# Deep learning:

AI - Artificial Intelligence:

→ Simulation of Human Intelligence in Machine. that are programmed think and act like Human.

→ Learning
→ Reasoning
→ Problem-solving.

Machine learns from experience.

## Machine Learning:

→ Subset of AI
→ imitate the way that human learn. gradully improving Accuracy.

Types:

→ Supervised
→ Semi-supervised.
→ unsupervised.
→ Reinforcement

Application of DL:

→ Translation between languges
→ Voice Recognization.
→ Handwriting Recognition.
→ context understanding.
→ customer support.
→ Medical care.
→ Self Driving cars

## Perceptron:

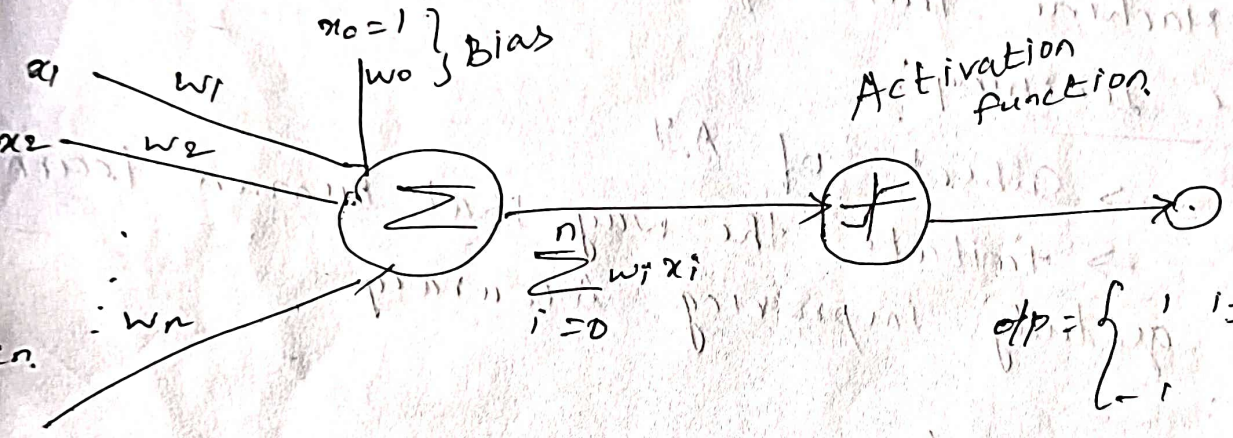
(ANN)

→ It used to build a ANN s/m.

→ Takes real world inputs → find linear combination of outputs

$$output(x_1 \dots x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n > 0 \\ -1 & \text{o.w} \end{cases}$$

$w_i$ → Real-valued constant




$$o/p = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 & \text{o.w} \end{cases}$$

→ Weights are changes → modify the perceptron weights.

→ Weights modified each step According to perceptron training Rule.

→ The process Repeated until the perceptron classifies all traing examples correctly.

→ $w_i$ associated with $x_i$ according to the Rule.

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \eta (t - o) x_i$$

## Machine Learning:

Subset of AI

It builds a model to learn the pattern from the structured data.

**Limitations:**

High Dimensionality of Data.

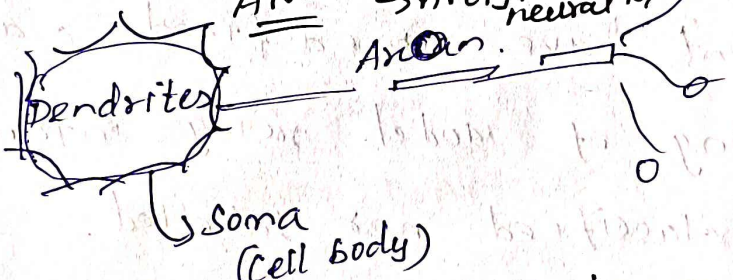
Not able to solve NLP

Big challenges in feature extraction.

complex problems.

## Deep learning: *Neural Network with Human Intervention

ANN → Artificial neural N/w




* Training the Machine by huge Data.
* It is the complex Neural N/w, which Learns from the given unstructured data and predict the O/P

Dendrites → Input layer

Soma → Neuron.

Axon → Interconnection layer

Synaps → neuron in next layer.

Bias → Added with sum of inputs:
→ To shift the activation function towards positive and Negative side.

**Limitations of DL:**

① Data: Massive amount of unstructured Data

② computational power: Training NN requires GPU → Graphical Processing unit. It is expensive

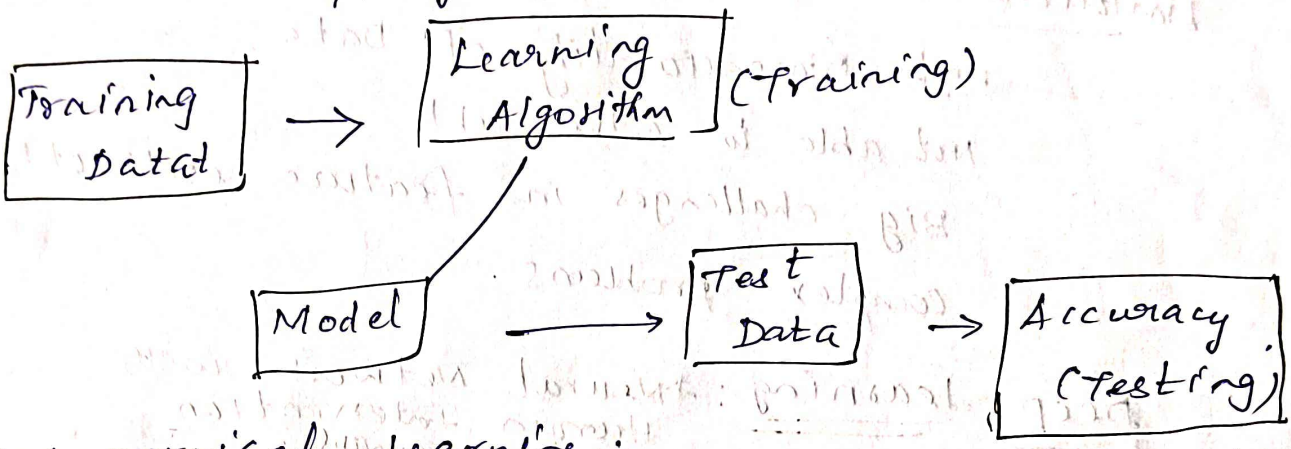
③ Training Time: It increased due to number of layers in Network

① Supervised learning:

→ input variable

→ output variable

Learn the mapping function - Algorithm (I/p to o/p)




② Unsupervised Learning:

It don't have predefined classes

Training of model using information that neither classified nor labelled.

Data

class 1 — High similarity.

class 2. — Low-class similarity.

Ex Marketing:

Reinforcement Learning:

→ It learn from past experience.

→ Learning interacting with an Environment

Types of Learning :

Supervised Learning
Unsupervised "
Memory based Learning
Hebbian "
Competitive "
Boltzman "
Error correction "

3 Memory Based Learning :

* classified based upon similarities with already learned Training and Test Data

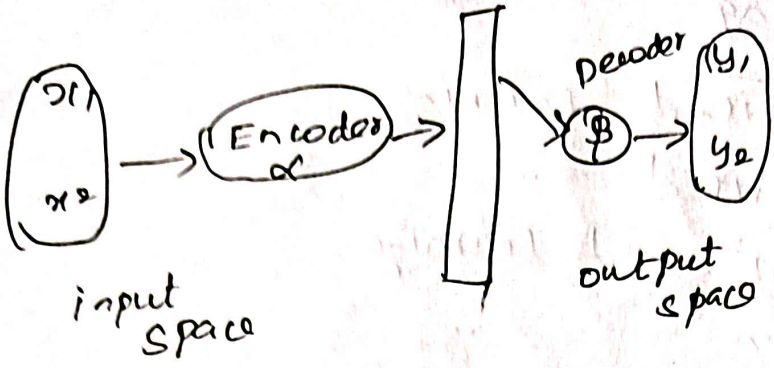
→ Instance Based Learning

→ Non parametric
No stronger pre determination
No fixed number of parameters

→ Identified based on similarities to previously seen concepts

→ compares the relevant features of data elements and calculate similarity Rate.

storage:
→ Traing Data size is large it becomes costly and difficult to store and handle the Data




Essential factors:

① criteria that is used for determining the local neighbourhood of input value.

② Type of Learning Rule

Ex KNN

## ⑦ Hebbian Learning:

Hypothesis based Learning.

simplest NN (Threshold neuron)

Donald O-Hebb proposed the Mechanism to update the weights between neurons

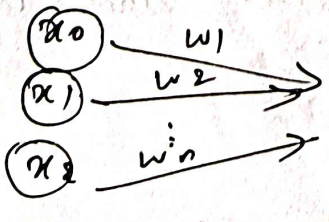
Weight updation enables the neuron to learn.

$$\Delta w \propto X \cdot Y = \Delta w = \beta x y$$

Learning Takes place;

→ Repeated activation of weakly connected neurons

→ change the strengths and pattern of weights, Leading to stronger connections




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

It has

① Input layer

② Association layer

③ Response layer

## Hebbian Learning:

Two kinds of Learning:

1. Parameter Learning:

changing and updating the connecting weights in NN

2. Structure Learning:

changing the structure or Architecture of NN

Hebbian Learning: (Hebb's Postulate of Learning)

Rule:

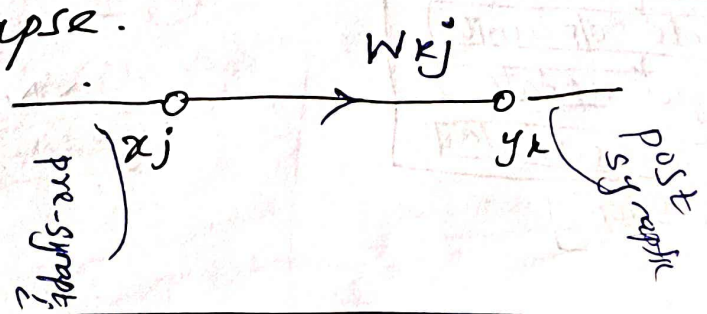
① If two Interconnected neurons on either side of synapse activated at same time (synchronously)

Synaptic weight between them should be increased.

② If two neurons on either side of synapse are fixed (or) Activated at different Time (Asynchronously)

weight of the synaptic connections is decreased.

That synaps is called as Hebbian synapse.



Hebbian Learning is mathematically expressed as,

$$\boxed{\Delta W_{kj}(n) = \eta\, y_k(n)\, x_j(n)}$$

$x_j$ - Pre-synaptic signal

$y_k$ - Post-synaptic signal.

$\eta$ - Learning Rate

$\Delta W_{kj}$ - change in synaptic weight

Competitive Learning:

It is the easiest learning Algorithm

The Learning occur due to change in synaptic gap.

1949 - Donald Hebb

unsupervised Learning.

Applications:

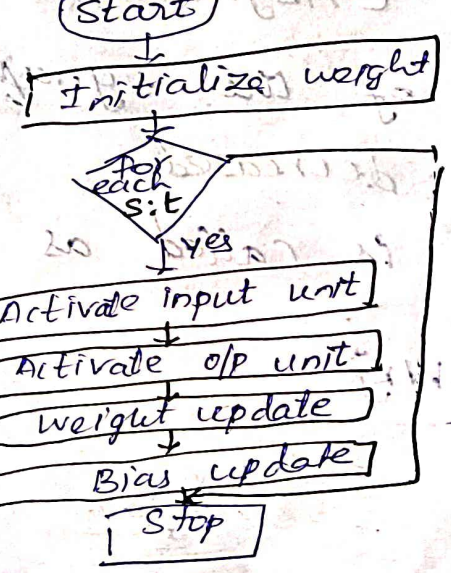
Pattern association, categorization and classification.

$$w_i(new) = w_i(old) + \underbrace{x_i y}_{\text{changes in the synaptic}}$$

$x_i$ - input vector

$y$ - output vector

Flowchart:



## Training Algorithm:

Step 1 : Initialize the weights & Bias

Step 2 : For each training i/p & o/p pairs perform step 3-5

Step 3 : $x_i = s_i$ (Activation)

Step 4 : $y = t$ (Activation)

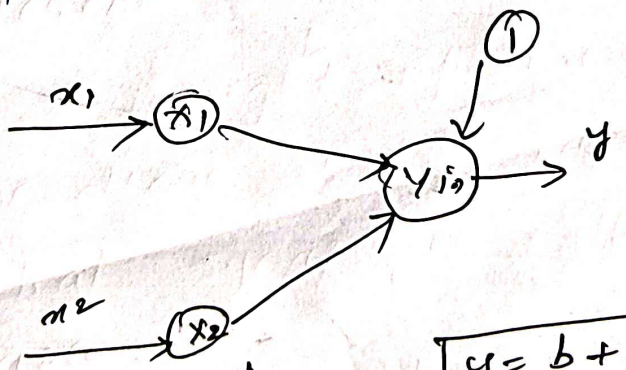
Step 5 : weight updation:

$w_i(new) = w_i(old) + x_i y$

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

Design Hebb network to implement the logical AND function. use bipolar inputs and outputs:

| Inputs | | | Targets |
|---|---|---|---|
| $x_1$ | $x_2$ | b | y |
| 1 | 1 | 1 | 1 |
| 1 | -1 | 1 | -1 |
| -1 | 1 | 1 | -1 |
| -1 | -1 | 1 | -1 |




$b(new) = b(old) + \Delta b$

$\boxed{\Delta b = y}$

$\boxed{y = b + x_1 w_1 + x_2 w_2}$

$\boxed{w_1 = w_2 = b = 0}$

$\boxed{w_i(new) = w_i(old) + \Delta w}$

$\boxed{\Delta w = x_i y}$

| Inputs | | | | weight changes | | | New weight | | |
|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_2$ | $b$ | $y$ | $\Delta w_1$ | $\Delta w_2$ | $\Delta b$ | $w_1(new)$ | $w_2(new)$ | $b(new)$ |
| | | | | $x_1 y$ | $x_2 y$ | $y$ | $w_1(old) + \Delta w_1$ | $w_2(old) + \Delta w_2$ | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | $0+1=1$ | $0+1 = 1$ | $0+1 = 1$ |
| 1 | -1 | 1 | -1 | -1 | 1 | -1 | $1+(-1) = 0$ | $1+1 = 2$ | $1+(-1) = 0$ |
| -1 | 1 | 1 | -1 | 1 | -1 | -1 | $0+1 = 1$ | $2+(-1) = 1$ | $0+-1 = -1$ |
| -1 | -1 | 1 | -1 | 1 | 1 | -1 | $1+1 = 2$ | $1+1 = 2$ | $-1+-1 = -2$ |

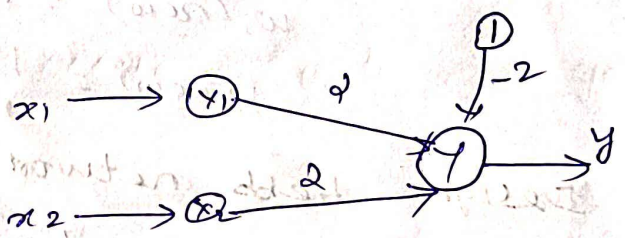
Final weights

$w_1 = 2$

$w_2 = 2$

Final Bias

$b = -2$




$$y = b + w_1 x_1 + w_2 x_2$$

(1,1)

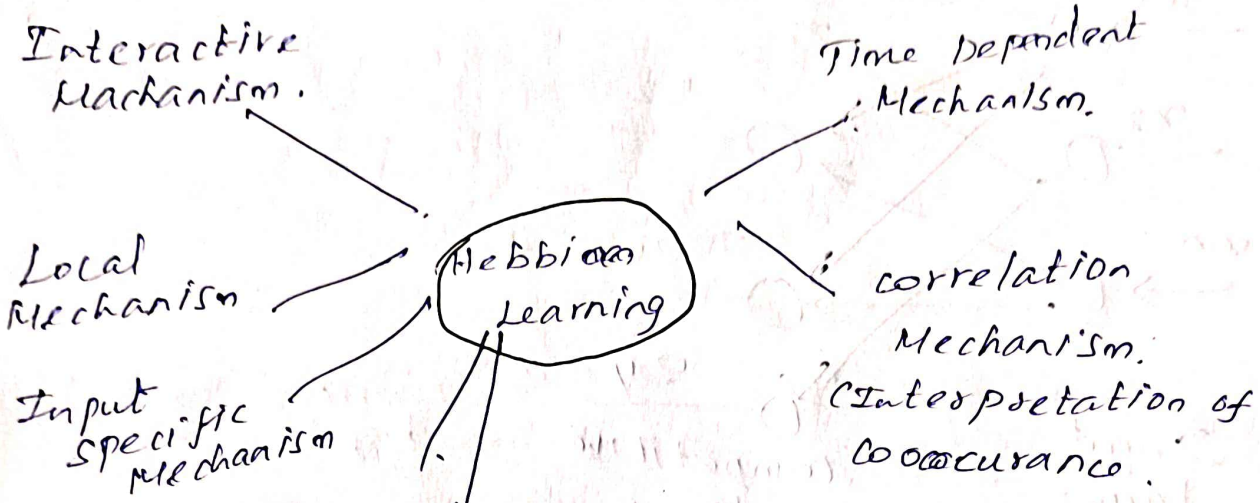
$$= -2 + 2(1) + 2(1)$$

$$= -2 + 2 + 2$$

$$\boxed{y = 2}$$

(4)

## Hebbian learning Mechanism:




I → If 2 neurons present on two sides connections are activated synchronously strength of connections increases

II → Activated Asynchronously → strength Decreases

5) competetive Learning:

It forms unsupervised learning

Winner Takes All

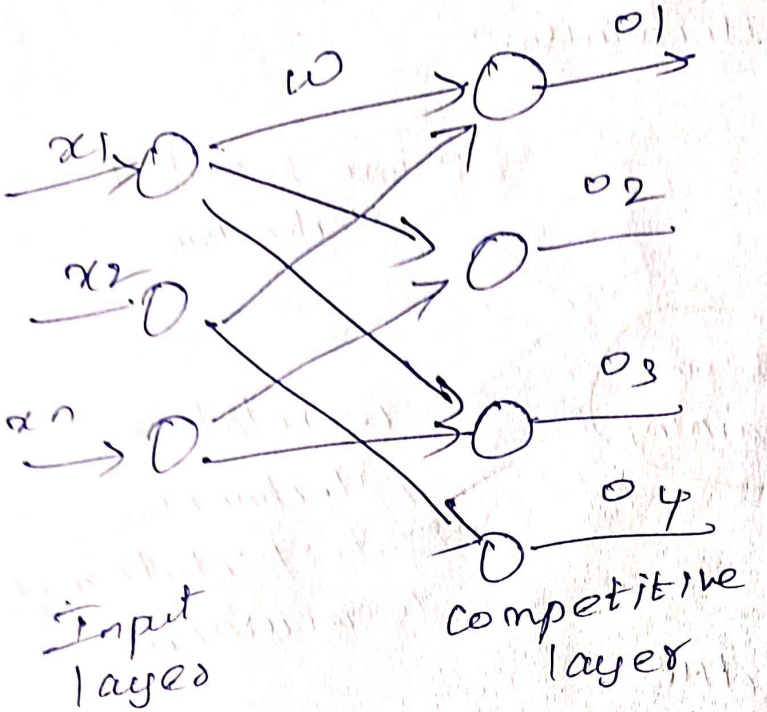
o/p node compete with each other to represent the input pattern

winner is declared According to the node having most output -

→ Remaining given as "0"

only one winner is activated

It update the weights, the Rest Remain changed.




Fundamental components:

① Group of neurons are similar
synaptic weights are Randomly distributed
→ Respond differently

② Limitation on each neuron's capacity

③ Allow the neuron to compete for privilage to Respond inputs.

Local variable "k" for Input pattern "x"

The output of $k^{th}$ winning neuron

$y_k \Rightarrow 1$

Remaining All 0

$$y_k = \begin{cases} 1 & \text{if } V_k > U_j \text{ for All } i, j \neq k \\ 0 & \text{o.w} \end{cases}$$

$$\Delta w_{kj} = \begin{cases} \eta (x_j - w_{kj}) & \Rightarrow x \\ 0 & \Rightarrow \text{if neuron } k \text{ losses the competition} \end{cases}$$

$V_k$ - net Input of $k^{th}$ neuron

Competitive Learning:

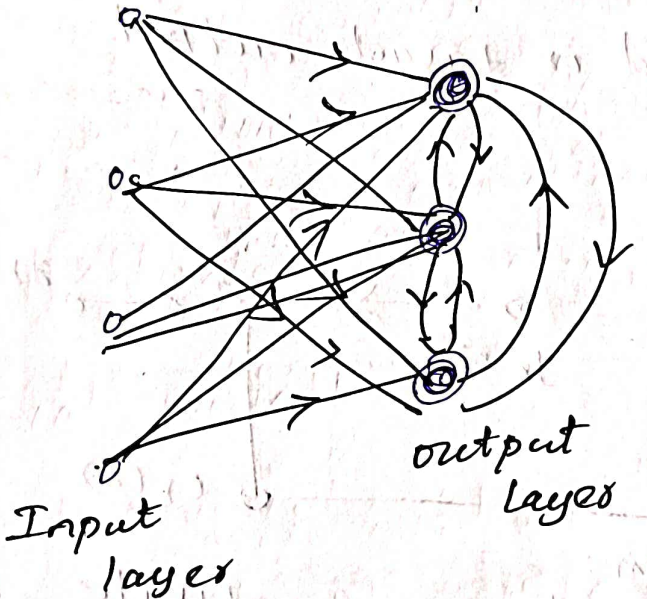
Neural Network is stimulated by the Environment.

Experience changes

NN consist of single layer of output neurons

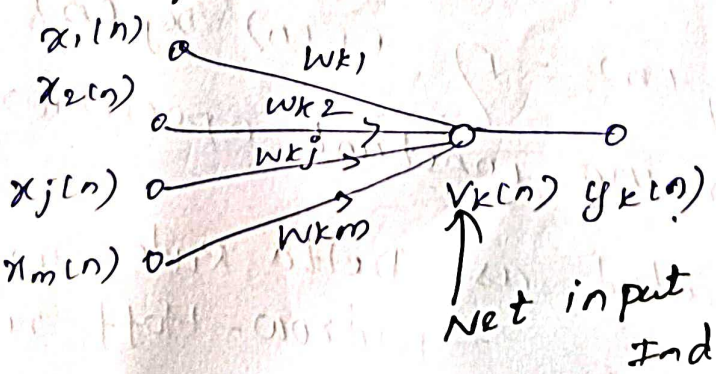
All output neurons are fully connected to input neurons.

All output neurons compete against each other for Activated




Maximum net input neuron is the winning neuron.

Winner takes All




Net input (or) combined input
Induced local field.

Winning Neuron:

output neuron to win the competition the Net input should be Maximum among all other neurons.

o/p of winning Neuron set to "1"

Remaining o/p neuron set to "0"

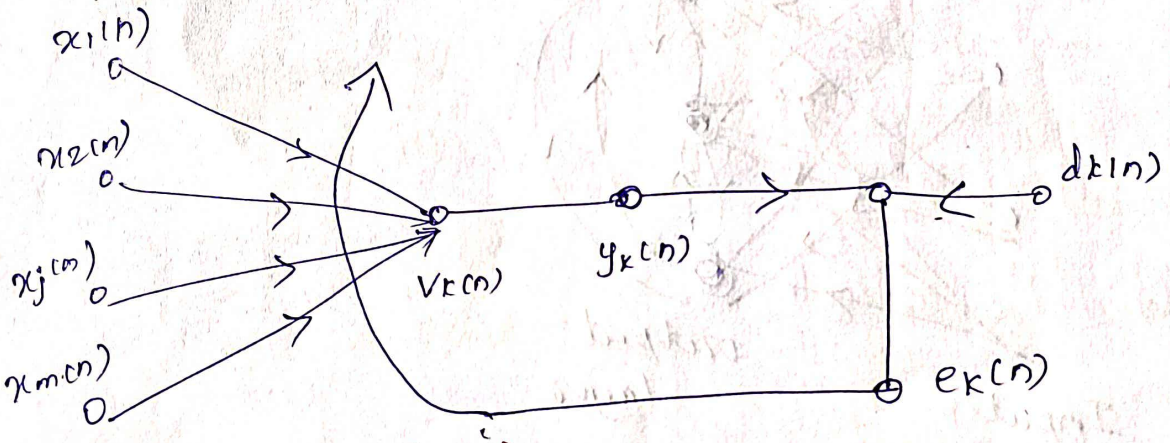
Error correction Learning: (Parameter Based Learning)

Input and output layers of neuron

Input signal

comparator

Desired or target output




Control Mechanism:

Error signal $e_k(n)$ activate control Mechanism.

Change in synaptic weights

$$\Delta w_{kj}(n) = \eta \, e_k(n) \, x_j(n)$$

$\eta$ - Learning constant (positive) value

It is called as Delta Rule (or) Widrow-Hoff rule

## Error correction Learning:

used with Supervised Learning

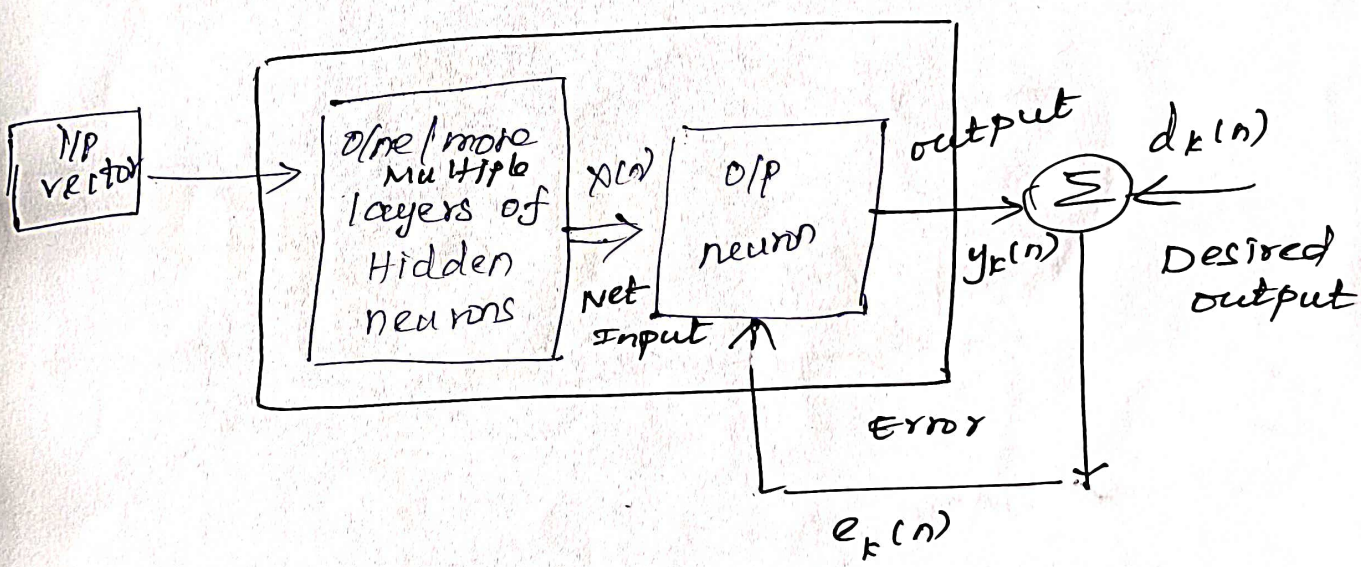
Comparing the system output and desired output

Types:

→ Backward error correction
→ Forward error correction.

## Backword error correction:

Adjusting or correcting the weights to Reached the Minimised loss function




Multi layer feed forward Network

The Error values can be used to directly adjust the tap weights using an Backward propogation Algorithm.

$$\boxed{e_k(n) = d_k(n) - x_k(n)}$$

$d_k(n)$ → Desired O/P
$x_k(n)$ = Actual O/P

$$e = d - y$$

K → controlled by signal vector X(n)
$y_k(n)$ denotes the feedback signal of neuron "k"

Back propogation passes error signals backward through the N/w during Training to update The weight of Network

Bidirectional Dataflow - Training.

Learning Rate:

Not to large :→ oscillate

not to Small → Long Time to converge.

Ex

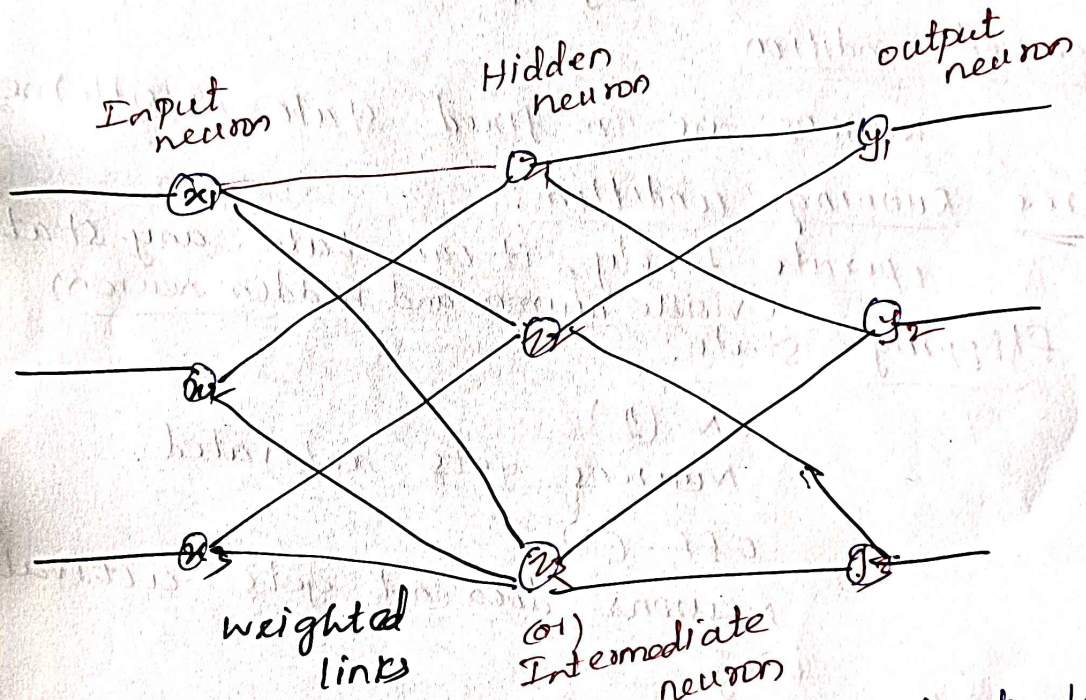
control system Applications

Boltzmann Learning:

Stochastic Learning Algorithm.

By Ludwig Boltzmann.

Neural Network based on Boltzmann Learning called Boltzmann Machine.




NN have Input layer, output layer and multiple hidden layers.

Neurons operate in Binary Manner

ON state → denoted by +1

OFF state → denoted by −1

It operates by Randomly choosing a neuron, and flipping its state.

Till the error becomes "zero", state of changes going on

Boltzmann Machine characterized by Energy function

$$E = \frac{-1}{2} \sum_{j} \sum_{k} w_{kj} x_k x_j$$

where $j \neq k$

$x_k \rightarrow$ state of neuron k

① Clamped condition

Neurons are in fixed state. (visible) neuron)

② Free Running condition:

operate freely, it can take any state. (visible neuron and hidden neuron)

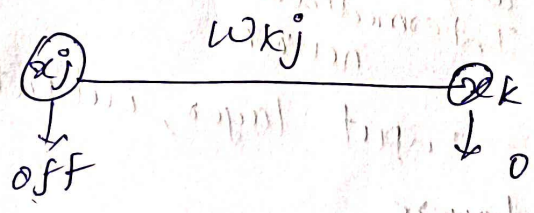
Flipping state:

ON-(1)

Neurons gets activated.

OFF-(-1)

Neurons does not gets activated.




probability of state change:

$$P = \frac{1}{1 + \exp\left(\frac{-\Delta E_k}{T}\right)}$$

$\Delta E \rightarrow$ change in energy

$T \rightarrow$ constant

① Restricted BM $\rightarrow$ removes the link b/w neurons in same layer which makes the training easier

② unrestrited BM

$\rightarrow$ Difficult problem in training BM

- no practical use.

Learning Tasks are:

1. Pattern Association
2. Pattern Recognization and function approximation
3. control and filtering
4. Beam forcing
5. Memory
6. Adaptations.

1. Pattern Association:
   1. Auto Association
   2. Hetero Association

Hetero Association:

pairs arbitrary
→ Input pattern to output pattern

supervised learning is used.

Let,

$x_k$ → key pattern

$y_k$ → Memorized pattern

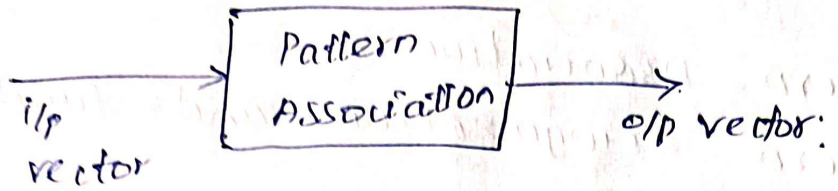
pattern Association is

$$x_k \rightarrow y_k \quad k = 1, 2 \cdots q$$

$x_k$ → Specifies where the Memorized pattern are stored → key to retrieving it

In Auto association $y_k = x_k$ → Input & o/p space having same dimensionality

## Learning Task: Pattern Association:




Association Memory is collective Memory that continuously learns through association

Association Memory has long been Recognized - character of Human Memory

Two Types:

Auto Association

Hetero "

① Auto Association:

NN - save and store a series of patterns in auto association repeatedly introducing them into network

It is given to Partial Explanation or noisy version of initial Pattern That it has stored.

Goal is to Recover/Recall the Pattern.

unsupervised learning

In Hetero Association:

The output may/may not be equal to input space.

Association Memory - phases:

1. storage phase
   → Traing phase.
2. Recall phase
   → Response phase
   → Retrieving Memorized patterns

2. Pattern Recognization and function optimization:

Pattern Recognization:

→ It is the process of where a received pattern (or) signal is assigned to one of the prescribed number of classes.

- Training
- New pattern → Assign to new class

Where same type of patterns exists.

Two parts:

Feature Extraction - unsupervised

classification - supervised.

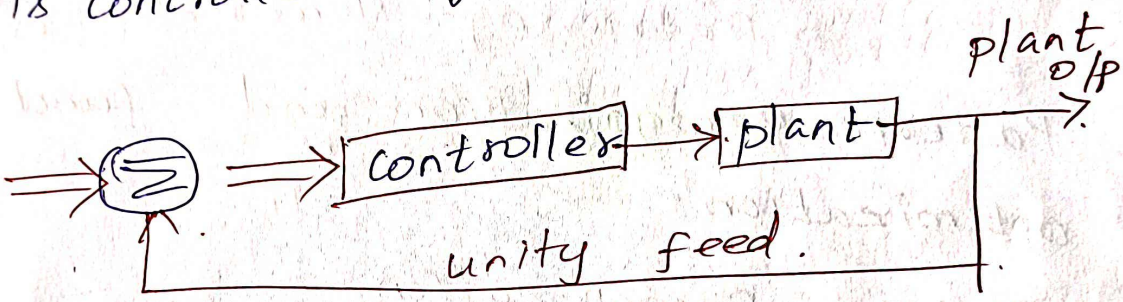
Function Approximation:

$$\boxed{d = f(x)}$$

$f \rightarrow$ function that maps input to output.

3) Control and filtering:

plant $\rightarrow$ critical parts of the system that is controlled by certain condition




Direct learning.

Indirect learning.

4) Filtering:

System/Algorithm to retrive information about a specified quantity of interest from a collection of noisy data.

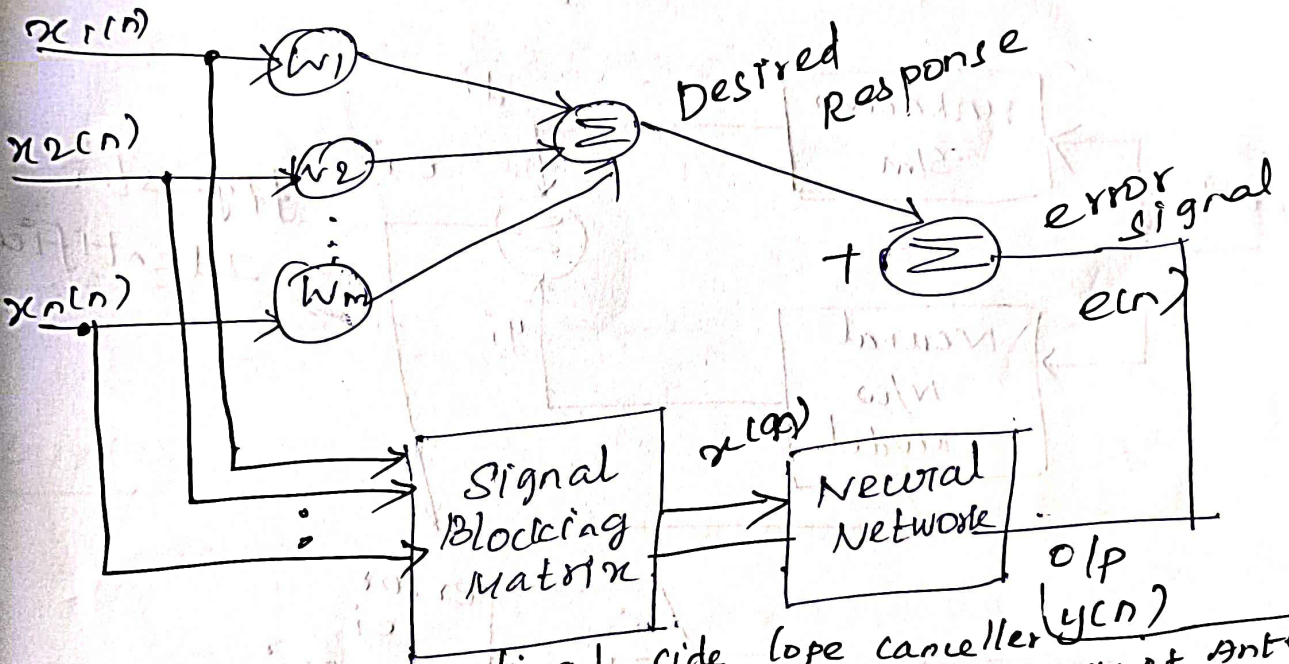
① Filtering

② Smoothing – A statistical approach of eliminating outliers from Dataset to make the pattern

③ Prediction.

5. Beamforming:

* Type of spatial filtering
* used to discriminate between target signal to spatial and background noise.
* used in Radar and sonar systems to identify the monitor a point of interest in the Receiver Noise.

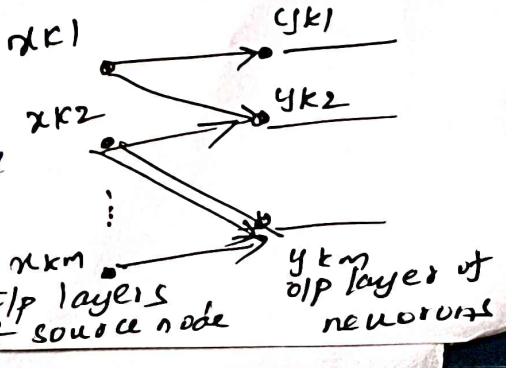



GSLC - generalized side lope canceller
→ Array of Antenna elements
→ Linear combiner
→ Signal Blocking Matrix

→ It is more difficult
→ signals comes from unidentified source.
→ there is no priori information.

6) Memory:

Based on Time frame.
① short term Memory
② Long term Memory
Associative Memory



Adaptations:

→ Learning in theoritical environment

→ " uncontrolled Environment

* Train the N/w
  ↳ small window - Pseudo-stationary
* import new data sample : Refresh the window.
* Retrain the N/w
* Repeat on frequency basis.

Pattern Recognition and function approximation.

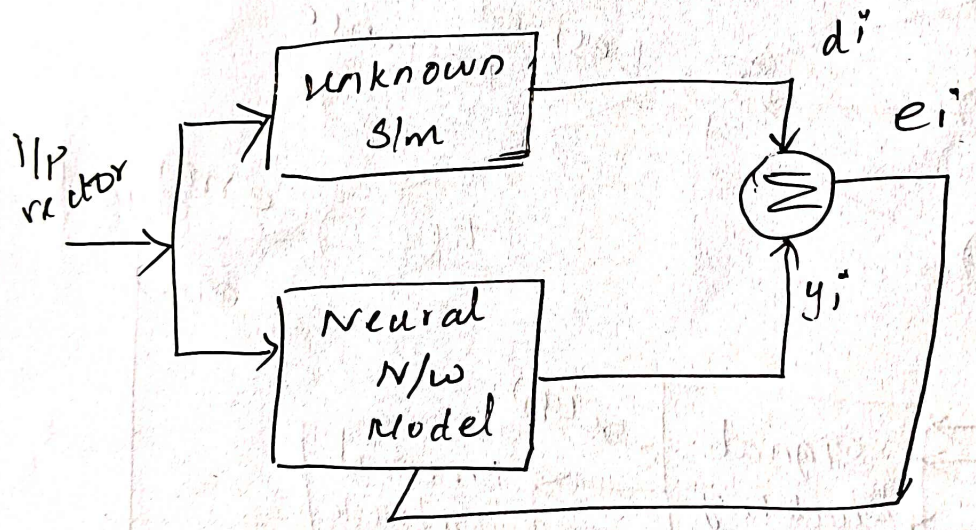



fig. S/m Identification.

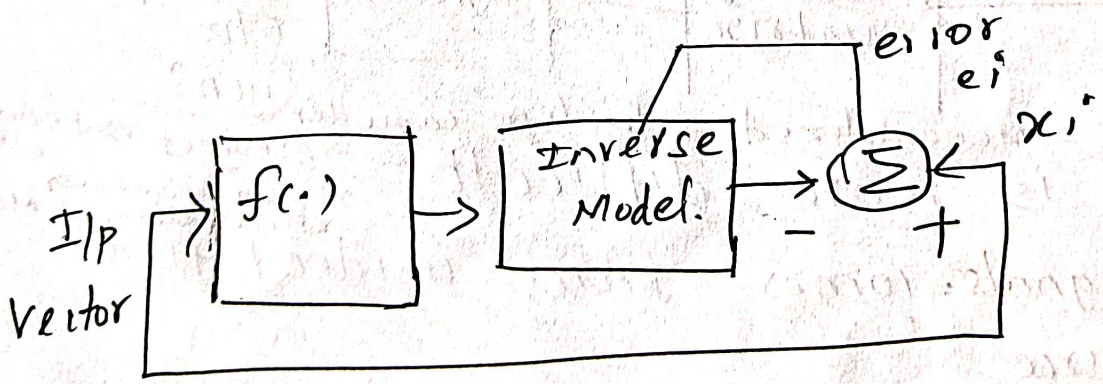
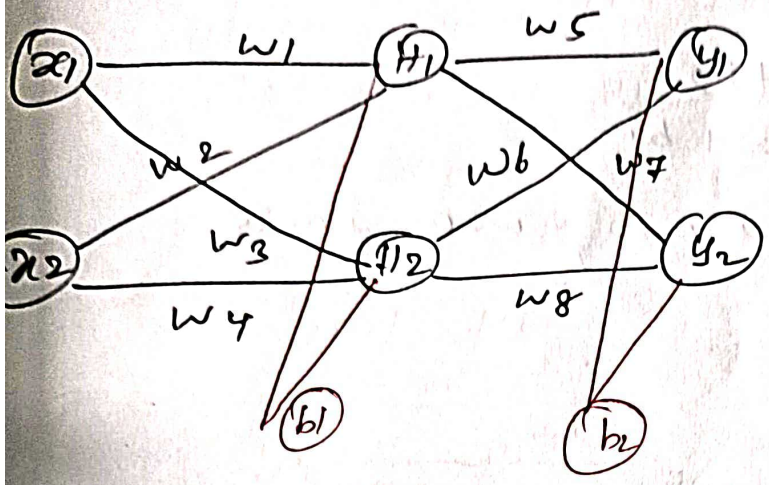



Fig. Inverse S/m

Back propogation problem (13)




Solution:-

$$H_1 = x_1 w_1 + x_2 w_2 + b_1$$

Activation function is sigmoid $= \frac{1}{1+e^{-x}}$

at $H_1$

$$\text{out } H_1 = \frac{1}{1-e^{-H}}$$

Given:

$x_1 = 0.05$ | $b_1 = 0.35$ | Learning rate $= 0.5 = \eta$

$x_2 = 0.10$ | $b_2 = 0.60$

Target values : $T_1 = 0.01$

$T_2 = 0.99$

Final weights:

$w_1 = 0.65$

$w_2 = 0.20$

$w_3 = 0.25$

$w_4 = 0.30$

$w_5 = 0.40$

$w_6 = 0.45$

$w_7 = 0.50$

$w_8 = 0.55$

Forward pass:

calculate out $H_1$, out $H_2$

$H_1 = x_1 w_1 + x_2 w_2 + b_1$

$= 0.05 * 0.15 + 0.10 * 0.20 + 0.35$

$\boxed{H_1 = 0.3775}$

$$\text{out } H_1 = \frac{1}{1+e^{-H_1}} = \frac{1}{1+e^{-0.3775}}$$

$$\boxed{\text{Out } H_1 = 0.593269992}$$

$H_2 = x_1 w_3 + x_2 w_4 + b_1$

$= 0.05 * 0.25 + 0.10 * 0.30 + 0.35$

$$\text{out } H_2 = \frac{1}{1+e^{-H_2}} =$$

$$\boxed{\text{out } H_2 = 0.596884378}$$

Calculate out $y_1$, out $y_2$:

$y_1 = \text{out } H_1 * w_5 + \text{out } H_2 * w_6 + b_2$

$= (0.4 * 0.593269992) + (0.596884378 * 0.45) + 0.6$

$y_1 = 1.105905967$

$$\text{out } y_1 = \frac{1}{1+e^{-y_1}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

$$\boxed{\text{out } y_1 = 0.75136507}$$

$$\text{out } y_1 = \frac{1}{1+e^{-y_1}} = 0.75136507$$

$$y_2 = \text{out } h_1 * w_7 + \text{out } h_2 * w_8 + b_2$$

$$= 0.59326992 * 0.50 + 0.596884378 * 0.55 + 0.6$$

$$=$$

$$\text{out } y_2 = \frac{1}{1+e^{-y_2}}$$

$$\boxed{\text{out } y_2 = 0.772928465}$$

Calculating the Error:

$$E_{Total} = \sum \frac{1}{2} (\text{Target} - \text{output})^2$$

$$E_{Total} = \frac{1}{2} (T_1 - \text{out } y_1)^2 + \frac{1}{2} (T_2 - \text{out } y_2)^2$$

$$= \frac{1}{2} (0.01 - 0.75136507)^2 + \frac{1}{2} (0.99 - 0.772928465)^2$$

$$= 0.274811033 + 0.023560026$$

$$\boxed{E_{Total} = 0.298371109}$$

$$E_{Total} = E_1 + E_2$$

$$E_1 = \frac{1}{2} (T_1 - \text{out } y_1)^2 = 0.274811033$$

$$E_2 = \frac{1}{2} (T_2 - \text{out } y_2)^2 = 0.023560026$$

...tion:

(182)

Part-2:

calculating Backward propogation Error

output layer → Hidden Layer

$w_5, w_6, w_7$ and $w_8$

First adjust the $w_5$

Error at $w_5$:

$$w_5^* = w_5 - \eta \frac{\partial E_{Total}}{\partial w_5}$$

$$\frac{\partial E_{Total}}{\partial w_5} = \frac{\partial E_{Total}}{\partial out\, y_1} \times \frac{\partial out\, y_1}{\partial y_1} \times \frac{\partial y_1}{\partial w_5}$$

Find:

$$\frac{\partial E_{Total}}{\partial out\, y_1}$$

$$E_{Total} = \frac{1}{2}(T_1 - out\, y_1)^2 + \frac{1}{2}(T_2 - out\, y_2)^2$$

$$\frac{\partial E_{Total}}{\partial out\, y_1} = \frac{1}{\cancel{2}} * \cancel{2}(T_1 - out\, y_1)^{2-1} * (-1) + [0]$$

$T_1 = 0.01$

$out\, y_1 = 0.75136507$

$$= -(T_1 - out\, y_1)$$

$$= -(0.01 - 0.75136507)$$

$$= 0.74136507$$

$$\frac{\partial E_{Total}}{\partial out\, y_1} = 0.74136507.$$

$$\frac{\partial y_{1out}}{\partial y_1} \Rightarrow y_{1out} = \frac{1}{1+e^{-y_1}}$$

$$\frac{\partial}{\partial y_1}\left(\frac{1}{1+e^{-y_1}}\right) \Rightarrow \frac{\partial}{\partial y_1}\left(1+e^{-y_1}\right)^{-1}$$

$$= (-1)\left(1+e^{-y_1}\right)^{-1-1} * e^{-y_1} * (-1)$$

$$= \left(1+e^{-y_1}\right)^{-2} * e^{-y_1}$$

$$\left(1+e^{-y_1}\right)^{-2} = \frac{1}{\left(1+e^{-y_1}\right)^2} = \left(y_{1out}\right)^2$$

$$\frac{\partial y_{1out}}{\partial y_1} = \left(y_{1out}\right)^2 * e^{-y_1}$$

$$y_{1out} = \frac{1}{1+e^{-y_1}}$$

$$1+e^{-y_1} = \frac{1}{y_{1out}}$$

$$e^{-y_1} = \frac{1}{y_{1out}} - 1$$

$$\boxed{e^{-y_1} = \frac{1-y_{1out}}{y_{1out}}} = \frac{1-0.75136507}{0.75136507}$$

$$\boxed{e^{-y_1} = \cancel{0.1868}\ 0.33091095}$$

$$\frac{\partial y_{1out}}{\partial y_1} = \left(y_{1out}\right)^2 * e^{-y_1}$$

$$= (0.75136507)^2 * 0.33091095$$

$$\boxed{\frac{\partial y_{1out}}{\partial y_1} = 0.18681560}$$

...tion :

$$\frac{\partial y_1}{\partial w_5} \Rightarrow$$

$$\frac{\partial}{\partial w_5} (h_{1out} * w_5 + h_{2out} * w_6 + b_2)$$

$$\frac{\partial y_1}{\partial w_5} = h_{1out}(1) + 0 + 0 = h_{1out}$$

$$\boxed{\frac{\partial y_1}{\partial w_5} = h_{1out} = 0.5932699 92}$$

$$\frac{\partial E_{Total}}{\partial w_5} = \frac{\partial E_{Total}}{\partial y_{1out}} * \frac{\partial y_{1out}}{\partial y_1} * \frac{\partial y_1}{\partial w_5}$$

$$= 0.74136507 * 0.186815601 * 0.5932699 92$$

$$\boxed{\frac{\partial E_{Total}}{\partial w_5} = 0.082167041}$$ change in $w_5$

updating the $w_5^*$

$$w_5^* = w_5 - \eta * \frac{\partial E_{Total}}{\partial w_5}$$

$$= 0.40 - (0.5 * 0.082167041)$$

$$\boxed{w_5^* = 0.35891648}$$

(16)

update $w_6$ : $w_6^* = w_6 - \eta \frac{\partial E_{Total}}{\partial w_6}$

$$\frac{\partial E_{Total}}{\partial w_6} = \frac{\partial E_{Total}}{\partial y_{1out}} \times \frac{\partial y_{1out}}{\partial y_1} \times \frac{\partial y_1}{\partial w_6}$$

Find : W.K.T

$$\frac{\partial E_{Total}}{\partial y_{1out}} = 0.74136507$$

$$\frac{\partial y_{1out}}{\partial y_1} = 0.186815601$$

Find $\frac{\partial y_1}{\partial w_6}$ =>

$$\Rightarrow \frac{\partial}{\partial w_6}(H_{1out} * w_5 + H_{2out} * w_6 + b_2)$$

$$\Rightarrow \frac{\partial}{\partial w_6}(H_{out} * w_5) + \frac{\partial}{\partial w_6} H_{2out} * w_6 + \frac{\partial}{\partial w_6} b_2$$

$$\Rightarrow 0 + H_{2out} * (1) + 0$$

$$\frac{\partial y_1}{\partial w_6} = H_{2out} = 0.772928865$$

$$\frac{\partial E_{Total}}{\partial w_6} = 0.74136507 * 0.186815601 * 0.772928865$$

$$= 0.1070495330$$

$$w_6^* = 0.45 - 0.5 * 0.1070495330$$

$$\boxed{w_6^* = 0.3964752} \approx 0.40866$$

update $w_7$ $\Rightarrow w_7^* = w_7 - \eta \frac{\partial E_{Total}}{\partial w_7}$

$$\frac{\partial E_{Total}}{\partial w_7} = \frac{\partial E_{Total}}{\partial y_{2out}} * \frac{\partial y_{2out}}{\partial y_2} * \frac{\partial y_2}{\partial w_7}$$

Find: $\frac{\partial E_{Total}}{\partial y_{2out}} = \frac{\partial}{\partial y_{2out}} \left[ \frac{1}{2} (T_1 - y_{1out})^2 + \frac{1}{2} (T_2 - y_{2out})^2 \right]$

$$= 0 + \frac{1}{\cancel{2}} \times \cancel{2} (T_2 - y_{2out})^{2-1} * (-1)$$

$$= (0.99 - 0.772928)(-1) = -0.217072$$

$$\frac{\partial E_{Total}}{\partial y_{2out}} = -0.217072$$

Find: $\frac{\partial y_{2out}}{\partial y_2} \Rightarrow \frac{\partial}{\partial y_2} \left( \frac{1}{1+e^{-y_2}} \right) \Rightarrow \frac{\partial}{\partial y_2} (1+e^{-y_2})^{-1}$

$$= (-1)(1+e^{-y_2})^{-1-1} * e^{-y_2} * (-1)$$

$$= \frac{1}{(1+e^{-y_2})^2} * e^{-y_2}$$

W.K.T $\frac{\partial y_{2out}}{\partial y_2} = (y_{2out})^2 * e^{-y_2}$

$$y_{2out} = \frac{1}{1+e^{-y_2}} \Rightarrow 1+e^{-y_2} = \frac{1}{y_{2out}}$$

$$e^{-y_2} = \frac{1}{y_{2out}} - 1$$

$$e^{-y_2} = \frac{1 - y_{2out}}{y_{2out}}$$

$$e^{-y_2} = \frac{1 - 0.772928465}{0.772928465}$$

$$\boxed{e^{-y_2} = 0.2937808}$$

$$\frac{\partial y_{2out}}{\partial y_2} = (y_{2\,out})^2 * e^{-y_2}$$

$$= (0.72928465)^2 * 0.2937808$$

$$= 0.59741841 * 0.2937808$$

$$\boxed{\frac{\partial y_{2out}}{\partial y_2} = 0.1755100}$$

Find $\frac{\partial y_2}{\partial w_7} = \frac{\partial}{\partial w_7}(h_{1out} * w_2 + h_{2out} * w_8) + b_2$

$$\frac{\partial y_2}{\partial w_7} = h_{1out} * 1 + 0 + 0 = h_{1out}$$

$$h_{1out} = 0.5932699 92$$

$$\boxed{\frac{\partial y_2}{\partial w_7} = 0.593269952}$$

$$w_7^* = w_7 - \eta \frac{\partial E_{Total}}{\partial w_7}$$

$$\frac{\partial E_{Total}}{\partial w_7} = (-0.217072 * 0.2937808 * 0.1755100$$

$$\frac{\partial E_{Total}}{\partial w_7} = (-0.0111 92555)$$

$$w_7^* = 0.50 - (-0.0111 92555)(0.5)$$

$$= 0.50 + 0.005596277$$

$$\boxed{w_7^* = 0.505596277} \simeq 0.511301270$$

$$\boxed{w_7^* = 0.511301270}$$

$$\boxed{w_8^* = 0.561370121}$$

Hidden layer updation $w_1, w_2, w_3, w_4$.

$$\frac{\partial E_{Total}}{\partial w_1} = \frac{\partial E_{Total}}{\partial H_{1out}} * \frac{\partial H_{1out}}{\partial H_1} * \frac{\partial H_1}{\partial w_1}$$

Find: $\frac{\partial E_{Total}}{\partial H_{1out}} \Rightarrow \frac{\partial E_1}{\partial H_{1out}} + \frac{\partial E_2}{\partial H_{2out}}$ $= 0.0055399425$
$0.0200$
$0.0144600$

$$\frac{\partial E_1}{\partial H_{1out}} \Rightarrow \frac{\partial E_1}{\partial y_1} * \frac{\partial y_1}{\partial H_{1out}}$$

$$\frac{\partial E_1}{\partial y_1} \Rightarrow \frac{\partial E_1}{\partial y_{1out}} * \frac{\partial y_{1out}}{\partial y_1}$$

$$\frac{\partial E_1}{\partial y_{1out}} \Rightarrow \frac{\partial}{\partial y_{1out}} \frac{1}{2}(T_1 - y_{1out})^2$$

$$\Rightarrow \frac{1}{2} * 2 (T_1 - y_{1out})^{2-1}$$

$$\Rightarrow (T_1 - y_{1out})(-1)$$

$$\Rightarrow (0.01 - 0.75136507)(-1)$$

$$\Rightarrow -0.74136507 * (-1)$$

$$\frac{\partial E_1}{\partial y_{1out}} = +0.74136507.$$

$$\frac{\partial y_{1out}}{\partial y_1} = \frac{\partial}{\partial y_1} (h_{1out} * w_5) + h_{2out} * w_6 + b_2$$

$$= \frac{\partial}{\partial y_1} \left( \frac{1}{1+e^{-y_1}} \right) \Rightarrow 0.18681560 1$$

WKT

$$\frac{\partial E_1}{\partial y_1} = 0.74136507 * 0.18681560$$

$$\frac{\partial y_{1out}}{\partial y_1} \Rightarrow \boxed{\frac{\partial E_1}{\partial y_1} = 0.138498562}$$

$$\frac{\partial y_1}{\partial h_{1out}} = \frac{\partial}{\partial h_{1out}} (h_{1out} * w_5) + h_{2out} * w_6) + b_2$$

$$= 1 * w_5 = w_5$$

$$\boxed{\frac{\partial y_1}{\partial h_{1out}} = 0.40}$$

$$\frac{\partial E_1}{\partial H_{1out}} = \frac{\partial E_1}{\partial y_1} * \frac{\partial y_1}{\partial h_{1out}}$$

$$= 0.138498562 * 0.40$$

$$\boxed{\frac{\partial E_1}{\partial H_{1out}} = 0.055399425}$$

Find $\frac{\partial E_2}{\partial H_{2out}} = \frac{\partial E_2}{\partial y_2} * \frac{\partial y_2}{\partial H_{2out}}$

$$\frac{\partial E_2}{\partial y_2} = \frac{\partial E_2}{\partial y_{2out}} * \frac{\partial y_{2out}}{\partial y_2}$$

$$\frac{\partial E_2}{\partial y_{2out}} = \frac{\partial}{y_{2out}}\left(\frac{1}{2}(T_2 - y_{2out})^2\right)$$

$$= \frac{1}{\cancel{2}} \times \cancel{2}\,(T_2 - y_{2out})^{2-1}(-1)$$

$$= \cancel{1}\,(0.99 - 0.772928465)(-1)$$

$$\boxed{\frac{\partial E_2}{\partial y_{2out}} = -(0.217071535)}$$

$$\boxed{\frac{\partial y_{2out}}{\partial y_2} = 0.17555100}$$

$$\frac{\partial E_2}{\partial y_2} \Rightarrow (-0.217071535 * 0.17555100)$$

$$\boxed{\frac{\partial E_2}{\partial y_2} \Rightarrow -0.0381071250}$$

$$\frac{\partial y_2}{\partial H_{2out}} = \frac{\partial}{\partial H_{2out}}(h_{1out} * w_7) + h_{2out} * w_8 + b_2)$$

$$= 0 + (h_{2out} * 1) + 0$$

$$\boxed{\frac{\partial y_2}{\partial H_{2out}} = 0.772928465}$$

$$\frac{\partial E_2}{\partial H_{2out}} = (-0.0381071250) * 0.772928465$$

$$= -0.029465$$

$$\boxed{\frac{\partial E_2}{\partial H_{2out}} = -0.029}$$

(19)

$$\frac{\partial y_1}{\partial H_{1out}} = \cancel{\frac{1}{5}} = 0.40$$

$$\frac{\partial E_2}{\partial H_{1out}} = -0.019$$

$$h_{1out} = \frac{1}{1+e^{-h_1}}$$

$$\frac{\partial H_{1out}}{\partial H_1} = H_{1out}(1 - H_{1out})$$

$$= 0.5932\,(1 - 0.5932)$$

$$\frac{\partial H_{1out}}{\partial H_1} = 0.241300709$$

$$H_1 = w_1 * x_1 + w_2 x_2 + b_1$$

$$\frac{\partial H_1}{\partial w_1} = x_1 = 0.05$$

$$\frac{\partial E_T}{\partial w_1} = \frac{\partial E_{tot}}{\partial H_{1out}} * \frac{\partial H_{1out}}{\partial H_1} * \frac{\partial H_1}{\partial w_1}$$

$$= 0.03635 * 0.241300 * 0.05$$

$$= 0.0004385 68$$

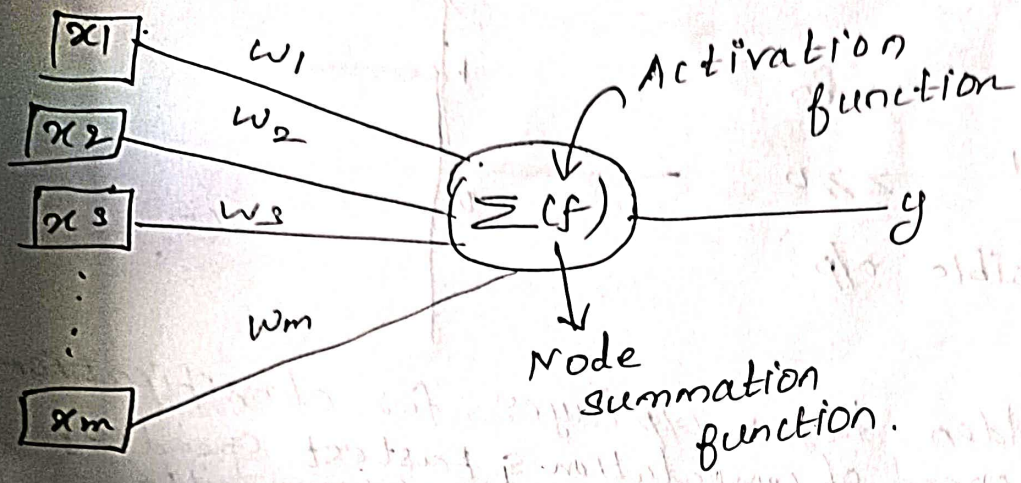
updating

$$w_{1new} = w_1 - \eta \frac{\partial E_T}{\partial w_1}$$

$$= 0.15 - 0.5 * 0.000438$$

$$\boxed{\begin{aligned} w_{1new} &= 0.149780716 \\ w_{2new} &= 0.19956143 \\ w_{3new} &= 0.24975114 \\ w_{4new} &= 0.29950229 \end{aligned}}$$

# Activation function:



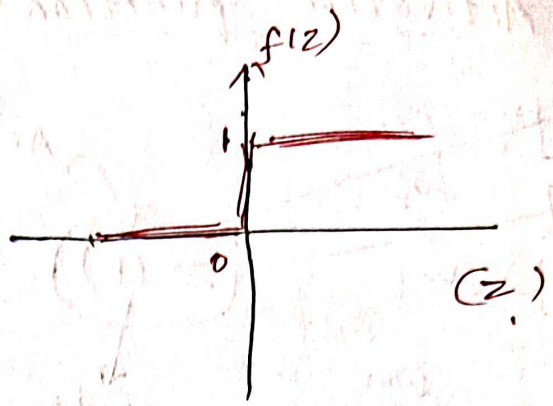

* Based upon this value neuron is activated or not.
* without activation function → It simply performs linear transformation of applying weight & Bias to the input.

Commonly used activation functions are

* Step function
* signum (sign (or) sgn) function.
* Linear function
* Rectified Linear Unit function
* Leaky Rectified Linear unit function
* Hyperbolic tangent (tanh(z))
* Sigmoid function
* softmax function

① Step function:

$$f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$$

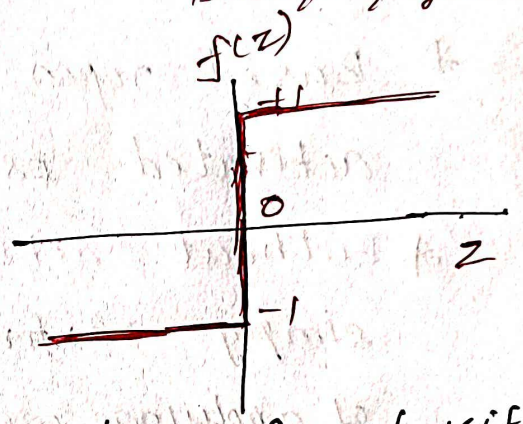



Range = possible o/p
$\{0, 1\}$

used in → Hidden layer, o/p layer for classification
⇒ Fastest speed of computation & Fastest Speed of Back propogation.

② signum (sign) function

$$f(z) = \begin{cases} -1 & z < 0 \\ 1 & z > 0 \end{cases}$$

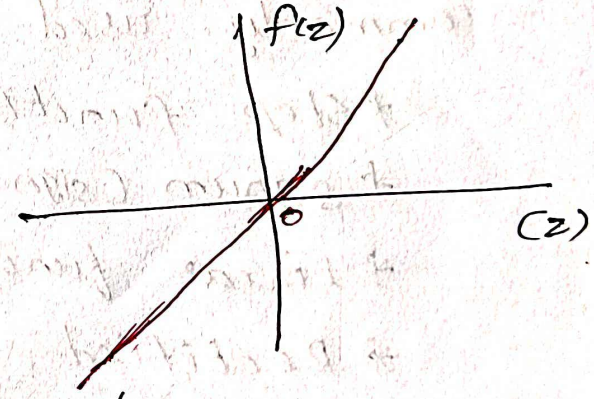



Range = possible o/p
$\{-1, 1\}$

used in → Hidden layer, o/p layer for classification.

③ Linear function:

$f(z) = z$

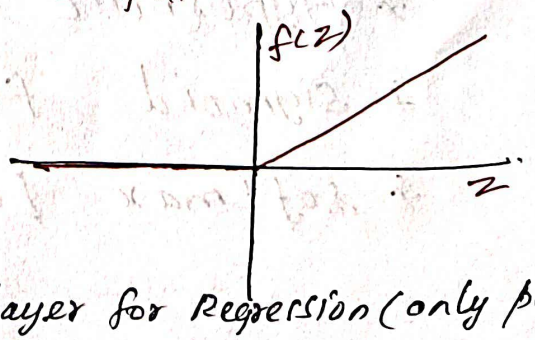



Range ⇒ possible o/p
$\{-\alpha, \alpha)$

used in → Hidden layer, o/p layer for Regression

④ Rectified Linear Unit (ReLU) function

$$f(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$$

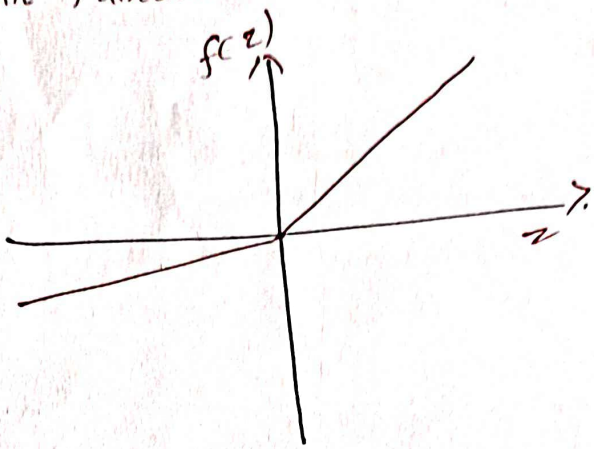



Range ⇒ possible o/p
$\{0, +\alpha)$

used in → Hidden layer, o/p layer for Regression (only positive o/p)

⑤ Leaky Rectified Linear Unit function

$$f(z) = \begin{cases} a\,z & z<0 \\ z & z \geq 0 \end{cases}$$




a is small + number

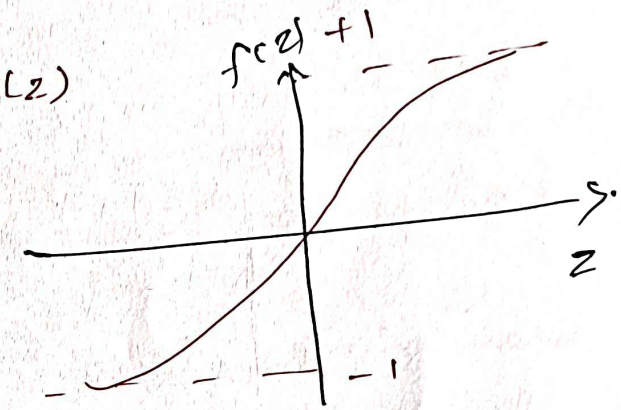
Range = Possible o/p

$(-\infty, +\infty)$

used in - Hidden layer.

⑥ Hyperbolic Tangent : tanh(z)

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



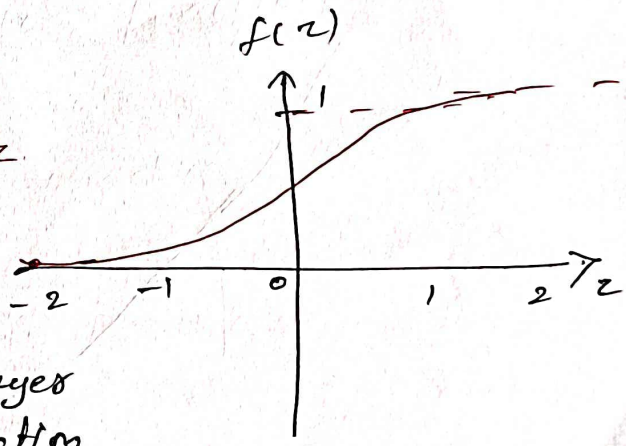

Range = Possible o/p

$(-1, +1)$

used in Hidden layer, o/p layer for classification

⑦ sigmoid function

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




Range → possible o/p

$(0, 1)$

used in → Hidden layer, o/p layer for classification

⑧ softmax function :⇒ slow → speed of computation / speed of Backpropogation

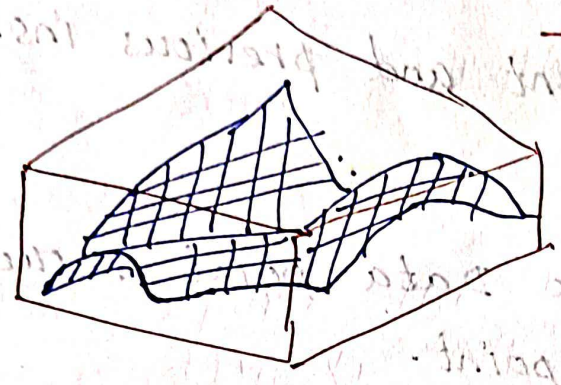
$z_1, z_2, \ldots, z_n$

used in → o/p layer for multi class classification

$$f(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + \ldots + e^{z_n}}$$

$$f(z_i) = \frac{e^{z_i}}{e^{z_1} + e^{z_2} + \ldots + e^{z_n}} = \boxed{\frac{e^{z_i}}{\sum_{k=1}^{n} e^{z_k}} = f(z_i)}$$

# Optimizers in Neural Networks:



→ [Optimizers are Algorithms or methods used to change the attributes of the neural N/w such as weights and learning rate to reduce the losses]

- once loss function is measured
- How the learning parameter are updated to fine-tune the network
- Gradient descent, having an optimizer optimize aggressively made lead to losing a minimum for loss.
- Too slowly leads to delayed learning and never converging.
- Aspect of controlling how aggressively a neural network optimizers is called learning rate.

Types of optimizer:

1. Momentum
2. Nesterov
3. Adagrad
4. Adadelta
5. RMSProp
6. Adam — Adaptive moment optimization
7. Nadam
8. Adamax
9. Gradient descent
10. Stochatic Gradient descent
11. MiniBatch SGD

Momentum:

Data variation - asymmetric
Factors (current and previous instants)

Nestarov:

Reckons future Data point rather than current data point.

Adagrad:

Gradient Algorithm is adaptive
Tensors involved are sparse.
Big updates for non-zero data points
Small updates otherwise.
It is adaptive.

Ada Delta:

Toned down version of Adagrad.
Smaller number of samples for slowing the rate of learning
Previous and current samples are factored for calculation.

RMS prop:

unpublished optimization Algorithm.
Adapt the learning rate with choice of unique learning rate for different parameters.
Every update is performed uniquely

**Adam:**

Adaptive moment optimization.
Mixture of both momentum and RMSprop.
Average the exponents of gradient.

**Nadam:**

Nesterov + Adam optimizer

**Adamax:**

Nuanced version of adam optimizer
The norms computation tends to infinite.

# Data preparation and Label preparation.

Train the Model.

Method in keras called the fit

It reports the traing accuracy and loss

→ The network iterates on data for Training in small Batches

→ Data is picked in one iteration called an epoch

Exp

→ Model calculate the differential of the instant loss about the weight.

→ After "n" epoch, lot of changes to the weight of the N/w.

→ Low loss

→ Classification Highly accurate. (98+%)

→ perform on Test Dataset.

→ Test Accuracy is good but lower than training accuracy.

→ Possibilities formulating by differentiation's chain rule.

→ How the network parameters vary as a function of measured loss. How they altered for correction.

## Hessian Matrix :-

It is square Matrix Number of Rows = Number of Colum

→ order of matrix is $n \times m$

→ used for Global minimization.

→ Find the second order derivative for given function

→ Based upon input Hessian Matrix is expanded.

Let consider, $f(x, y, z)$

Hessian matrix

$$f(x,y,z) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix}$$

$$f(x,y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

**Prob:-** $f(x,y) = 3x^3 + 2xy + y^3$ (2x2 Hessian matrix)

Find :- $\frac{\partial^2 f}{\partial x^2} = \frac{\partial^2}{\partial x^2}\left(3x^3 + 2xy + y^3\right)$

$= \frac{\partial}{\partial x}\left(3 \times 3x^2 + 2y(1) + 0\right)$

$= \frac{\partial}{\partial x} = \left[9x^2 + 2y\right]$

$$\frac{\partial^2 y}{\partial x^2} = 9 \times 2x + 0$$

$$\frac{\partial^2 y}{\partial x^2} = 18x$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2}{\partial x \partial y}(3x^3 + 2xy + y^3)$$

$$= \frac{\partial}{\partial y}(9x^2 + 2y)$$

$$= [0 + 2] = 2$$

$$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial^2}{\partial x \partial y}(3x^3 + 2xy + y^3)$$

$$= \frac{\partial}{\partial x}(0 + 2x + 3y^2)$$

$$= 2 + 0 = 2$$

$$\frac{\partial^2 f}{\partial y^2} = 6y$$

$$f(x, y) = \begin{bmatrix} 18x & 2 \\ 2 & 6y \end{bmatrix}$$

$$= 2\begin{bmatrix} 9x & 1 \\ 1 & 3y \end{bmatrix}$$

(2) $f(x, y, z) = 4x^3yz + 16xy^4z + 12x^2yz^3$ $(f(2, 1, 0)$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}(4x^3yz + 16xy^4z + 12x^2yz^3)$$

$$= \frac{\partial}{\partial x \partial y}(12x^2yz + 16y^4z + 24xyz^3)$$

$$\frac{\partial^2 f}{\partial x \partial y} = (12x^2z + 64y^3z + 24yz^3).$$

$$\Rightarrow \frac{\partial^2 f}{\partial x^2} = 0 \quad \therefore \frac{\partial^2 f}{\partial x \partial y} = (48(0) + 0 + 0) = 0$$

1) kernal regression and relation its RBF network.
2) Learning strategy in RBF network
3) Simulated annealing and differentiable NLP and SLP

① Kernal regression is a non-parametric technique in statistics and ML used to estimate the relationship between a dependent variable and one or more independent variables. Unlike traditional parametric regression models kernal regression does not assume a specific form for the relationship between variables. Instead, It uses a kernal function to weigh observations according to their distance from the point of interest.

Key Concepts:-

1) Kernal function:-

→ A kernal function determines how much influence each observation has on the estimation of dependent variable at given point.

→ Common kernals include Gaussian (RBF), Epanechnikov, and Polynomial kernals. The choice of kernal can affect the smoothness of estimated function.

2) Bandwitch parameter:

→ The bandwidth controls the width of the kernal which in turn controls the smoothness of estimated function. A smaller bandwith leads to more flexible, but potentially overfitted model, while a larger bandwidth produces a smoother.

Relation to Radial Basis function (RBF):

The Radial Basis function (RBF) is a specific type of kernal, often used in kernal regression and other kernal-based methods like support vector machines. The most common RBF is Gaussian kernal as $K(x, x') = \exp\left(-\frac{|x-x'|^2}{2\sigma^2}\right)$

The estimated value of $y$ is

$$y'(x) = \frac{\sum K(x, x_i) \cdot y_i}{\sum_{i=1}^{n} K(x, x_i)}$$

(2) Learning strategies in RBF Network :-

RBF network learning strategies typically involve determining the parameters of network, which include the centers of RBF, their width of o/p layer

1) Center Selection :-

→ Random selection: choose centers randomly from input data

→ K-means clustering is used to find cluster centers.

→ Supervised learning : Adjust centers via supervised learning.

2) width (spread) Determination :

→ fixed width : set a constant width for all RBFs

→ Adaptive width : Adjust width based on distance to nearest centers.

→ Cross - validation : use validation set to optimize widths.

3) Output layer weight learning :

→ Linear Regression : Determine weights by minimizing squared error.

→ Gradient Descent : Iteratively optimize weights using gradient descent.

→ Regularization :- Apply techniques like Ridge Regression to prevent overfitting.

4) Training Strategy:-

→ Hybrid Training :- Combine unsupervised (for Centers) and Supervised (for weights) learning.

→ online learning : update parameters incrementally with new data.

③ Simulated Learning Strategies in RBF Network.

1) Simulated Annealing:-

→ Gradually lower the "temperature" to find optimal RBF Centers, width, and weights by exploring the Parameter space and escaping local minima.

② Genetic Algorithms : use evolutionary process like selection, crossover, and mutation to evolve RBF Centers, width and weights over generations to minimize error.

③ Particle Swarm optimization (PSO)

④ Bayesian optimization

⑤ simulated Annealing + Gradient Descent (Hybrid)

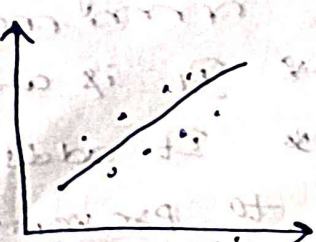
Regularization:-

* Regularization is a technique to overcome the model overfitting

* overfitting:-

* overfitting is the drawback in which the model perform well during the training, but it performs Poorly during testing. This is because it learns to mu-ch from become noise.

overfitting



Good fitting

* So inorders to reduce the overfitting, regularization technique is used.

Types of Regularization:

1) $L_1$ Regularization
2) $L_2$ Regularization.

$L_1$ Regularization:-

* It is also called LASSO Regularization.
* LASSO - Least Absolute Shrinkage selection operation
* LASSO regression add 'absolute coefficients' to the loss function.
* LASSO regression reduce the overfitting & it inclu-des all feature in the model.

$$Loss = (y - \hat{y})$$

where the regularization adds the 'absoulte weight' to the loss function to prevent it from overfitting

$$L_1 = Loss + \lambda |w|$$

Let us consider the Logistic regression model

$$Loss = y \log(\hat{y}) + (1-y) \log(1-\hat{y})$$

To regularize above model we all the Absoulte coeffi-cient as follows,

$$L_1 = y \log(\hat{y}) + (1-y) \log(1-\hat{y}) + \lambda |w|$$

$$= y \log(wx+b) + (1-y) \log(1-(wx+b)) + \lambda |w|$$

where $\boxed{\hat{y} = wx+b}$

$y \rightarrow$ actual o/p, $\hat{y}$ = predicted o/p

(2) $L_2$ Regularization:-

* It is also called Ridge Regression
* It adds the 'squared weight' to the loss function to prevent overfitting
* Ridge regression reduce the overfitting & the feature selection processes. $\boxed{L_2 = Loss + \lambda (w)^2}$

Let $\lambda = 1$, $w = 1.42$, Loss function $= 0.32$, Apply $L_1$ & $L_2$

$L_1 = \text{Loss} + \lambda |w|$

$L_1 = 0.32 + 1(1.42) = 1.74$

$= 1.74$

$$\begin{array}{r} 1.42 \\ 0.32 \\ \hline 1.74 \end{array}$$

$L_2 = \text{Loss} + \lambda |w|^2 = 0.32 + 1(1.42)^2 = 0.32 + (1.42)^2$

$= 2.336$

① Dropout layer :- The units (Neurons) which is set to 0 (or) the neurons are the neurons with unwanted o/p is said drop from the layers. This prevents the model to learn from noisy data.

② Batch normalization :- It is used to model overfitting in which the mean of an unit is zero and variance is (1) one.

③ Ensembling methods :-

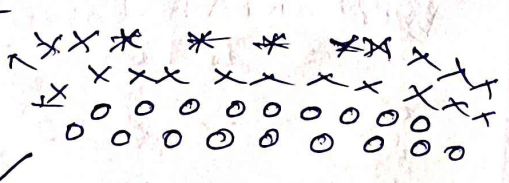
* RADIAL Basis functions :-

→ used in ANN

→ has only one hidden nodes

→ we need to convert non linearly seperately data to linearly seperable data

Example :-



2-classes (star and circle)

data is not linearly separable

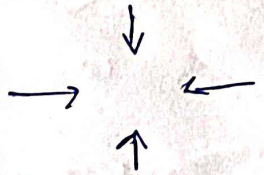
→ 2 steps.

1) Increase the dimensionality (2D - 3D)

(but this step is not mandatory only based on req)

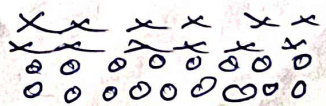
✓ ② expand the direction (Horizontal)

Compress the direction (vertical)

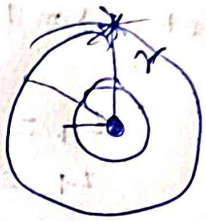


after this

Resultant dataset is



How RBF does this ?




→ Consider one center randomly (any (0,*))

→ draw concentric circles

To expand / compress, we use ③ functions

1) multiquadric :-

$$\phi(r) = (r^2 + c^2)^{1/2} \quad \therefore c > 0 \Rightarrow \text{Constant}$$

② Inverse multiquadric $\phi(r) = \dfrac{1}{(r^2+c^2)^{1/2}}$

③ ✓ Gaussian function :-

$r$ = radius

$$\phi(r) = \exp\left[\frac{-r^2}{2\sigma^2}\right]$$