23/7/18

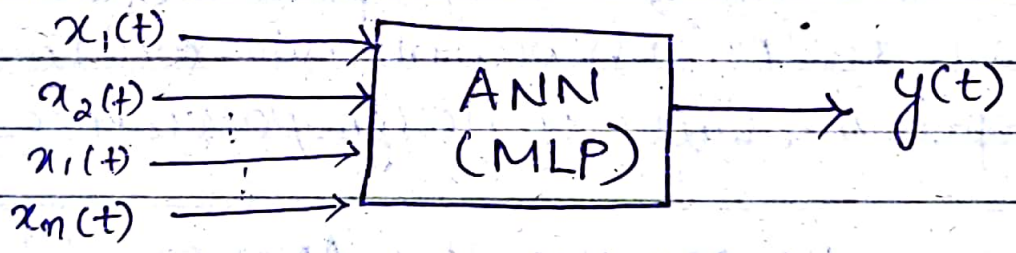
ANN Model (MLP - Multi-layer perceptron).

(i) Architecture.
(ii) Kind of learning — unsupervised - used for reg. and fn approx.
                     — supervised.
                     — Reinforced
(iii) Application area.

(1) Feed-forward — MLP
                 — Kohnen self-organising

(2) Feed back network
    Hopfield network
    Recurring network.

(1) Training phase.
(2) Testing phase.

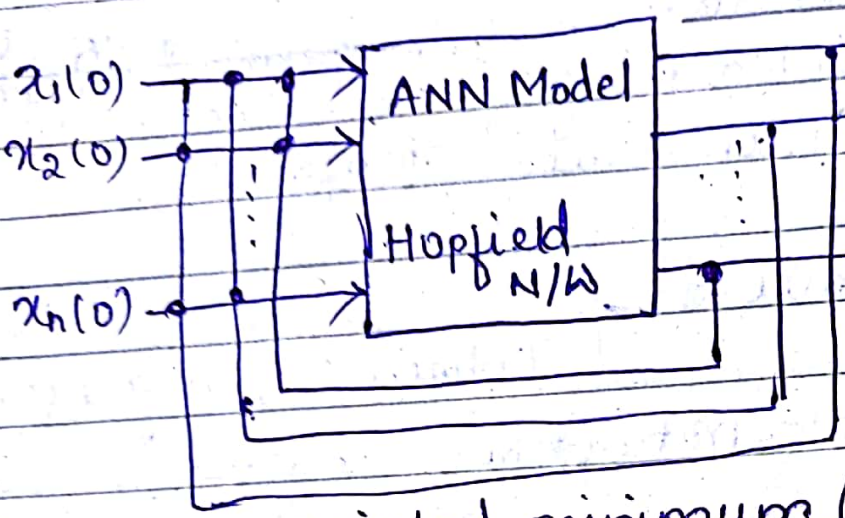


Kohonen Self-organizing feature map.

(1) feed forward network.
(2) Unsupervised learning.
(3) Pattern recognition or pattern matching or pattern classification.

24/7/18

Hopfield network



→ Energy associated minimum (then it stops).
→ Used in memory recall & optimization
→ Doesn't have training phase.

We use this network in an e.g.:- like designing a product in which we minimize its cost. There are certain design parameters and certain set of constraints. We are looking for optimum value of design parameters.

In general, we can use this ANN model in optimization problem.

→ In some ANN mode self feedback is allowed and in some it is not allowed.

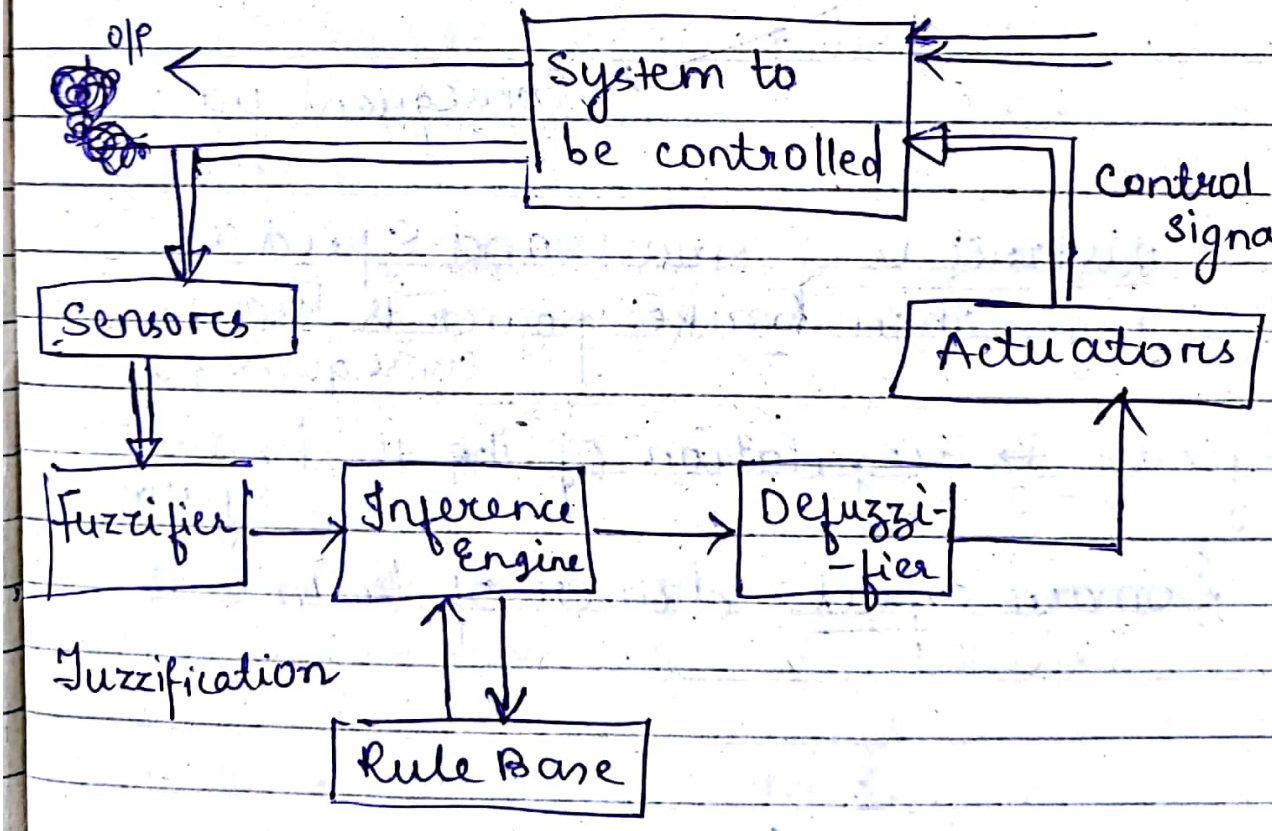
→ Model will give exact form of the data which is the corrupted data given as

I/P → memory recall.

Some models which are hybrid of these 3 basic ANN models (MLP, Kohnen, Hopfield).

① Radial basis f^n N/W → Hybrid of MLP + KSOM.
→ First it goes through this classification state & then interpolation state.
→ i.e., 1st it goes through supervised learning and then unsupervised learning

Fuzzy System:



→ Braking system.

In conventional control system → mathematical model.
In fuzzy " " → algorithms.

Disadv. of ccs → mathematical model take much more time & calculations.

Rule base → based on the expert modelling. (domain's expert).
So, it need alot of tuning, as to reach precision.

30/7/18

Rule base — Antecedant part.
→ Consequent part.

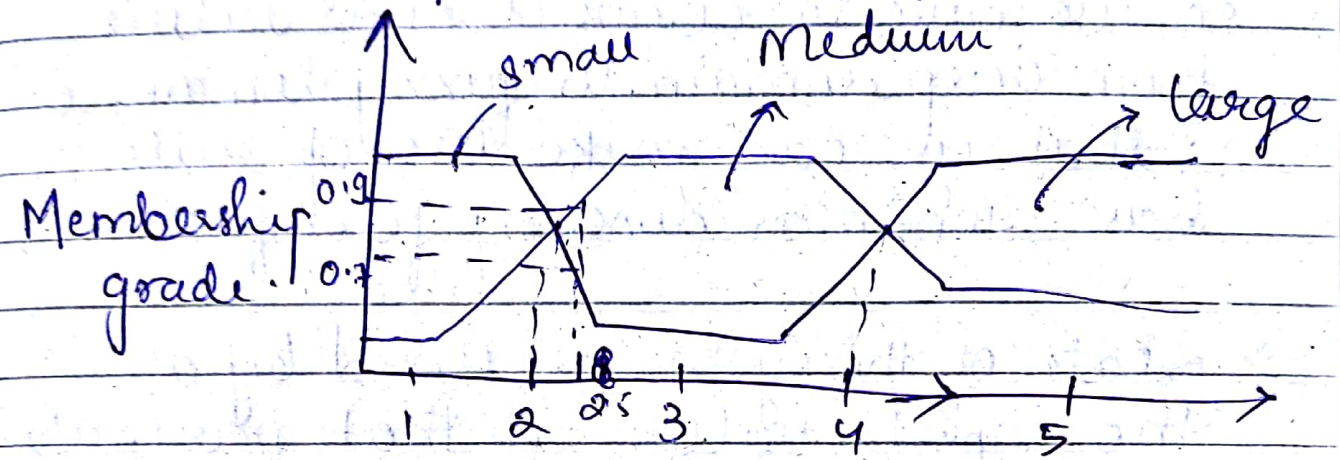
If distance u is small and speed is high, then brake power is hiard.
Antecedant part ← (If distance u is small and speed is high,) (then brake power is hiard.) → Consequent part.

Objective → regulation of the the brake system.

Domain expert - driver of train.

→ small, hard, high → fuzzy values.

→ To what extent a particular physical quantity is small, large or medium is known as degree of membership or membership grade.



Our universe of discourse is 0 to 5 km and universe of discourse consist of fuzzy sets. Here we have 3 fuzzy sets (small, medium & large).

As here at 2.5 km, we can have small value or medium value.
So we will see membership grade.
→ For small here it is 0.7 and for medium it is 0.9.
So, 2.5 belongs to medium fuzzy set.

→ The domain expert will make rule base according to the fuzzy levels and he will always talk about fuzzy levels and his knowledge is in fuzzy domain

→ Our rule base is in fuzzy domain, so we have to convert our values from crisp domain to fuzzy domain, so that we can make use of rule base; which is done by fuzzifier.

→ State of the system is served by all the inputs values at that particular time. So now again we want crisp values, so our fuzzy values are needed to be converted to the crisp values.
So the whole system is designed in this manner.

→ Rule base is written for the entire system, not for one state.
At one state, some rules are relevant and some are irrelevant.

→ Degree of firing of a particular rule → related to the relevance of the rule depending on the state of the system.

→ Degree of firing is b/w 0 and 1.
A rule which is relevant will have a value close to 1 and will highly influence the control signal; a rule which is irrelevant it will have a value close to 0 and its influence on control signal is invisible.

31/7/18

Rule-base.

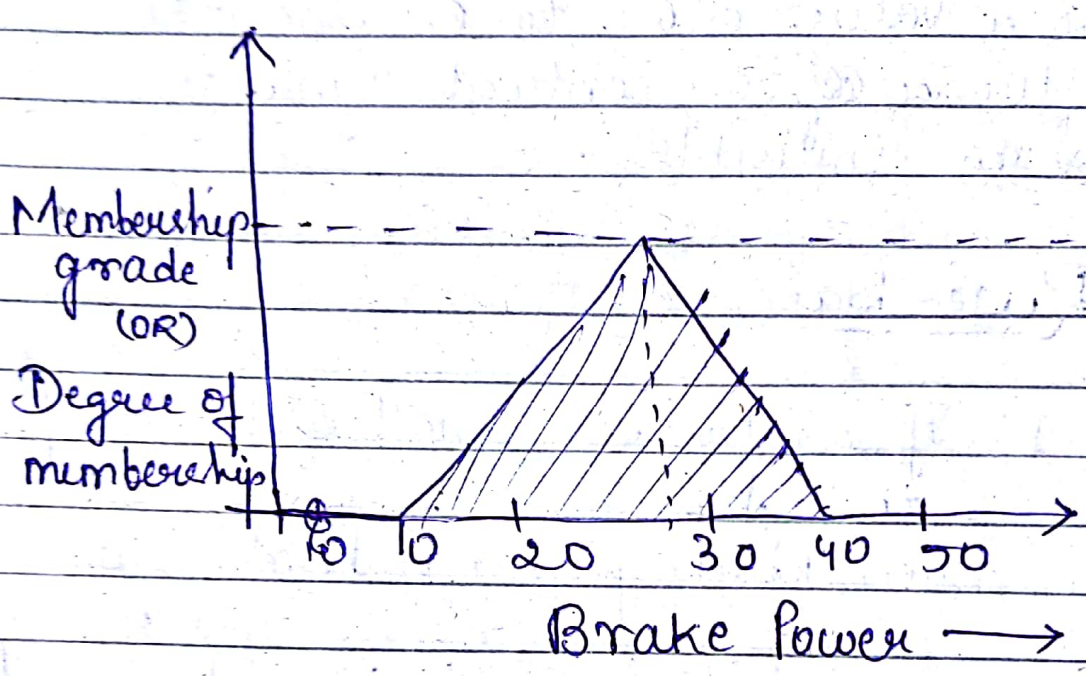
Rule 1: If distance is small AND Speed is high.
Then brake power is hard.

→ State of the system is being identified by the crisp values of the I/P variables.

→ If Rule 1 is totally irrelevant to the current state of the system, then Rule 1 will have value close to 0; and then the break power is hard to a lesser extent & vice versa.

→ Every rule will give inferied fuzzy outcome, and this outcome will be on higher side, if DOF is on the higher side.

→ We will find the inferred fuzzy outcome of all the rules and together they will give aggregated inferred fuzzy outcome.



→ Converting or giving a crisp value to the aggregated inferred fuzzy outcome is known as defuzzification.

There are many methods to do this :-

1. Centre of Area or Centroid method.
2. Mean of maxima.
3. Method of heights.

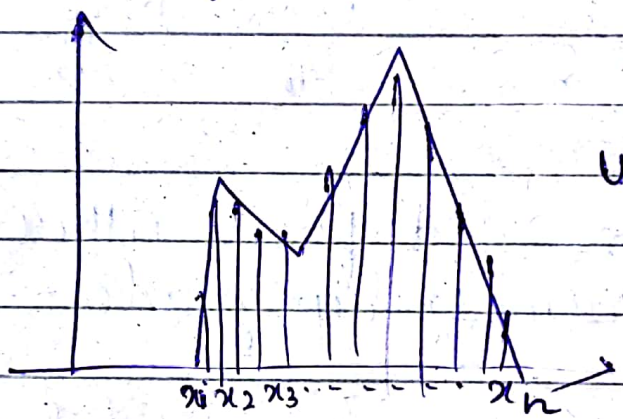
① Centroid method:

$$y = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

Numerator: summation of weights multiplied by the members to whom that weight is assigned.

Denominator: Sum of the weights

$x_i \rightarrow$ we divide our fuzzy sets into a number of members.



we divide our fuzzy system into n parts.

→ In case of fuzzy set it can take any value from 0 to 1 (not 0), not like in case of crisp values which can have value either 0 or 1.
(either membership exist - 1
no membership exist - 0)

Eg:- $X = \{a, b, c, \ldots\ldots\ldots x, y, z\}$

$A = \{a, e, i, o, u\}$ → will have crisp values

$\tilde{B} = [0.1/a, 0.2/b, 0.9/c, 0.5/d, 0.7/f]$

→ fuzzy set ; can have value b/w 0 and 1.

Here, a have membership grade of 0.1
b " " " " 0.2
c " " " " 0.9

Leaving a, b, c, d, f all other members will have membership grade = 0.

→ Fuzzy sets are used to model situations where there is uncertainty.

$\tilde{C} = [0.2/a, 0.4/b, 0.6/c, 0.8/d, 0.9/g]$

Both $\tilde{C}$ & $\tilde{B}$ are defined over a common universal of discourse.

→ The most fuzzier set will have a Memebership grade of 0.5 (as we cant decide whether it is a member or not)

Complement of $\tilde{B}$ is known as negation of $\tilde{B}$

$$\tilde{B}' = [0.9/a, 0.8/b, 0.1/c, 0.5/d, 0.7/f]$$

→ Most fuzzy → when fuzzyset and negation of fuzzyset is almost identical

→ If a fuzzyset and its complement of a fuzzy set are far from one another (the distance is maximum) then this is the case of lesser set being fuzzy.

Tuning includes :-

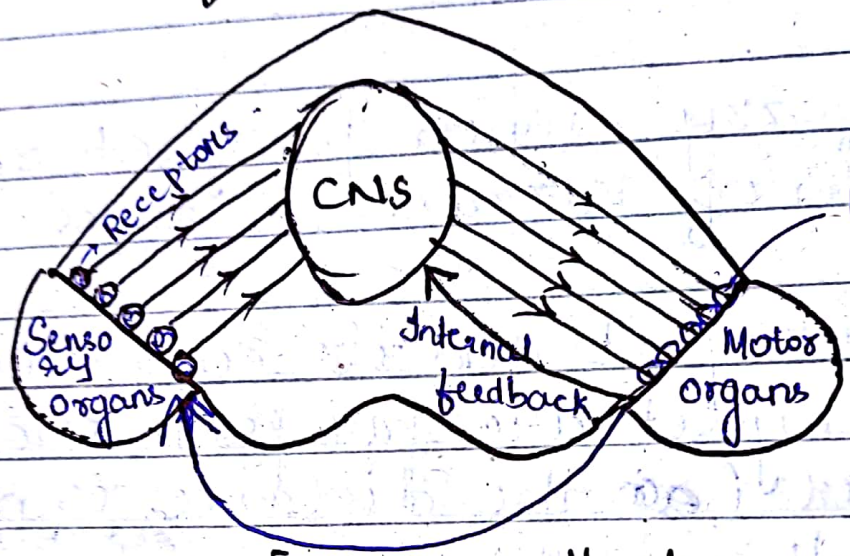
→ Length of the rule base.
→ No:- of the fuzzy sets.
→ Position of the entire fuzzy set.

— X —

2/8/18

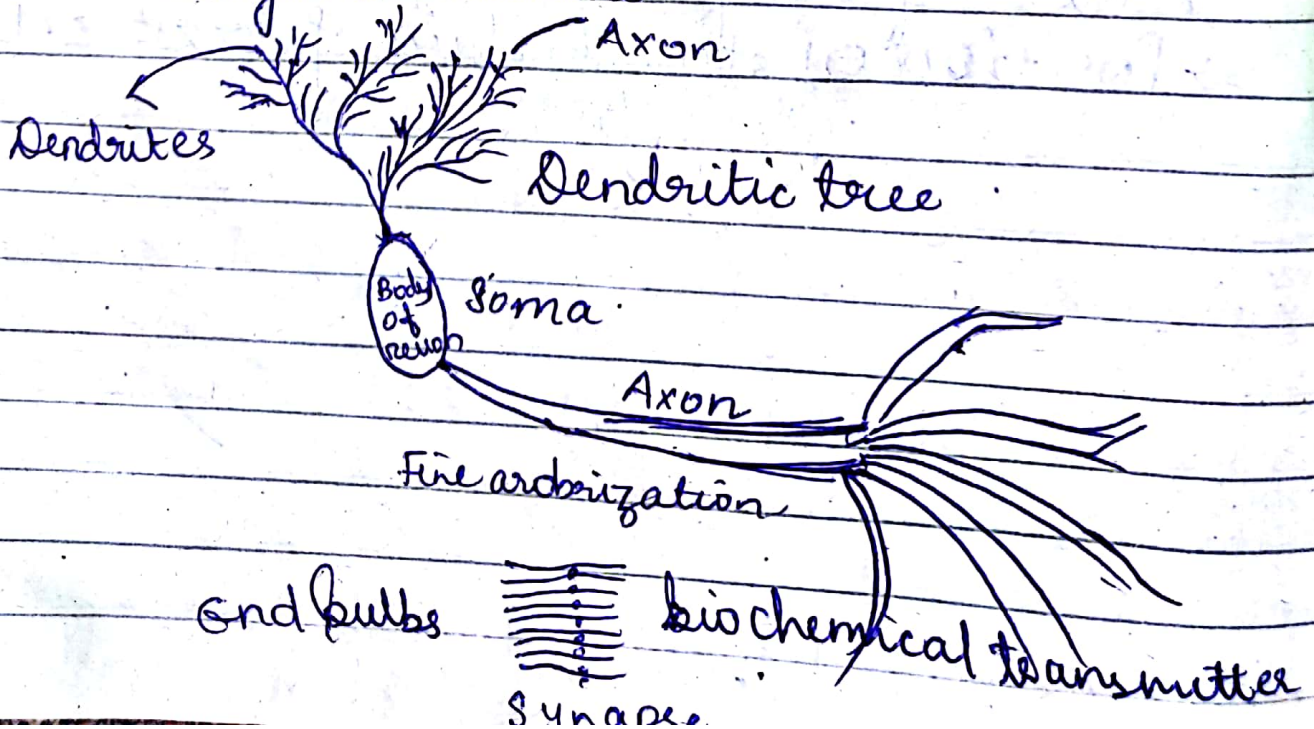
Functioning of Human Brain.

~~External feedback flow~~



information flow in nervous system.

→ The main or the basic processing element or the fundamental building blocks is known as biological neurons.



Because when the electrical pulse is carried to the end points, generation of chemical transmitters and these transmitters will be sensed by dendrites & carried to neighbouring neuron. This is why it is called of synapse

① whether signal is received over a specific of time (period of latent submission).

Aggregat signal > membrane potential of neuron (threshold)
then the ~~signal will be fired~~ or firing of neuron.

Signals.

Excitory Inhibitory

(If all the signals or I/P being received by neurons are excitory in nature; then there will be maximum chances of firing).

→ If arrival of signal is uniformly distributed over period of latent submission then chances of firing will be on lower side.

→ The biochemical transmitter has the ability to make signals inhibitory of excitory.

⇒ inhibitory → -1 (synaptic wt).
excitory → +1 (").

Refractory period → Within this period there will no generation of electrical pulse / O/P is dead.

Functioning of the biological neuron -

Signals received from outside world or neighbouring neurons, over a certain point of time

↓

Aggregate of signals

↓

Compared to membrane potential

↓

If aggregate > membrane potential

No ← No firing

Yes

↓

Firing of neuron

↓

Electrical impulse is Carried to the end of axon

↓

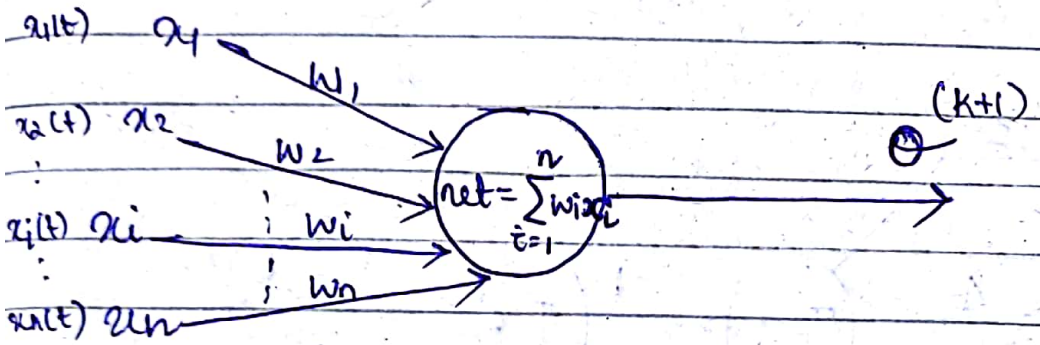
Generation of biochemical transmitter.

↓

Make some signal excitory and some signal & inhibitory.

Mathematical model of biological neuron

McCulloch & Pitts Model of biological neuron



$x_1, x_2 \cdots x_n$ are inputs

$w_1, w_2 \cdots w_i, \cdots w_n$ are synaptic strength.

If w_i have to make inhibitory; they are given value -1.

If w have to make excitory; then they are given a value of $+1$.

$$\theta^{(k+1)} = 1 \text{ if } \sum_{i=1}^{n} w_i x_i^{(k)} \geq T.$$

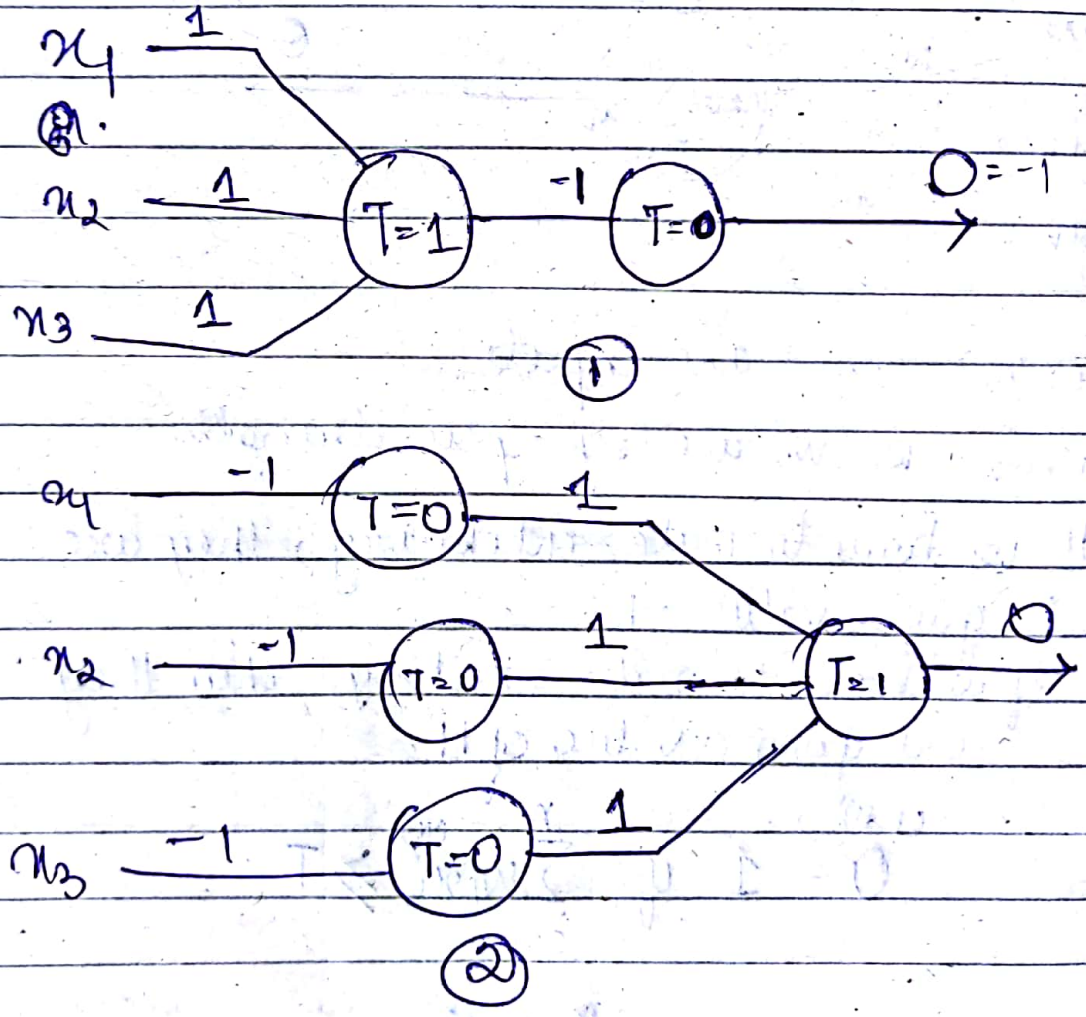
$$= 0 \text{ if } \sum_{i=1}^{n} w_i x_i^{(k)} < T$$

Where $T \rightarrow$ membrane potential (threshold).

→ This mathematical model can be

used for simulation of AND, OR, etc gates.

→ If dominance is shown by excitory I/P then the firing chance is higher else if inhibitory I/P shows dominance then firing chance is lower.



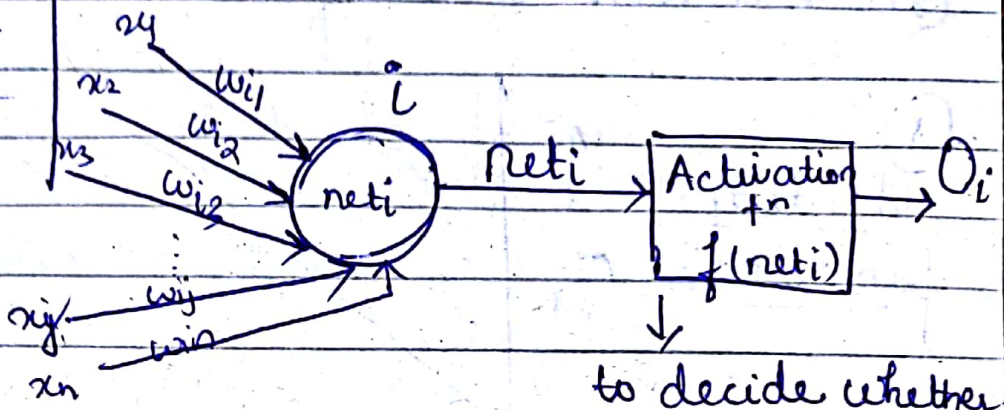
What kind of gates these are?

3/8/18

1 1 1 1

x_1	x_2	x_3
1	0	0
1	0	1
1	1	0
1	1	1

Modified or refined version of neuron model :-



to decide whether it should be fired or not

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix} \quad W = \begin{bmatrix} w_{i1} \\ w_{i2} \\ w_{i3} \\ w_{ij} \\ w_{in} \end{bmatrix}$$

(Input Vector)

(Weights)

$$neti = \sum_{j=1}^{n} w_{ij} x_j$$

(or)

$$X(t) = \begin{bmatrix} x_{t1} \\ x_{t2} \\ \vdots \\ x_{tj} \\ x_{tn} \end{bmatrix}$$

Input at certain time (t).

Types of Activation function

① Identity activation function

$$f(neti) = neti$$

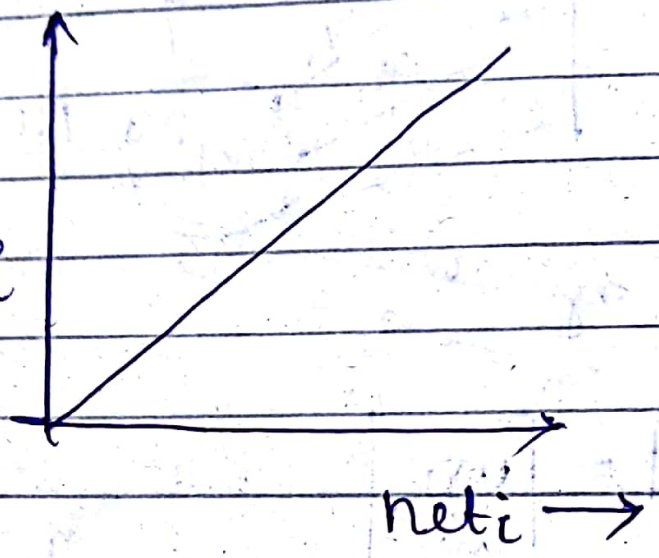
$$\because f(x) = x$$

② Unipolar binary activation function

③ Bipolar binary activation function.

④ Unipolar sigmoidal function.
⑤ Bipolar sigmoidal function.
⑥ Piecewise linear function.

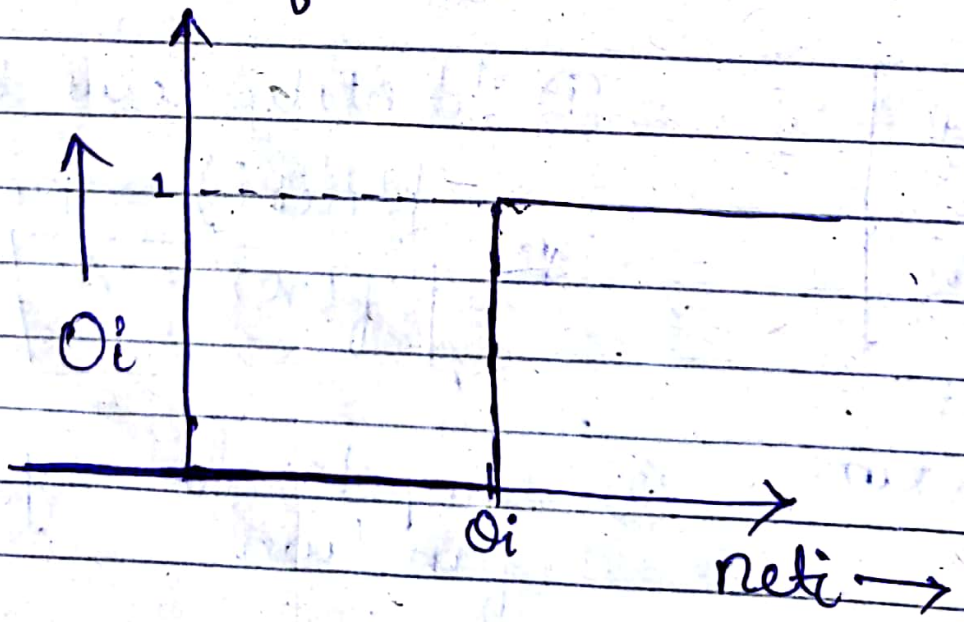
① Identity function:-



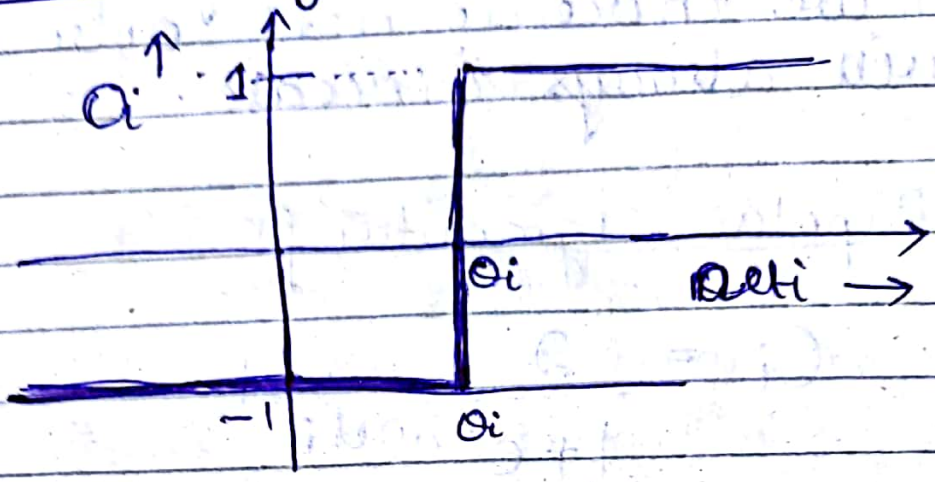
② Unipolar binary AF.

$O = 1$ if $net_i \geq \theta_i$

O if $net_i < \theta_i$

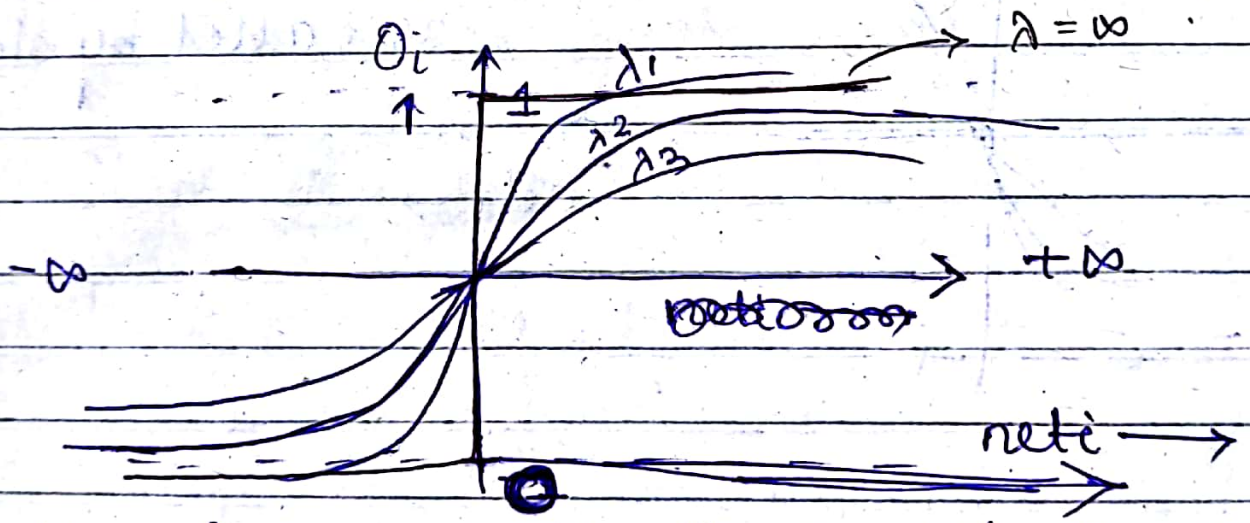


(3) Bipolar binary AF.



(4) Unipolar sigmoidal function.

$$O_i = f(net_i) = \frac{1}{1+e^{-\lambda net_i}}$$



Here $\lambda \rightarrow$ controls the slope / gain of the sigmoidal function when $net_i = 0$.

$\lambda_1 > \lambda_2 > \lambda_3$

When $\lambda = \infty$; hard limiter function

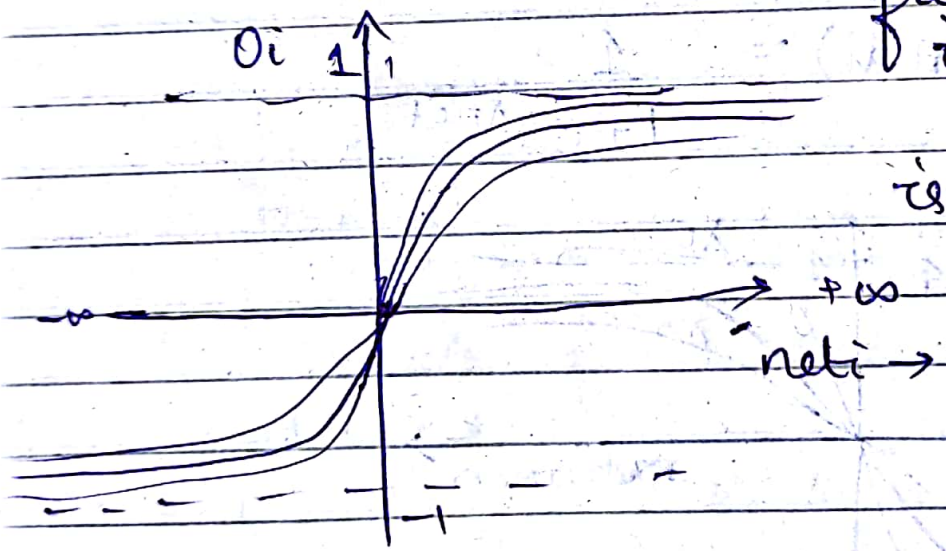
O_i [0 1] → it has 1 level so it is known as Unipolar.

→ This is an increasing function, as we increase neti our O/P will always increase.

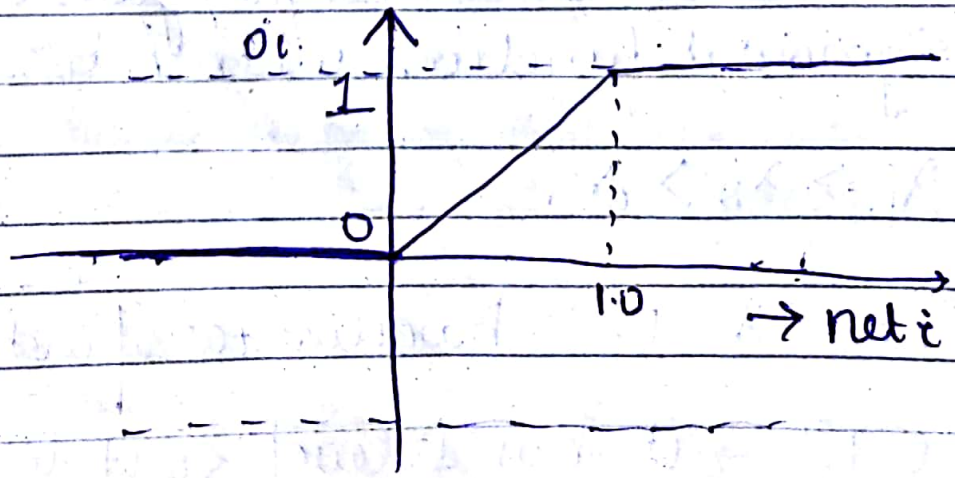
(5) Bipolar sigmoidal fn.

$$O_i = \frac{2}{1+e^{-\lambda net_i}} - 1$$

$O_i : [-1, 1]$ can have value from -1 to 1 i.e, it has 2 levels so it is called bipolar



(6) Piece-wise linear fn



$O_i = 1$ for $net_i \geq 1$

$O_i = net_i$ for $net_i < 1$

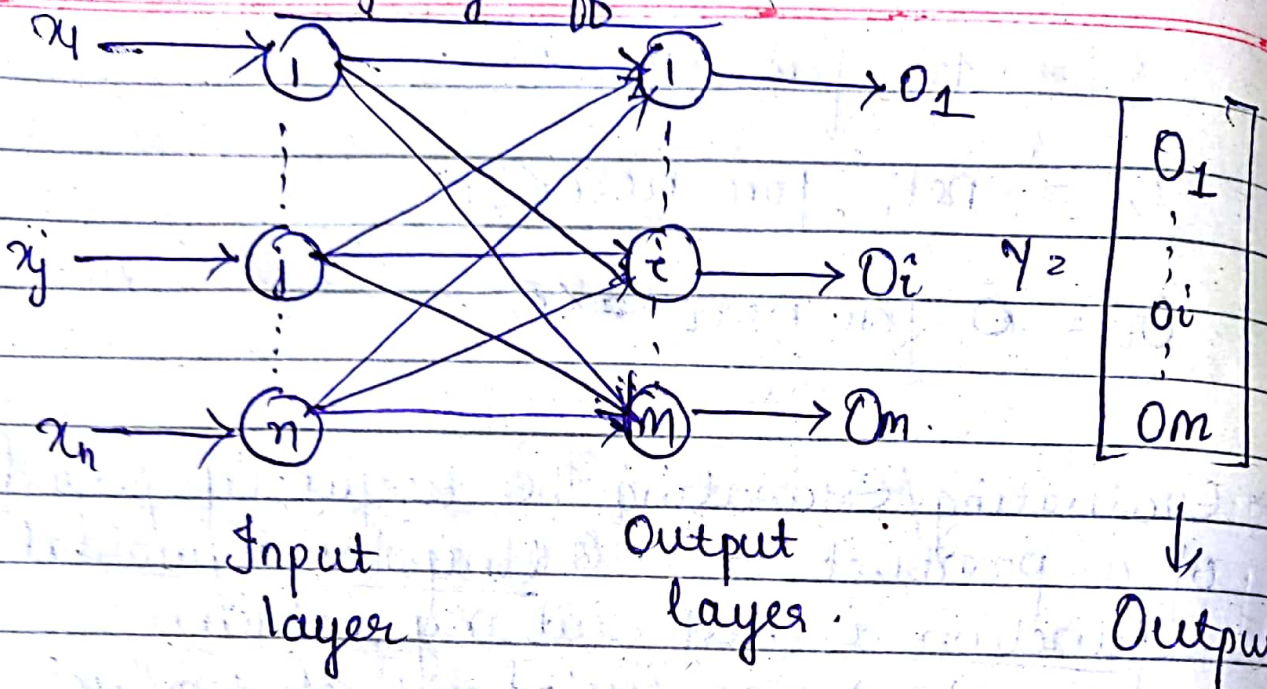
$O_i = 0$ for $net_i < 0$.

→ Evaluating/Estimating the useful life period of a product → Unipolar sigmoidal function is best suit my problem.
as Bipolar is ruled out as product life can never be -ve; and it can never be either 0 or 1, it might have continous value, so that's why unipolar sigmoidal function is best for this case.

Types of Neural Network

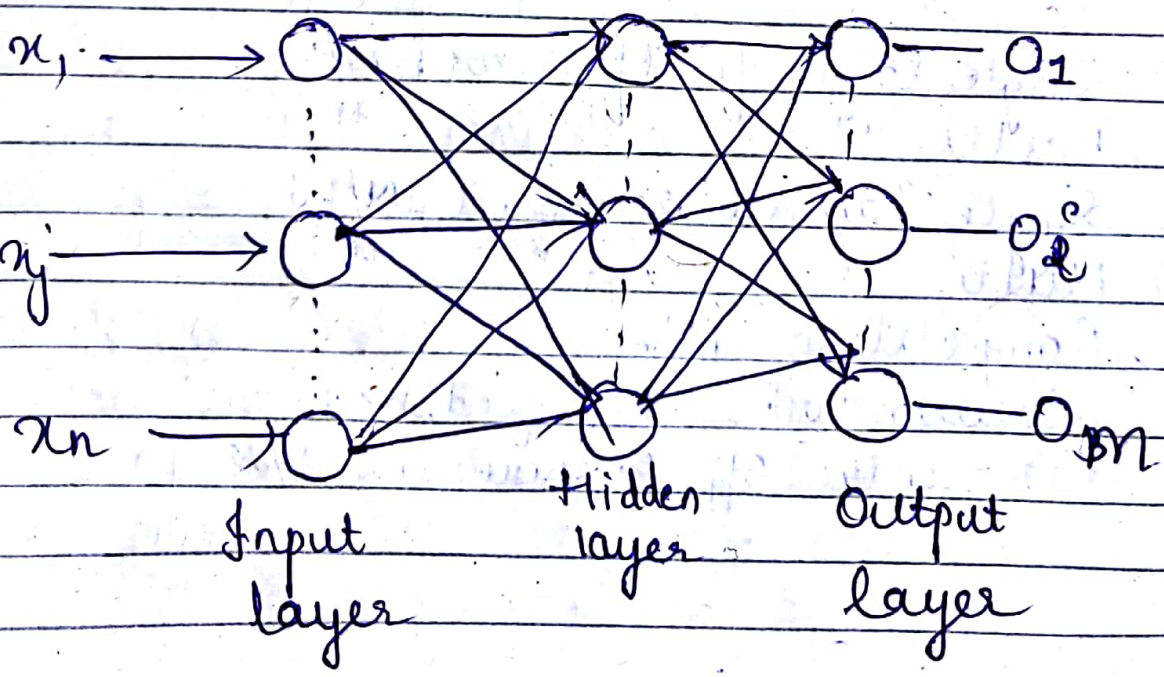
(1) Single layer feedforward N/W.
(2) Multi " " "
(3) Single layer recurrent N/W.
(4) Multi " " ".
(5) Competitive N/W.
(6) Neuron with self feedback.
(7) On-centre Off surrounding N/W.

Sing layer ff n/w

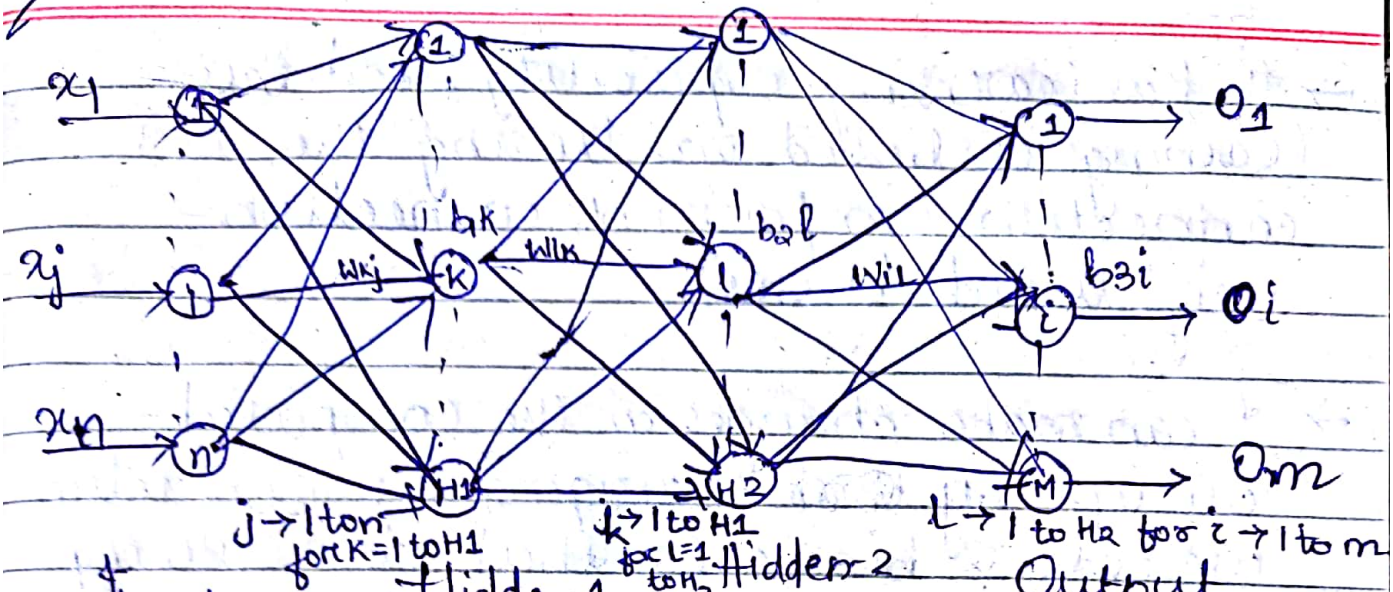


$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix} \rightarrow \text{Input Vector}$$

Multi-layer ff. N/W. with 1 Hidden layer.



6/8/18



$$X = \begin{bmatrix} x_1 \\ x_j \\ x_n \end{bmatrix}$$

$$Y = \begin{bmatrix} O_1 \\ O_i \\ O_m \end{bmatrix}$$

→ Net input to the kth node, will be compared to the activation function and the output will be ~~calculate~~ carried to the next layer and so on.

→ Whatever the final output or computed output is compared to the desired output.

→ N/W is exposed to a particular environment so that it adapt to the environment such that for a particular I/P, the O/P is closed to the desired O/P.

→ It has learning capability, and this learning is based on tuning the connection (in form of connection weights and biases).

→ I can make changes in the computed output by changing the connection weights & biases so that its learning capability become better & better or the tuning of n/w becomes better or wts & biases becomes more refined & refined.

→ No:- of I/P vectors & O/P vectors are problem dependent.

→ Depending on degree of non-linearity b/w the output & input vector, the no:- of hidden layer & no:- of nodes per hidden layer depends.

→ If degree of non-linearity is more, then we will be requiring more no:- of nodes or more no:- of hidden layers.

→ Why increasing the no:- of hidden layers or no:- of neurons per layer will help?

Ans There will be more flexibity or the parameters will increase so by tuning this we get our desired O/P.

In simple ways we have more no:- of adjustable parameters; our computational volume will increase but our accuracy will increase.

Only disadvantage → more computational burden

---

① Stepwise learning.
② Batch-wise learning.

X[1] → D[1]
X[2] → D[2]
⋮
X[p] → D[p]

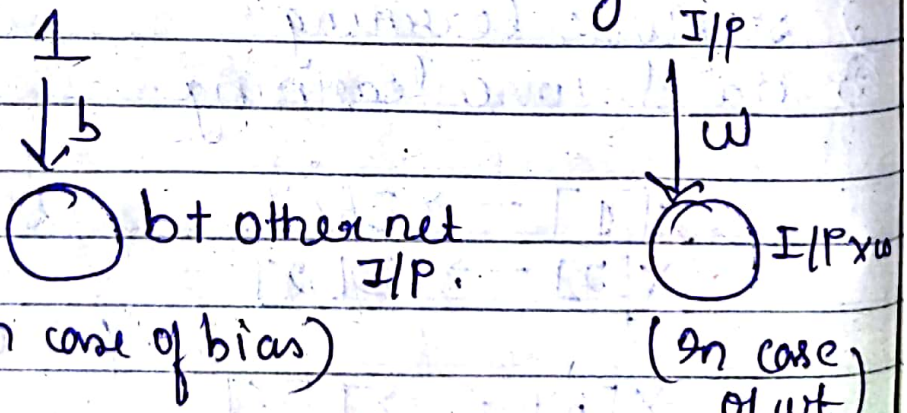
} set of input/O/P patten. or set of examples or set of instances

To learn the relation b/w O/P I/P pattern by specific learning algorith → purpose of MLP.

→ If you give an I/P X[1] and get the value of D[1] and compare it with desired O/P and take the error and make adjustment to correction wts & biases or it is known as step-wise learning.

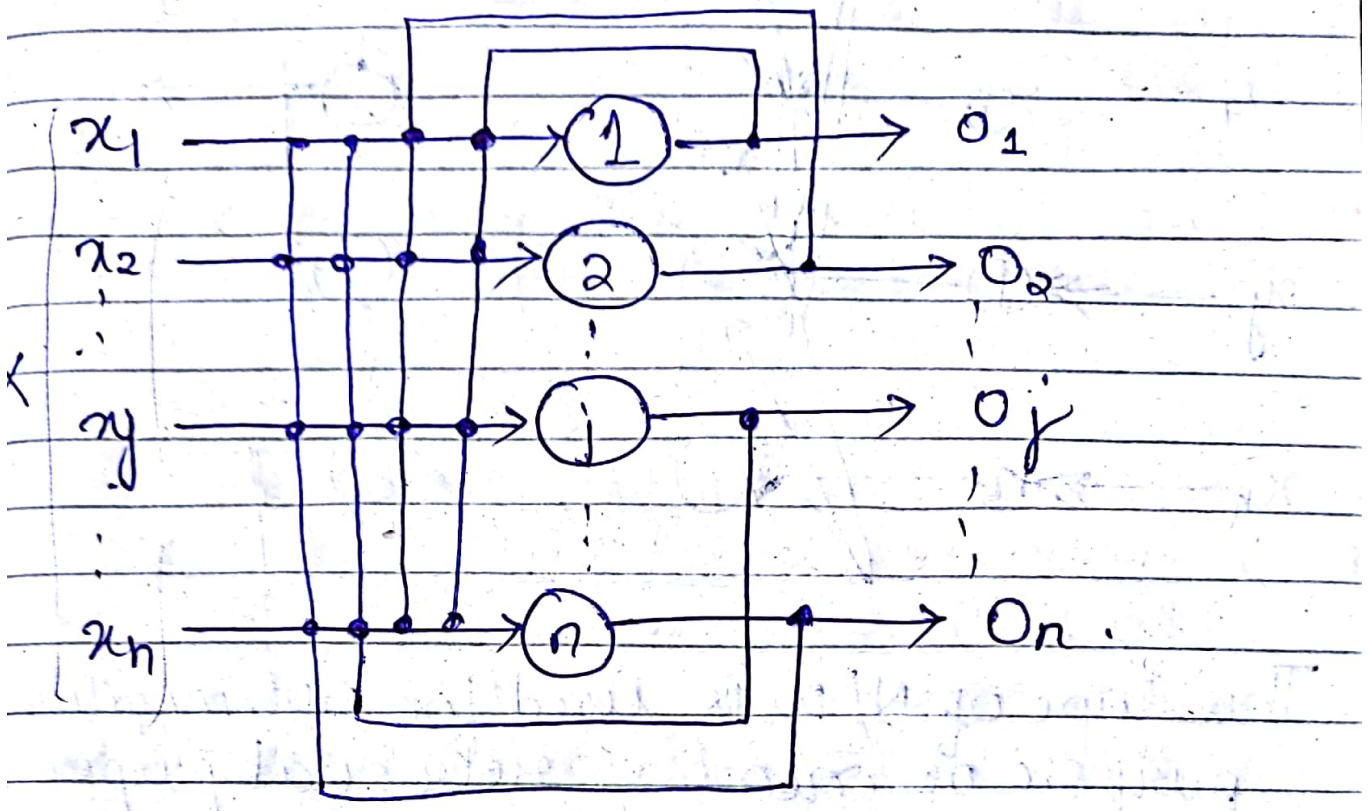
→ If we store error of D[1] and then go for X[2] presentation and store the error of D[2] and so on and finally make adjustment to connection wts and biases is known as batch-wise learning.

→ In case of bias I/P will always be 1



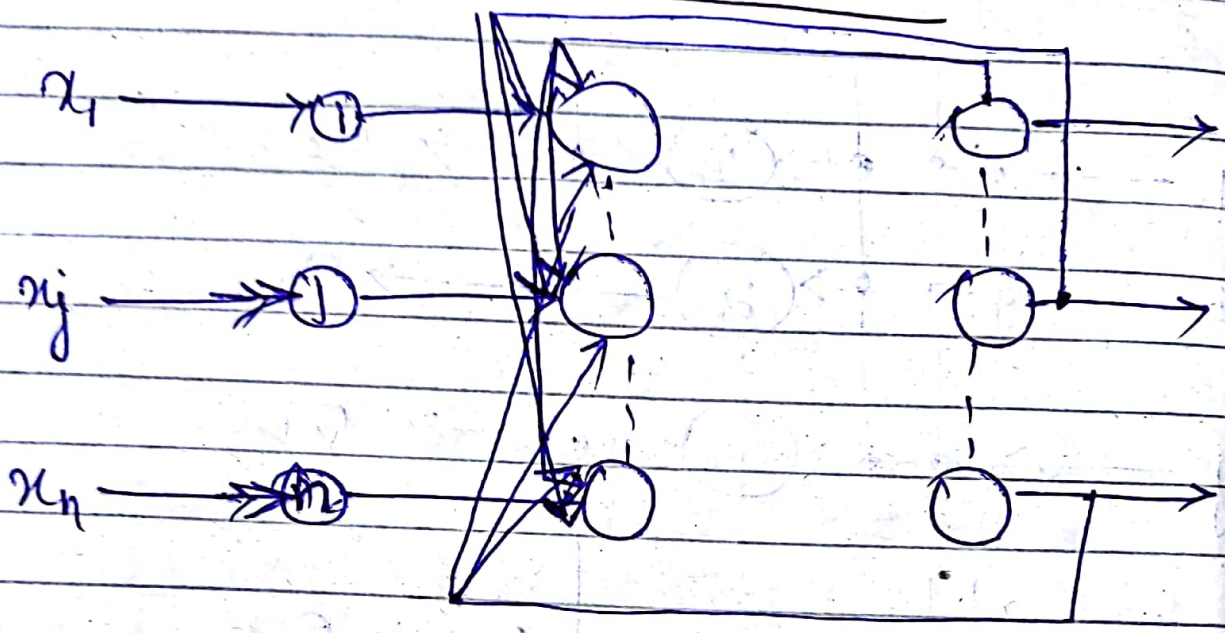
When we add bias to the net I/P, this becomes the case of bias else we will only have net input, which we get by multiplying weights & inputs.

Single layer recurrent N/W

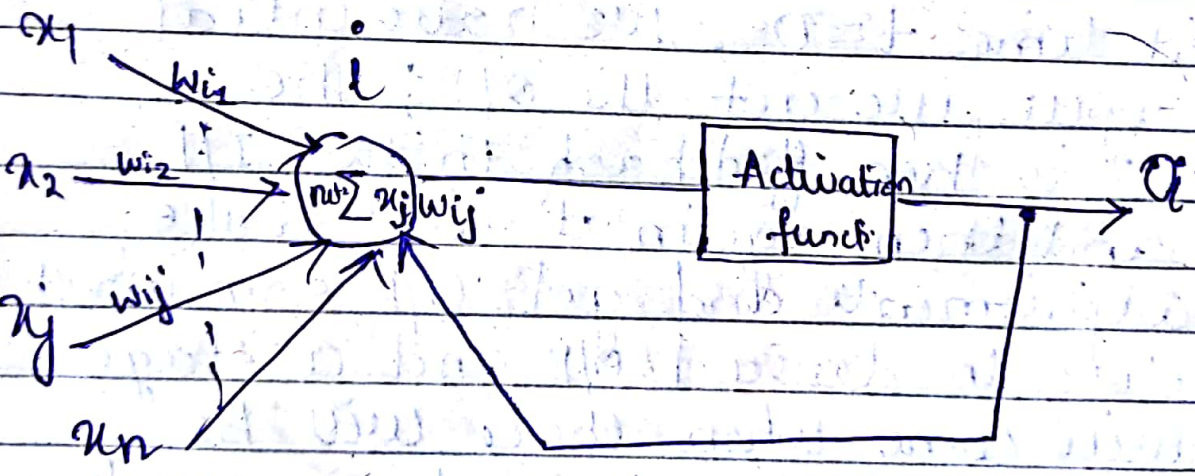


→ At time $t=0$, we have intial input, we get the O/P; the O/P is then fed back to the I/P and then again it will make adjustments and get O/P somewhat close to desired O/P and a stage will come when there will be no change in O/P and we reach close to the desired O/P.
Thus, we are going from some initial state to final state, coming close to our desired O/P.

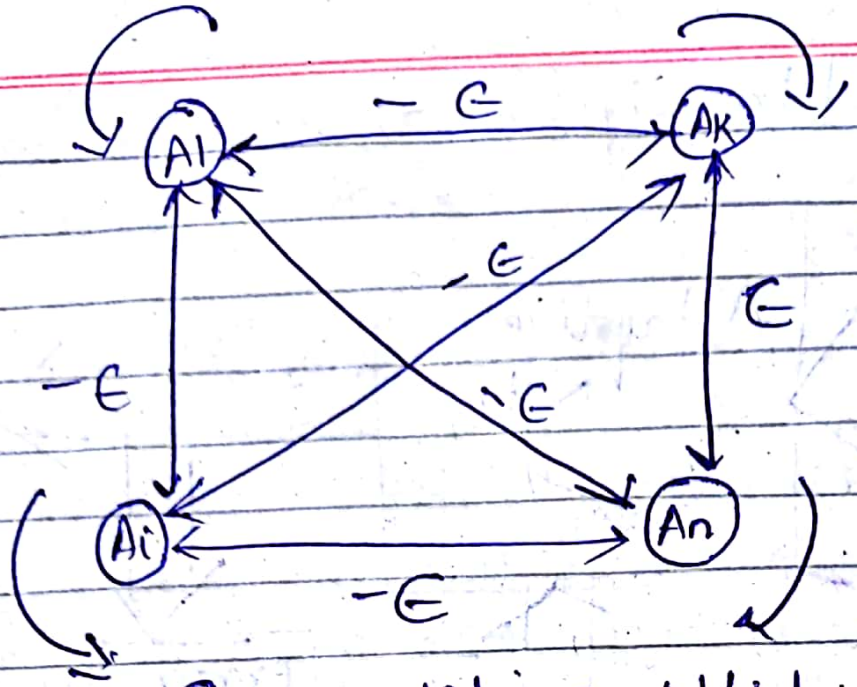
Multi-layer recurrent N/W.



This type of N/W is used for optimization purpose or memory retrieval purpose.



One neuron with self feedback.



Competitive N/W.

One will be the winner node; and it will adjust its wt acording to its neighbour node.

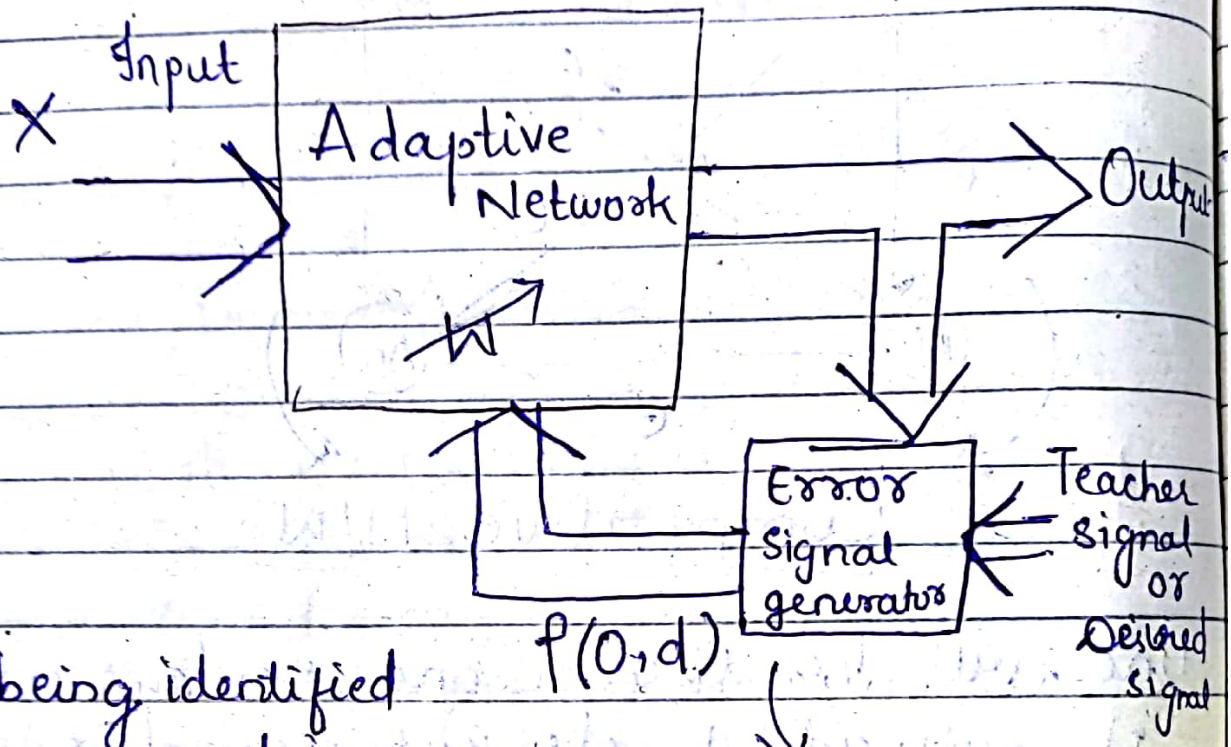
7/8/18

Neural Learning :-

(1) Supervized learning.
(2) Unsupervized learning.
(3) Reinforced learning.

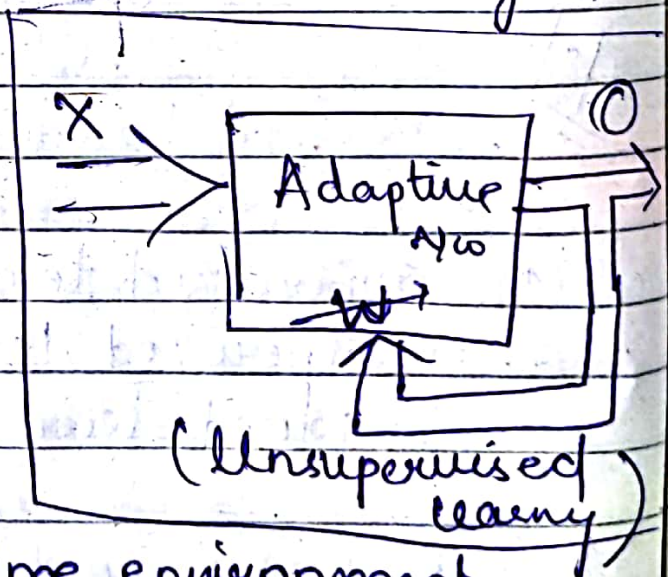
But here, we won't be studying reinforced learning (in this course).

Supervised learning:



↳ It is being identified by the connection weights & biases and W is written for net input and according to error signal, W is changed so that for input X; the O/P is close to the desired O/P.

function of O/P & desired signal



↗ When it is expose to some environment, it adapts to the environment, in form of tuning of connection weights & biases. That is why it is known as adaptive n/w.

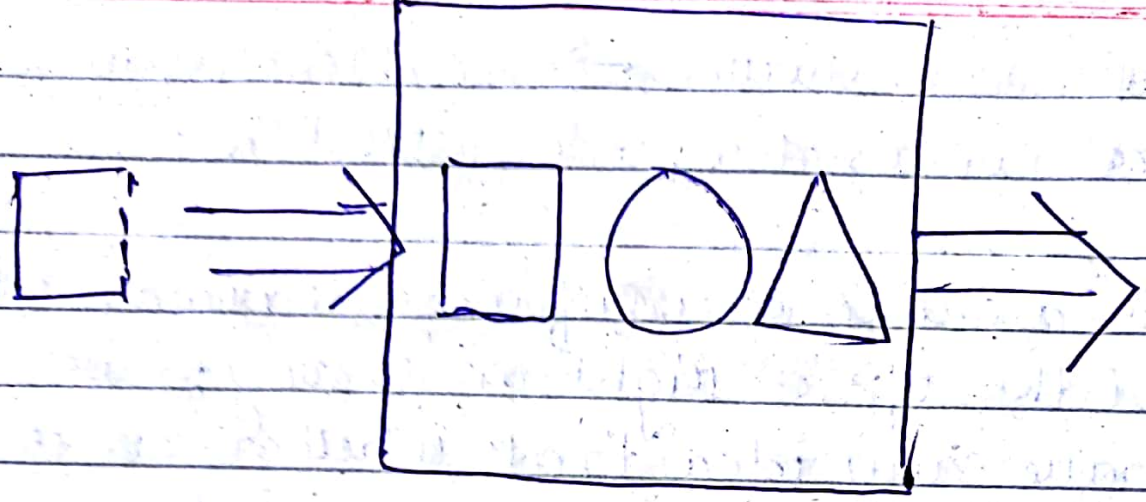
⇒ Here we have shown ⇒ double arrow, as I/P is not a scalar, it is a vector.

→ There is a critic in reinforced learning which tells whether the O/P is right or wrong; we won't have any idea that whether it is more than the desired value or less or how close it is correct or wrong, there is no idea of this concept.

→ In @ unsupervised learning, there is no teacher signal; instead the N/W adapts according to the O/P value; which is again fed to the adaptive network.

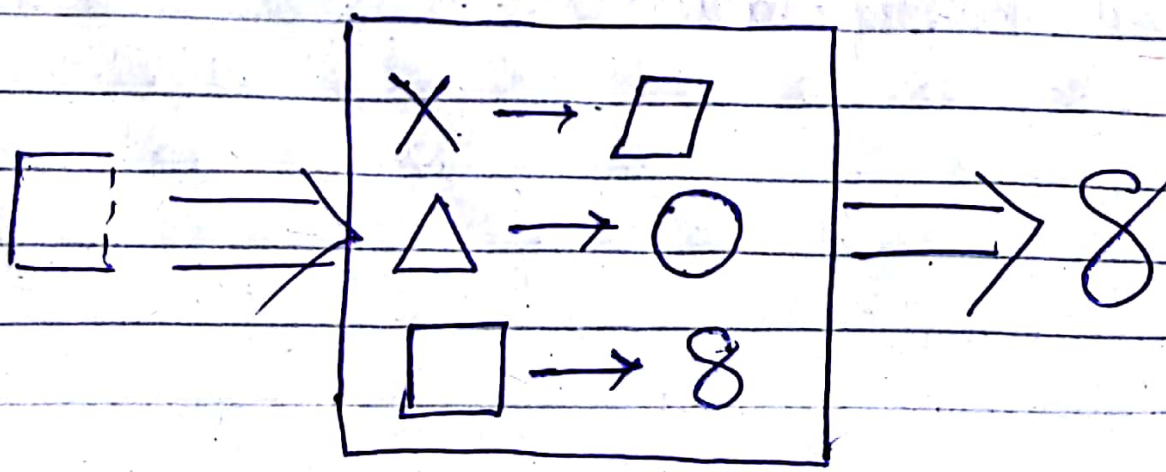
→ Auto Association.
→ Hetro-Association
→ Recall.
→ Generalization.
→ Good interpolation
→ Bad interpolation.

} Some related terms.



→ We are storing 3 patterns in ANN model. When this model is provided with some member pattern and if this N/W is able to repeat the exact pattern of that pattern (input). This is cable of implementing the auto association.

→ If it is presented with some noise of the I/P pattern, if the N/W is able to retrive the exact pattern, then it can implement auto association.



→ Here we are storing pattern pairs

→ If it is presented with the corrupted version of one of the member, if it is able to retrieve the associated part of that member then it is said to implemented hetro-association.

→ When NN is being presented with some I/P pattern and it gives some O/P pattern then the O/P pattern is called recall.

→ The pattern is not stored in shapes form but a pattern consisting of 1 and -1.
They are stored in ANN model and are also known as fundamental memory.

→ All the shapes comprises of binary number 1 and -1 and are stored in that form.
and, we get the O/P in that form only.

→ O/P pattern can be 1 or -1.
The O/P tells us whether it is in class 1 or class 2 or any other class in a binary form.
This is known as classification.

→ If it belongs to class 1 ; then O/P will be $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

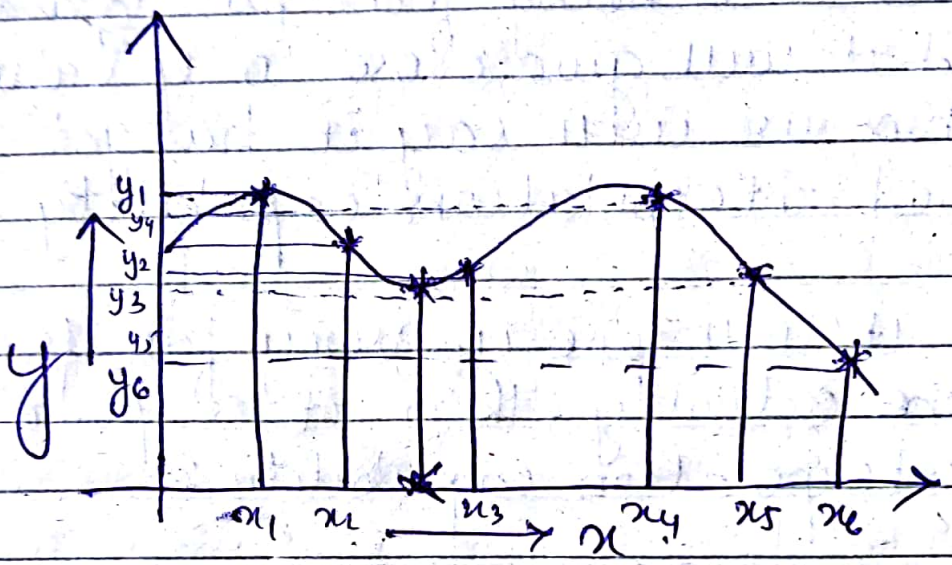
Class 2 - $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$

Class 3 - $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

→ When you are just matching the features with one of the classes ; then it is classification.
But if you are seeing that whether the feature is there (close matches) or you are recognising that whether it is there or not then this is recognition

MLP:

→ Good interpolation capability.
→ Poor extrapolation.



→ When the value of x is x_4; then value of y is y_1. Similarly,

$x_1 \rightarrow y_1$
$x_2 \rightarrow y_2$
$x_3 \rightarrow y_3$
$x_4 \rightarrow y_4$
$x_5 \rightarrow y_5$
$x_6 \rightarrow y_6$

} I/P output patterns.

We have an ANN model, which is to learn the relationship b/w x_4 & y_1.

→ With the set of examples we have trained the ANN model. That when it is being fed by x_4 then it will get O/P near y_4 and so on

→ When it is being fed b/w x_6 and x_9 and it will give close to y value, then we will say it has got good interpolation capability.

If it will give y, very far from the actual y, then it is poor interpolation capability.

→ If I have fed the ANN model, with x_7 which is beyond the operating domain, then it gives a poor y value, then we can say the poor extrapolation capability.

→ Interpolation → when we in b/w training domain; and the value is very good or close then we will say good interpolation capability.

→ Extrapolation :- When we pick value beyond the training domain then the performance is poor, then we will say poor extrapolation capability.

→ Generalization

If your trained model is being fed by any of the trainig pattern then the computed O/P is close to desired O/P or if for any unseen training pattern is being fed and the performance is good, then we will say it is good generalization.

→ If we have a good interpolation and bad extrapolation then our model have a poor generalization.

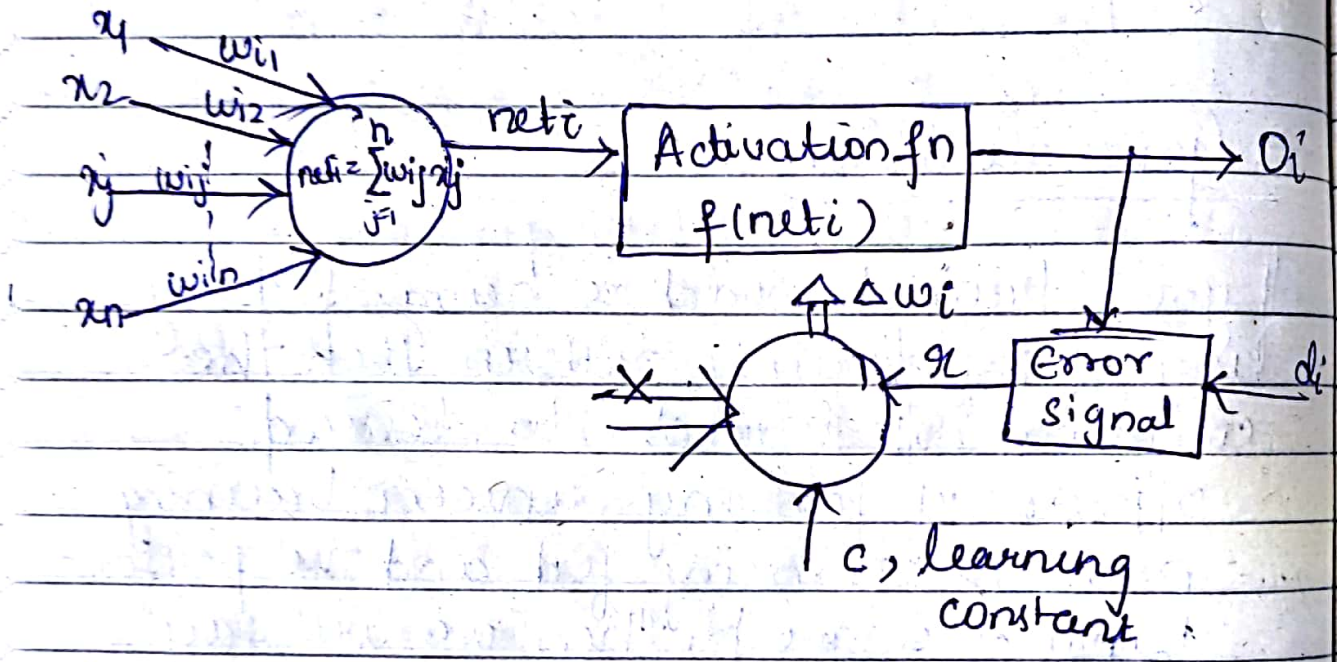
*Learning rule or learning process is a method or a mathematical logic. It improves the ANN performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a N/W when a N/W simulates in a specific data environment.

9/8/18

Applying learning rule is an iterative process. It helps NN to learn from the existing conditions & improve its performance

Neuron learning rule

→ This neuron has got many inputs (n inputs) and one of these inputs can be bias input as well.



→ It calculate neti → passes through activation → compute the output and then compare with desired signal (di).

→ When neuron is exposed to some environment and produce some O/P which is compared with di. It is supposed to adapt itself according to this by changing the connection weight and this adjustment occurs in the form of Δw.

→ What kind of activation function is there

* it depend for what purpose neuron is used.

→ Adjustment is made in form of Δw_i to get the desired output.

→ Here, learning signal $r = \mathcal{R}(w, x, d_i)$ (d_i ↑ desired signal if available; x ↓ input vector; w ↑ weight vector)

→ d_i is not available at the time of unsupervised learning.

→ Correction is represented in the form of weight vector or we can say that adaptation is represented as:-

$$\Delta w_i(t) \propto \mathcal{R}(w, x, d_i)$$

$$\Delta w_i(t) = c \cdot \mathcal{R}(w, x, d_i) \cdot x$$

Δweight = Error * INPUT * c

c ↓ constant of proportionality or learning constant.

$0 < c < 1$

→ If c becomes close to 1 then learning process may become unstable, because the weight will change more drastically. This may help us to arrive at a solution more quickly, but with such large changes in weight it's

possible we will overshoot the optimal weights. With a small learning constant, the weights will be adjusted slowly, requiring more training time but allowing the network to make very small adjustments that could improve the network's overall accuracy.

→ Now update value of weight vector.

$$w_i(t+1) = w_i(t) + \Delta w_i(t)$$

$$w_i(t+1) = w_i(t) + c \cdot r \cdot x$$

(or)

New weight = weight + error * input * learning constant

where $w_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{ij} \\ \vdots \\ w_{in} \end{bmatrix}$ { correction is to be made in individual connection wt }

→ When the neuron is subjected to another environment; the change in w_{ij} is

$$\Delta w_{ij} = c \cdot r \cdot x_j$$

→ In discrete instant of time t.

$$\Delta w_{ij}(t) = c \cdot r \cdot x_j(t) \quad \} \; j = 1 \text{ to } n$$

$$w_{ij}(t+1) = w_{ij}(t) + c \cdot x \cdot x_j \quad \forall j = 1 \text{ to } n$$

This correction is then applied to the connection weights.

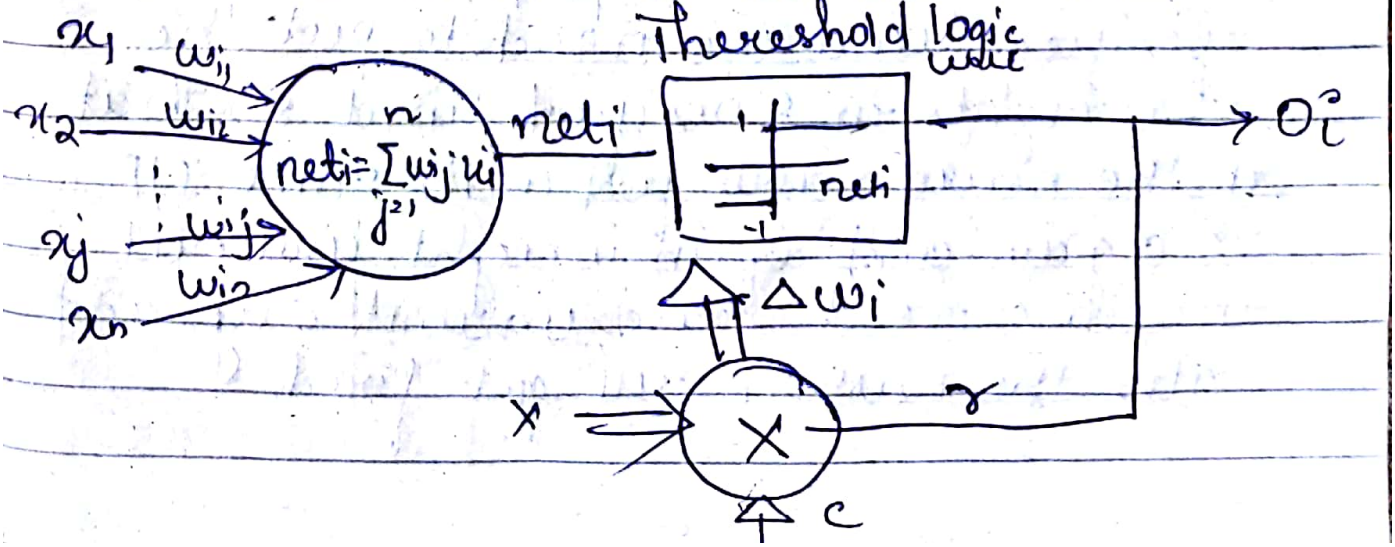
→ We are trying to implement the interpretation of a very classical statement that is :-

"If a neuron A is very close to firing its neighbouring neuron B and if it does it consistently and repeatedly then a metabolic change takes place in neuron A or B or both that increase the efficiency of neuron A in firing neuron B"

Hebbian postulate

→ To make this learning rule more easier we use :-

Hebbian learning Rule :-



→ It is for unsupervised learning and no di is there:

$$r = \text{sgn}(net_i)$$
$$\Delta w_i = c \cdot r \cdot x$$
$$\Delta w_i(t) = c \cdot r \cdot x(t)$$
$$\Delta w_i(t+1) = w_i(t) + c \cdot r \cdot x(t)$$

$$\Delta w_{ij} = c \cdot O_i \cdot x_j$$
$$w_{ij}(t+1) = w_{ij}(t) + c \cdot O_i \cdot x_j \quad \Big]_{j=1 \text{ to}}$$

→ If O_i and X_j are similar in polarity then Δw_{ij} will be +ve otherwise it will be -ve and connection weight will decrease.

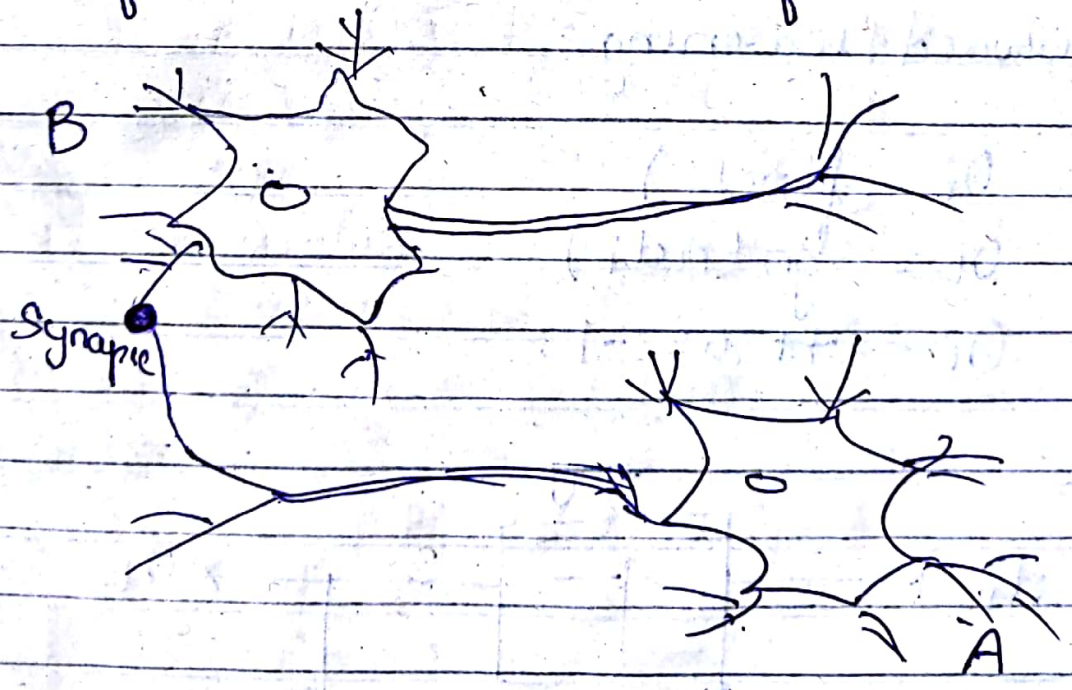
∴ If w_{ij} will increase, then the prob. of neuron getting fired will also increase.

→ Here in order to keep O/P in between specified limit we need to put the weights in specified limit so that if the neuron will fed with same I/P again and again weight will not increase or decrease unlimitedly and also the neuron will get fired.

* Initially the weights are initialized or they are taken closer to zero. They can be +ve and -ve. It is kept close to zero because to give enough room to the neuron for learning. (to vary from the minimum and maximum value).

→ The absolute values of the weights are usually proportional to the learning time, which is undesired.

Aug → Hebb's postulate in other words:- when a weight contributes to firing a neuron, the weight is increased. (If the neuron doesn't fire then it is not)



Generalised Hebb Rule

→ When a weight contributes to firing a neuron, the weight is increased

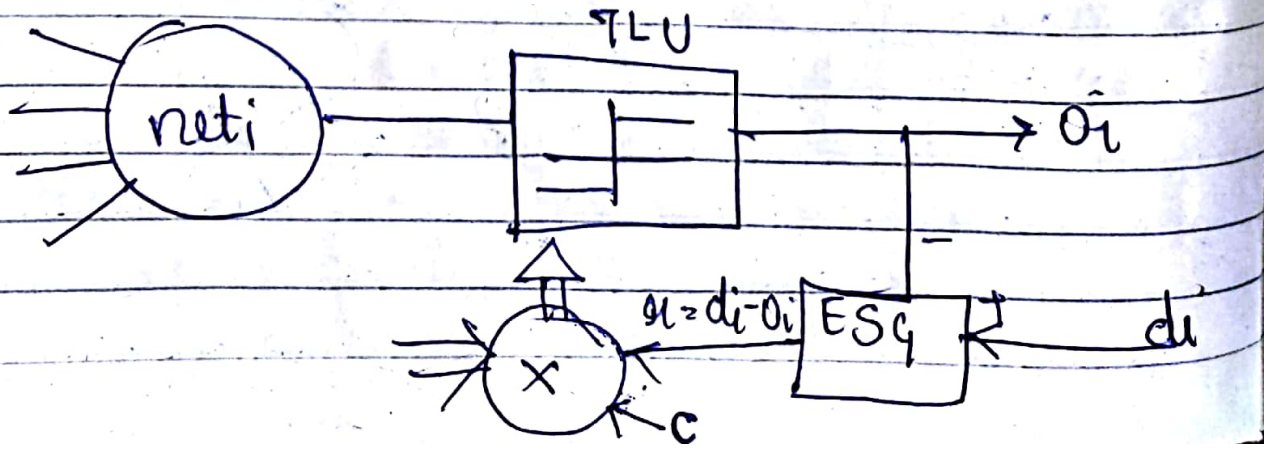
→ When a weight acts to inhibit the firing of a neuron, the weight is decreased

* The Hebbian learning is unsupervised because weights are strengthened by the actual response to a stimulus.

② Perceptron learning

→ Supervised learning.

$$O_i = f(net_i)$$
$$O_i = sgn(net_i)$$
$$O_i = +1 \text{ or } -1$$



$$\Delta w_i = c \cdot (d_i - O_i) x_i$$

$$w_i(t+1) = w_i(t) + c(d_i - O_i) \cdot x_i$$

$$\begin{cases} \Delta w_{ij} = c(d_i - O_i) \cdot x_j \\ w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \end{cases}$$

→ $j = 1$ to n.

(In this case there will be learning) $\Delta w_i = \pm a \cdot c \cdot x$. (When they have disagreement) ~~(In this case there will be no learning)~~

→ Learning will occur only when there is disagreement b/w d_i and O_i.

→ Δw_i and Δw_{ij} will be 0 in case d_i and O_i have agreement.

When d_i and O_i will have agreement, $d_i = O_i$

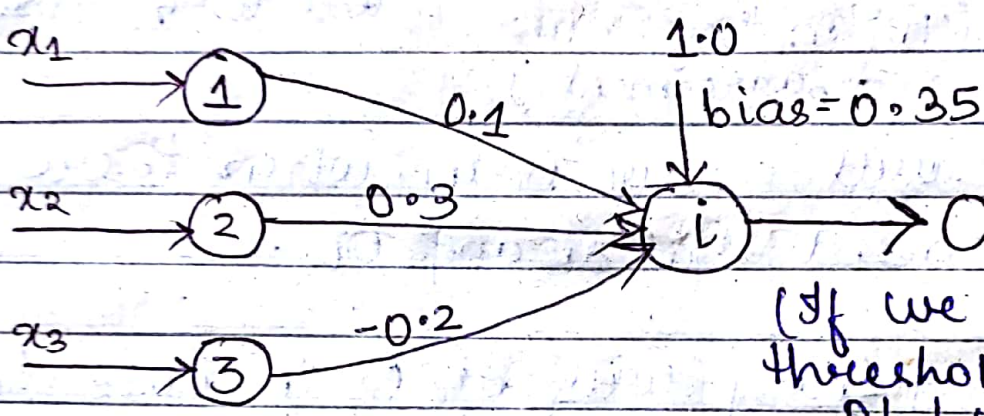
$$\boxed{d_i - O_i = 0}$$

$$\Delta w_{ij} = c \cdot 0 \cdot x_j$$

$$= 0$$

10/8/18

① Calculate the output or Obtain the output of a neuron i for the network shown in Fig. 1 using following activation functions
① Unipolar sigmoidal fn
② Bipolar " "
Take gain unity at $net_i = 0$.



(If we need threshold then instead of 1 we will write t)

At any point of time, $x_1 = 0.8$, $x_2 = 0.6$ and $x_3 = 0.4$

or $X = \begin{bmatrix} 0.8 \\ 0.6 \\ 0.4 \end{bmatrix}$

(In this ques:- If activation threshold is given then we don't need bias & we give I/P for bias as -1)

Sol $net_i = 0.8 \times 0.1 + 0.6 \times 0.3 - 0.2 \times 0.4 + 1 \times 0.35$

$= 0.08 + 0.18 + 0.35 - 0.08$

$= 0.53$

Role of bias → It provides flexibility to bring computed O/P close to desired O/P (so that it gives more flexibility to change wts).

Q2 Implement AND logic gate using Mc Cullock Pitts method.

Consider the following truth table

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	

Q3 Implement XOR gate using Mc Cullock Pitts neuron model.

Q4 Obtain linear separability line for OR gate using bipolar binary data.

* If bias is +ve then it will help in facilitating firing of neuron; if bias is -ve then it will help in inhibiting the firing of

(1) $net_i = 0.8 \times 0.1 + 0.6 \times 0.3 - 0.3 \times 0.$
$+1 \times 0.35$

$= 0.18 + 0.35$
$= 0.53$

* Note Sometimes in question, bias is not given directly rather it is given as 4th ~~value~~ input with value = 1 given $\lambda = 1$

$$O_i = \frac{1}{1+e^{-1 \times 0.53}}$$

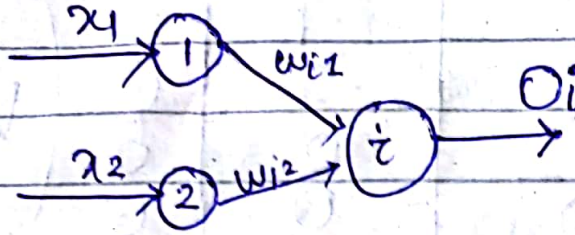
$= 0.629$ (BSF).

$$O_i = \frac{2}{1+e^{-0.53}} - 1$$

$= 2 \times 0.629 - 1$
$= 0.258$ (BSF)

lesson

(2)



Assume threshold = 0.

	w_{i1}	w_{i2}
(1)	1	1
(2)	-1	1
(3)	1	-1
(4)	-1	-1

(1) 1 and 1 . need y=1

① neti = (2) y = 1.
② neti = 0 y = 1
③ neti = 0 y = 1
④ neti = -2 y = 0.

(2) 1 and 0 need y=0

① neti = (1); y = 1
② neti = -1; y = 0
③ neti = 1; y = 1
④ neti = -1; y = 0

(3) 0 and 1 need y=0.

① neti = (1); y = 1
② neti = 1; y = 1
③ neti = -1; y = 0
④ neti = -1; y = 0

(4) 0 and 0 .

① (0); y = 0
② 0; y = 0
③ 0; y = 0
④ 0; y = 0

Set threshold ≥ 2

		neti	Oi	y
1	1	2	1	1
1	0	1	0	0
0	1	1	0	0
0	0	0	0	0

(✓)

③ Implementation of exclusive-OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive OR is a non-linear problem. so it cannot be implement with single neuron as

$w_1 = 1 ; w_2 = 1$

0	0	0
0	1	1
1	0	1
1	1	1

(X) No such threshold exist which

$w_1 = 1 ; w_2 = -1$

0	0	0
0	1	-1
1	0	1
1	1	0

(X)

$w_1 = -1 ; w_2 = 1$

0	0	0
0	1	1
1	0	-1
1	1	0

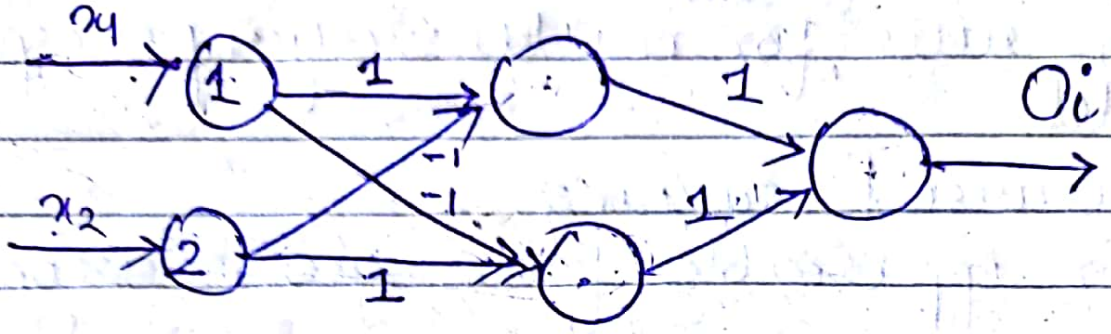
(X)

$w_1 = -1 ; w_2 = -1$

0	0	0
0	1	-1
1	0	-1
1	1	-2

(X)

Structure for XOR



$$w_{i1}x_1 + w_{i2}x_2 + w_{i3}x_3$$

$w_1 = -1$; $w_2 = -1$; $w_3 = 1$

		net$_i$
0	0	0
0	1	-1
1	0	-1
1	1	-2

$w_1 = 1$; $w_2 = -1$
$w_3 = 1$

0	0	0
0	1	-1
1	0	1
1	1	0

$w_1 = 1$; $w_2 = 1$; $w_3 = 1$

0	0	0
0	1	1
1	0	1
1	1	2

$w_1 = -1$; $w_2 = 1$
$w_3 = 1$

0	0	0
0	1	1
1	0	-1
1	1	0

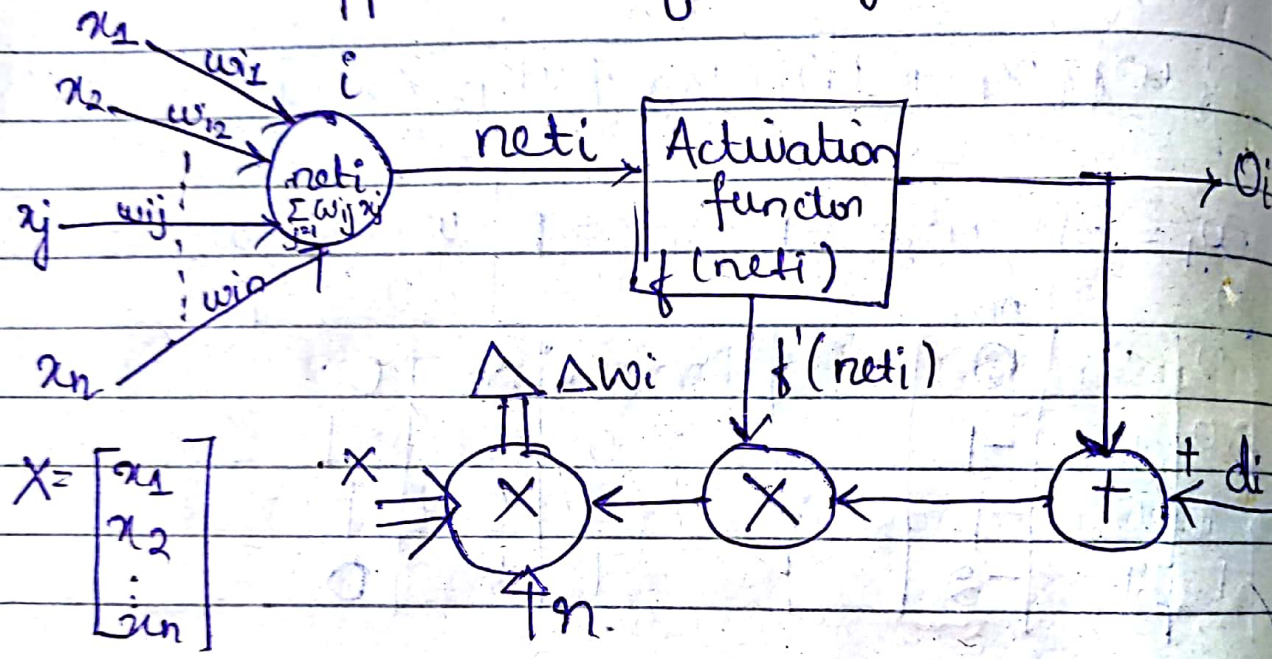
13/8/18

Imp Delta learning rule

→ Same rule is used for multi-layer perceptron model.

→ Supervised learning

→ Also applicable for single neuron.



→ Connection weights are entries within the weight vector.

→ W_i = is the weight vector associated with neuron i.

→ The connection weights should be made in such a way that for the same input vector, computed O/P come closer to the desired Output.

→ The tuning of the connection vector comes in the form of adaptation of the neuron or learning of the neuron.

→ The learning will be there only when there is disagreement b/w Oi and di.

→ If ~~error~~ on decreasing the w_{ij} the error is increasing then I have to increase w_{ij} and vice-versa

So we need to define our error function.

$$E = \frac{1}{2}(d_i - O_i)^2$$

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{2} * 2(d_i - O_i)\left(-\frac{\partial O_i}{\partial w_{ij}}\right)$$

but $O_i = f(net_i)$

$$= \sum_{j=1}^{n} w_{ij} x_j$$

$$\frac{\partial E}{\partial w_{ij}} = -(d_i - O_i) \cdot f'(net_i) \cdot x_j$$

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}} = \eta (d_i - o_i) \cdot f'(net_i) \cdot x_j$$

$\eta \rightarrow$ learning rate.

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

$$\boxed{w_{ij}(t+1) = w_{ij}(t) + \eta (d_i - o_i) \cdot f'(net_i) \cdot x_j \quad \text{for } j \text{ 1 to } n}$$

→ Delta LR is applicable to continuous O/P not binary O/P.
so A/C fn used can be unipolar sigmoidal or bipolar sigmoidal.

$$\boxed{f'(net_i) = \lambda o_i (1 - o_i)} \quad \text{in case of unipolar}$$

$$f'(net_i) = \begin{cases} \lambda o_i (1 - o_i) & \\ \frac{1}{2} \lambda (1 - o_i^2) & \text{in case of bipolar} \end{cases}$$

$$f(net_i) = \frac{1}{1 + e^{-\lambda net_i}}$$

$$f(net_i) = \left[1 + e^{-\lambda net_i}\right]^{-1}$$

→ Weights can be intialized anywhere but generally it is intialized b/w

→ In the set of IN OUT patteren, we find some are +ve and some are -ve then we will go for bipolar; and when O/P is +ve only then we will go for unipolar sigmoidal fn.

$$-0.5 \leq w_{ij} \leq 0.5$$

(OR)

$$-1 \leq w_{ij} \leq 1$$

by the use of random number generator.

→ Set of input-output patters.

$X[1] \rightarrow Y[1]$

$X[2] \rightarrow Y[2]$

⋮

$X[p] \rightarrow Y[p]$

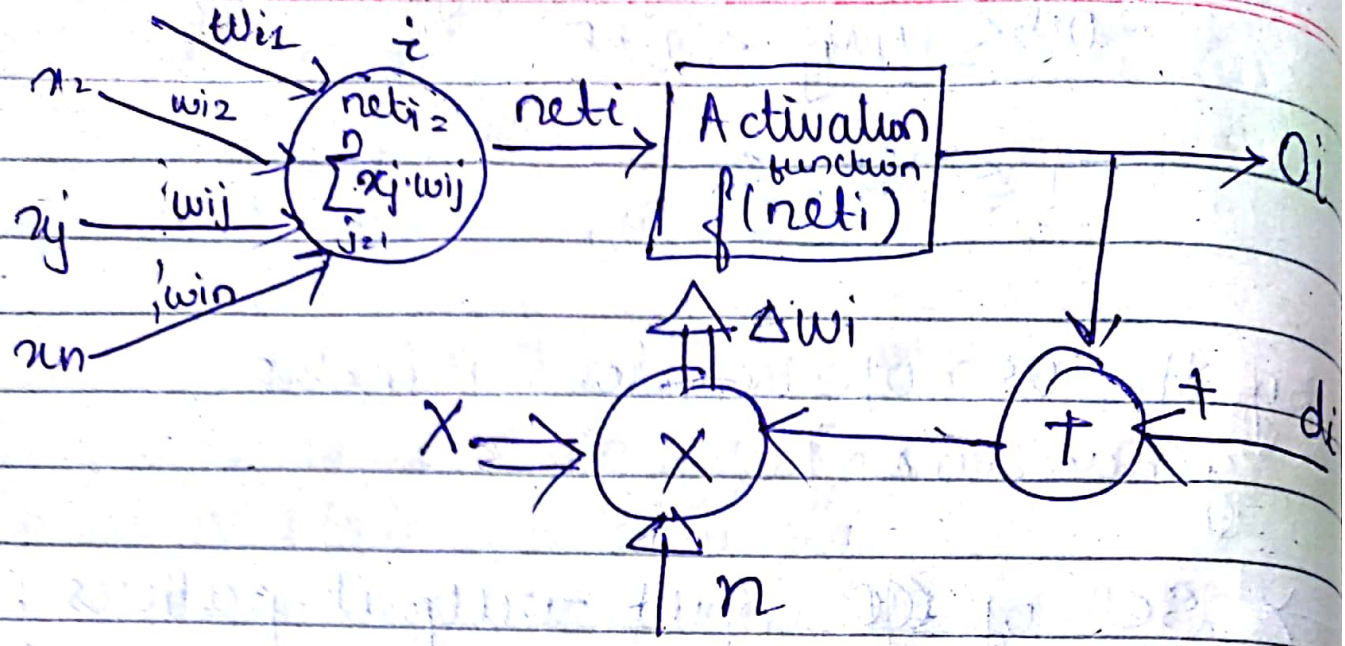
⋮

$X[P] \rightarrow Y[P]$

→ RMS error b/w O_i and d_{pi} is somewhat very less or acceptable range then our learning stops.

Widrow-Hoff learning rule:-

→ Also known as least mean square rule



→ It is for supervised learning.

$$\Delta w_{ij} = \eta (d_i - O_i) x_j$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$$

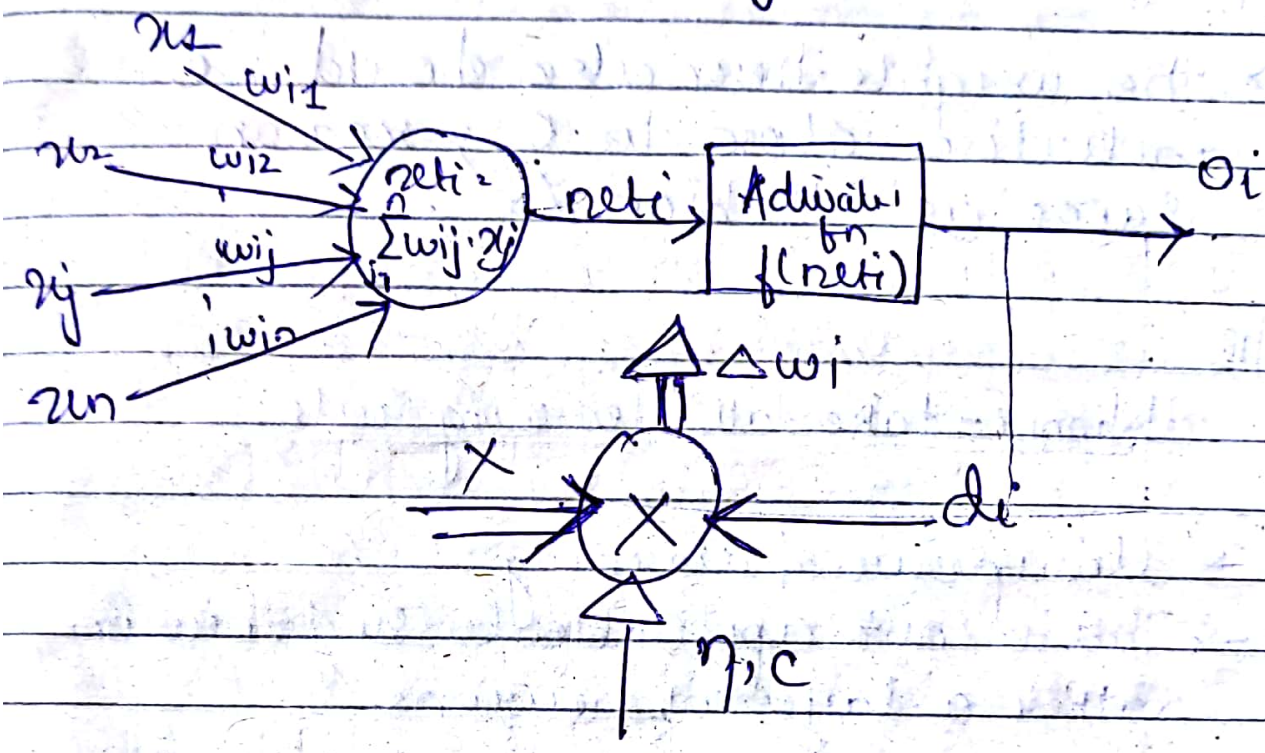
$$w_{ij}(t+1) = w_{ij}(t) + \eta (d_i - O_i) x_j$$

for $j = 1$ to n.

→ When this $f(net_i) = net_i$ (identity activation function) appling ~~or~~; then ~~LMS learning rule becomes~~ Delta learning rule becomes LMS learning rule.

→ Correlation learning rule.

→ Supervised learning.



$$\Delta w_{ij} = \eta \cdot d_i \cdot x_j$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

$$\boxed{w_{ij}(t+1) = w_{ij}(t) + \eta\, d_i\, x_j}$$

η lies b/w 0 and 1

→ If x_j and d_i are in agreement in terms of polarity then w_{ij} should be increased, if in disagreement then it should be decreased

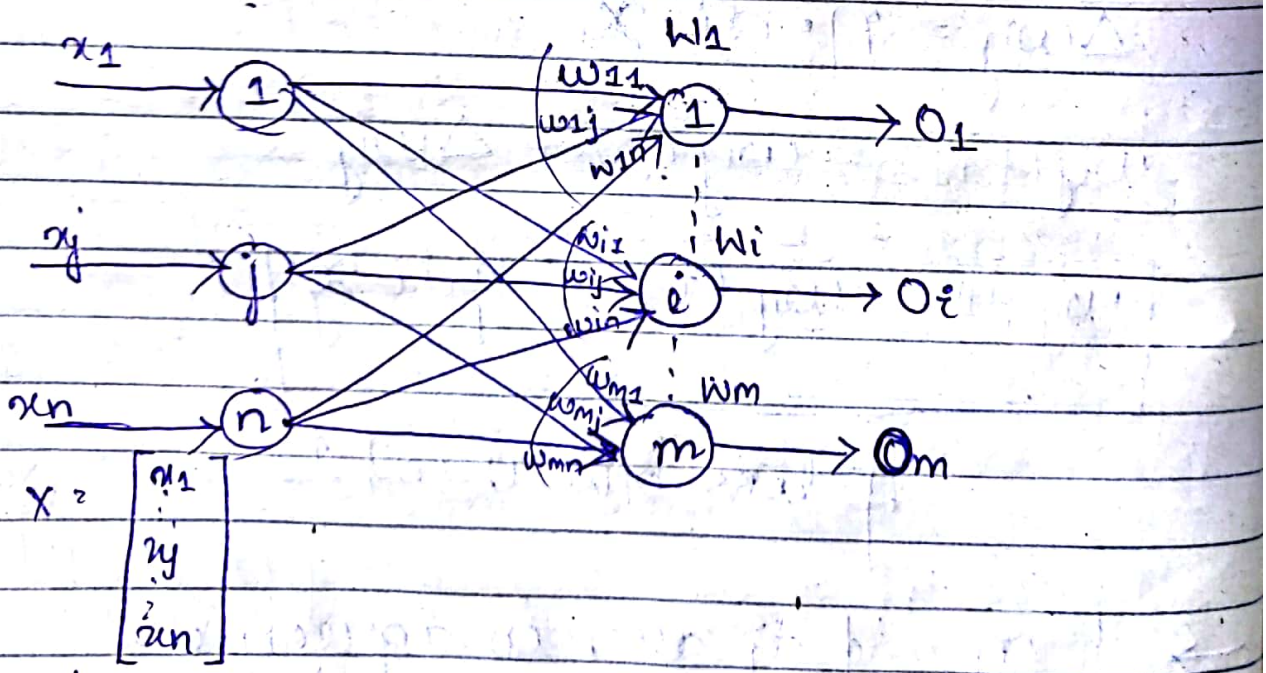
→ This is almost same as Hebian learning rule but it is unsupervised and this is supervised.

→ The weights here also should be initialise close to 0; reason same as Hebian LR.

14/8/18

Winner-take-all learning rule.

→ Unsupervised learning
→ This is not applicable to single neuron rather a layer of neurons.



→ Input neuron → they are not modifying the input signal, they just act as a carrier

Or distributor.

- They don't do any kind of computations.

→ The node that have maximum value of O/P, declared as winner node for a presentation input.

→ When we take some nodes, near the winner node then it is known as follow the leader approach.

→ Only the winner node makes changes in its weight and the neighbouring nodes also make changes in its weight in follow the leader approach.

→ Whenever we make presentations, this input vector will get one O/P class, whose value is maximum (whose wt vector is similar to I/P vector) and is declared as winner.

→ • The no:- of classes = no:- of O/P nodes.

→ • The cardinality of the I/P vector = the cardinality of the weight vector.

→ $W_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{ij} \\ \vdots \\ w_{in} \end{bmatrix}$

One metric

→ Measuring the distance b/w the weight vectors (W_1, W_i ... W_n) with the input vector. The one with the minimum Euclidian distance will declare as the winner.

2nd metric

Taking dot product of the two vectors (of unit length).
The one giving the highest value of the dot vector declared as winner giving that the vectors are of unit length.

1st metric

Euclidean distance b/w X and W_i

$$Ded = \sqrt{(x_1 - w_{i1})^2 + (x_2 - w_{i2})^2 + \cdots + (x_i - w_{ii})^2 + \cdots (x_n - w_{i2})^2}$$

(2) Inner product criterion (provided participating vectors are of unit length).

$$a_i = [W_i]^T [X] = w_{i1}x_1 + w_{i2}x_2 + \cdots w_{ij}x_j + \cdots w_{in}x_n$$

(activation)

Converting the vector into unit length vector

$$x_j = \frac{x_j}{\sqrt{x_1^2 + x_2^2 + \cdots x_j^2 + \cdots x_n^2}} \quad \text{for } j = 1 \text{ to } n$$

→ The one that is the winner node will make modifications to its weight vector.

$$\Delta w_{ij} = \eta_i (x_j - w_{ij})$$

↳ learning rate.

$$w_{ij}(t+1) = w_{ij}(t) + \eta_i (x_j - w_{ij})$$

$$= w_{ij}(t) + \Delta w_{ij}(t)$$

for $j = 1$ to n.

→ But in case of follow the leader approach we will go for the neighbouring nodes of the winner nodes and make changes in the weight vector of the neighbouring nodes.

→ In this FTLA; the neighbouring nodes will try to make changes to the wt. vectors to look similar to their leader node (winner node).

→ This FTLA will be used in Kohonen self-organizing feature.

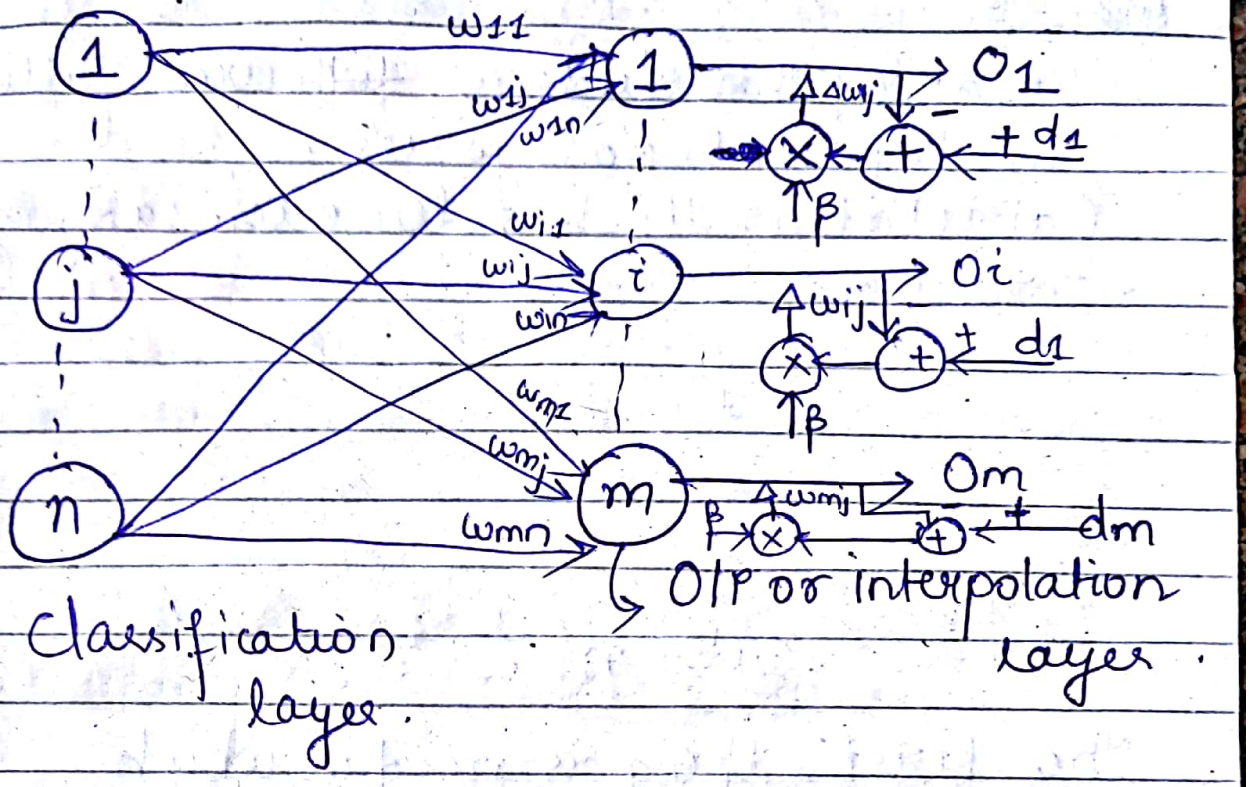
In-star learning rule (Same as Winner-take-all).

→ All the wt. connections that are terminating at a winner node; will change or modify its synaptic strength or wts.

Out-star learning rule.

→ Supervised learning.

→ Applicable to group of nodes.



→ For a given presentation, and based on that I/P there will be a winner node in classification layer and its O/P will be 1 and for the rest it will be 0.

(whose wt. vector is similar to I/P node)

→ The connection weights arising/outstaring from the winner nodes they are supposed to be updated or refined.

→ $$\Delta w_{ij} = \beta (d_i - O_i)$$
$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$
$$= w_{ij}(t) + \beta (d_i - O_i)$$

for $i = 1$ to m

Eg:-

Some of the students following falling in different categories. Calculation the fees for each category is different.

Cat – 1

Cat 2

:

Cat – m.

The first thing is need in which category he is falling that is tell by classification layer and further fees calculation is done by interpolation layer of Out-star learning.

→ This is a two step problem.

① Finding in which class the student lies based on the attributes of the student taken as I/P vector. so first step is classification.

② After we found the category of student, we do fees calculation which is done by interpolation

16/8/18

Flute ① We can classify it in
Hockey Stick ② 3 classes.
Pen ③
Sarod ①
Football ②
Pencil ③
Volleyball ②
Rubber ③

→ Just by looking at the attributes and comparing them with different items we can classify them in different classes.

→ Sets of attributes associated with certain items can be shown as:-

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix}$$

Kohnen Self-organising feature map (KSOFM)

→ Used for classification, pattern recog-nition, pattern matching
→ It's an ANN Model.
→ Imp is to know the architecture, processing of the features, Application area, etc

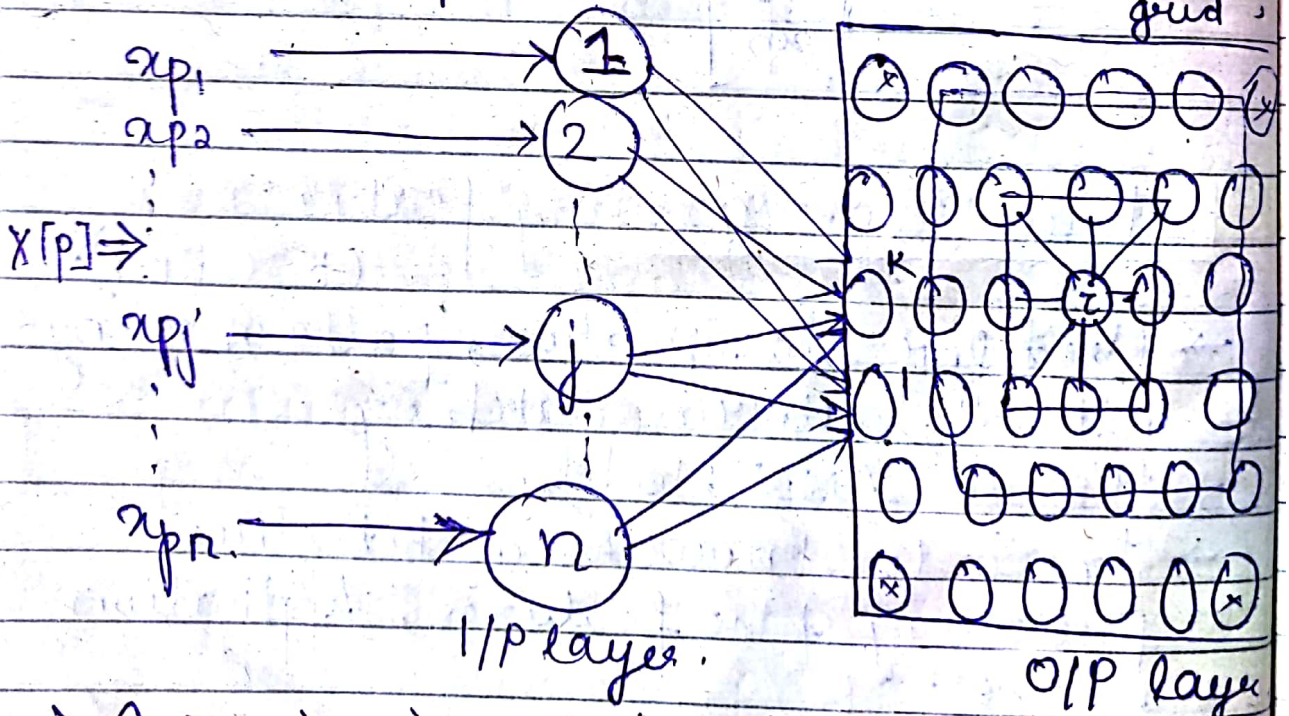
→ It employs unsupervised learning
→ Feed-forward architecture.

→ Criteria for learning :- follow the leader approach.

$X[p]$ → one of the feature pattern.
$x_{p1}, x_{p2} \cdots x_{pn}$ → attributes of the p feature pattern.

$$X[p] = \begin{bmatrix} x_{p1} \\ x_{p2} \\ \vdots \\ x_{pj} \\ \vdots \\ x_{pn} \end{bmatrix}$$

$X[1], X[2], \cdots X[p]$:- $X[p]$ are set of I/P patterns contain n attribute.



→ Criteria is ~~used~~ for classification is :- Two I/P pattern which are similar to each other fall in one class and two dissimilar will fall in another class

→ Our work is to classify these items in certain no:- of classes.

→ O/P is arranged in 2-D space; every node is connected to each of the I/P node.

→ Two factors which affects the O/P or which identify the O/P.

→ What is a wt vector associated with a particular node (i) and the neighbouring node to that node or the neighbourhood

→ They are the neighbourers of the 1st order. (i is the 0th order) and similarly the outer to the 1st order neighbourers are the neighbourers of the 2nd order.

→ W_i → weight vector associated with i. Whose dimension is same as that of no:- of I/P neurons $= n$.

$$W_i = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{ij} \\ w_{in} \end{bmatrix}$$

→ Our O/P layer is similar to as K-map.

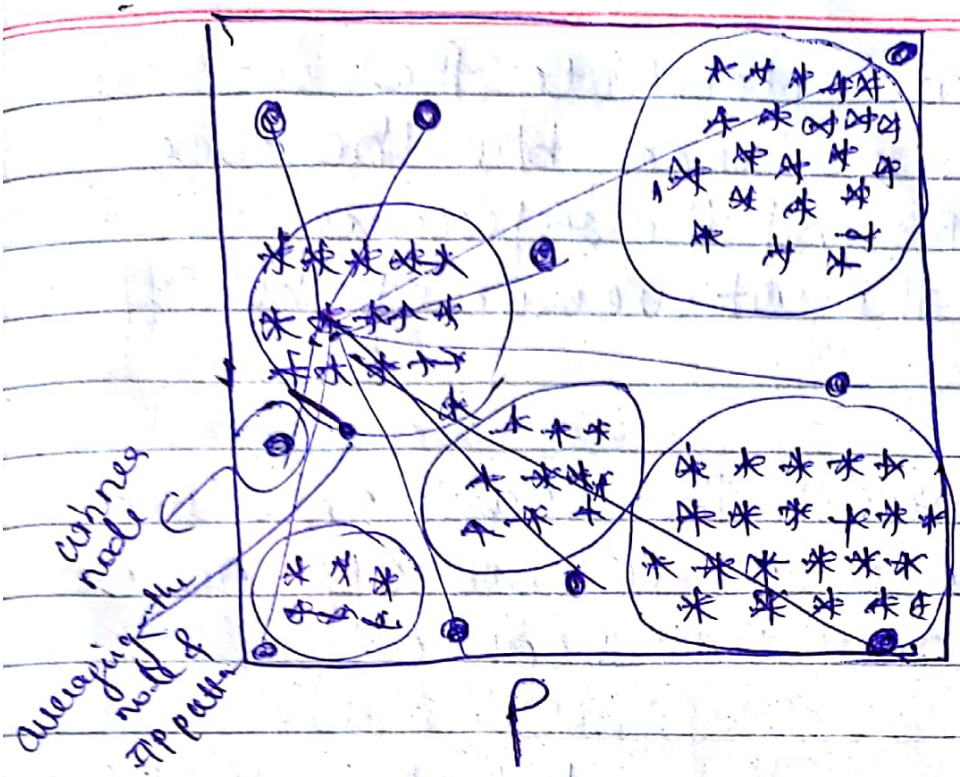
→ I/P pattern having same statistical value fall in one class and vice-versa.

→ Every O/P node calculate its activation, the node with the maximum activation will be the winner node. So there will be change in wts of the winner node along with the neighbouring nodes following the follow the leader approach.

→ We are given set of input patterns with n attributes.

→ We make presentation to the model, the pth pattern with n attributes.

→ Then the calculation of O/P as mentioned above takes place.



→ There are P stars or P points and are placed in n-dimensional space and are then transformed in 2-Dimensional ~~space~~ plane.

→ Criteria for transferring/transformation
→ which are they are neighbourhood or there should be no chage in environment in 2-dimensional ~~space~~ plane.

→ We are transforming an input pattern with n dimension over a 2-dimensional space in an ANN model.

→ We consider the reference point which is equal to the no:-of nodes.

→ We then calculate the eludian distance b/w the one of the I/P and the reference point which the wt vector of the O/P node.

→ the one with the least eludian distance is the winner O/P node or the winner wt. vector.

→ Now we will update the weight vectors as the average of the I/P presentation and the winner node wt. vectors.

→ It is the weight vectors or the reference points are moving over the 2-dimensional grid. Their location will change! The wt vector which is the winner node will move across the 2-D grid.

→ After doing this we will give our next input pattern and again calculate all the steps as above and calculate the winner node & so on.

→ At the end of the day, there will be the least movement or no movement in the wt vector and our training process will be over.

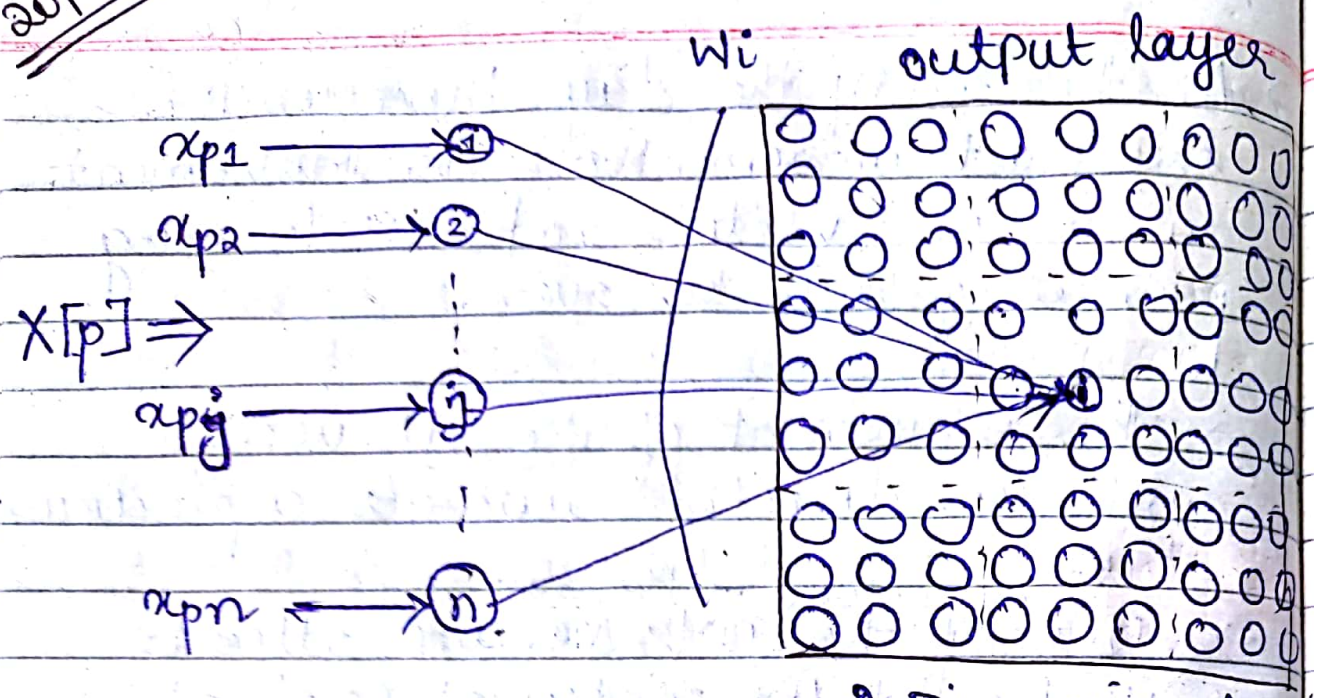
→ The movement of the wt vector will be directed towards a particular class. At the end of the day,

→ One of the wt vector atleast will be at the centre of each of the class and that particular node will make one class and have least euclidean distance with patterns falling in but class, still some of nodes will be left out as in previous we don not know how many classes will be there.

→ Some nodes will be lying in the Original position and is known as the virgin node.

→ Generally we take no. of nodes greater than the number of classes.

20/8/18.



2-Dimensional grid.

$$Wi = \begin{bmatrix} wi_1 \\ wi_2 \\ \vdots \\ wij \\ \vdots \\ win \end{bmatrix}$$ for i = 1 to M.

Input data.

$$X[1] = \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1j} \\ \vdots \\ x_{1n} \end{bmatrix} ; \quad X[2] = \begin{bmatrix} x_{21} \\ x_{22} \\ \vdots \\ x_{2j} \\ \vdots \\ x_{2n} \end{bmatrix} ; \quad X[p] = \begin{bmatrix} x_{p1} \\ x_{p2} \\ \vdots \\ x_{pj} \\ \vdots \\ x_{pn} \end{bmatrix} ..$$

$$X[p] = \begin{bmatrix} x_{p1} \\ x_{p2} \\ \vdots \\ x_{pj} \\ \vdots \\ x_{pn} \end{bmatrix}$$

→ Input :- We have given a set of I/P pattern i.e. X[1], X[2], ... X[p], ... X[P]

→ A set of attributes forms a particular input for a certain period of time.

→ ~~Purpose~~
How many nodes are to be there in I/P layer ? (architecture concern).

→ Once the presentation has been made, then we will calculate value for all O/P, the one with the highest value will be the winner node.

→ 1 iteration constitute of all presentation of the Input pattern. It is also known as epoch.

→ Two similar pattern should fall in one class & two dissimilar pattern will be in different class :- condition for classification.

→ We can make some guess about the class and we will take no:- of O/P nodes more than that of no:- of classes.

→ O/P nodes is always less than the classes because if they are same then what is the use of classification.

→ We have to make more computation for a presentation when we have more no:- of O/P nodes, but generally to be in safe side, we will take that more than no:- of classes.

→ Every node in O/P layer is identified by its no:- of neighbouring nodes and its weight vector.

→ For initialising the connection weight so they belong to same I/P space as you have picked the input pattern.

→ P → points in n dimensional space.
So we have to pick our wts from these n dimensional space.

→ Scann along the jth dimension, and see the minimum & maximum co-ordinate and so for every dimension (1st, 2nd ... n) we will calculate minimum & maximum.

x_{jmax} x_{jmin}] $j = 1$ to n.

→ Now by joining these above x_{jmax}, x_{jmin} we can find our I/P space.

→ We will assign values to our wt. vector from the calculated I/P space so that it covers all the I/P values (or we can say cover all the I/P space uniformly).

Simply, we have to cover our I/P space uniformly for calculation of wts.

→ For covering the I/P space uniformly

$$w_{ij} = x_{jmin} + \gamma (x_{jmax} - x_{jmin}) \quad (0)$$

where $\gamma \rightarrow$ random number between 0 & 1.

where $j = 1$ to n and $i = 1$ to M.

On $\gamma = 0$; $\gamma = 1$

$w_{ij} = x_{jmin}$ $\quad$ $w_{ij} = x_{jmax}$.

→ Preprocessing of the data is to be done after weight calculation.

Preprocessing of the data :

$$x_{pj} = \frac{x_{pj} - x_{jmin}}{x_{jmax} - x_{jmin}} \quad j = 1 \text{ to } n \rightarrow p = 1 \text{ to } P.$$

→ Every input value should lie b/w 0 and 1. (It is not necessary to fix range b/w 0 and 1 but it can have range)
We do this to give equal wtage to all the attributes (because if some I/P will have value in a wider range then it will affect the O/P more and we want that all are attributes are imp to us).

(There are other methods to calculate ~~distance~~ to the winner node; finding the eclidian distance b/w wt vector and input vector and one with the minimum eclidian distance is declared as

→ We are converting vector into unit length to get it ready for the calculation of activation function

→ Now after this all our I/P vector should be converted to have unit length.

$$x_{pj} = x_{pj} \Big/ \left[\sum_{j=1}^{n} x_{pj}^2\right]^{1/2} \qquad \begin{matrix} j = 1 \text{ to } n \\ p = 1 \text{ to } P \end{matrix} \quad (2)$$

same

→ Now we convert wt. vector into vector of unit length.

(To find out the winner node we have to use activation function that is why we have converted into vector of unit length)

$$w_{ij} = \frac{w_{ij}}{\left[\sum_{j=1}^{n} w_{ij}^2\right]^{1/2}} \qquad \begin{matrix} j = 1 \text{ to } n \\ i = 1 \text{ to } M \end{matrix} \quad (3)$$

→ Calculate the activation of output node.

$$a_i = \sum_{j=1}^{n} w_{ij} x_{pj} \qquad i = 1 \text{ to } M \quad (4)$$

And one with the higher a_i value will be declared as winner node.

(winner node)

→ The winner node means the wt vector of that winner node is most similar to that input pattern.

21/8/18

Updation of connection weights

w_{ij} = synaptic link originating from jth node and terminating at the ith node.

$$\Delta w_{ij}(t) = \eta(t)\,[x_{pj} - w_{ij}] \quad \text{—— (5)}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad \text{—— (6)}$$

$j = 1$ to n.

$i \in$ set of nodes in the neighbourhood of winner node along with winner node.

Flowchart for the training of SOFM.

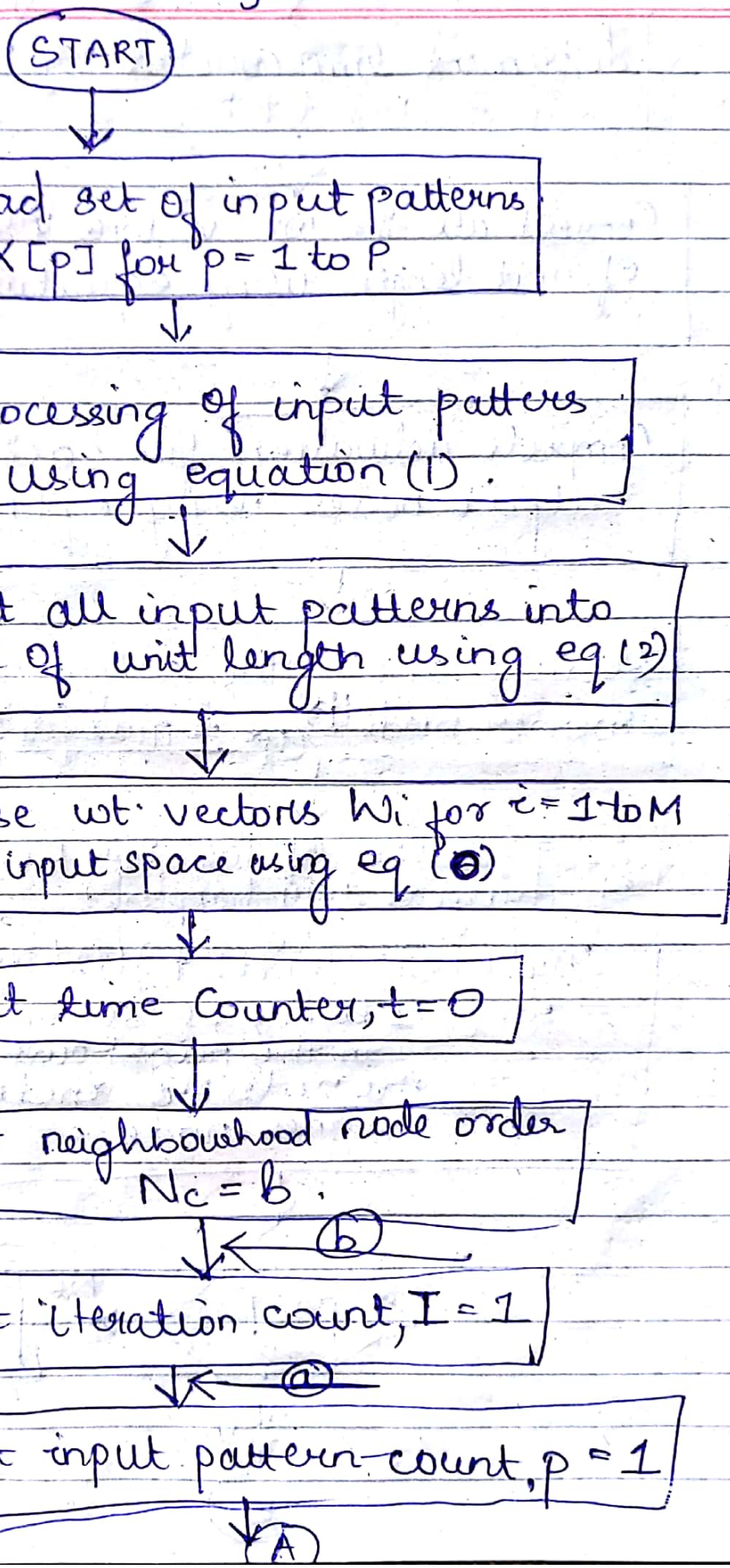
At any instant of time,

$$\eta(t) = \frac{\eta(0)\, e^{(-A/B)(t)}}{(1 + N_e)}$$

A & B are constants

The const. A/B here signifies that initially the the value of η will be max and further it

{(OR) initially the learning is more and as we proceed further, learning is done in smaller steps}.



will go on decreasing}