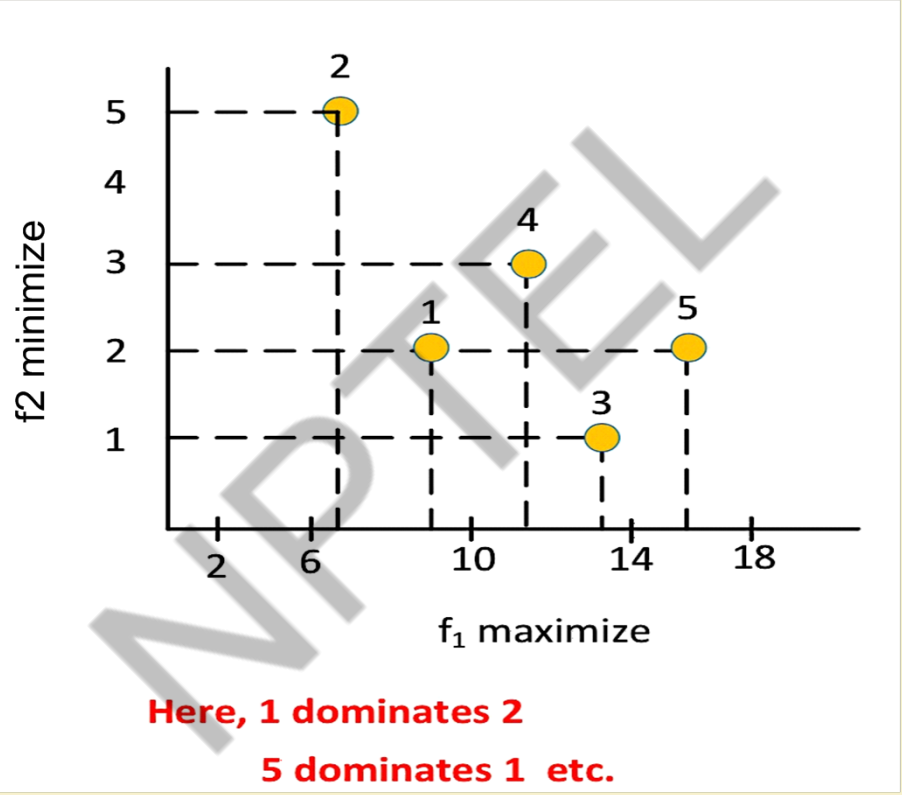
3) While $P_k = \phi$ do

   1. [*Initialize*] $Q = \phi$ (for sorting next non-dominated solutions)
   2. For each $x_i \in P_k$ and For each $x_j \in S_i$ do
      - Update $n_j = n_j - 1$
      - If $n_j = 0$ then $x_j \; in \; Q$ else $Q = Q \cup x_j$
   3. Set $k = k + 1$ and $P_k = Q$ (Go for the next front)

Non-dominated Sorting Procedure

Note: Time complexity of this procedure is $O(mn^2)$, where m is the number of objective and n is the population size.

An Illustration



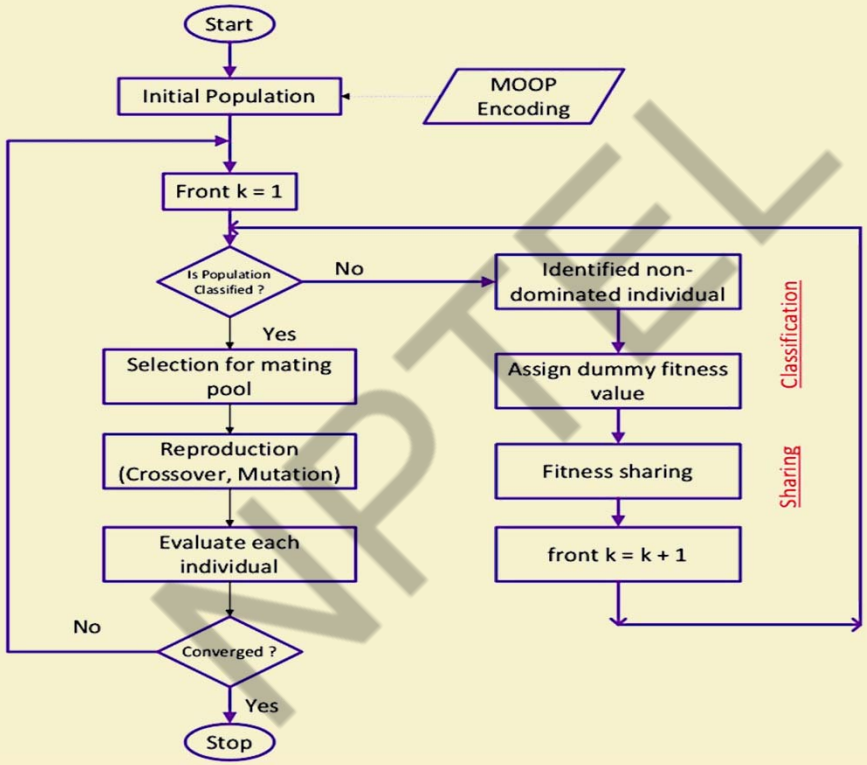
Non-dominated Sorting Procedure

- Three non-dominated fronts as shown.
- The solutions in the front [3, 5] are the best, follows by solutions [1, 4].
- Finally, solution [2] belongs to the worst non-dominated front.
- Thus, the ordering of solutions in terms of their non-domination level is : [3, 5], [1, 4], [2]

NSGA Technique

1. NSGA is based on several front of classification of the individuals.
2. It finds a front (of non dominated individuals) at particular rank and then assign a fitness value (called dummy fitness value) to each solution in the front followed by sharing of fitness value.
3. It then selects some individuals based on shared fitness values of the individuals for mating pool.
4. Reproduce offspring for next generation and thus completes an iteration.

Flowchart of NSGA



Assign dummy fitness value

- Once all non dominated individual are identified for a front, it assigns a unique fitness value to each belongs to that front is called dummy fitness value.
- The dummy fitness value is a very large number and typically is proportional to the number of population (which includes the number of individuals in all fronts including the current front).
- The proportional constant (≥ 1) is decided by the programmer.

Assign dummy fitness value

Note :

- The same fitness value is assigned to give an equal reproductive potential to all the individual belong to a front.
- A higher value is assigned to individuals in an upper layer front to ensure selection pressure, that is, having better chance to be selected for mating.
- As we go from an upper front to next lower front, count of individuals are ignored (i.e., logically remove from the population set) and thus number of population successively decreases as we move from one front to the next.

Sharing the fitness value

- In the given front, all individuals are assigned with a dummy fitness value which is proportional to the number of population.
- These classified individuals are then shared with their dummy fitness values.
- Sharing is achieved by dividing the dummy fitness value by a quantity proportional to the number of individuals around it. This quantity is called niche count.

Sharing the fitness value

- This quantity is calculated by computing Sharing coefficient denoted as $sh(d_{ij})$ between the individual x_i , x_j belong to the front under process, as per following.

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{T_{shared}}\right)^2 & , if\ d_{ij} < T_{shared} \\ 0 & , otherwise \end{cases}$$

In the above, the parameter d_{ij} is the phenotype distance between two individuals x_i and x_j in the current front.

Sharing the fitness value

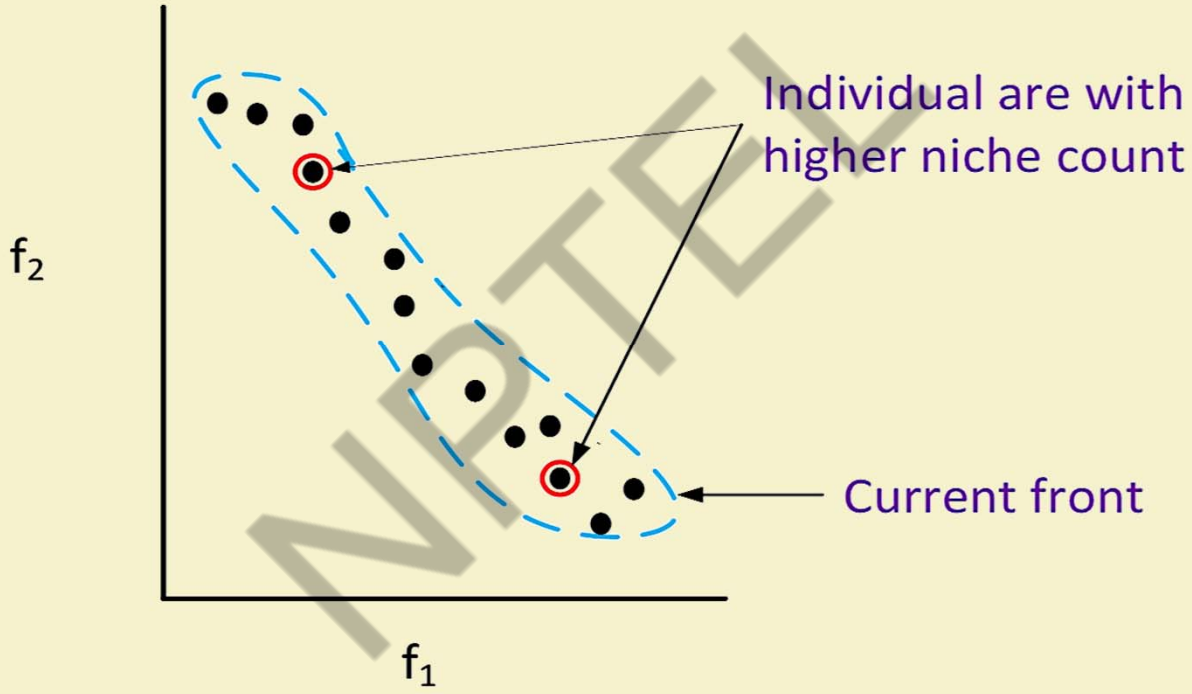
- T_{share} is the maximum phenotype distance allowed between any two individuals to become members of a niche. This value should be decided by the programmer.
- Niche count of x_i then can be calculated as

$$\gamma(x_i) = \sum_{\forall x_j \in P_k} sh(d_{ij})$$

where P_k denotes the set of non dominated individual in the current front k).

- The shared fitness value of x_i is then $\bar{f}_i = \frac{f}{\gamma(x_i)}$, where f is the dummy fitness value.

NSGA Technique



Sharing the fitness value

Note :

- The sharing is followed to ensure the population diversity. That is, given a fair chance to those individuals which are not so niche.
- $Sh(d_{ij}) \neq 0$, and it always greater than or equal to 1 ($= 1$ when $d_{ij} = 0$).
- After the fitness sharing operation, all individuals in the current front are moved to a population set to be considered for mating subject to their selection.
- The assignment of dummy fitness value followed by fitness sharing is then repeated for the next front (with a smaller population size).

NSGA Selection

- Once all fronts are processed, we obtained a set of population with their individual shared fitness values.
- This population is then through a selection procedure.
- NSGA follows stochastic remainder proportionate selection, which is as follows.

  1) Calculate cumulative probabilities of all individuals as in Roulette-Wheel scheme.
  2) Generate a random number (for first time only) r_i (this is the probability to get an individual to be selected).

NSGA Selection

3) If $P_{j-1} \leq ri < P_j$ (where P_j and P_{j-1} are two cumulative probabilities, then consider j-th individual for the selection.
4) Let $E_j = P_j \times N$ (N, the population size and E_j denotes expected count).
5) If integer part in E_j is non-zero, then select j-th solution for mating.
6) The fractional part is used as the random number for the selection of next solution.
7) Repeat Step 3-6 until the mating pool of desired size is not reached.

Reproduction and generation of new population

Follow the standard reproduction procedure as in Simple GA.

Remark on NSGA

- Several comparative studies reveal that NSGA is outperformed by both MOGA and NPGA.
- NSGA is also highly inefficient algorithms because of the way in which it classified individuals (it is $O(mn^3)$ time complexity).
- It needs to specify a sharing parameter T_{share}.
- It is a non-elitism approach.
- Deb et al. proposed an improved version of NSGA algorithm called NSGA-II.

Thank You!!

Debasis Samanta
CSE
IIT Kharagpur

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Soft Computing

Pareto-based approaches to solve MOOPs

Prof. Debasis Samanta
Department of Computer Science & Engineering
IIT Kharagpur

NSGA-II : An improvement of NSGA

- NSGA is also highly inefficient algorithms because of the way in which it classified individuals (it is $O(mn^3)$ time complexity).
- It needs to specify a sharing parameter T_{share}.
- It is a non-elitism approach.
- NSGA-II algorithm has been proposed addressing the limitations in NSGA algorithm.

Elitist Multi-objective Genetic Algorithm : NSGA-II

Debasis Samanta
CSE
IIT Kharagpur

Elitist Multi-objective Genetic Algorithm : NSGA-II

K. Deb, A. Pratap, S. Agarwal and T. Meyarivan 2002.

Reference :

A Fast and Elitist Multi-objective Genetic Algorithm : NSGA-II by K. Deb, A. Pratap, S. Agrawal and T. Meyarivan in *IEEE Transaction on Evolutionary Computation*, Vol.6, No.2, Pates 182-197, April 2002.

Non-dominated sorting in NSGA-II

The following approach of finding non-domination front is followed in NSGA.

Notation:

- P = A set of input solution
- x_i and x_j denote any two solutions
- P' = set of solutions on non-dominated front.

Non-dominated sorting in NSGA-II

Approach:

- In this approach, every solution from the population P is checked with a partially filled population P' (which is initially empty).
- To start with, the first solution from P is moved into initially empty P'. Thereafter, each solution $x_i \in P$ one by one is compared with all member of the set P'.
- If the solution x_i dominates any member of P', then that solution is removed from P'.

Non-dominated sorting in NSGA-II

- Otherwise, if solution x_i is dominated by any member of P', the solution x_i is ignored.
- On the other hand, if solution x_i is not dominated by any member of P' it is moved in to P'. This is how the set P' grows with non dominated solutions.
- When all solutions of P are checked, the remaining members of P' constitute the solutions on non-dominated front.

Non-dominated sorting in NSGA-II

This front is removed and the same steps are repeated with the remaining solutions to find the next non-domination front until *P* is empty. This approach is precisely stated in the following.

Steps :

$P = x_1, x_2, \ldots, x_N$

1) Initialize $P' = x_i$ set solution counter $j = 2$.
2) Let $j = 1$.
3) Compare solution x_i with x_j from P for domination
4) If $x_i \leq x_j$, delete x_j from P'
5) If $j < |P'|$, then increment j by one and go to step 3. Else Go to step 7.

Non-dominated sorting in NSGA-II

6) If $x_j \leq x_i$, then increment i by one and go to step 2.

7) Insert x_i in P' *i.e.* $P' = P' \cup i$

8) If $i < N$, increment i by one and go to step 2.

9) Output P' as the non-dominated from with current population

10) if $P \neq \phi$, repeat Step 1-9.

11) Stop.

Note: Time complexity of this procedure is $O(mn^2)$ and in worst case $O(mn^3)$ (when one front contains only one solution.)

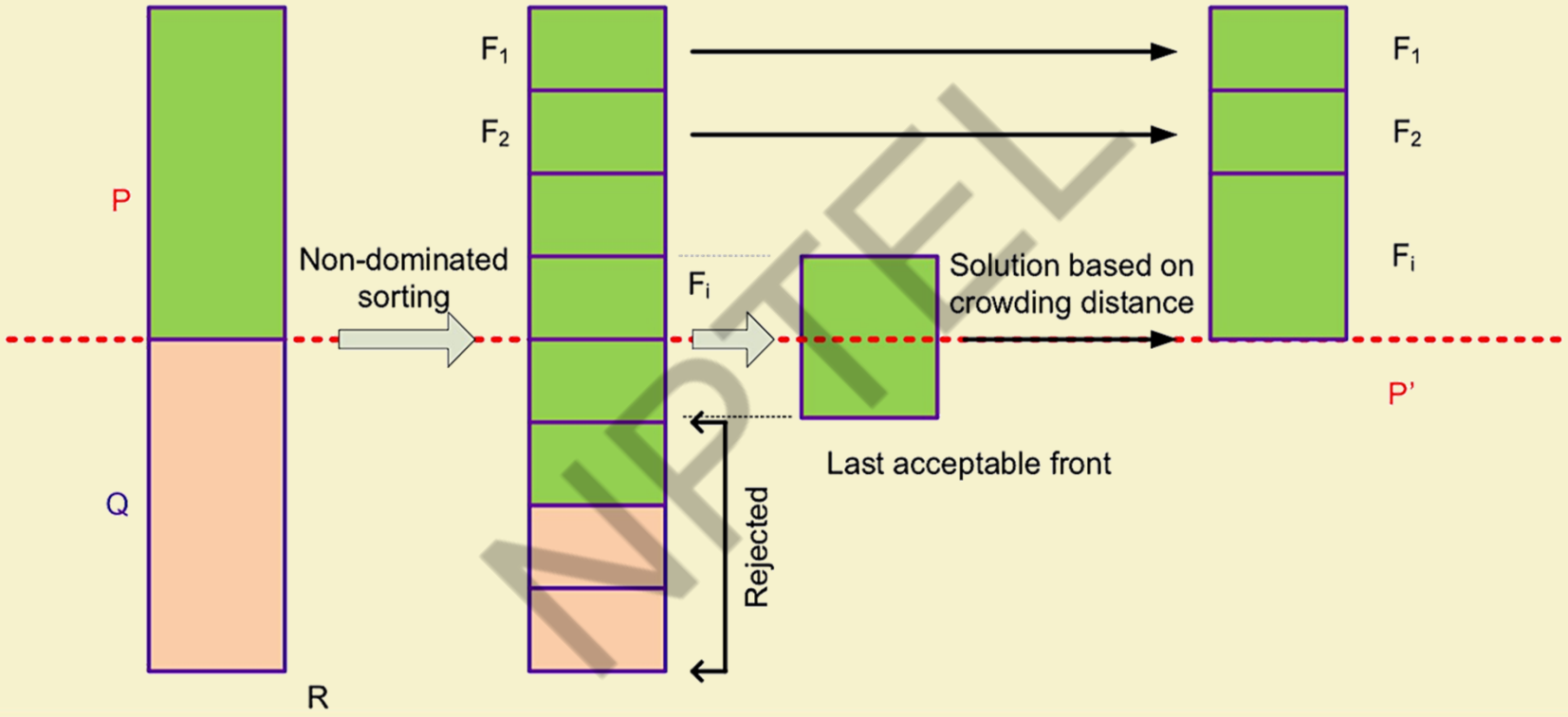
Overview of NSGA-II

- Let P be the current population.
- Create offspring population Q from P using the genetic operators (mating pool, crossover and mutation).
- Two population are combined together to form a large population R of size of $2N$.
- Apply non-dominating sorting procedure to classify the entire population R.
- After the non-domination sorting, the new population P is obtained by filling the non-dominated front one-by-one until the $|P'| < |P|$
  - The filling starts with the best non-dominated front followed by the second non-dominated front, followed by the third non-dominated front and so on.

Overview of NSGA-II

- Since the total population size is $2N$, not all fonts may be accommodated in the fronts available in P'. All fronts which could not be accommodated are simply rejected. When the last allowed front is being considered, there may exist more solution in the last front than the remaining slots in the new population.
- Instead of arbitrarily discarding some members from the last acceptable front, the solution which will make the diversity of the selected solutions the highest are chosen.
- This is accomplished by calculating **crowding distance** of solutions in the last acceptable front.
- This way a new generation was obtained and the steps are repeated until it satisfies a termination condition.

NSGA Technique



Basic Steps : NSGA-II

Steps:

1) Combine parent (P) and offspring (Q) populations and obtain $R = P \cup Q$
2) Perform a non-dominated sorting to R and identify different fronts of non-dominated solutions as F_i, $i = 1, 2, \cdots, etc.$
3) Set new population $P' = \phi$. Set a new counter $i = 1$ (to indicate front to be allowed to fill P'.
4) Fill P' until $|P'| + |F_i| < N$ that is, $P' \cup F_i$ and $i = i + 1$.

Basic Steps : NSGA-II

5) Perform **Crowding sort** (F_i) and include the most-widely spread $N - |P'|$ solution by using the crowding distance values in the sorted F_i to P'.

6) Create offspring population Q' from P' by using the crowded-tournament selection, crossover and mutation operator.

7) $P = P', Q = Q'$.

8) Repeat step 1-7 until the convergence of solutions.

Thank You!!

Debasis Samanta
CSE
IIT Kharagpur

IIT KHARAGPUR

NPTEL ONLINE CERTIFICATION COURSES

Introduction to Soft Computing

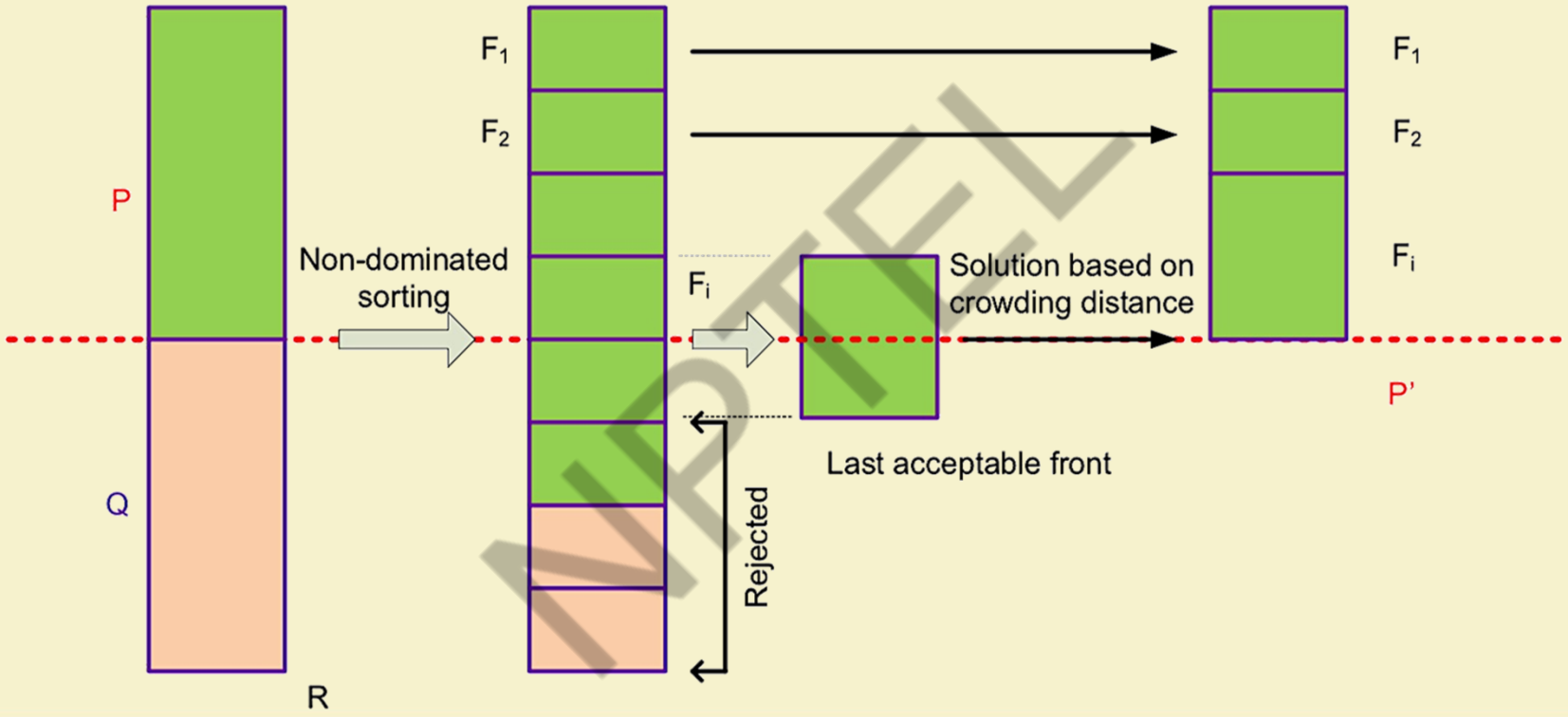
Pareto-based approaches to solve MOOPs

Prof. Debasis Samanta
Department of Computer Science & Engineering
IIT Kharagpur

Elitist Multi-objective Genetic Algorithm : NSGA-II

Debasis Samanta
CSE
IIT Kharagpur

NSGA Technique



Crowding sort procedure

The crowding sort follows the following two concepts.

- Crowding distance metric (d) and
- Crowded comparison operator $<_e$

Note :

- The crowding distance d_i of a solution x_i is a measure of the search space around x_i which is not occupied by any other solutions in the population. (It is just like a population density : d_i is lower for a dense population).
- The crowded comparison operator $<_e$ (a binary operator) to compare two solutions and returns a winner.

Crowding sort procedure

Definition : Crowding comparison operator

A solution x_i wins over a solution x_j, iff any of the following conditions are true

- If solution x_i has a better rank (i.e. dominance rank). That is $rank(x_i) < rank(x_j)$
- If they have the same rank but solution x_i has better crowding solution than x_j.
- That is $rank(x_i) = rank(x_j)$ and $d_i > d_j$(where d_i and d_j are crowding distance of x_i and x_j, respectively).

Crowding sort procedure

Note:

- The first condition ensures that x_i lies on a better non-domination front.
- The second condition resolve the tie of both solution being on the same non-dominated front by deciding on their crowding distances.

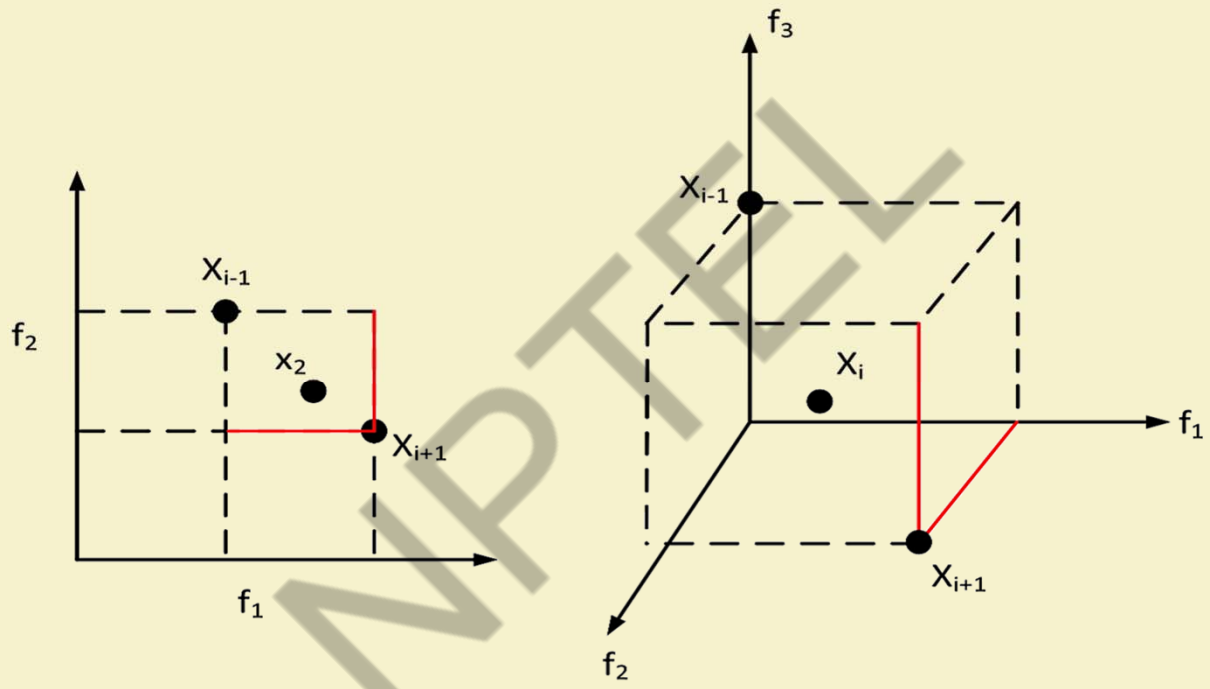
(**Note:** For NSGA-II, only the second condition is valid as all solutions belong to one front only.)

Crowding sort procedure

Definition : Crowding distance measure

The crowding distance measure d_i is an estimate of the size of the largest cuboid enclosing the point x_i (i-th solution) without including any point in the population.

Crowding sort procedure



(a) Solutions in 2D space

(b) Solutions in 3D space

Crowding sort procedure

- In Fig.(a), the crowding distance is the average side length of the rectangle (so that two nearest solutions of x_i and x_{i-1} at two opposite corners).
- In Fig.(b), the crowding distance is the sum of three adjacent edges of a cuboid formed by the two neighbors x_{i-1} and x_{i+1} of x_i at two opposite corners of the cuboid.

Crowding distance calculation

Given a set of non-dominated solutions say F, and objective functions $f = f_1, f_2, \ldots, f_M$, the procedure to calculate the crowding distance of each solution $x_i \in F$ is stated below.

Steps :

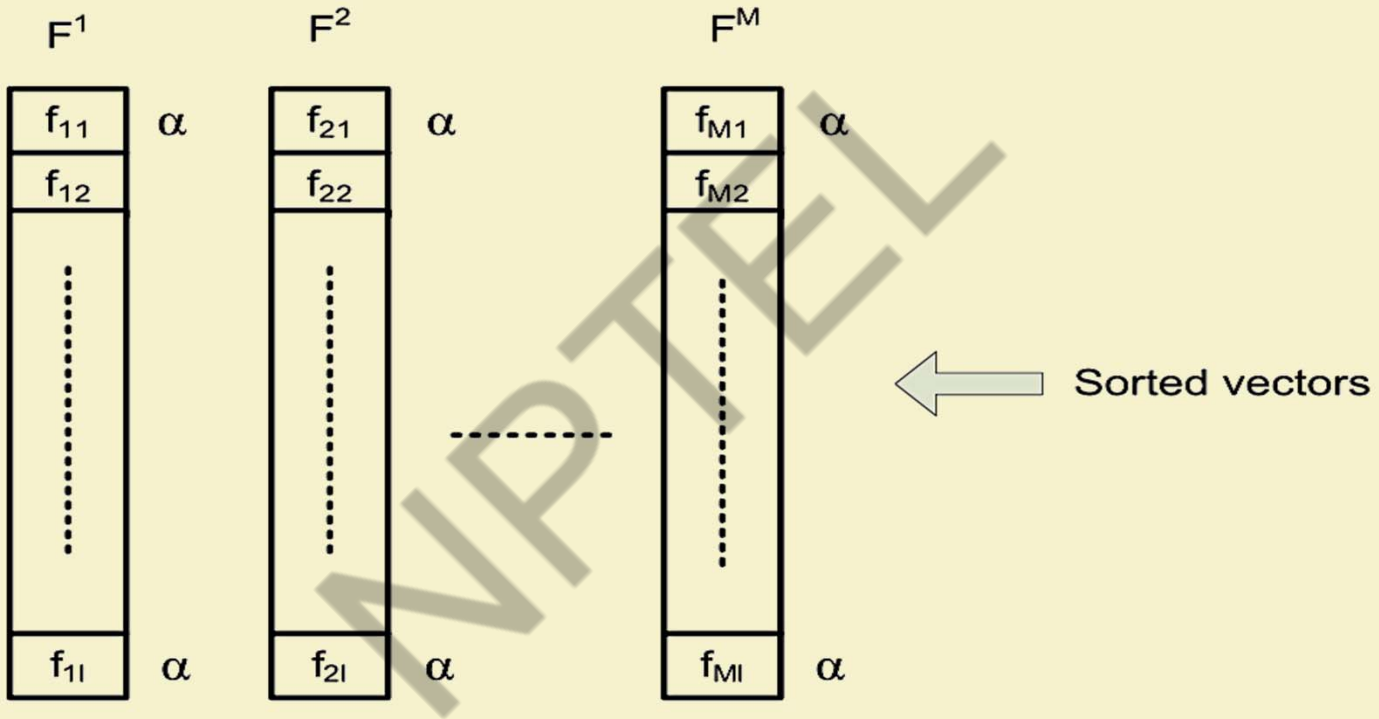
1) Let $I = |F|$ (number of solutions in F)
2) For each $x_i \in F$, set $d_i = 0$ (Initialize the crowding distance)

Crowding distance calculation

3) For each objective $f_i \in f$
   - $F^i = sort(F, i)$
     - Sort all solutions in F with respect to objective values f_i
     - This will result $F^1, F^2, \ldots, F^m$ sorted vector.
     - The sorting is performed in ascending order of objective values f_i

The sorted vectors are shown in the fig.

Crowding distance calculation

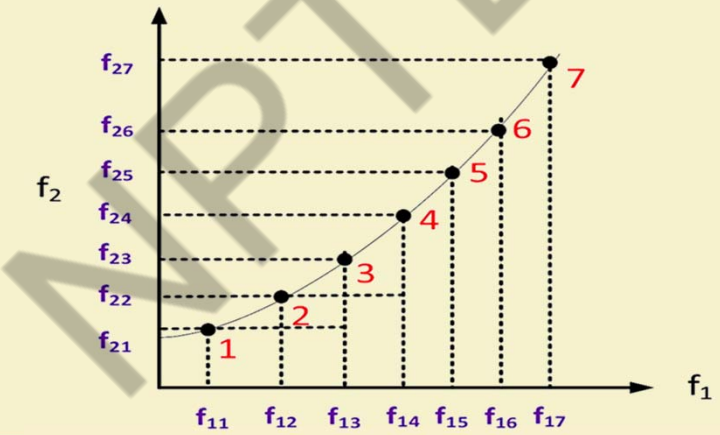


Crowding distance calculation

- For each solution $j = 2$ to $I - 1$ do

$$d_j = \sum_{k=1}^{M} \frac{f_{k,j+1} - f_{k,j-1}}{f_k^{MAX} - f_k^{MIN}}$$

- $d_i = d_I = \propto$ (Each boundary solutions are not crowded and hence large crowded distance)



Crowding distance calculation

Note:

- All objective function are assumed to be minimized.
- Time complexity of the above operation is $O(mn \log_2 n)$, is the complexity of m sorting operation with n elements.
- The parameters f_k^{MAX} and f_k^{MIN}, $k = 1,2,\cdots M$ can be set as the population-maximum and population-minimum values of $k-th$ objective function and is used here to normalize the objective values.

Crowding Tournament

- Once crowding distance of all solutions in the last acceptable front F is calculated we are to select $N - |P'|$ solutions from F to complete the size of $|P| = N$.
- To do this, we are to follow tournament selection operation according to $<_c$ (Crowding comparison operator). That is we select x_i over x_j ($x_i <_c x_j$) if $d_i > d_j$.
- Note that we prefer those solutions which are not in the crowded search space. This ensures a better population diversity.

Crowding Tournament

Remarks :

- NSGA-II is an elitist approach.
- It consider a faster non-dominated sorting procedure with time complexity $O(mn^2)$ compared to $O(mn^3)$ in NSGA.
- It does not require explicit sharing concept, rather use a crowding tournament selection with time complexity $O(mn \log n)$.
- Thus, the overall time complexity of NSGA-II is $O(mn^2)$.

Thank You!!

Debasis Samanta
CSE
IIT Kharagpur

IIT KHARAGPUR

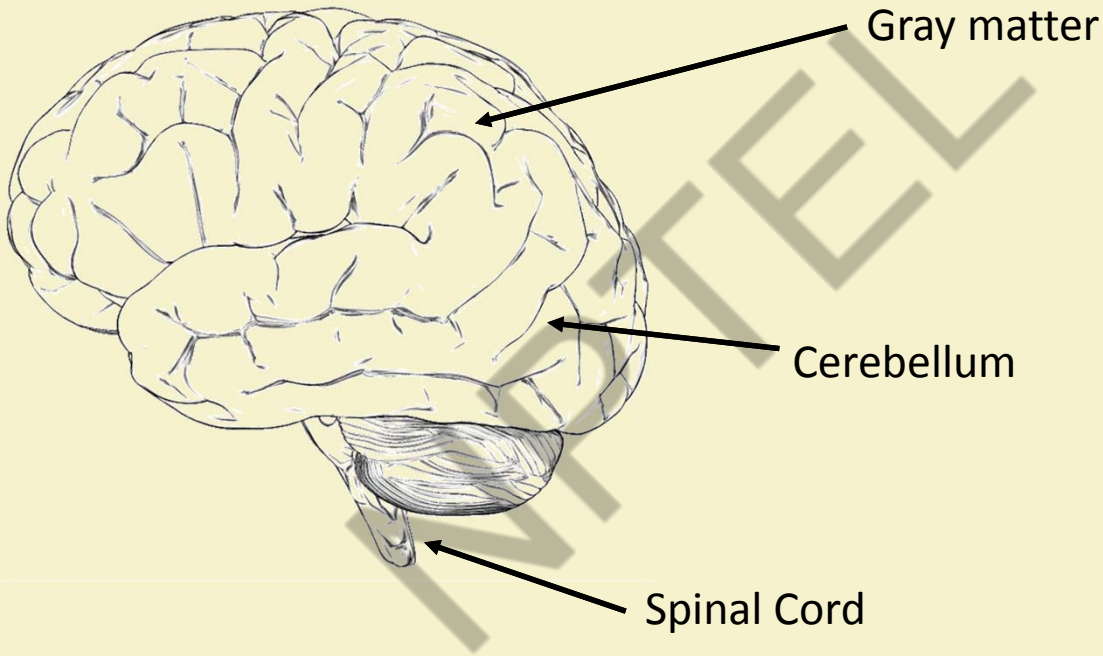
NPTEL ONLINE
CERTIFICATION COURSES

Introduction to Soft Computing

Artificial Neural Network : Introduction

Debasis Samanta
Department of Computer Science and Engineering
IIT KHARAGPUR

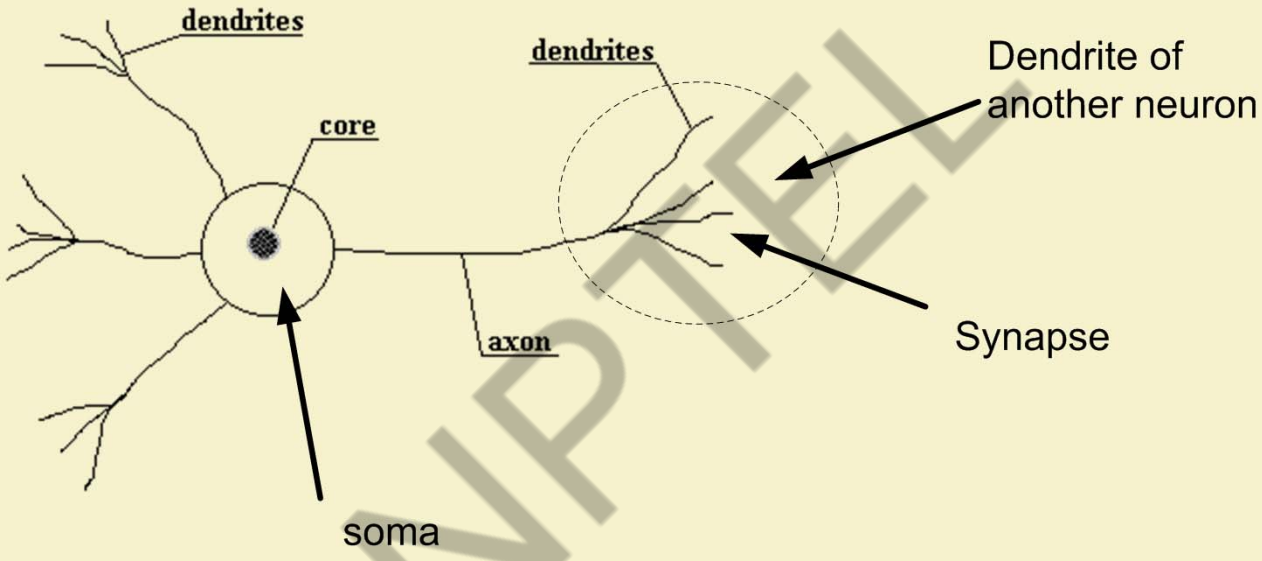
Brain: Centre of the nervous system



Biological nervous system

- Biological nervous system is the most important part of many living things, in particular, human beings.
- There is a part called **brain** at the centre of human nervous system.
- In fact, any biological nervous system consists of a large number of interconnected processing units called **neurons**.
- Each neuron is approximately 10μ long and they can operate in parallel.
- Typically, a human brain consists of approximately 10^{11} neurons communicating with each other with the help of **electrical impulses.**

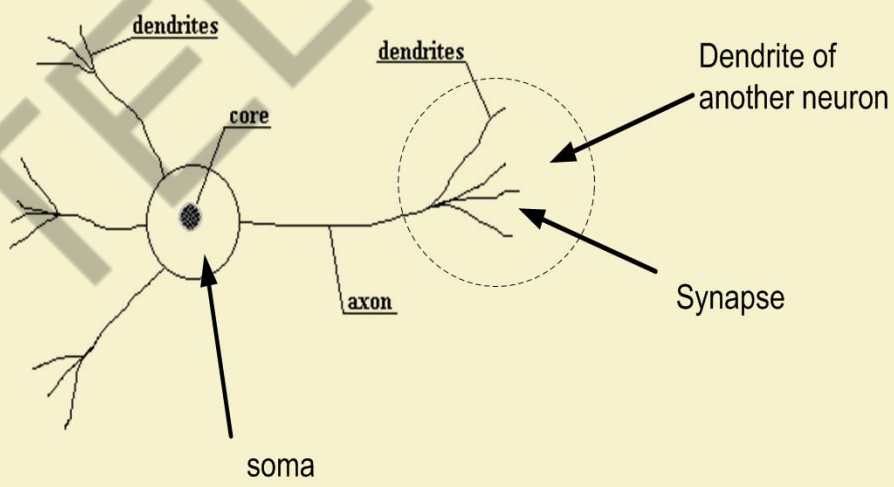
Neuron: Basic unit of nervous system



Neuron and its working

Figure shows a schematic of a biological neuron. There are different parts in it

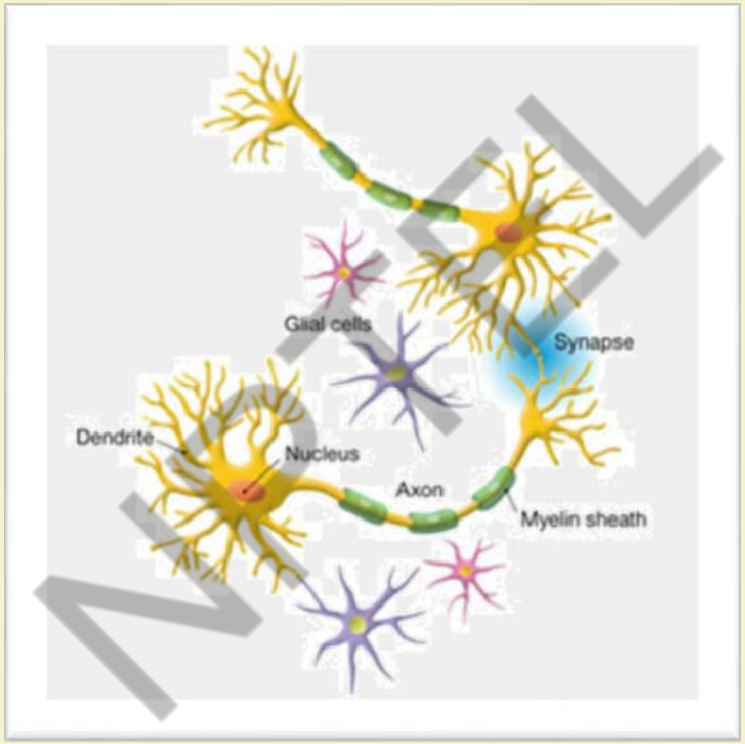
- **Dendrite** : A bush of very thin fibre.
- **Axon** : A long cylindrical fibre.
- **Soma** : It is also called a cell body, and just like as a nucleus of cell.
- **Synapse** : It is a junction where axon makes contact with the dendrites of neighbouring dendrites.



Neuron and its working

- There is a chemical in each neuron called neurotransmitter.
- A signal (also called sense) is transmitted across neurons by this chemical.
- That is, all inputs from other neuron arrive to a neurons through dendrites.
- These signals are accumulated at the synapse of the neuron and then serve as the output to be transmitted through the neuron.
- An action may produce an electrical impulse, which usually lasts for about a millisecond.
- Note that this pulse generated due to an incoming signal and all signal may not produce pulses in axon unless it crosses a **threshold value**.
- Also, note that an action signal in axon of a neuron is commutative signals arrive at dendrites which summed up at soma.

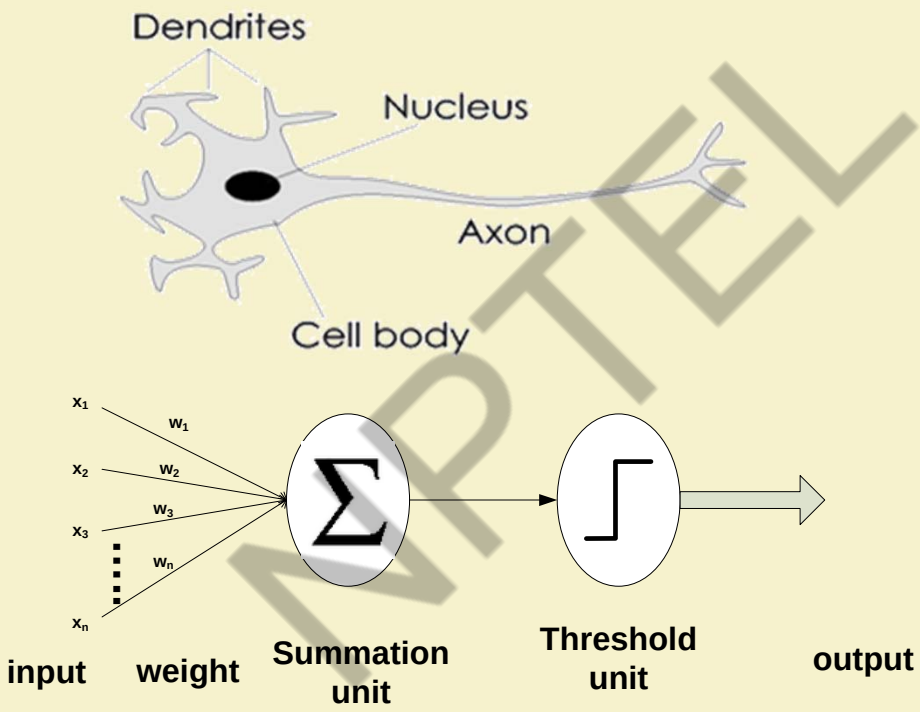
Neuron and its working



Artificial neural network

- In fact, the human brain is a highly complex structure viewed as a massive, highly interconnected network of simple processing elements called **neurons**.
- Artificial neural networks (ANNs) or simply we refer it as neural network (NNs), which are simplified models (i.e. imitations) of the biological nervous system, and obviously, therefore, have been motivated by the kind of computing performed by the human brain.
- The behaviour of a biological neural network can be captured by a simple model called artificial neuron or **perceptron**.

Analogy between BNN and ANN



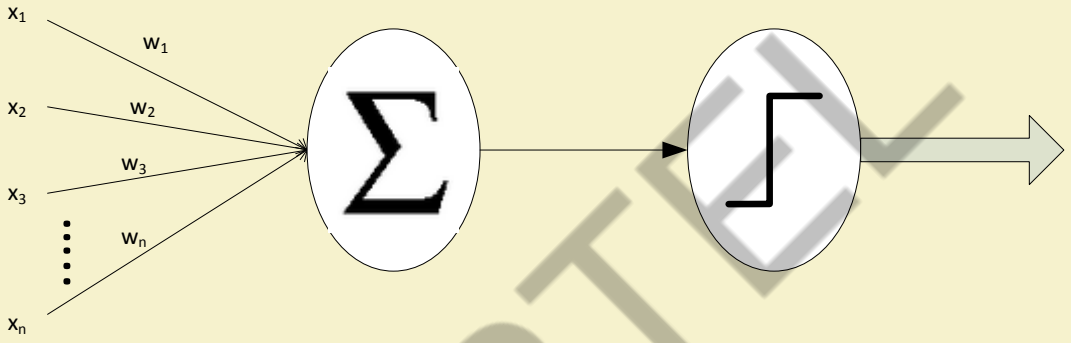
Artificial neural network

We may note that a neutron is a part of an interconnected network of nervous system and serves the following.

- Compute input signals
- Transportation of signals (at a very high speed)
- Storage of information
- Perception, automatic training and learning

We also can see the analogy between the biological neuron and artificial neuron. Truly, every component of the model (i.e. artificial neuron) bears a direct analogy to that of a biological neuron. It is this model which forms the basis of neural network (i.e. artificial neural network).

Artificial neural network



Here, $x_1, x_2, \ldots \ldots x_n$ are the n inputs to the artificial neuron.
$w_1, w_2, \ldots \ldots w_n$ are weights attached to the input links.

Artificial neural network

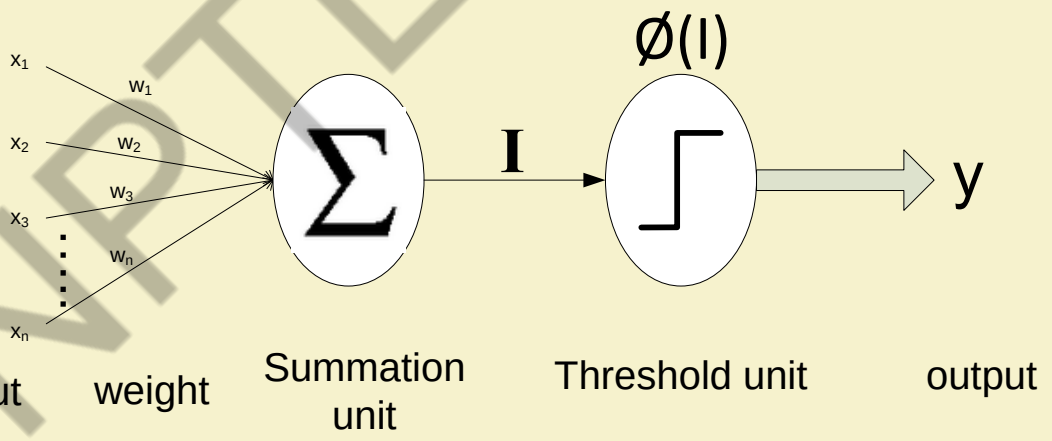
- Note that, a biological neuron receives all inputs through the dendrites, sums them and produces an output if the sum is greater than a threshold value.
- The input signals are passed on to the cell body through the synapse, which may accelerate or retard an arriving signal.
- It is this acceleration or retardation of the input signals that is modelled by the **weights**.
- An effective synapse, which transmits a stronger signal will have a correspondingly larger weights while a weak synapse will have smaller weights.
- Thus, weights here are multiplicative factors of the inputs to account for the strength of the synapse.

Artificial neural network

Hence, the total input say I received by the soma of the artificial neuron is

$$I = W_1X_1 + W_2X_2 + \ldots\ldots\ldots\ldots + W_nX_n = \sum_{i=1}^{n} W_iX_i$$

To generate the final output y, the sum is passed to a filter ϕ called transfer function, which releases the output.
That is $y = \phi(I)$



Artificial neural network

- A very commonly known transfer function is the **thresholding function** denoted as ϕ.
- In this thresholding function, sum $(\text{i. e.}, I)$ is compared with a threshold value say θ.
- If the value of *I* is greater than θ, then the output is 1 else it is 0 (this is just like a simple linear filter).
- In other words,

$$y = \phi\left(\sum_{i=1}^{n} W_i X_i - \theta\right)$$

where

$$\phi(I) = \begin{cases} 1 & if\ I > \theta \\ 0 & if\ I \leq \theta \end{cases}$$

Such a ϕ is called step function (also known as Heaviside function).

Transformation functions

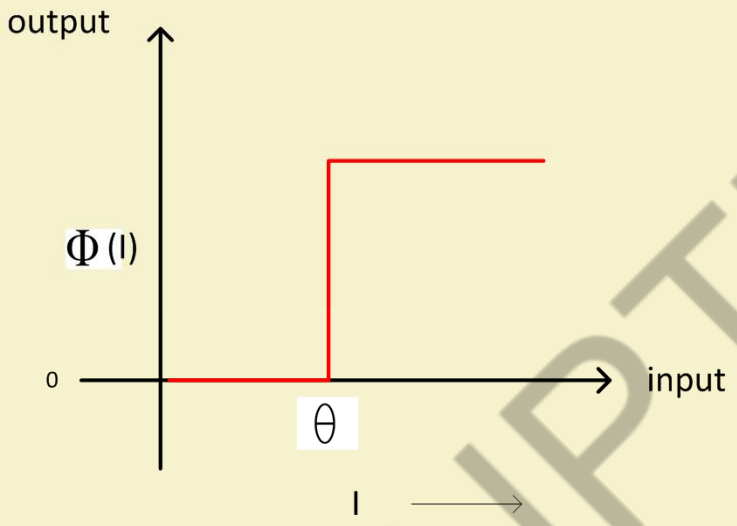
- **Hard-limit transfer function** : The transformation we have just discussed is called hard-limit transfer function. It is generally used in perception neuron.
  In other words,

$$\phi(I) = \begin{cases} 1 & ,if\ I > \theta \\ 0 & ,if\ I \leq \theta \end{cases}$$

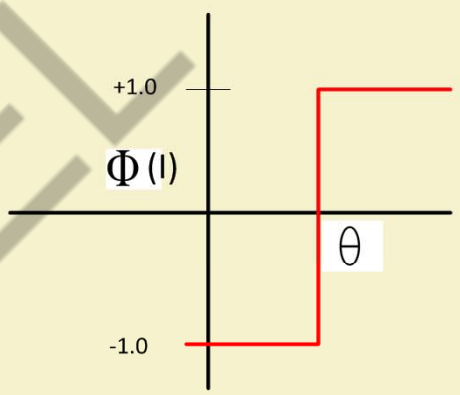
- **Linear transfer function :** The output of the transfer function is made equal to its input (normalized) and its lies in the range of -1.0 to $+1.0$. It is also known as Signum or Quantizer function and it defined as

$$\phi(I) = \begin{cases} +1 & ,if\ I > \theta \\ -1 & ,if\ I \leq \theta \end{cases}$$

Artificial neural network



(a) Hard-limit transfer function



(b) Signum transfer function

Transformation functions

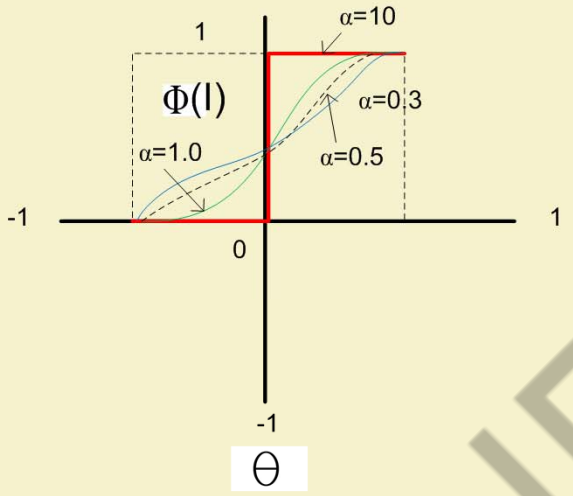
- **Sigmoid transfer function** : This function is a continuous function that varies gradually between the asymptotic values 0 and 1 (called log-sigmoid) or -1 and +1 (called Tan-sigmoid) threshold function and is given by

$$\phi(I) = \frac{1}{1 + e^{-\alpha l}} [log - Sigmoid]$$

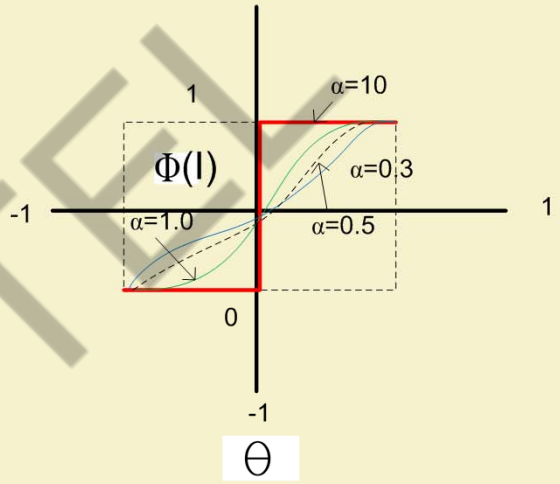
$$\phi(I) = \tanh(I) = \frac{e^{\alpha l} - e^{-\alpha l}}{e^{\alpha l} + e^{-\alpha l}} [\tan - Sigmoid]$$

Here, α is the coefficient of transfer function.

Transfer functions in ANN



(a) Log-Sigmoid transfer function



(b) Tan-Sigmoid transfer function

Advantages of ANN

- ANNs exhibits mapping capabilities, that is, they can map input patterns to their associated output pattern.

- The ANNs learn by examples. Thus, an ANN architecture can be trained with known example of a problem before they are tested for their inference capabilities on unknown instance of the problem. In other words, they can identify new objects previous untrained.

- The ANNs posses the capability to generalize. This is the power to apply in application where exact mathematical model to problem are not possible.

Advantages of ANN

- The ANNs are robust system and fault tolerant. They can therefore, recall full patterns from incomplete, partial or noisy patterns.
- The ANNS can process information in parallel, at high speed and in a distributed manner.
- Thus a massively parallel distributed processing system made up of highly interconnected (artificial) neural computing elements having ability to learn and acquire knowledge is possible.

Thank You!!

IIT KHARAGPUR

NPTEL ONLINE
CERTIFICATION COURSES

Introduction to Soft Computing

ANN Architectures

Debasis Samanta
Department of Computer Science and Engineering
IIT KHARAGPUR

Neural network architectures

There are three fundamental classes of ANN architectures:

- Single layer feed forward architecture
- Multilayer feed forward architecture
- Recurrent networks architecture

Before going to discuss all these architectures, we first discuss the mathematical details of a neuron at a single level. To do this, let us first consider the AND problem and its possible solution with neural network.

The AND problem and its Neural network

The simple Boolean AND operation with two input variables x_1 and x_2 is shown in the truth table.

Here, we have four input patterns: $00, 01, 10$ and 11.

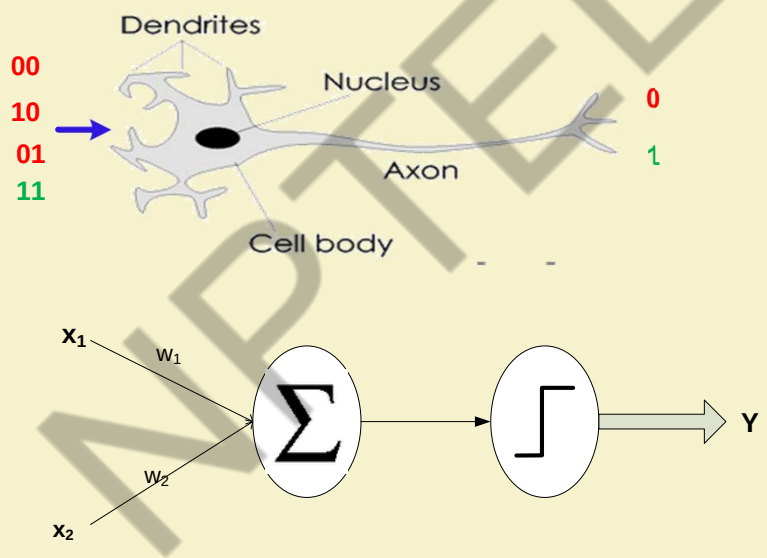
For the first three patterns output is 0 and for the last pattern output is 1.

Inputs x_1	Inputs x_2	Output (y)
0	0	0
0	1	0
1	0	0
1	1	1

The AND Logic

The AND problem and its Neural network

Alternatively, the AND problem can be thought as a perception problem where we have to receive four different patterns as input and perceive the results as 0 or 1.

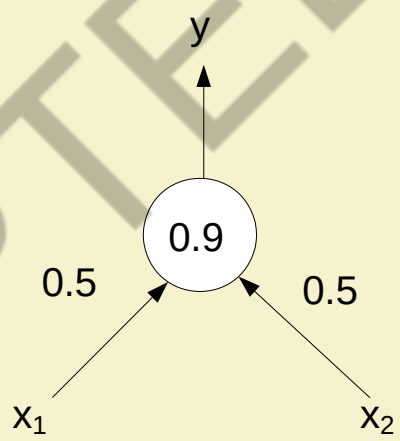


The AND problem and its Neural network

A possible neuron specification to solve the AND problem is given in the following. In this solution, when the input is 11, the weight sum exceeds the threshold (= 0.9) leading to the output 1 else it gives the output 0.

Inputs x_1	Inputs x_2	Output (y)
0	0	0
0	1	0
1	0	0
1	1	1

The AND Logic



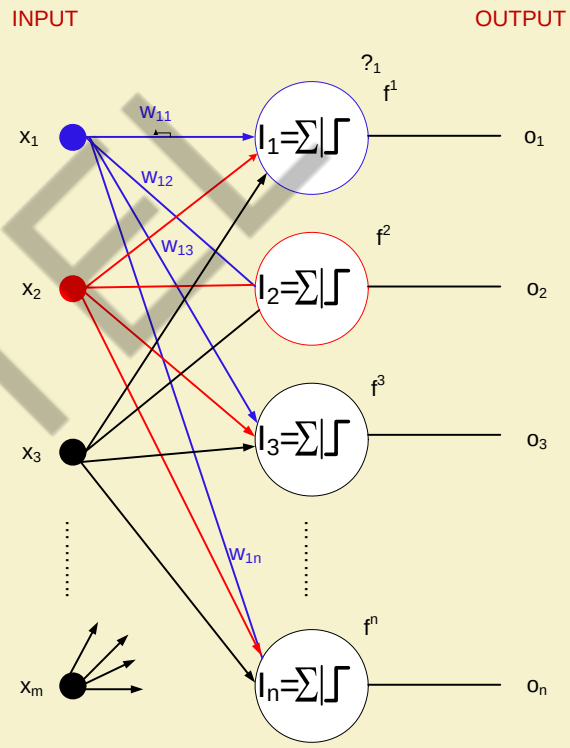
A single neuron

Here

$$y = \sum w_i x_i - \theta$$
$$w_1 = 0.5$$
$$w_2 = 0.5$$
$$\theta = 0.9$$

Single layer feed forward neural network

The concept of the AND problem and its solution with a single neuron can be extended to multiple neurons.



Single layer feed forward neural network

- We see, a layer of n neurons constitutes a single layer feed forward neural network.
- This is so called because, it contains a single layer of artificial neurons.
- Note that the input layer and output layer, which receive input signals and transmit output signals are although called layers, they are actually boundary of the architecture and hence truly not layers.
- The only layer in the architecture is the synaptic links carrying the weights connect every input to the output neurons.

Modeling SLFFNN

In a single layer neural network, the inputs $x_1, x_2, \ldots\ldots\ldots x_m$ are connected to the layers of neurons through the weight matrix W. The weight matrix $W_{m\times n}$ can be represented as follows.

$$W = \begin{vmatrix} W_{11} & W_{12} & W_{13} \cdots & W_{1n} \\ W_{21} & W_{22} & W_{23} \cdots & W_{2n} \\ \ldots & \ldots & \ldots \cdots & \ldots \\ W_{m1} & W_{m2} & W_{m3} \cdots & W_{mn} \end{vmatrix} \quad (1)$$

The output of any k^{th} neuron can be determined as follows.

$$O_k = f_k\left(\sum_{i=1}^{m}(W_{ik}X_i) + \theta_k\right)$$

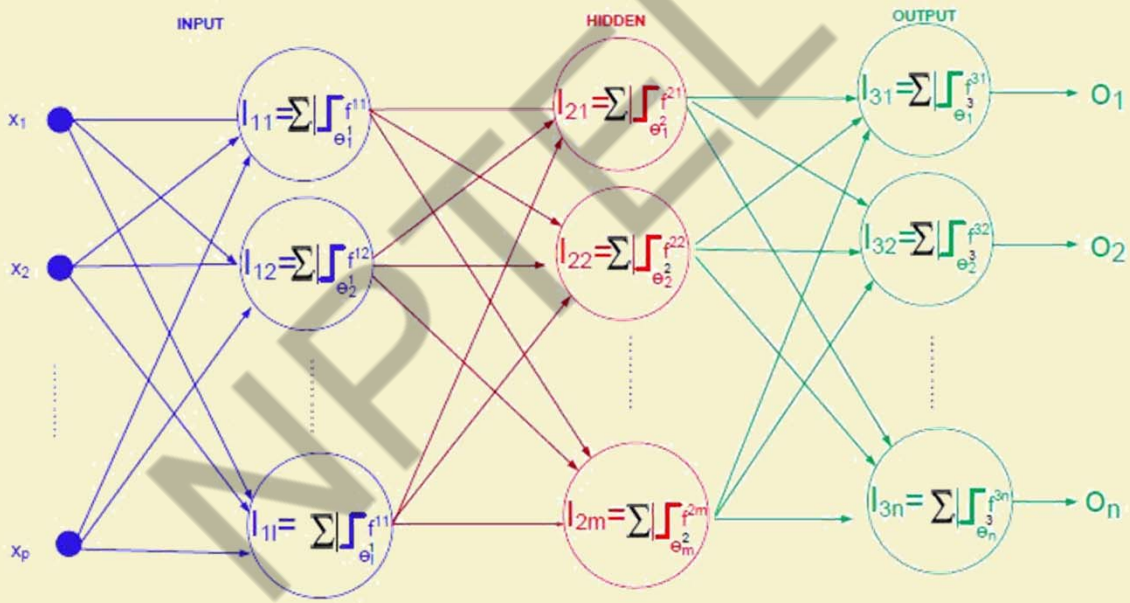
where $k = 1, 2, 3, \ldots\ldots, n$ and θ_k denotes the threshold value of the k^{th}neuron. Such network is feed forward in type or acyclic in nature and hence the name.

Multilayer feed forward neural networks

- This network, as its name indicates is made up of multiple layers.
- Thus architectures of this class besides processing an input and an output layer also have one or more intermediary layers called hidden layers.
- The hidden layer(s) aid in performing useful intermediary computation before directing the input to the output layer.
- A multilayer feed forward network with l input neurons (number of neuron at the first layer), $m_1, m_2, \ldots\ldots m_p$ number of neurons at i^{th} hidden layer $(i = 1, 2, \ldots\ldots, p)$ and n neurons at the last layer (it is the output neurons) is written as $\mathrm{l} - m_1 - m_2 - \ldots\ldots - m_p - n$ MLFFNN.

Multilayer feed forward neural networks

Figure shows a schematic diagram of multilayer feed forward neural network with a configuration of $l - m - n$.



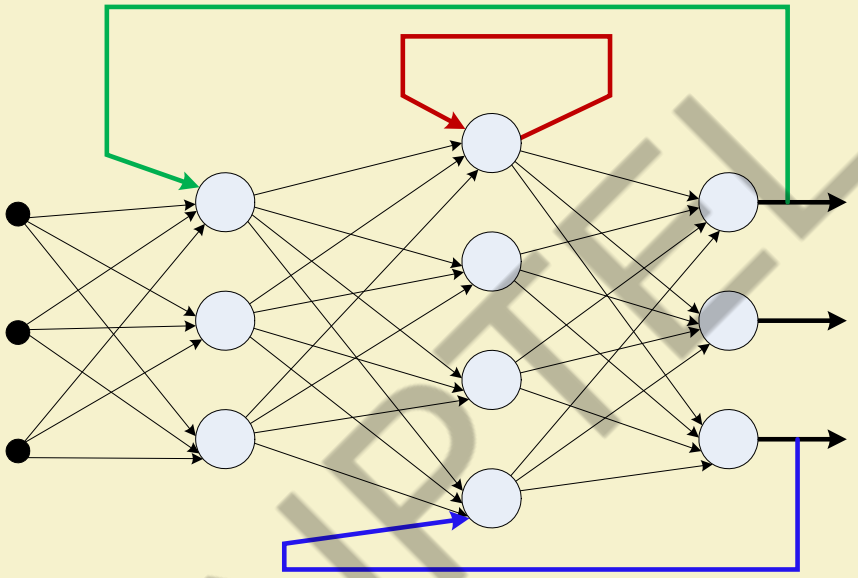
Multilayer feed forward neural networks

- In $l - m - n$ MLFFNN, the input first layer contains l number of neurons, the hidden layer contains m number of neurons and the last (output) layer contains n number of neurons.
- The inputs $x_1, x_2, \ldots \ldots x_p$ are fed to the first layer and the weight matrices between input and the first layer, the first layer and the hidden layer and those between hidden and the last (output) layer are denoted as W^1, W^2 and W^3 respectively.
- Further, consider that f^1, f^2 and f^3 are the transfer functions of neurons lying on the first, hidden and the last layers, respectively.
- Likewise, the threshold values of any i^{th} neuron in j^{th} layer is denoted by θ_i^j.
- Moreover, the output of i^{th}, j^{th}, and k^{th} neuron in any l^{th} layer is represented by $O_i^l = f_i^l(\sum X_l W^l + \theta_i^l)$ where X_l is the input vector of l^{th} layer.

Recurrent neural network architecture

- The networks differ from feedback network architectures in the sense that there is at least one "feedback loop".
- Thus, in these networks, there could exist one layer with feedback connection.
- There could also be neurons with self-feedback links, that is, the output of a neuron is fed back into itself as input.

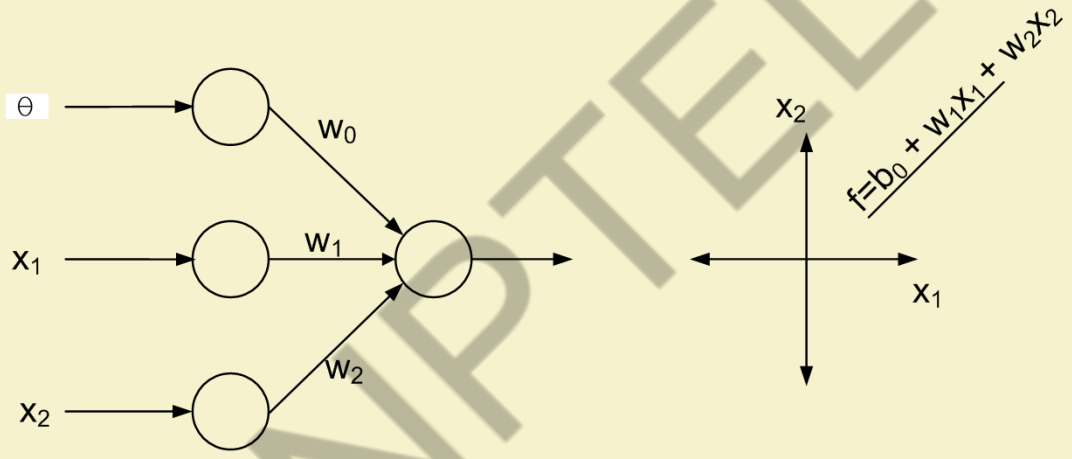
Recurrent neural network architecture



Depending on different type of feedback loops, several recurrent neural networks are known such as Hopfield network, Boltzmann machine network etc.

Why different type of neural network architectures?

To give the answer to this question, let us first consider the case of a single neural network with two inputs as shown below.



$$f=w_0\theta + w_1x_1 + w_2x_2$$
$$=b_0 + w_1x_1 + w_2x_2$$

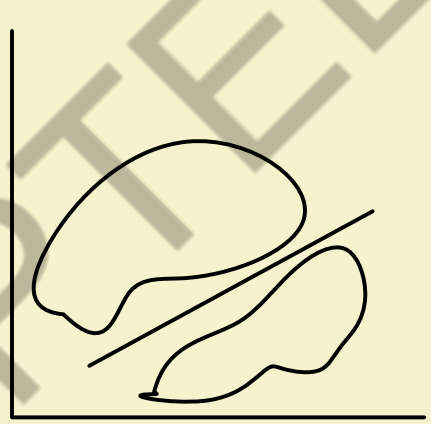
Revisit of a single neural network

Note that $f = b_0 + w_1 x_1 + w_2 x_2$ denotes a straight line in the plane of $x_1 - x_2$ (as shown in the figure (right) in the last slide).

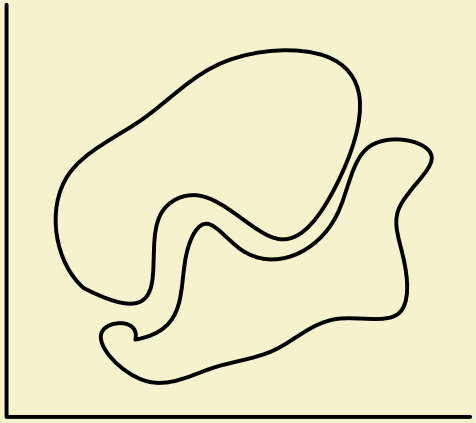
Now, depending on the values of w_1 and w_2, we have a set of points for different values of x_1 and x_2.

We then say that these points are linearly separable, if the straight line f separates these points into two classes.

Linearly separable and non-separable points are further illustrated in Figure.



Linearly Seperable

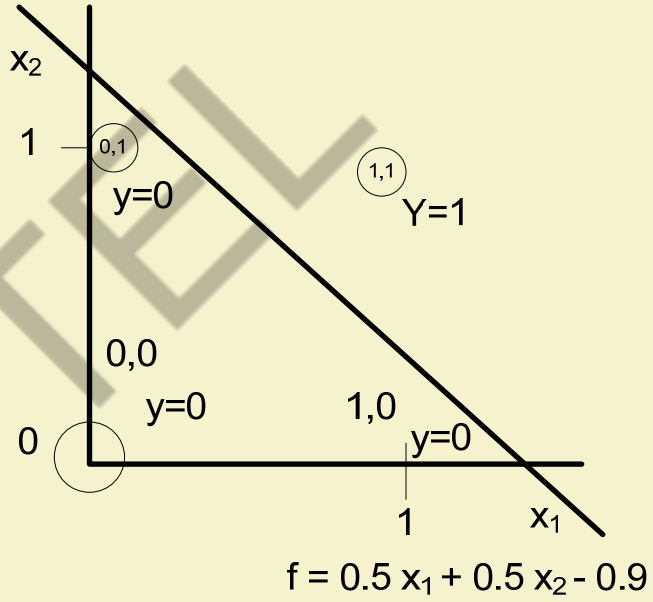


Linearly Non-Seperable

AND problem is linearly separable

x_1	x_2	Output (y)
0	0	0
0	1	0
1	0	0
1	1	1

The AND Logic

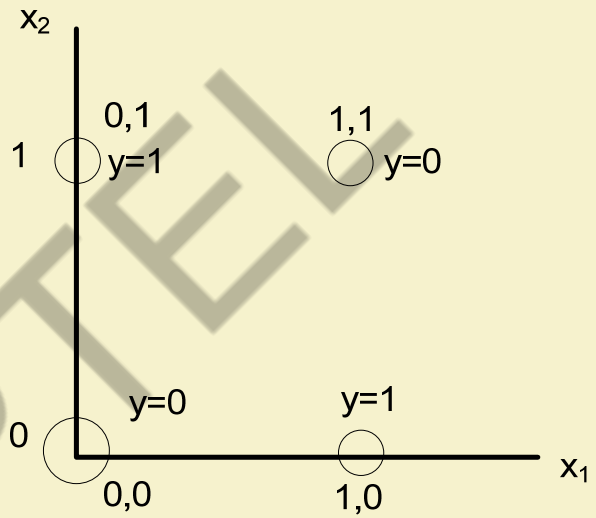


AND-problem is linearly separable

XOR problem is linearly non-separable

x_1	x_2	Output (y)
0	0	0
0	1	1
1	0	1
1	1	0

XOR Problem



XOR-problem is non-linearly separable

Our observations

- From the example discussed, we understand that a straight line is possible in AND-problem to separate two tasks namely the output as 0 or 1 for any input.
- However, in case of XOR problem, such a line is not possible.

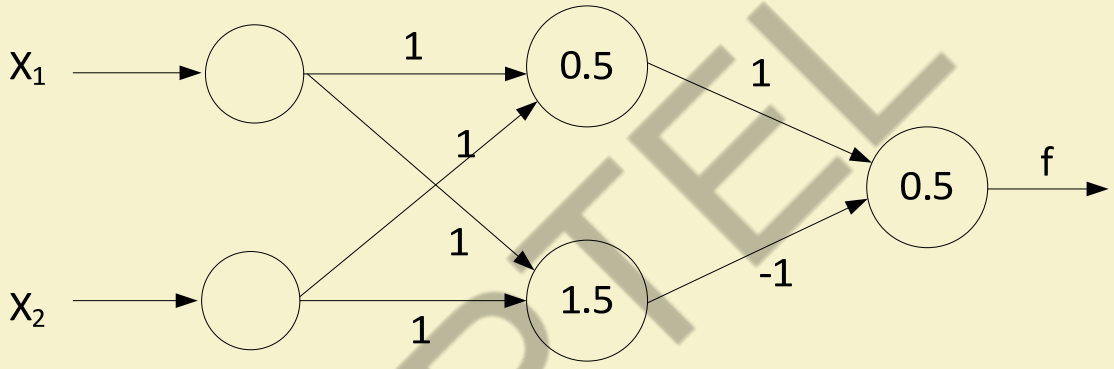
Note:

Horizontal or a vertical line in case of XOR problem is not admissible because in that case it completely ignores one input.

Example

- So, far a 2-classification problem, if there is a straight line, which acts as a decision boundary then we can say such problem as linearly separable; otherwise, it is non-linearly separable.
- The same concept can be extended to n-classification problem. Such a problem can be represented by an n-dimensional space and a boundary would be with $n-1$ dimensions that separates a given sets.
- In fact, any linearly separable problem can be solved with a single layer feed forward neural network. For example, the AND problem.
- On the other hand, if the problem is non-linearly separable, then a single layer neural network can not solves such a problem. To solve such a problem, multilayer feed forward neural network is required.

Example: Solving XOR problem

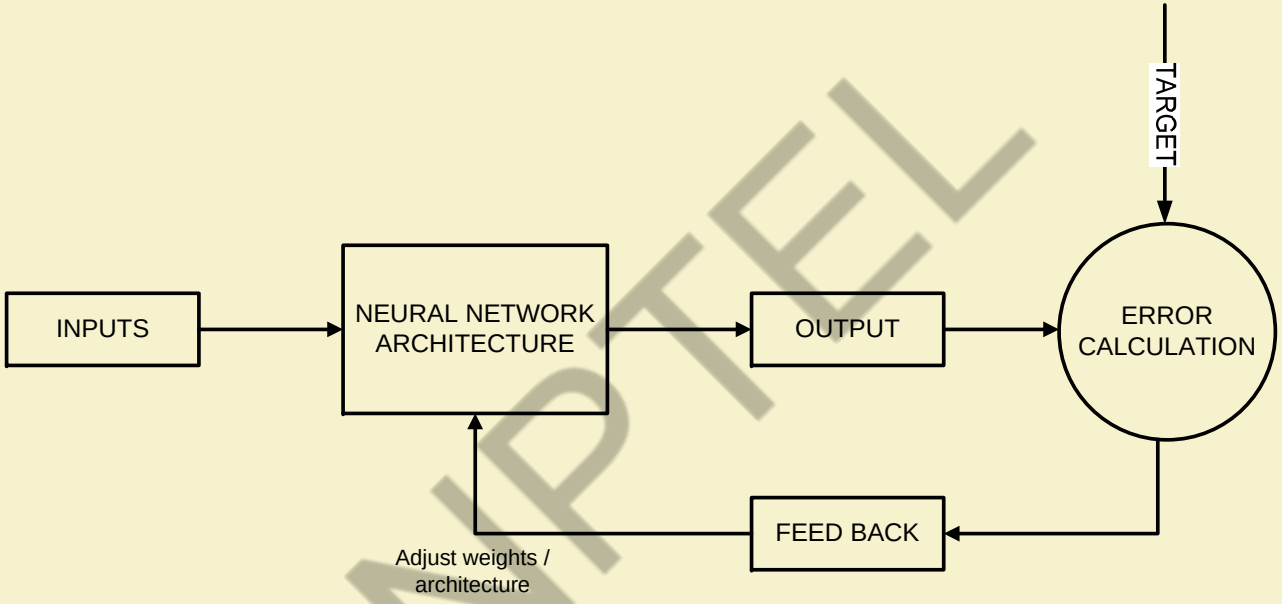


Neural network for XOR-problem

Dynamic neural network

- In some cases, the output needs to be compared with its target values to determine an error, if any.
- Based on this category of applications, a neural network can be static neural network or dynamic neural network.
- In a static neural network, error in prediction is neither calculated nor feedback for updating the neural network.
- On the other hand, in a dynamic neural network, the error is determined and then feed back to the network to modify its weights (or architecture or both).

Dynamic neural network



Framework of dynamic neural network

Dynamic neural network

From the above discussions, we conclude that

- For linearly separable problems, we solve using single layer feed forward neural network.
- For non-linearly separable problem, we solve using multilayer feed forward neural networks.
- For problems, with error calculation, we solve using recurrent neural networks as well as dynamic neural networks.

Thank You!!