Date
Page

Introduction to ANN.

Introduction to ANN, History of Neural Network &, Structure & working of Neural Network, Biological, Neural Net, Architecture Topology of neural network architecture, features Characteristics Types, Activation functions, Models of Neuron - MC Culloch & pitts Model, perceptron adaline model, Basic learning laws, Applications of neural networks, Comparison of BNN & ANN.

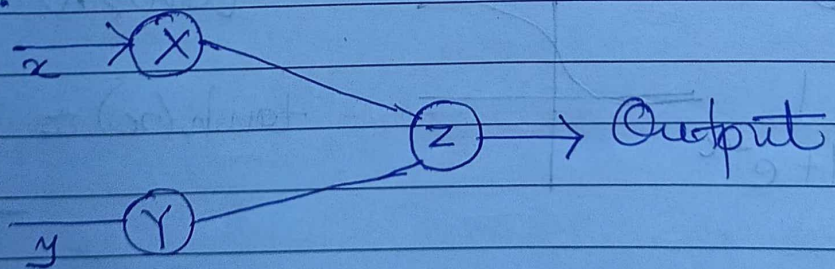
#1. Characteristics of ANN:

i. It is neurally implemented mathematical model.

ii. Interconnected processing elements called neurons to do all operations.

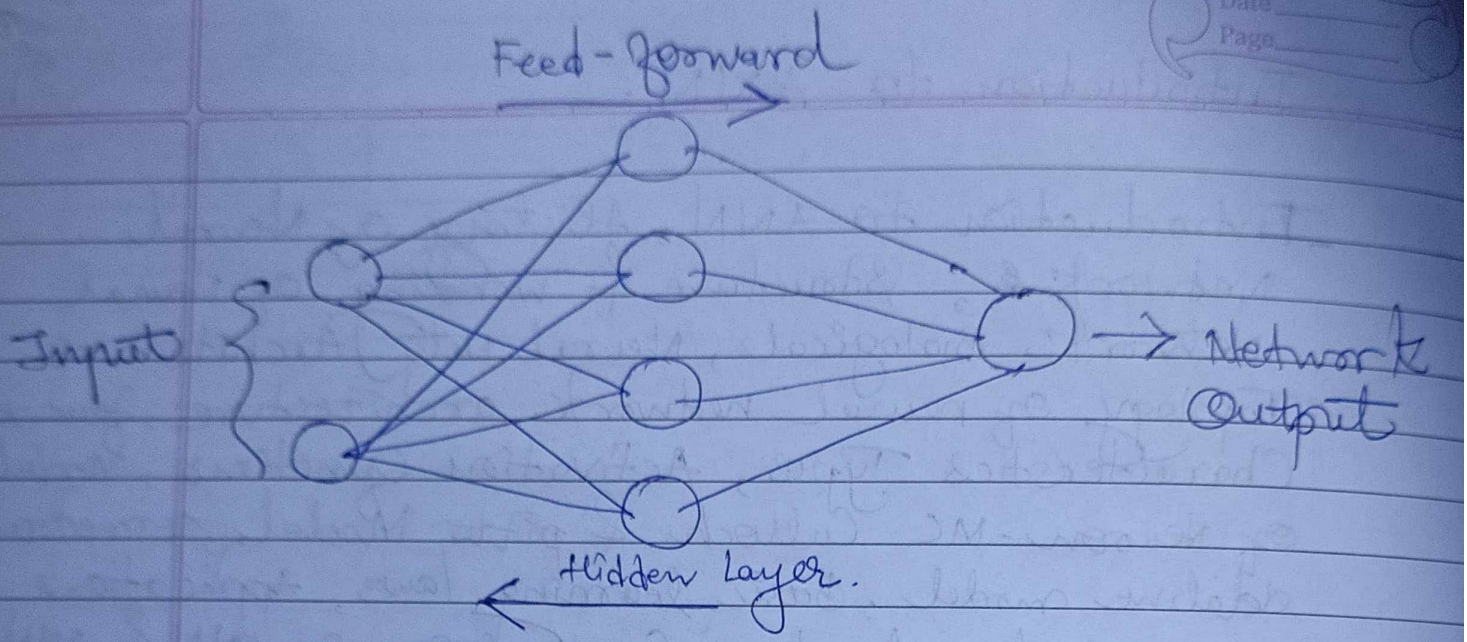
iii. Information is stored basically with weight linkage of neurons.

iv.



v. ANN algorithm accepts only numeric & structured data.

vi. CNN Convolutional Neural Network & RNN accepts unstructured & non-numeric data.



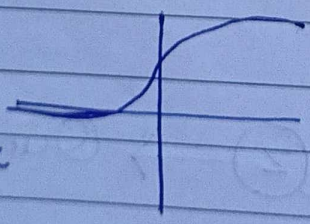
Hidden layer or distillation layer extracts some most relevant patterns & sends ahead for further analysis.

It accelerates & improves efficiency of network by recognizing imp info & discards redundant data.

#2 Activation functions:

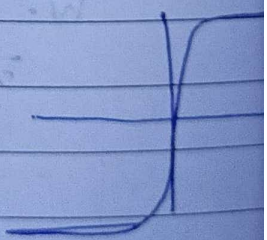
Sigmoid

$$\phi(x) = \frac{1}{1+e^{-x}}$$



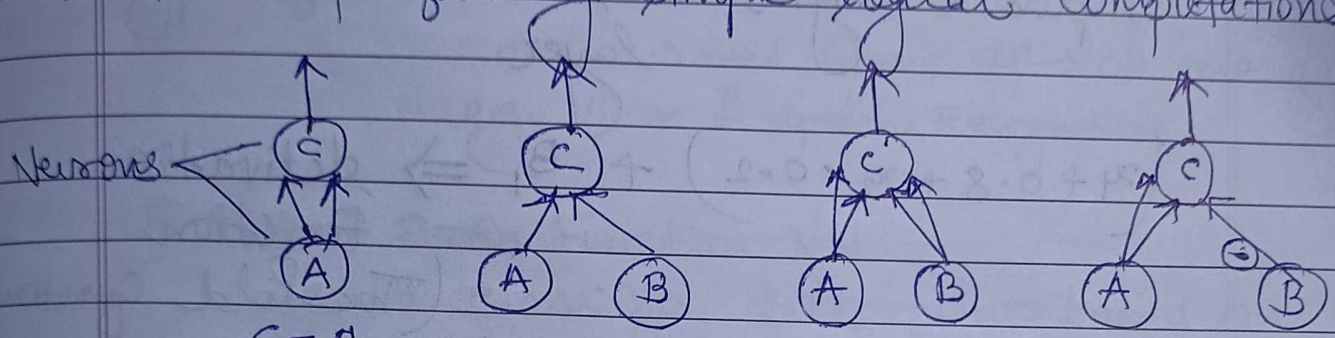
tanh

$$\tanh(x) =$$



#3.

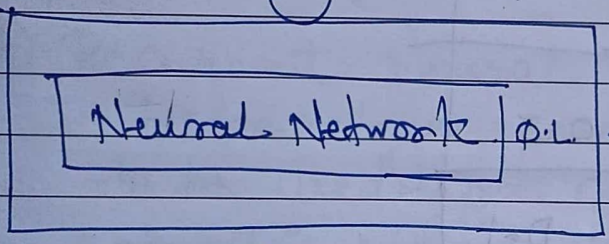
#4. ANN's performing simple logical Computations.



* Applications of Neural Networks.

Google Translation English to Marathi

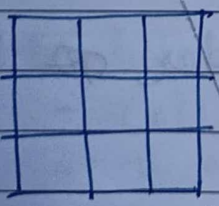
Face Recognition, Music Composition



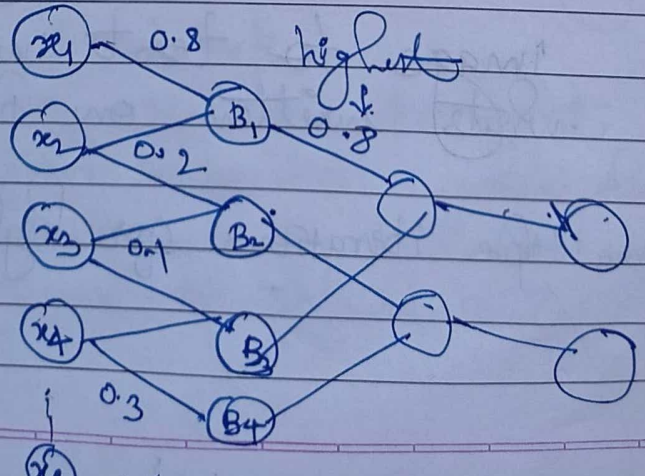
Algorithms are inspired by structure of human brain

Data → NN → output.

training of data



9 pixels



$(x_1 \cdot 0.8 + x_2 \cdot 0.2 + \ldots$

Inputs x weight = Send to hidden layer

$(x_1 \cdot 0.8 + x_2 \cdot 0.2) + B_1 \Rightarrow$ activation function (Threshold function)

→ activation function will decide whether neuron to be ~~op~~ activated or not.

Actual outputs	Error
1	0.5
0	-0.1
0	-0.5

* Deep learning : Magic Box

Optical Character Recognition is used for image to text in order to identify ~~whats~~ written on plates.

* multiple iterations for high accuracy.

* Types of ANN ① Feed forward NN.
data travel from IP → OP one direction
eg. Vision & Speech Recognition.

② Radial Basis Function NN
model classifies data point based on its distance from a centre point. power restoration sys

③ Kohonen Self Organizing NN:
- Vectors of random dimension are input to Discrete map comprised of neurons.
- medical Data analysis → pattern Recognition.

④ Recurrent Neural Network:
- Hidden layer saves its output to be used for future Prediction.
- Robotics: flying drone to check what was the last move
- text to speech conversion

⑤ Convolutional Neural Network:
- Inputs are taken in batches.
- used in image & signal processing.

⑥ Modular Neural Network:
- Collection of NN to get output.
- Photos facial recognition, when there are lot many people, one network used for separation, next to figure out.

- Research.

Python packages

Import:

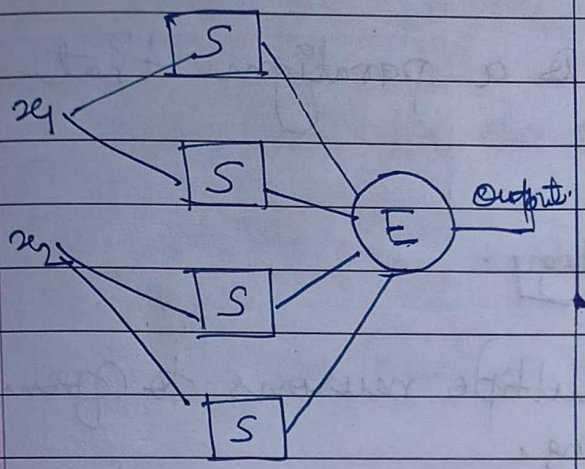
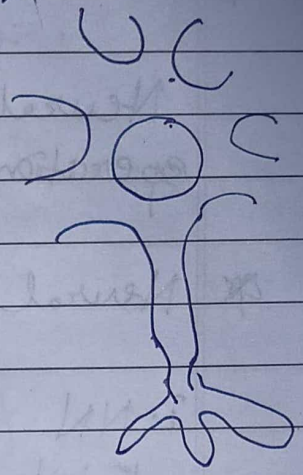
```
from keras.models import Sequential
from keras.layers import Conv2D
                 import MaxPooling2D
                 import Flatten
                 import Dense
```

Classifier → used for classification
Dog / Cat, true / False, Yes / No

* ERROR → Actual - Predicted

* loss → [actual output − (predicted output)]^2 ^{2}

* Backpropagation: For reducing error of prediction & update weights of network.

Artificial Neural Network	Biological Neural Network
(1) 	
(2) Processing element is carried out by neurons.	(2) It works on feed-forward strategy.
(3)	
Parameters: input, weights, output hidden layer.	Dendrites, Synapse, axon, cell body.
Learning: Complex network	Simple network
Processor: High speed i.e. efficiency	low speed.
Memory: localized	distributed.
Computing: Sequential manner	parallel manner
Environment: Well-Constrained	Un-Constrained
Separate from processor.	Integrated into processor.
Nodes and Edges	cells.

यशराज देवरात.

Date
Page

* Hebbian Learning Rule:

i. used for pattern classification.

ii. One input & one output

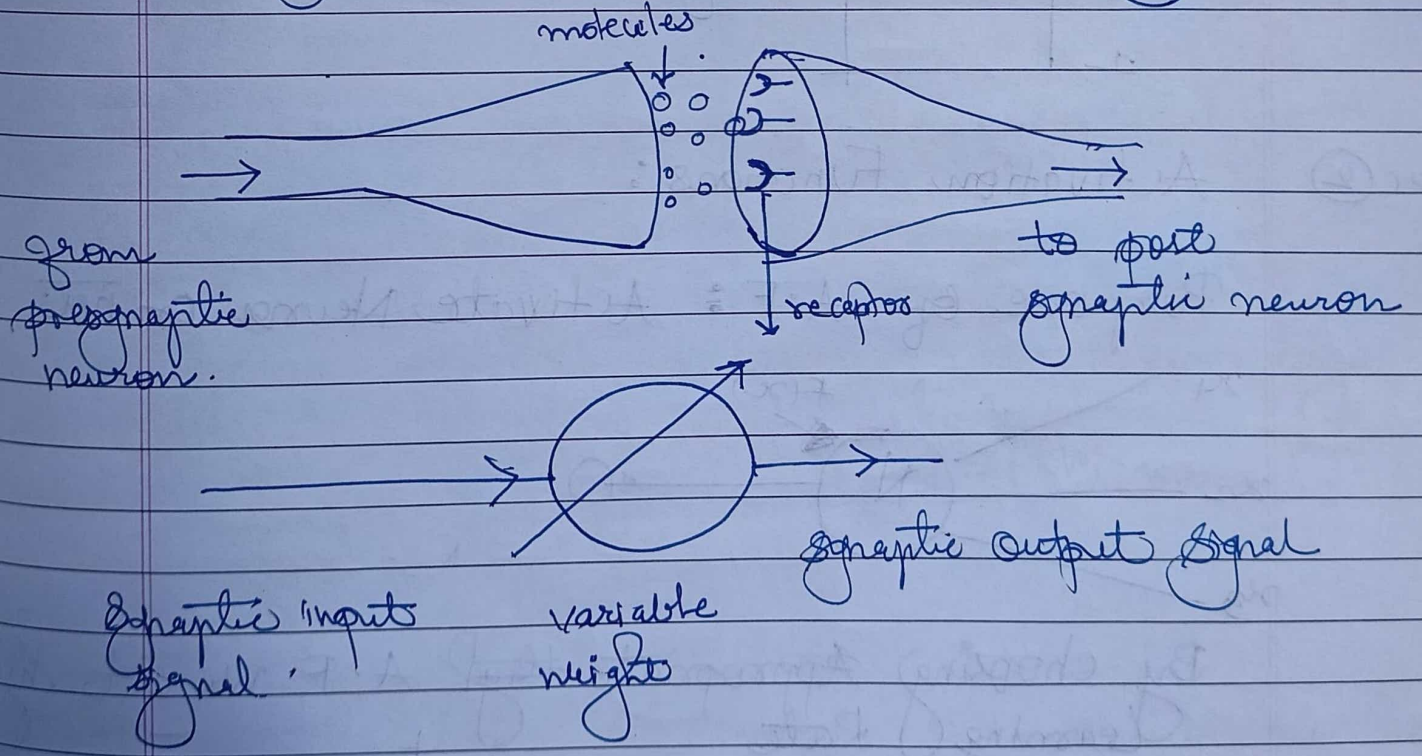
iii. weights are updated

$$W_{ia}(new) = W_i(old) + x_i y$$

$$b(new) = b(old) + y$$

Presynaptic → transmits chemical changes.

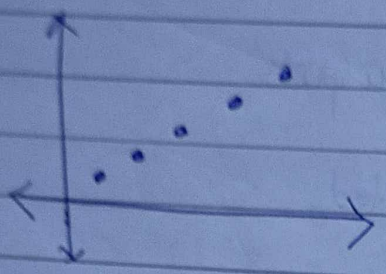
Postsynaptic → receivers chemical changes.



Summary

Lec ①

$$n_1 = (W_{11}x_1 + W_{21}x_2)\ O_1$$

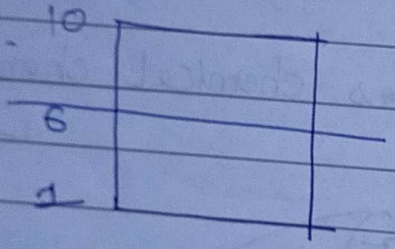


Equilibrium Rate $= \sum_{i=1}^{4} O_i$

Color	Length	Weight
Brown	5	10
Red	6	13

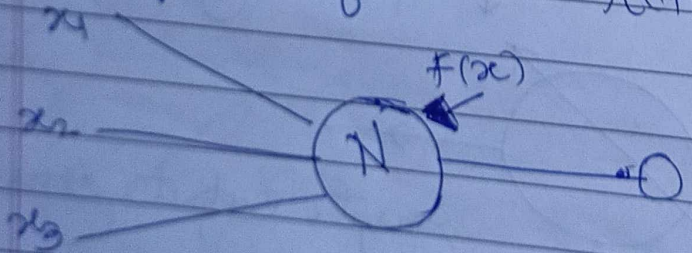
Condition for sorting accordingly

Color → Brown & length ≤ 7



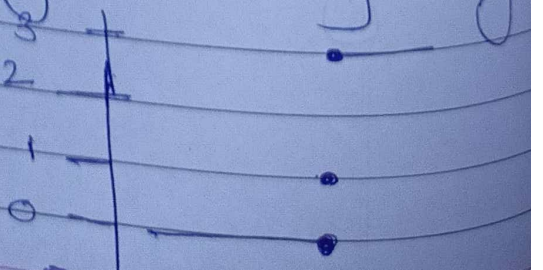
Lec ② Activation Functions:

Purpose of A.F: Activate Neuron or not



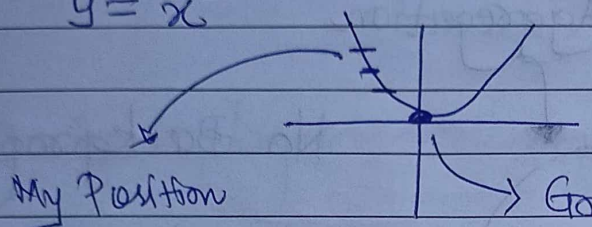
By choosing Appropriate A.F we get high Learning Rate.

Binary: $f(x) \rightarrow 1/0$



$f'(x) = 0 \longrightarrow$ No Learning

$y = x^2$



$y = mx + c$

Fixing the direction to ~~act~~ achieve goal.

* Produce Non-linearity output

① Sigmoid : $\frac{1}{1+e^{-x}}$, $y'(x) = e^{-x}(1+e^{-x})^{-2}$

② tanh : $\frac{e^x - e^{-x}}{e^x + e^{-x}}$, $y'(x) = (1 - (g(x))^2)$

③ Relu $f(x) = \max(0, x)$, $y'(x) = \begin{cases} 1, & x > 0 \\ 0 & \text{other} \end{cases}$

* ④ ~~Softmax~~ $f(x) = x * (x \cdot \text{Sigmoid}(x))$.

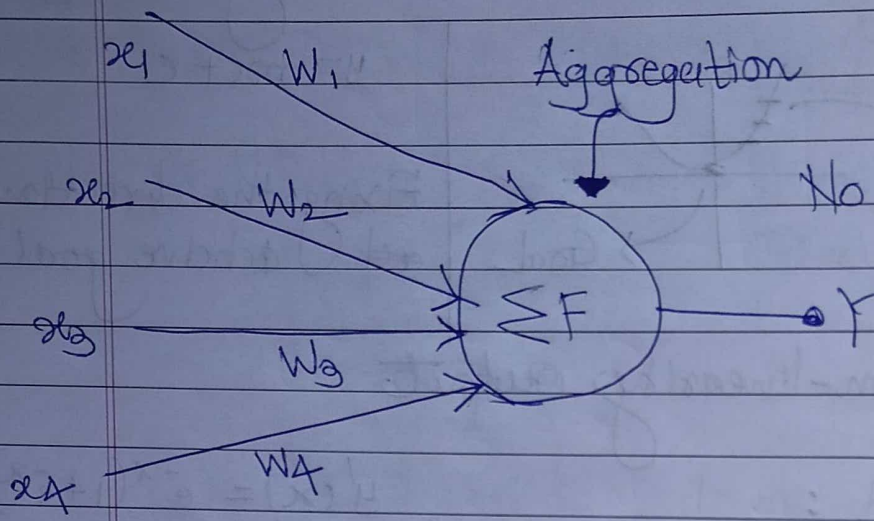
- ~~Softmax~~ works better at ~~Output~~ layer
- other functions are better at hidden layer.

Mean = $\frac{1}{N} \sum_{i=1}^{N} P_i W_i$, $S_i = (P W_i - \text{mean})^2$

* Hidden layer finds out the equilibrium stage.

MP Model Vs Adaline Model.

Lec ③ Mc Culloch Pitts Model (MP Model)



$$\sum_{i=1}^{n+1} x_1 w_1 + x_2 w_2 + x_n w_n \ldots\ldots$$

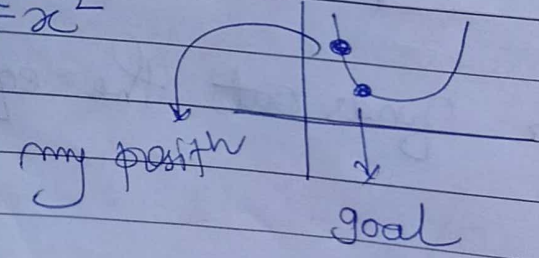
if $X > T$
fires 1 Excitatory Stage

else $X < T$
fires 0 Inhabitatory Stage

Lec ④ Gradient / derivative

if $f'(x)$ is zero → No Learning

$y = x^2$



$y = mx + c$, $m' = m - \delta m$

fix the direction.

Output layer → Softmax function.

Activation function produces non-linearity. It also selects neurons for achieving great learning.

$$\text{Mean} = \frac{1}{N} \sum_{j=1}^{N} \Phi w_j$$

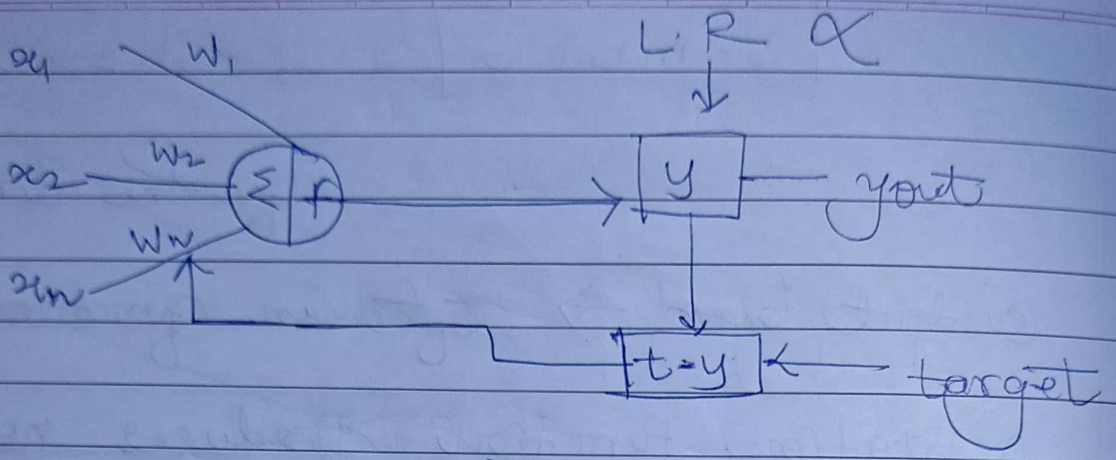
* MP Model : $u = \sum_{j=1}^{N} w_j y_j + \theta$

It has no Backpropagation

AND : $A \cdot B$
OR : $A + B$
XOR : $\overline{A+B}$
NOR : $\overline{A}$

* Adaline Model

MP Model → No backpropagation, No increase in efficiency
Linear Approach.
classification



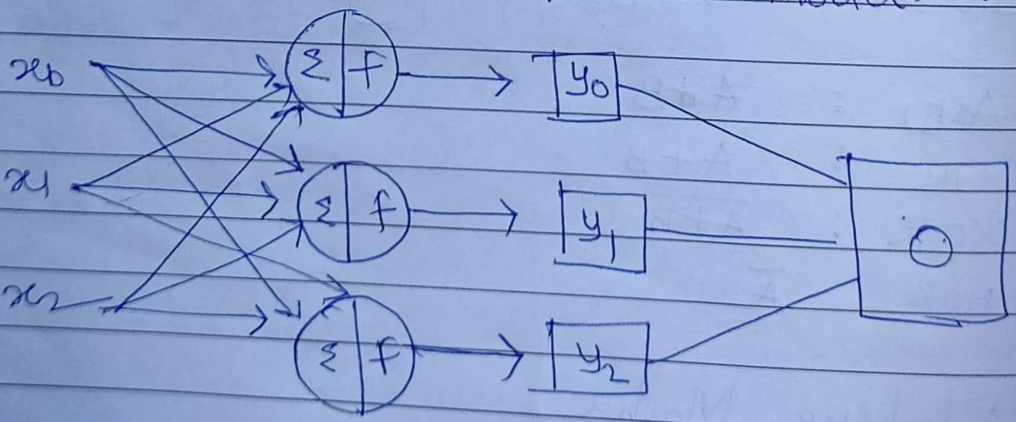
$$\text{Error} = t - y$$

$$X = \sum_{i=1}^{n} x_i w_i + b$$

$$W_{new} = W_{old} + \alpha (t-y) \cdot x_i$$

$$RMSE = \sqrt{(t-y)^2} \qquad (t-y)^2$$

* Madaline model : multilayer Adaptive linear Model.



* Learning Rules:

Delta Rule :

do Change in weights till it reaches threshold value

$$E = \sqrt{(t-y)^2}$$ Cummulative Effect

Calculate Rate of change in Error.

$$\nabla E = \frac{\partial E}{\partial W_i} = \frac{\partial}{\partial W_i} = x_1 w_1 + x_2 w_2 + b$$

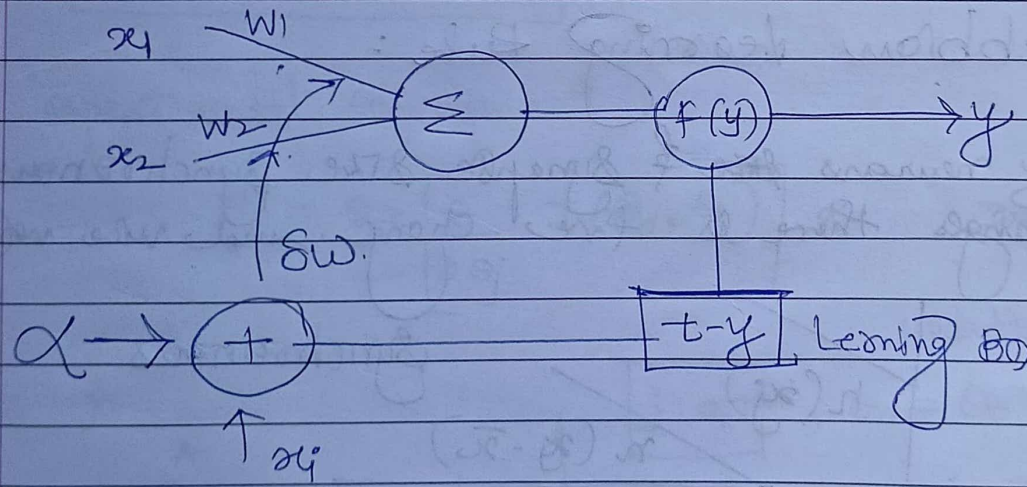
$$\nabla E = (t-y)^2 = t^2 - 2ty + y^2$$

$$\Delta E = -2t - 2y + 0$$

$$\nabla E = -2(t-y)$$

$$\nabla E = -\alpha (t-y)$$

$$\nabla E_{ij} = -\alpha (t - y_i)$$



Delta Rule = rate of change in weights.

If output comes Zero we will stop.

Summary mathematical Concept.

Lec (5)

x_1 x_2 t. $W_1 = 0.2$, $W_2 = 0.2$, $b = 0.2$

$\alpha = 0.2$

$$y_{in} = \sum x_i w_i + b$$

$$f_{y_{in}} = y = e = t_i - y$$

Current Error $= E = (t_i - y)^2$

$$n_1 w_1 = w_1 + \alpha (t_i - y) \cdot x_i$$

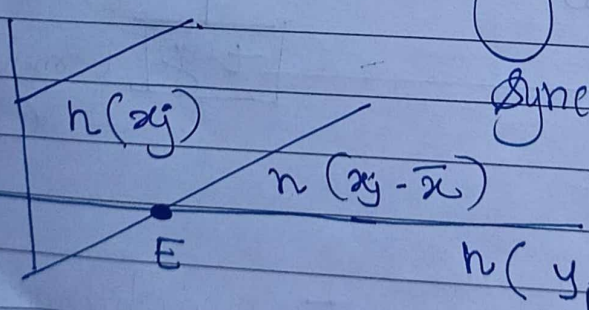
$$n_2 w_2 = w_2 + \alpha (t_i - y) \cdot x_i$$

$$E = E_1 + E_2 + E_3 \quad , \quad MSE = \sqrt{(t_i - y)^2}$$

* Hebbian learning Rule:

If neurons ~~fire~~ & Synapse fire synchronously, changes there is fire change. and vice versa

Asynchronous.



$$\Delta W_j = n (x_j - \bar{x}) (y_j - \bar{y})$$

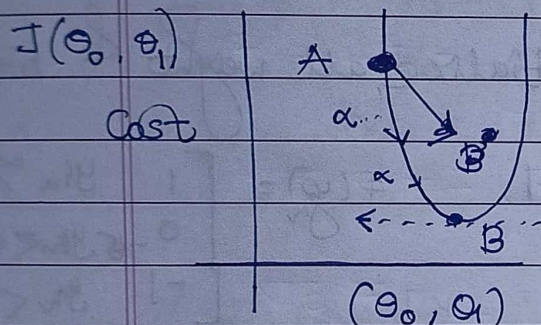
$$\Delta W_j = n f(x_j, y_i)$$ Hebbian Hypothesis.

Think X Academy

Activation Function: Activate Hidden layer node produces non-linearity

* Gradient Descent: update weight & bias to get minimize cost.

↓ Slope → Going down. → How algorithm will know B is minimum point.



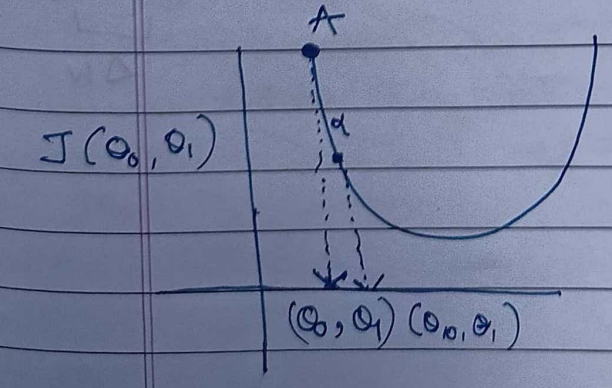
α = L.R

$$\frac{d}{d\theta}\left(J\theta_0, \theta_1\right) = 0$$

repeat until Convergence {

$$\theta_j = \theta_j - \left[\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)\right] \alpha$$ Point-B

}



updation

$j : 0, 1$, $\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} [J(\theta_0, \theta$

$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} [J(\theta_0, \theta$

}

repeat until Convergence {

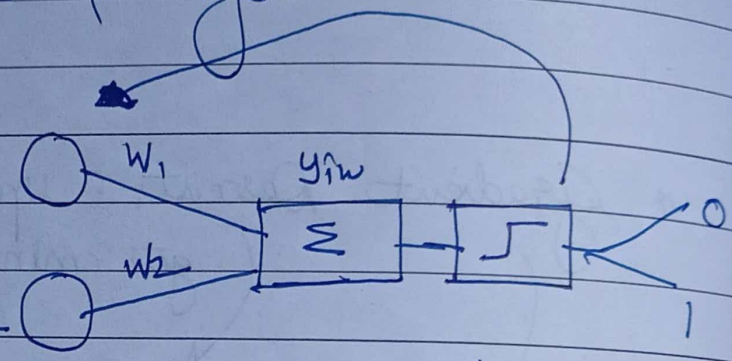
$\theta_0 : \theta_0 - \alpha \frac{1}{m} \left(\sum_{i=1}^{m} (h_\theta - y_i) \cdot x_i\right)$

$\theta_1 : \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta - y_i) \cdot x_i$

Date
Page

9. Implement AND function using perceptron network for bipolar inputs & targets.
(-1, 1)

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



$y = x \cdot w + b$

Step 1 : Initialize all weights & bias = 0

$w_{new} = w_{old} + \alpha \Delta w$, $\alpha = 1$

$$f(y_{in}) = \begin{cases} 1 & y_{in} > \theta \\ 0 & -\theta \le y_{in} \le \theta \\ -1 & y_{in} < \theta \end{cases}$$

Step 2 : $w_1 = w_2 = b = 0$, $\alpha = 1$

calculate new inputs : $y_{in} = x_1 w_1 + w_2 x_2 + b$

$y_{in} = 0$

Step 3 : Sigmoid Activation function : $y = f(y_{in})$

$f(0) = 1$, $f($

$y = 0$

as $t = 1$ move to step 4

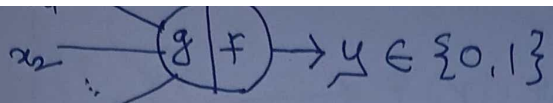
Step 4 : update the weights : $w_{new} = w_{old} + \alpha t x_i$ (Δw)

$b_{new} = b_{old} + \alpha t$

$w_1 = 0 + (1)(1)(1) = 1$

$w_2 = 0 + (1)(1)(1) = 1$

$b_{new} = 0 + (1)(1) = 1$



$x_n \in \{0,1\}$

MP model

Date
Page

Book 2nd epoch $x_1 = 1, \quad x_2 = -1, \quad b = -1$

$y_{in} = x_1 w_1 + w_2 x_2 + b$

$y_{in} = (1)(1) + (-1)(1) + 1 = 1$

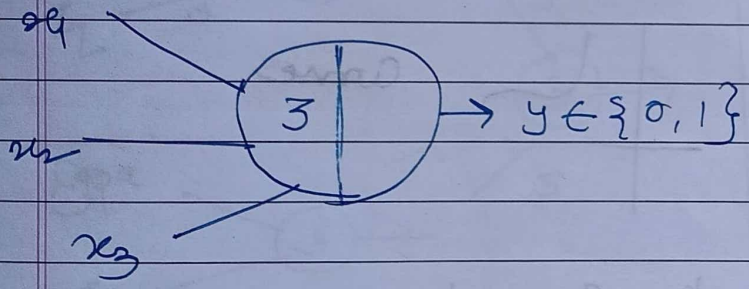
$f(y_{in}) = f(1) = 1$ as $t = -1$ move to Step 4

Step 4: $w_1 = 1 + (1)(-1)(1) = 0$

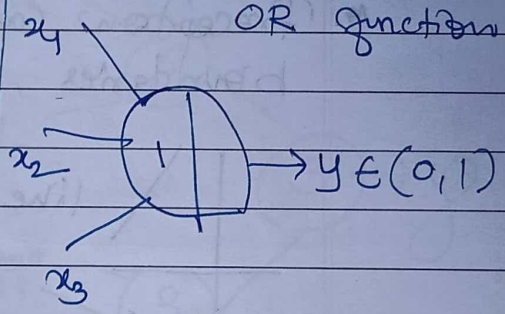
$w_2 = 1 + (1)(-1)(-1) = 2$

$b_{new} = 1 + 1(-1) = 0$

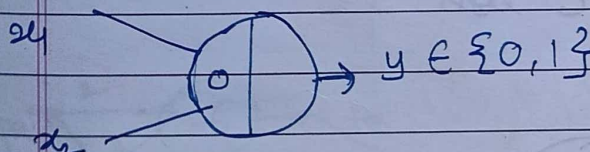
AND function:

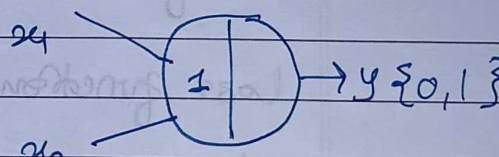


OR function



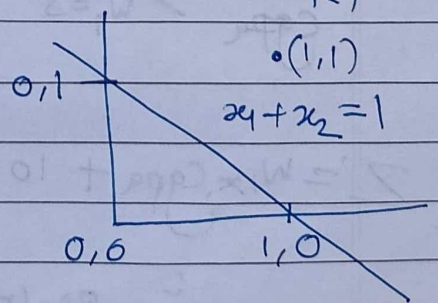
NOR function





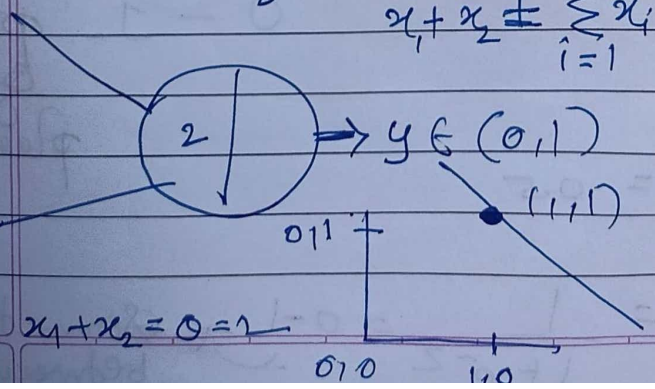
OR function

$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$



AND function

$x_1 + x_2 \geq \sum_{i=1}^{2} x_i \geq 2$



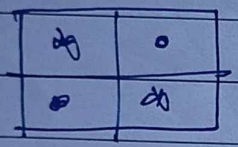
Campus X

* What is perceptron ? How to train ?

$Ax + By + C = 0$ Logistic Regression

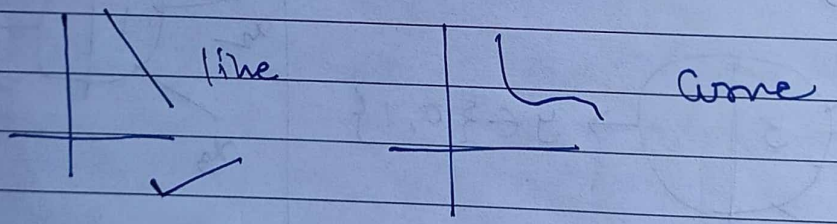
$y = mx + b$ Linear Regression

* Perceptron can not classify XOR input



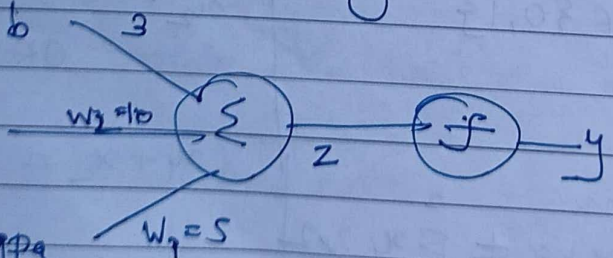
need of Multilayer perceptron.

* Preceptron cannot create nonlinear decision boundaries, it is linear model.



* Activation funt^{n}: Sigmoid.

Loss function: logistic loss



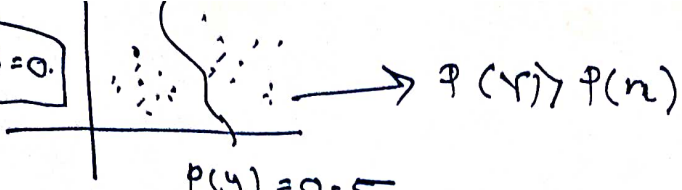
Act - Sigmoid.

0 - 1 probability of yes placement

$Z = W_1 * Cgpa + 10 * iq = 0.5$

$\therefore \quad Sig(z) = \frac{1}{1 + e^{-z}} = 0-1$ → Something between

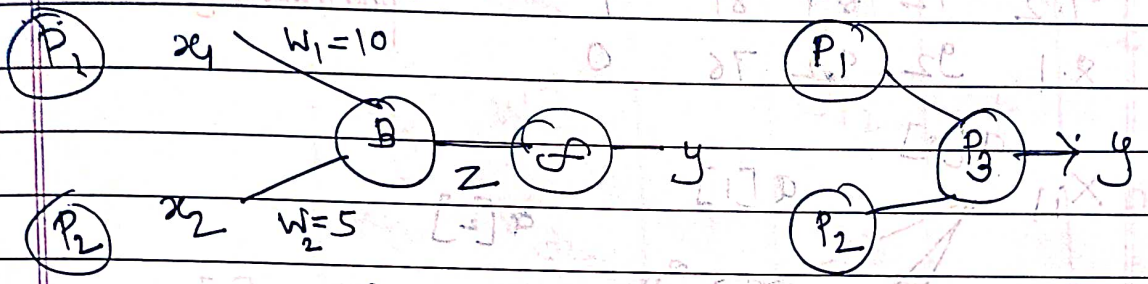
$5x + 10y + 3 = 0$

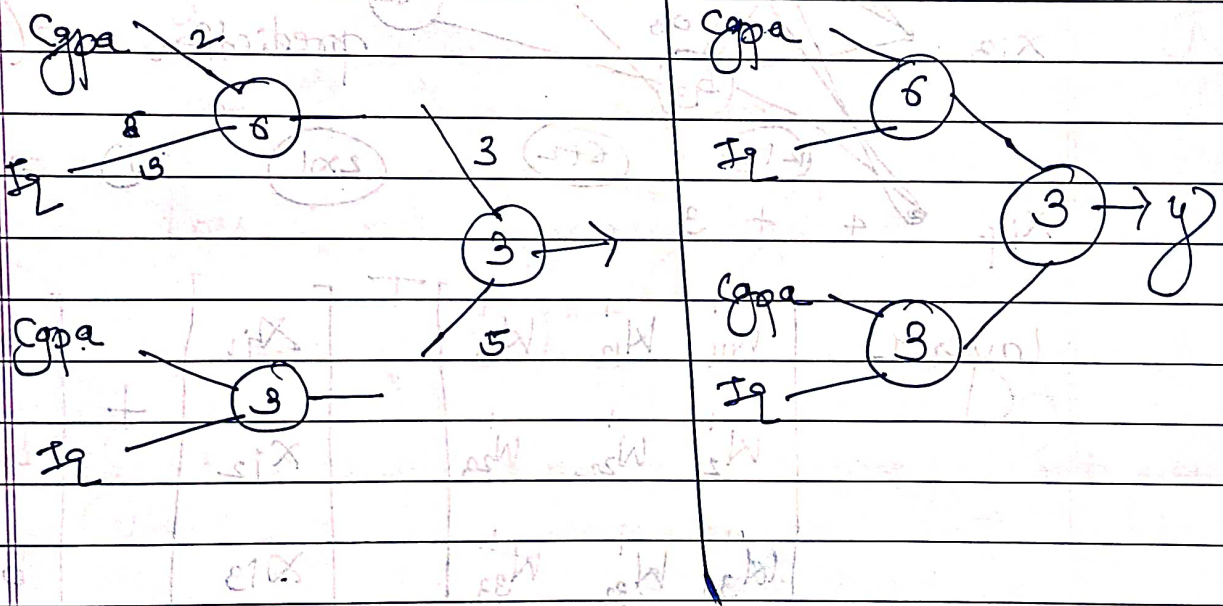


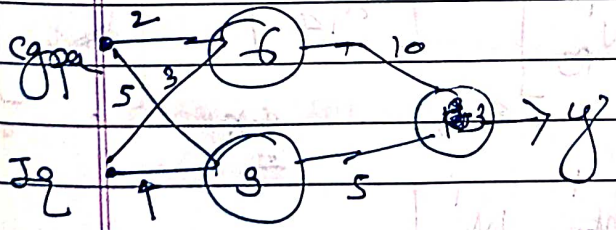
Date
Page

0.8 answer. yes → 0.2 probability of no placement.

MLP :







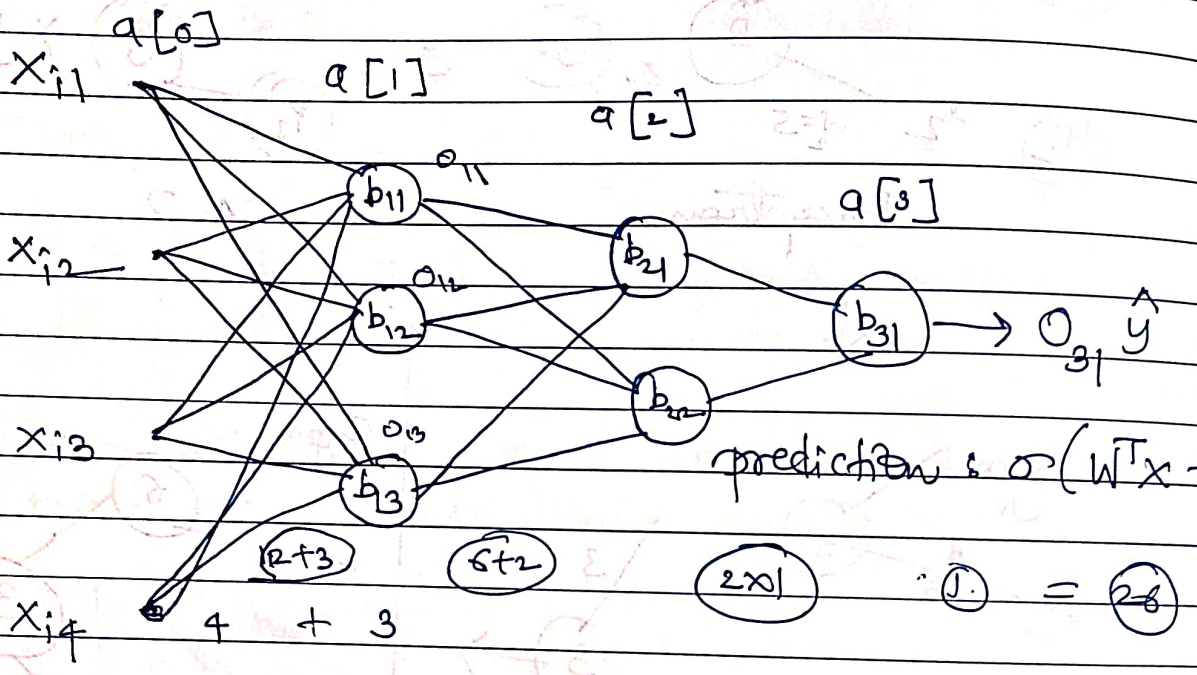
78-8K

Date
Page

* Forward Propagation :

cgpa	iq	10th	12th	placed
7.2	72	6.9	81	1
8.1	92	9n	76	0

trainable parameter



prediction = $\sigma(W^T x + b)$

Layer 1

$$\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \\ w'_{31} & w'_{32} & w'_{33} \\ w'_{41} & w'_{42} & w'_{43} \end{bmatrix}^T \begin{bmatrix} x_{i1} \\ x_{i2} \\ x_{i3} \\ x_{i4} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{14} \end{bmatrix}$$

4×3 $\quad 4\times1 \quad 3\times1$

$$= W^T = \begin{bmatrix} w'_{11} & w'_{21} & w'_{31} & w'_{41} \\ w'_{12} & w'_{22} & w'_{32} & w'_{42} \\ w'_{13} & w'_{23} & w'_{33} & w'_{43} \end{bmatrix}$$

3×4

O_{11}
O_{12}
O_{13}

$$\sigma\left(W_b^T x(x) + b\right) = O_{ij}$$

Layer 2

$$\begin{bmatrix} W_{11}^2 & W_{12}^2 \\ W_{21}^2 & W_{22}^2 \\ W_{31}^2 & W_{32}^2 \end{bmatrix}^T \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{22} \end{bmatrix}$$

$$\sigma\left(\left[W_{11}^3 O_{21} + W_{21}^2 O_{22} + b_{31}\right]\right) = \hat{y}_i \cdot O_{31}$$

* Loss function: it tells how dataset is modelling.

Loss ↑ → efficiency low

Loss ↓ → efficiency high

$$L = (y_i - \hat{y})^2$$

Why it is imp: you can improve what you measure.

Lost function Vs Cost function

single training example	overall dataset
$(y_{in} - \hat{y}_{in})^2$	$\frac{1}{n} \Sigma (y_i - \hat{y}_{in})^2$

LPA.

iq	cgpa	lpa
80	8	9
100	9	

(18) ↓

* Backpropagation :

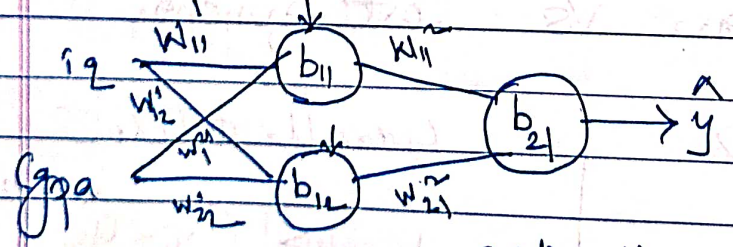
1. It is an algorithm to train NN.
2. Produce optimum weights & biases for given data for neural network.
3. How ? Gradient Descent
   Forward Propagation
   Regression.

Steps : 1. You may select points (row).

2. Predict lpa (Students) → forward propagation

3. choose error / loss function.

$$M.S.E = (y - \hat{y})^2$$



activation function : Linear Regression

$$M.S.E = (y - \hat{y})^2$$

3 - 18

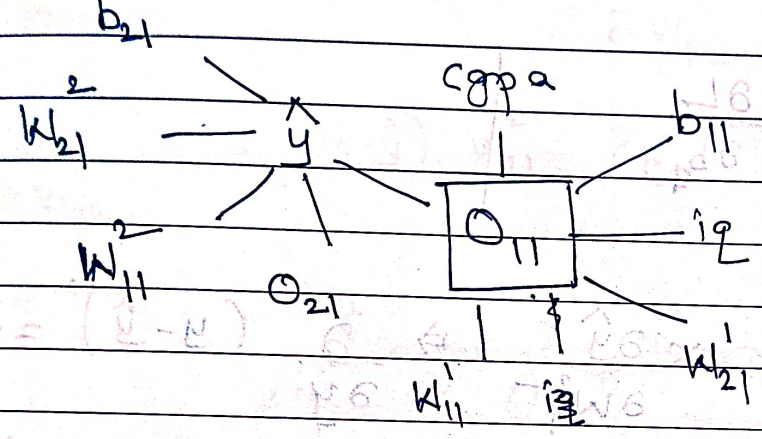
$$L = (15)^2$$

$$L = 225$$

(18) ↓ error

$$O_{21} = W_{11}^{2} O_{11} + W_{21}^{2} O_{21} + b_{21}$$

$$Loss_{min} = \hat{y} = O_{21}$$



⇒ Backpropagation

↑. weight & bias update

↳ Gradient Descent

$$W_{new} = W_{old} - \frac{\partial L}{\partial W_{old}} \alpha$$

$$b_{new} = b_{old} - \frac{\partial L}{\partial b_{old}} \cdot \alpha$$

weight W_{11}^{2}

$$W_{11_{new}}^{2} = W_{11_{old}}^{2} - \frac{\partial L}{\partial W_{11}^{2}} \alpha$$

$$b_{21_{new}} = b_{21_{old}} - \alpha \frac{\partial L}{\partial b_{21}}$$

$$W_{21_{new}}^{2} = W_{21_{old}}^{2} - \frac{\partial L}{\partial W_{21}^{2}} \alpha$$

$0 < x < 1$

Derivatives of Loss / Error w.r.t all weights

$$\frac{\partial L}{\partial W_{11}^2}, \frac{\partial L}{\partial W_{21}^1}, \frac{\partial L}{\partial b_{11}} \qquad \frac{\partial L}{\partial W_{11}^2}, \frac{\partial L}{\partial W_{21}^2}, \frac{\partial L}{\partial b_{21}}$$

$$\frac{\partial L}{\partial W_{12}^2}, \frac{\partial L}{\partial W_{22}^1}, \frac{\partial L}{\partial b_{12}}$$

$$\frac{\partial L}{\partial W_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W_{11}^2} \quad \Rightarrow \frac{\partial}{\partial y}(y - \hat{y})^2 = -2(y - \hat{y})$$

$$\frac{\partial \hat{y}}{W_{11}^2} = -2(y - \hat{y}) \frac{\partial}{\partial W_{11}^2}(O_{11} W_{11}^2 + O_{12} W_{21}^2 + b_{21}) = O_{11}$$

$$\boxed{\frac{\partial L}{W_{11}^2} = -2(y - \hat{y}) O_{11}}$$

$$\frac{\partial L}{\partial W_{21}^2} = \frac{\partial}{\partial W_{21}^2}(O_{11} W_{11}^2 + O_{12} W_{21}^2 + b_{21}) = O_{12}$$

$$\boxed{\frac{\partial L}{\partial W_{21}^2} = -2(y - y_{in}) O_{12}}$$

$$\boxed{\frac{\partial L}{\partial b_{21}} = -2(y - \hat{y})}$$

$$\frac{\partial L}{\partial W_{11}^1} = 2(y-\hat{y})\, W_{11}^2 \cdot X_{i1} \qquad \frac{\partial L}{\partial W_{21}^1} = -2(y-\hat{y}) W_{11}^2 X_{i2}$$

$$\frac{\partial L}{\partial b_{11}} = -2(y-\hat{y})\, W_{11}^2 \qquad \frac{\partial L}{\partial W_{12}^1} = -2(y-\hat{y})\, W_{21}^2 X_{i1}$$

$$\frac{\partial L}{\partial W_{22}} = -2(y-\hat{y})\, W_{21}^2 \cdot X_{i2} \qquad \frac{\partial L}{\partial b_{12}} = -2(y-\hat{y})\, W_{21}^2$$

Summary of Backpropagation Algorithm

epoch = 15 → (5)

for i in range (epochs):

for j in range (X, shape[0]):

→ select 1 row (random)

→ predict (using forward propagation)

→ calculate loss (using M.S.E)

→ update weights & bias using G.D.

→ $w_{new} = w_{old} - \alpha \frac{\partial L}{\partial w_0}$

Calculate loops for epochs: $\frac{L_1 + L_2 + L_3 + L_4}{4}$

average loss.

UNIT 2

Learning Algorithms

Learning and memory, Learning Algorithms, no. of hidden layers / nodes, Error correction, gradient descent rules, Perceptron learning algorithms, Supervised learning, Backpropagation, multilayer network architecture, Back propagation learning algorithm, feedforward & feedback neural networks, example & its applications

* Energy / potential

$$E(D) = \frac{k}{n} \log_2 \frac{k}{n} - \frac{n-k}{n} \log_2 \frac{n-k}{n}$$

$$E(D) = -\sum_{i=1}^{k} P_i \log_2 P_i$$

Competative learning → higher potential neuron will get fired

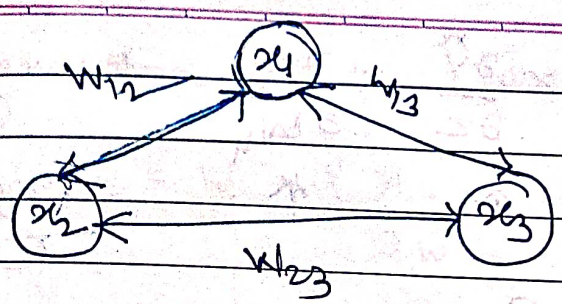
Change in Energy → Change in weights

$$N_h = \frac{N_s}{(N_o + N_i)\alpha}$$

α = Scaling factor

$$\beta = 0.7 (p)^{1/n}$$

Gain = info (D) − info (D) after 1



$e_i = x_1 = x_2 w_{12} + x_3 w_{13}$

$\sum e_i = e_1 + e_2 + e_3$

$\therefore E = -\frac{1}{2} (e_i x_i)$

* Backpropagation → Gradient Descent rule / generalized delta rule.

* Vanishing Gradient Problem

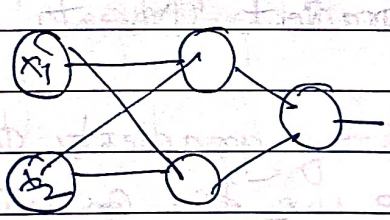
- Neural Network doesn't get trained.

1) $0.1 \times 0.1 \times 0.1 = 0.0001$
↳ V.G.V

2) more hidden layers: □□□□□

3) Sigmoid / tanh

Backpropagation

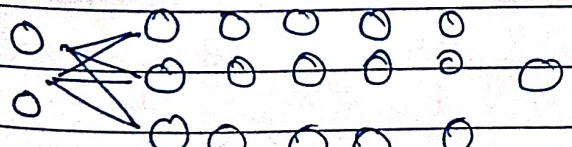


$\hat{y} - y = L / E$

$w_n = w_0 - \alpha \frac{\partial E}{\partial w}$

$w_n = 1 - 0.01 \times 0.0001$

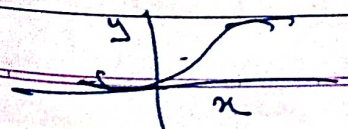
$w_n = 0.999$ Vanishing gradient small



$\frac{\partial L}{\partial w_{11}} = 0.00001$

Backpropagation model → x

training



$$\frac{\partial L}{\partial w_1'} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_1'}$$

↑ ↑ ↑

→ very small change

↓

very small loss

How to recognize ?

1) Loss goes → epoch → no changes → vgp

2)

Gradient = old weights − new weights / learning rate

How to reduce ?

Reduce Complexity

use Relu function $|x|x| = |$

Batch normalization → layer

Residual network

* Batch Gradient Descent (vanilla)

(3 variants of gradient descent, which differ in how much we use to compute gradient of objective function.

accuracy → time

(BGD)

epoch = 10

for i in range (10):

y_hat = np.dot(x, w) + b

50 values

Date
Page

y_hat ≈ y → loss

$$w_n = w_o - \alpha \frac{\partial L}{\partial w}$$

→ print loss

* SGD:

epochs = 10

for i in range (10):

for in range (x.shape[0]) → shuffle

↳ print random

↳ loss

↳ y_hat

avg_loss print → for epoch

frequency of weight update is higher

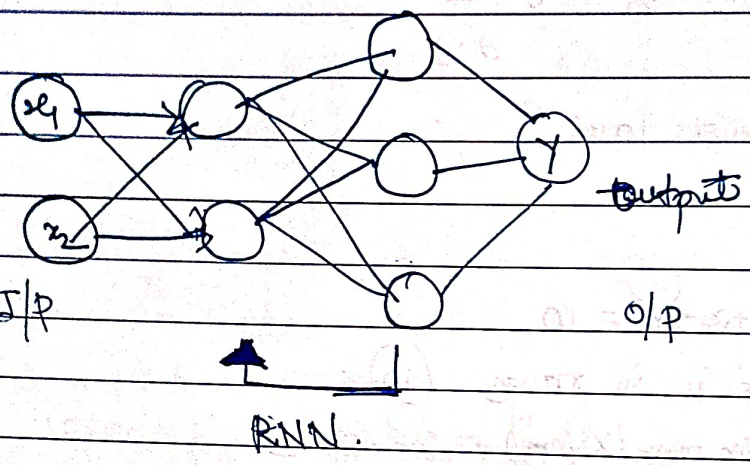
50 X 10 = 500

BGD	SGD
① Single epoch for all data points one time gradient calculate weight update	Single epoch, no of row times weights update.
② more faster	more weight updates.
③ For completing epoch BGD faster	slow, 10 hr
④ smooth curve for loss decreasing	for solution SGD is fast
	distorting curve for loss

10 epochs

Q1.

* Recurrent Neural Network:

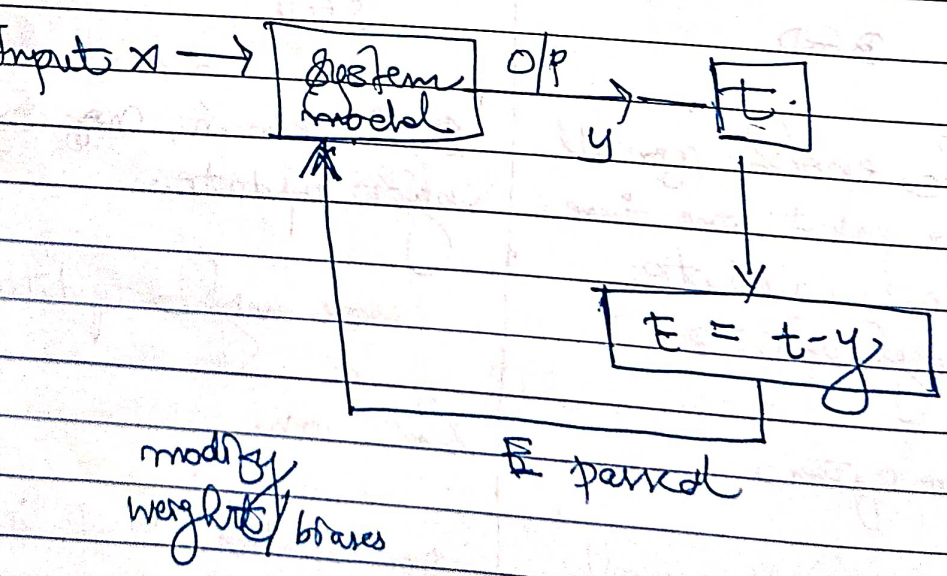


i. Information is stored within network.

ii. Classification / regression used in non-sequential data.

iii. used in NLP, image classification, speech recognition.

* ERROR Correction:



Vanishing Gradients Descents problem:

- When NN doesn't get trained.
- $0.1 \times 0.1 \times 0.1 = 0.0001$.
  ↳ very small

As update weights are,

$$W_n = W_0 - \alpha \frac{\partial E}{\partial W_0}$$

$W_n = 1 - 0.01 \times 0.001$

$W_n = 0.99$

Vanishingly small.

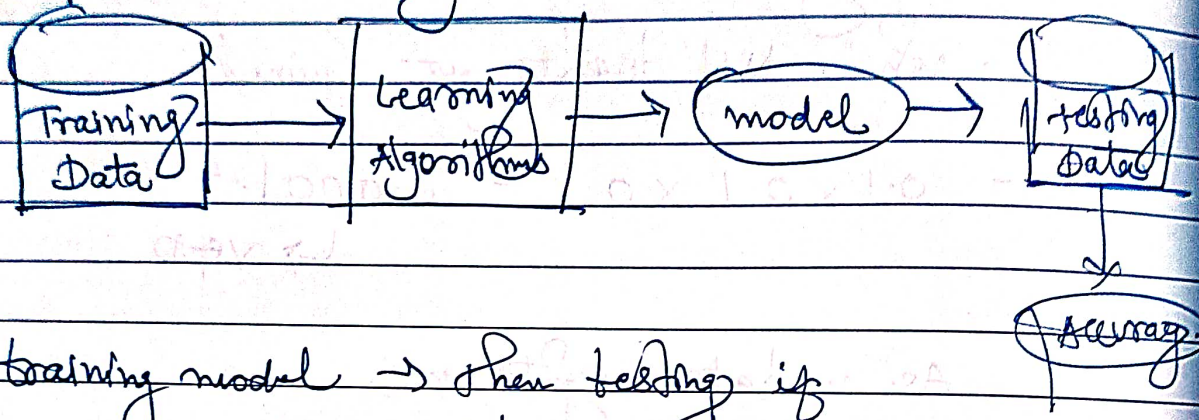
When partial derivatives have products with less value i.e. 0.0001.

At this situation gradient is small, so updates are small.

- How to solve this:
  - Good weights initialization.
  - use of LSTM network
  - Reduce Complexity
  - use of Relu $1 \times 1 \times 1 = 1$
  - Batch normalization - layers
  - more no of hidden layers.

Date
Page

Supervised learning:



training model → then testing if model is producing outputs.

Supervised → classification & regression

Supervised	Unsupervised
① Desired outputs is given	Desired targets not given
② Complex algorithm not processed	Complex algorithm
③ trainy data	No training data.
④ prediction of labeled data.	detection of relation in data
eg. character recognition	face recognition.

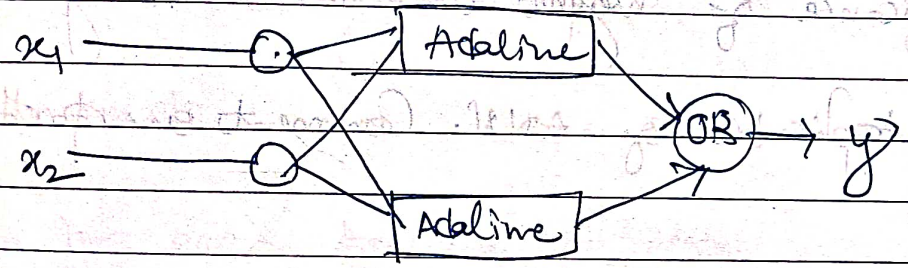
* Adaline model :

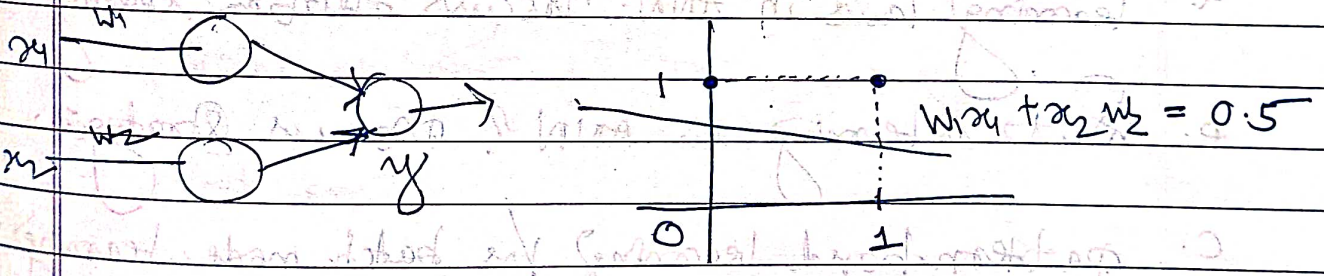
It is perceptron in which weights are modified in such way that MSE is diminished at every iteration.

Step function is has no role in training adaline.

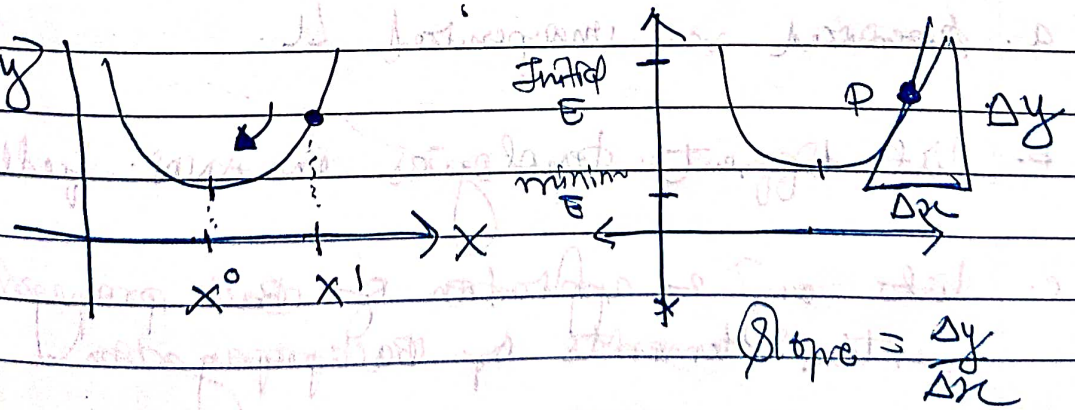
Adaline is same as that of perceptron when used for classification.

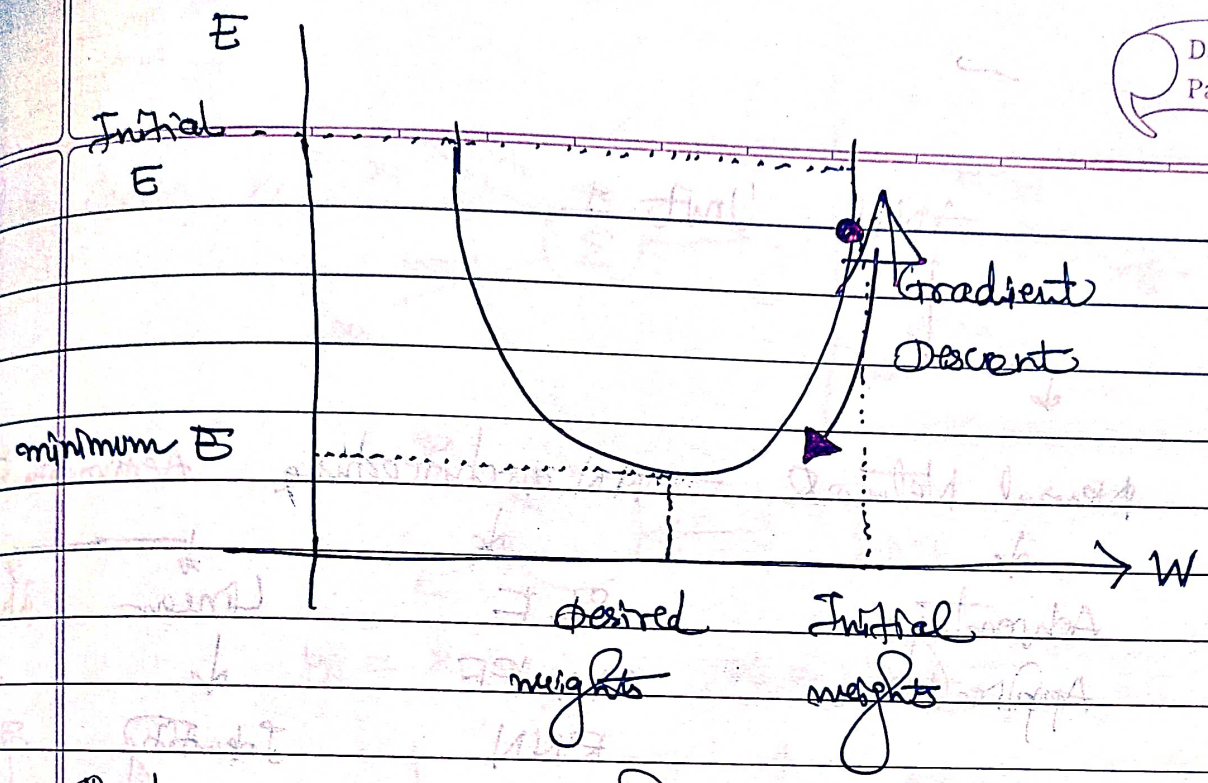
* madaline model





* Gradient Descent Rule :





Backpropagation algorithm:

initialize weights

repeat

for each training pattern

train on that pattern

end loop.

until error is low.

Date
Page

ANN Unit 1

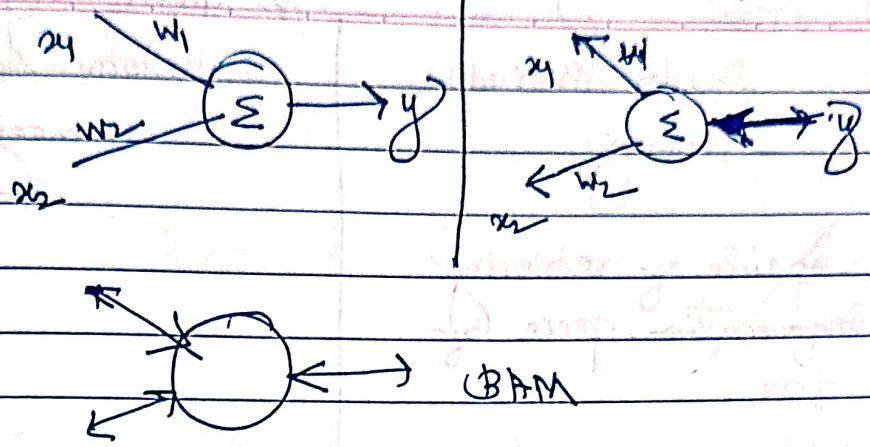
- History
- Introd^{n}
- Structure
- Neural Network
  - Advantages
  - Application
- NN architecture
  - SLF
  - ML FF
  - RNN
- Activation Fun^{n}
  - Linear
    - Identity
  - Non linear
    - Sigmoid

- Learning laws
- MP model
- perceptron
  - Single layer Perceptron
  - Multilayer Perceptron
  - Adaline / madeline

ANN Unit 2

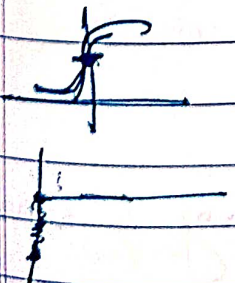
- Associative learning memory
  - Auto
  - Hetro
  - Hopfield network
  - BAM
- Hebbian learning, Competitive learning, Error Correct^{n}
- Backprop
  - Gradient descent rules → V.G.P
  - Supervised / unsupervised
  - feedback / feed forward

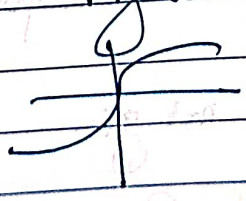
* BAM.



$y_i = x_i w_{ij}$ $\quad x_i = w_{ij} \cdot y_i \cdot T$

	Sigmoid	Tanh	Relu
Range	0-1	-1 to 1	0 to ∞
Diff	Y	Y	Y
Nature	Non linear	Non-linear	Linear
Zero centred	N	Y	N
A.f			
Eq	$\frac{1}{1+e^{-x}}$	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\max(0, z)$
Accuracy	Good	Very Good	Excellent



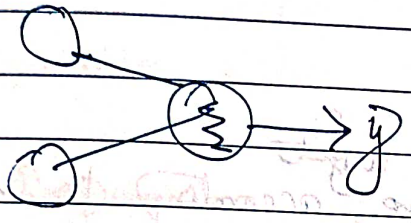


reconstructing input data Auto Associative memory	associating different types of i/p data. Hetero Associative memory
(25) Capable of retrieving information piece of data	
(1) Capable of retrieving partial information from piece of data.	Capable of retrieving one category data from another category.
(2) input & o/p vector are same.	input & o/p vector are different
(3) retrieves same pattern	retrieves stored pattern
(4) Recalls memory of same modality eg. Color Correction	Recalls memory of different Character from I/P PCA

Supervised	Unsupervised
Desired o/p given	not given
Use high level of learning.	High level of learning is achieved
Use of training data	No use

Date
Page

Introduction to ANN:



Use of ANN → ① Robotic ② Automatic car driving
③ Recognizing Visual Objects

History of NN:

i. 1943 mc Culloch & walter pitts

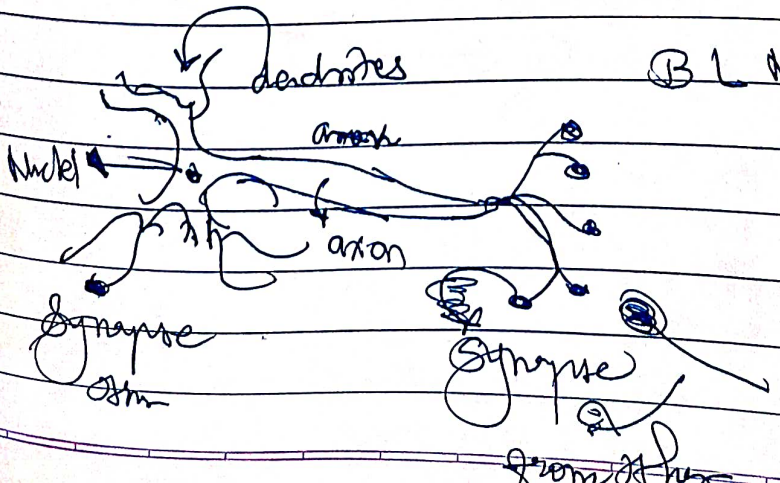
ii. Pitts Culloch wrote paper for describing ANN using brain biological structure.

iii. Design Electrical Circuits

iv. Frank Rosenblatt 1958 perceptron

v. ADALINE & MADALINE

vi. Backprop 1986.



Date
Page

* Properties of NN:

Non linearity:

adaptivity → change weight according to surrounding.

Contextual information → knowledge in structured form

V-LSI implementation → high efficiency.

* Application of NN

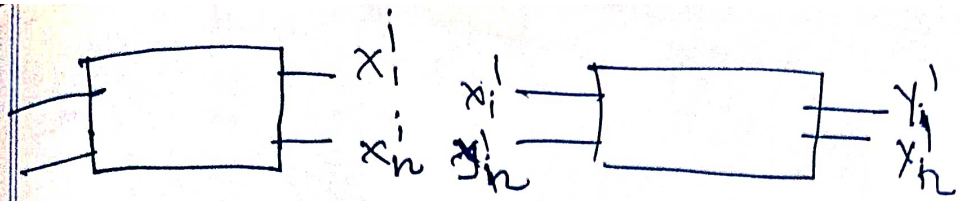
① Clustering ② Classification ③ Function Approx

④ Prediction.

* MP model

* Perceptron
  - SLP
  - MLP

* Basic learning law. → weight & bias



- Associative memory:

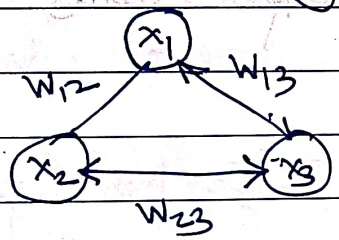
  Static :

  Dynamic :

  Hop-field model & BAM used Associative memory.

- Hopfield Network: Single layer RNN, No training

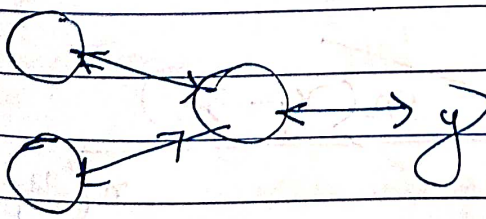
  Weights are initialized properly.



Discrete Hopfield model

Continuous Hopfield model.

Weights are trained in one shot fashion.



$y_i = x_i w_{ij}$

$x_i = y_i w_{ij}^T$

Date
Page

Competative learning Neural Network.

- Outputs Units are said to be Competitors for input patterns.
- During training, the output Unit that provides the highest activation to a given input is declared the winner and is moved closer to input pattern, whereas the rest of the neurons are left unchanged
- Building Competative LR

① Condition of winner / to be a

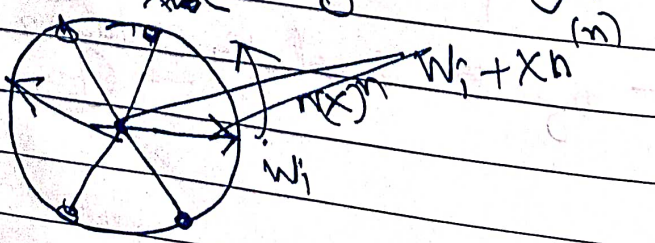
$$y_k = \begin{cases} 1 & v_k > v_j \quad , j \neq k \\ 0 & \text{otherwise} \end{cases}$$

② Condition of sum of total of weight:

$$\sum_k w_{kj} = 1 \quad \forall k.$$

③ Change of weights for winner

$$\Delta w_{kj} = \begin{cases} -\alpha (x_j - w_{kj}) & \text{wins} \\ 0 & \text{loses.} \end{cases}$$



$$g_i(x)^n = \sqrt{\sum_i (w_i - x_{i}^{(n)})^2}$$

Competitive learning Algorithm:

1. Normalize all input patterns.
2. Randomly select $(x)^n$ pattern.
   a. find winner neuron
   $$i = [W^{(n)} \cdot i]$$
   b. update winner neuron
   $$w_i = w_i + \eta (x)^n$$
   c. Normalize winner neuron
   $$w_i = \frac{w_i}{\|w_i\|}$$
3. Go to Step 2 until no changes occur in N epochs.

Date
Page

ART Network:

- It adapts new learning pattern without losing old pattern resonance.
- Vector classifies which accepts inputs vector and classifies it into one of the categories depending upon stored pattern resembles most.
- Addresses stability - plasticity.

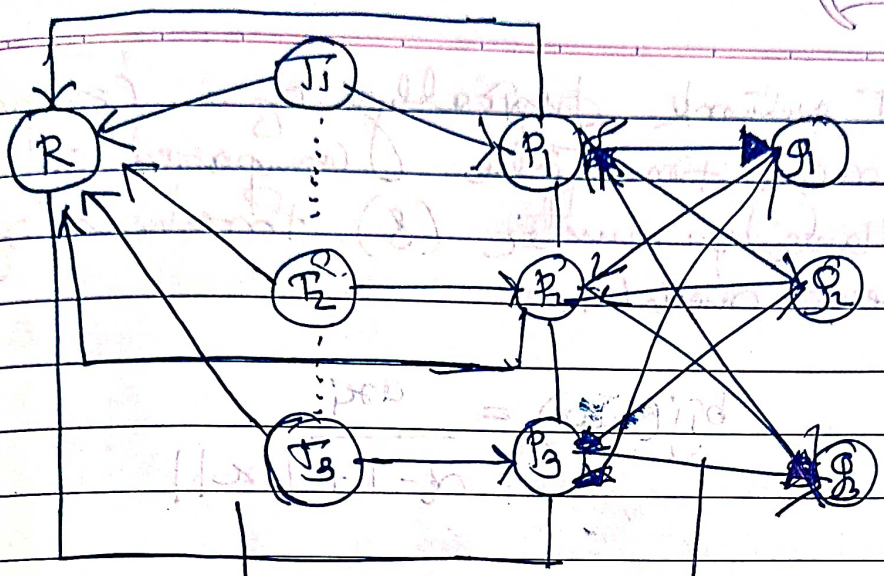
- Operations of ART-1 Network:

Recognition phase: Inputs vector is compared with classification presented at every node at output layer. It matches with classified one.

Comparison phase: Comparison of inputs vector is done to comparison layer vector.

Search phase: It will search for resets as well as match done in the above phases. If no resets match is good enough classify, otherwise repeated for the other stored patterns must be sent to find correct match

ART1 layer 1.



4(a) layer inputs portion

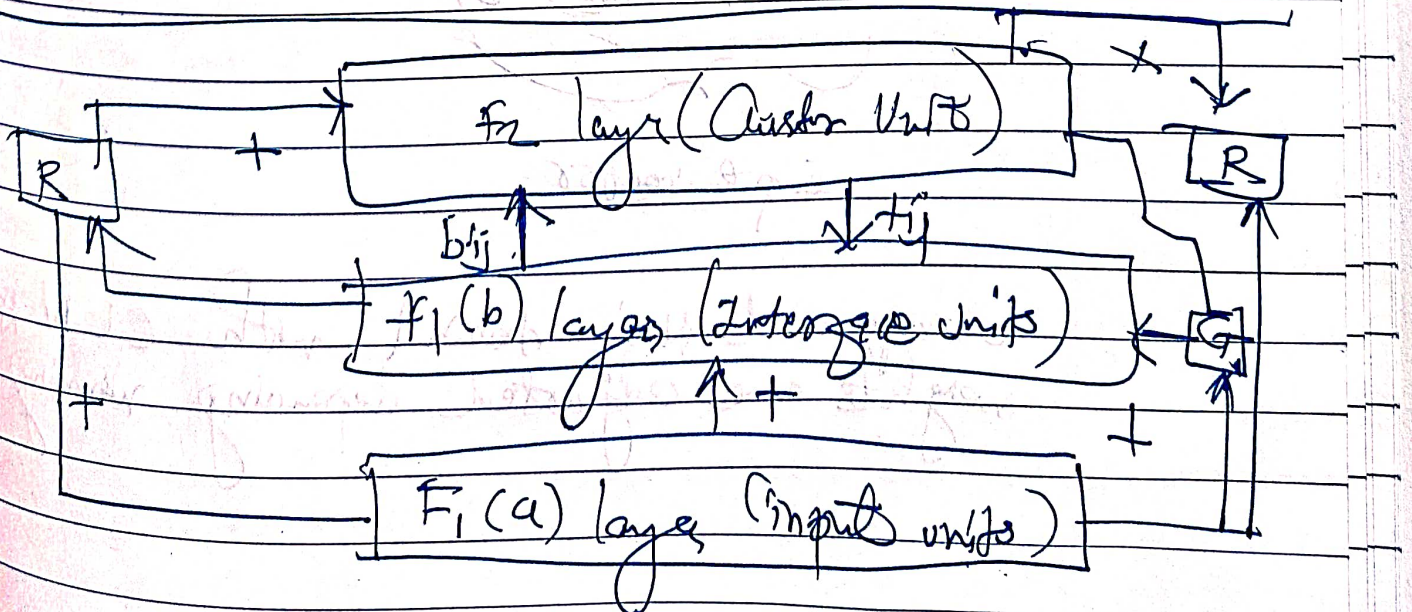
4(b) layer interface portion

4 (c) layer Cluster units.

No processing input vectors connected to F_1 b layer.

F_1 b connected to F_2 layer through bottom up weights b_{ij}

F_1 b layer through top down t_{ji}



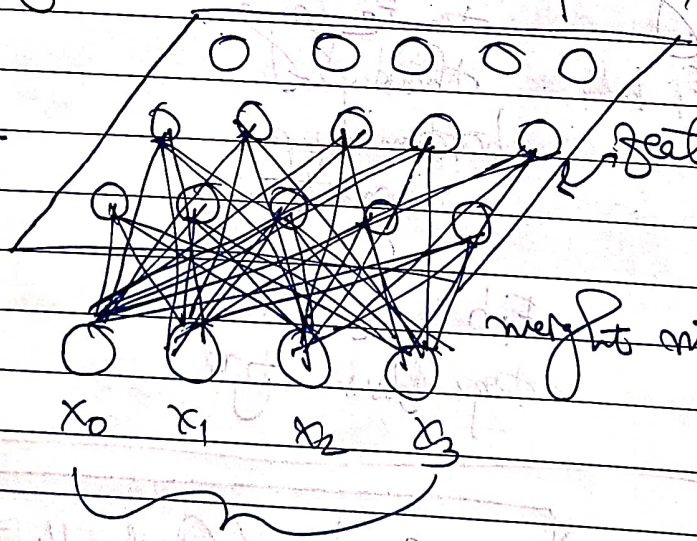
- ART Network typically of a comparisson field & recognition field composed of neurons, a vigilance parameter (ρ) threshold of recognition & reset module.

$$b_{ij}(new) = \frac{ax_i}{a-1+||x||}$$

$$t_{ij}(new) = x_i$$

- Self Organizing Maps (SOM):

This use neighborhood function to preserve the topological properties of input space.



Unsupervised NN, trained with CLN Algorithm weights are adjusted depending winning neuron.

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

Date
Page

Learning Vector Quantization :

- ANN algorithm that allows to choose no. of training instances to suspend & know exactly what those eg: look like.
- LVQ is collection of codebook vectors. They are randomly selected at beginning & are suitably for optimally summarizing the training data sets in multiple iterations of learning algorithm. Codebook can be used K-nearest neighbour.

Step 1 : Initialize weights vectors to the first m training vectors, where m is number of different categories.

Step 2 : While stopping condition false, do Step 3 to 6.

Step 3 : for each training input vector x, do step 4 to 4.

Step 4 : Find J so that D(J) is minimum.

update the weights of J neuron.

If $T = C_j$ THEN,

$W_j(new) = W_j(old) + \alpha(x - W_j(old))$.

Step 5 : Reduce α

Step 6 : Test stopping Condition :

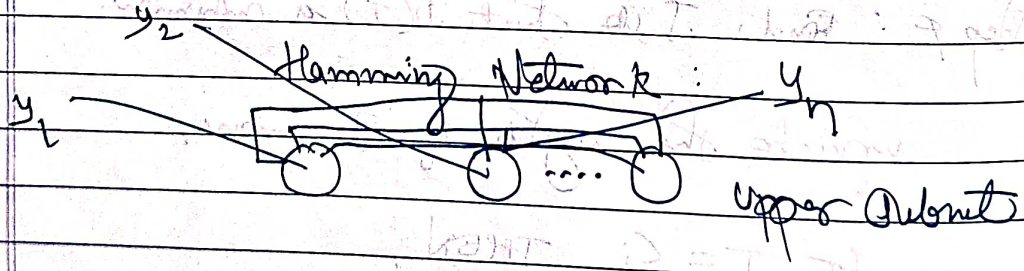
UNIT- 6.

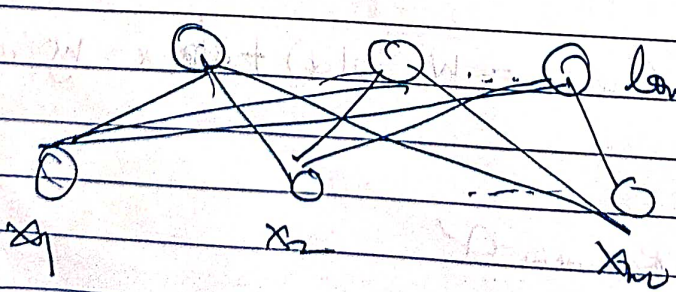
Appln of ANN.

* Pattern Classification.

- NN is best of P.C due to its ability of multilayer feedback NN to form complex D.R. in pattern space.
- Many pattern recognition, explicitly character, Number, symbols, word, vowel recognition.
- limitations are : if character, vowel are highly deformed, transformed, degraded it is hard to process.
- Typically are B & W pixels 128×128 points and 16×16 elements array.

Pattern Classification Process:





(1) Hamming Network is the maximum likelihood classifier for binary inputs & it proposes correlation matching between input & stored pattern.

(2) It consist of 2 subnets lower of M (128 × 128) input units, N output units, Corresp to N pattern classes.

(3) In lower subnets the connection weights between the input & output units is constant.

$$w_{ij} = \alpha_{ij}/2 \;, \qquad \theta_i = M/2$$

(3) In upper subnets, weights are fixed such that outputs units inhibit each other,

$$\gamma_{kl} = 1 \;, \quad k = l$$

$$\gamma_{kl} = -\varepsilon \quad k \neq l$$

(4) $$\delta_i = f\left[\sum_j w_{ij} x_j - \theta_i\right]$$

Date
Page

NETTALK- Convert English Text to Speech Application.

① Speech Signal: It is time varying vocal tract system excited by time-varying excitation signal.

② Articulators such as tongue, jaws, lips.

③ Vocal tract system is described by acoustic features such as frequency response, resonance & anti-resonance of the system.

④ ③ types of vocal system:

ⓐ Voiced Source. ⓑ Unvoiced Source

ⓒ Plosive Source.

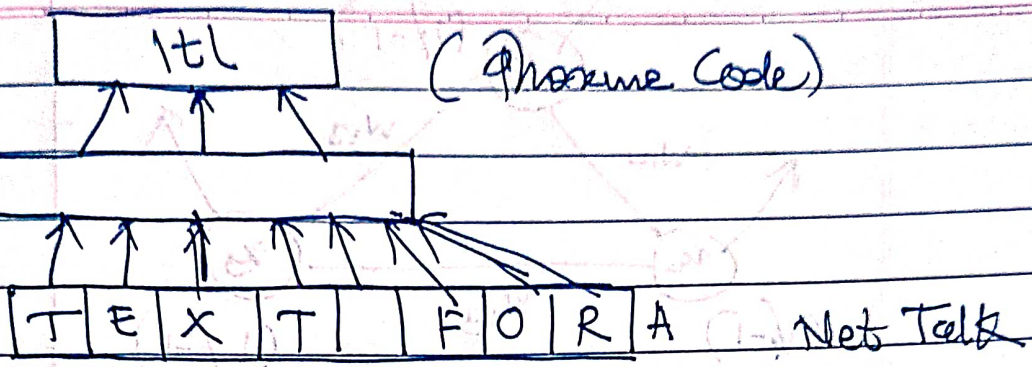
⑤ Segmental Features: Speech signal in short interval like 10-20 ms. speech

⑥ Speech is a message:

ⓐ Speech recognition:

ⓑ Speech synthesis:

ⓒ Speaker Identification:



Net Talk working

(1) It MLF network, developed generate pronunciation units or phonemes phenemes code from an input text.

(2) Phoneme Code is presented to inputs of the a speech synthesizer to produce speech corresponding to text.

(3) The network consists of an input layer 7×29 binary units, corresponding to 7 inputs (text characters) with punctuation marks, 1 hidden layer, 80 units, one output layer with 26 units.