Unit I :-

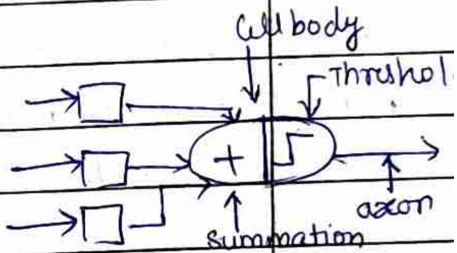
Basics of Artificial Neural Network.

Artificial Neural Network :- Analogous to Human Brain.
mp neuron, perceptron.

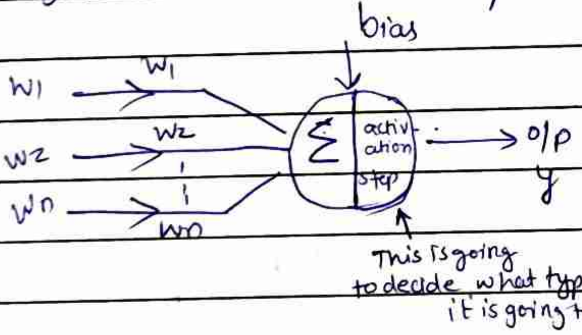
Association of biological neuron with Artificial neuron

dendrite :- receives signals from other neuron
synapse :-

Association of biological net with artificial net.



Biological neuron	Artificial neuron
cell	neuron
Dendrites	Weights or interconnectn
Soma	Sums the input
Axon	output



i/p vector $X = [x_1, x_2 \dots x_n]$

Weight vector $W = [w_1, w_2 \dots w_n]$

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

o/p Y is,

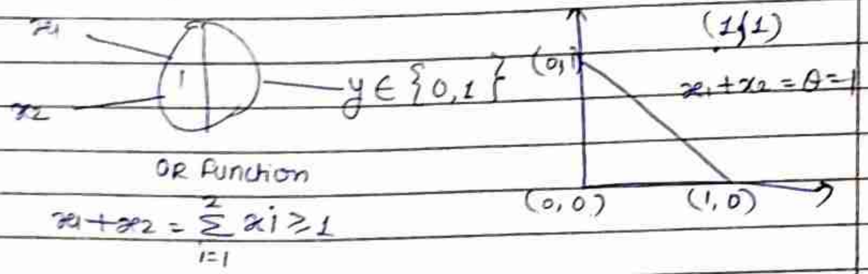
$$Y = f(y_{in})$$

MNIST Dataset

Training images $\longrightarrow$ 60,000

Testing images $\longrightarrow$ 10,000. $\quad 28 \times 28 \times \underbrace{1}_{\text{Grey scale}}$

Geometric Interpretation of M-Neuron

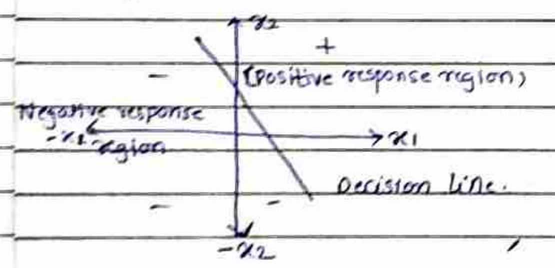


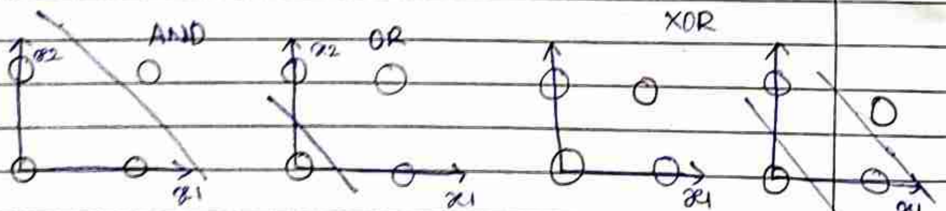
OR Function

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

- For xor no linear ~~fpt~~ separability

- Linear Separability :- concept wherein the separation of input space into regions is based on whether the network response is positive or negative $\bar{a}b+a\bar{b}$
- A decision line is drawn to separate positive and negative responses.
- A decision line ~~is~~ may also be called as decision making line
- necessity was felt to classify the patterns based upon their output responses





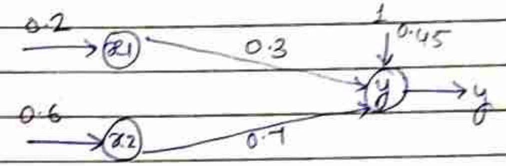
output :- 0 — ▬ 1 — ▬

No separation is possible

separation with 2 layers LTU's

$y_{in} = b + \sum_{i=1}^{n} \underbrace{w_i x_i}_{\text{feature extraction}}$

1. Calculate the net input for the network shown in figure with bias included in the network ?



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

$y_{in} = 0.45 + 0.2 \times 0.3 + 0.6 \times 0.7$

$y_{in} = 0.45 + 0.06 + 0.42 = \underline{\underline{0.93}}$

Activation function :- → to overcome disadvantage over ... perceptron ... → to handle non-linearity present at input of activation function → applied over the i/p to calculate ANN output

1. identify function.
2. Binary step function
3. Bipolar step function
4. Sigmoid function
   a. Binary sigmoid function
   b. Bipolar Sigmoid Function.
5. Ramp function

Activation function

Back propagation :- weight & Bias are going get updated based upon error calculation happening at the output (prediction done by neural network). If it does not matches the desired output calculate Mean square error

Forward Propagation :-

* Step activation :- $g(x) = 0$ for all x. for $x = 0$ $x < 0$ moguwa
derivation $x = 1$ $x > 0$

* Identity function :- one is to one input betn 2 correspondence

* Sigmoidal :- $\sigma(x) = \frac{1}{1+e^{-x}}$.

derivative = $g(x) = \sigma(x)(1-\sigma(x))$
starts approaching to 0 when we take derivative.

since the sigmoidal function is differential function so back propagate are hidden layers. So it is mostly used. And mostly used for non-linear data.

*RELU :- Rectified Linear Unit.

* tanh Activation function :- Differential function can ~~be~~ be used in back propagation.

$g(x) = 1-(\tanh(x))^2$ — derivative.

$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$

$f(x) = x \quad x > 0$ $\qquad$ $1, x >= 0$
$\qquad 0 \quad x < 0$ $\qquad$ $0. x < 0.$

$g(x) =$

Leaky Relu Activation function.

$f(x) = x$, when $x > 0$
αx, $x < 0$.

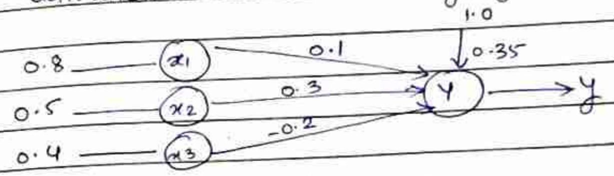
$g(x)$ $1, x >= 0$
$0; x < 0$.

Softmax

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Find out y_{in} you are getting it is passed through some activation function.
RSigmoid.

Q. Obtain the o/p of the ~~...~~ neuron Y for the network shown in Fig 3 using activation function as i) binary sigmoid, ii) Bipolar sigmoid



$y = b + x_i w_1 + x_2 w_2$
$= 0.35 + 0.8 \times 0.1 + 0.5 \times 0.3 + 0.4 \times 0.2 = 0.5$

i) Binary sigmoidal function.

$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.53}} = 0.62$

ii) Bipolar sigmoidal function.

$y = f(y_{in}) = \frac{2}{1+e^{-y_{in}}} - 1 = 0.25$.

Flowchart of Training & Testing of Perceptron

P	Hebb	Pg =
single feed	1949	3.2.1
forward n/w	Binary data or ~~...~~ used for processing binary data. Capable of handling bipolar data.	

Training (Single o/p classes) :- Random weights are generated according forward pass parameters betn (0-1)

1. Initialize the weight and bias Initialize learning rate (α)
value we choose for learning rate neural network is going to learn.

2.

2. Check for stopping condition, if it is false, perform step 2-6
3. perform steps 3-5 for each bipolar or binary training vector pair s:t
4. set activation (identity) for each input unit $i = 1$ to n:

$$\boxed{x_i = s_i}$$

5. Calculate output response of each o/p unit $j = 1$ to m; first, the net i/p is calculated as

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

6. weight & bias adjustment for $j = 1$ to m & $i = 1$ to n.

if $t_j \neq y_j$, then

$$\boxed{\begin{aligned} w_{ij}(new) &= w_{ij}(old) + \alpha t_j x_i \\ b_j(new) &= b_j(old) + \alpha t_j \end{aligned}}$$

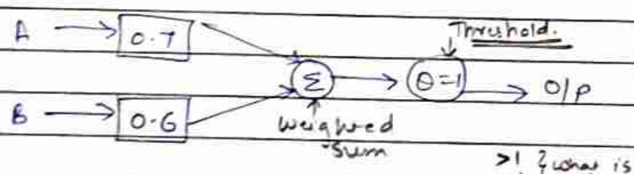
else, we have.

$$\boxed{\begin{aligned} w_{ij}(new) &= w_{ij}(old) \\ b_j(new) &= b_j(old) \end{aligned}}$$

7. Test for the stopping condition i.e.; if there is no change in weights then stop the training process, else start again from step 2

• Numerical

• Logic and gate perceptron training rule.



A	B	Target o/p A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

Whenever A & B are equal to 1 o/p must be 1, we need to have 2 weights, $w_1 = 1.2$, $w_2 = 0.6$, $\theta = 1$, $\alpha = 0.5$

i) $A = 0$, $B = 0$, Target $= 0$

$y_{in} = \sum_{i=1}^{2} w_i x_i = 0 \times 1.2 + 0 \times 0.6$

- This is not greater than threshold ∴ calculated o/p is 0

$y_{in} = 0 <$ threshold $= 1$ ∴ calculated o/p $y = 0$

- Since actual calculated value is 0 & targeted o/p is 0 ∴ no need of ~~the~~ weight updation.

ii) $A = 0$, $B = 1$

$= 0 \times 1.2 + 1 \times 0.6 = 0.6$

$y_{in} = \sum_{i=1}^{2} w_i x_i$

- Since, $y_{in} = 0.6$ is less than the threshold $= 1$ ∴ the o/p is 0. so no need of weight updation.

iii) $A = 1$, $B = 0$

$y_{in} = 1 \times 1.2 + 0 \times 0.6 = 1.2$

As the calculated o/p y is not match to target o/p so we need updation of weight

∴ weight are updated as per perceptron training rule.

$W_{ij}(new) = W(old) +$ learning rate · target o/p & actual o/p

$w_1 new = 1.2 + 0.5(0-1) \, 1 = 0.7$

$w_2 new = 0.6 + 0.5(0-1) \, 0 = 0.6$

$W new = W old + \alpha(t - 0) x_i$
targeted updation.

Taking new weights $w_1 new = 0.7$ & $w_2 = 0.6$.

i) $A = 0$, $B = 0$, Target $= 0$

$y_{in} = 0 \times 0.7 + 0 \times 0.6 = 0$.

$y_{in} <$ threshold.

ii) $A = 0$, $B = 1$, Target $= 0$

$y_{in} = 0 \times 0.7 + 1 \times 0.6 = 0.6$

No updation of weight.

iii) $A=1,\ B=0,\ y_{in}=0$

$y_{in} = 1\times0.7 + 0\times0.6 = 0.7$

no need of updation of weight

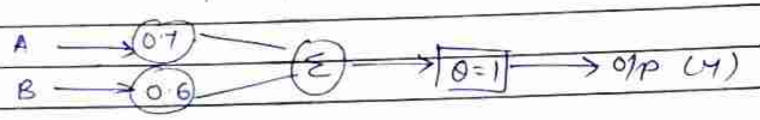
iv) $A=1,\ B=1 \quad y_{in}=1$

$y_{in} = 1\times0.7 + 1\times0.6 = 1.3$

No need of weight updation.

we have classified all the ~~traing~~ training i/p correctly with $w_1 = 0.7$ & $w_2 = 0.6$, threshold = 1, $\theta = 1$ & learning rate $\alpha = 0.5$

Modified perceptron And gate.



Initialize test sample.

$w_i \cdot x_i$

- Perceptron rule for XOR gate.

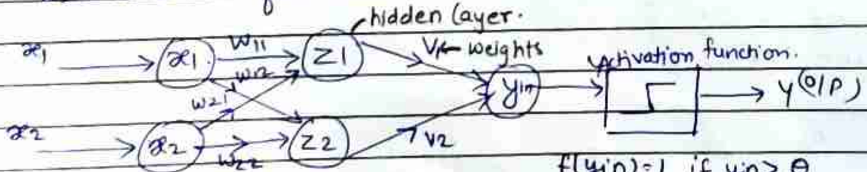
$y = x_1\bar{x_2} + \bar{x_1}x_2$

$y = z_1 + z_2$

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

$z_1 = x_1\bar{x_2}$ - function1

$z_2 = \bar{x_1}x_2$ function2



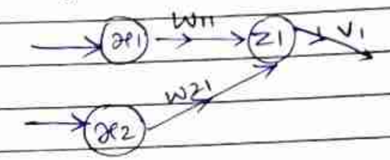
$f(y_{in}) = 1$ if $y_{in} \geq \theta$

$= 0$ if $y_{in} < \theta$

input = x_i

w_{ij}

1st function $z_1 = x_1\bar{x_2}$



x_1	x_2	$z_1 = x_1\bar{x_2}$
0	0	0
0	1	0
1	0	1
1	1	0

let initial weight $w_{11} = w_{21} = 1$, Threshold $\theta = 1$, Learning rate = 1.5

(x_1, x_2) — $w_{11} * x_1 + w_{21} * x_2$

$(0,0) = z_1\ in = w_{ij} * x_i = (1\times0)+(1\times0) = 0 \qquad 0<1$

$(0,1) = z_1\ in = w_{ij} \times x_i = (1\times0)+(1\times1) = 1 \qquad 1<1$ compared with o/p z_1

$(1,0) = z_1\ in = w_{ij} \times x_i = (1\times1)+(1\times0) = 1 \qquad 1<1$

$(1,1) = z_1\ in = w_{ij} \times x_i = (1\times1)+(1\times1) = 2. \qquad 2<1$

→ which is not same as target o/p ∴ weights are updated.

$w_{new} = w_{old} + \alpha(t - o)x_i$

$w_{11} = 1 + 1.5(\underset{\text{target}}{0} - \underset{\text{actual}}{1})0 = 1$

$w_{21} = 1 + 1.5(0 - \underset{\text{actual o/p}}{1})\underset{x_2}{1} = -0.5$.

Now, assume $w_{11} = 1$ & $w_{22} = -0.5$.

$x_1 = 0,\ x_2 = 0$

$y_{in} = (1\times0)+(-0.5\times0) = 0$

$x_1 = 0,\ x_2 = 1$

$z_1\ y_{in} = (1\times0)+(-0.5\times1) = -0.5$

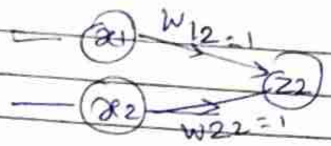
$x_1 = 1,\ x_2 = 0$

$z_1\ y_{in} = (1\times1)+(-0.5\times0) = 1$

$x_1 = 1,\ x_2 = 1$

$z_1\ y_{in} = (1\times1)+(-0.5\times1) = 1 - 0.5 = 0.5$.

∴ Calculated o/p ~~are~~ values are ~~are~~ same as target o/p z_1 with the updated weight $w_{11} = 1$ & $w_{21} = -0.5$



x_1	x_2	z_2
0	0	0
0	1	1
1	0	0
1	1	0

$x_1=0, x_2=0,$
$Z_2\,y_{in} = (1\times0)+(1\times0)=0$

$x_1=0, x_2=1$
$Z_2\,y_{in} = (1\times0)+(1\times1)= 1$

$x_1=1, x_2=0$
$Z_2\,y_{in} = (1\times1)+(0\times0)= 1$
∴ we need to updated the weight.

$w_{new} = w_{old} + \alpha(t-0)x_i$
$w_{12} = 1+1.5(0-1)1 = -0.5$
$w_{22} = 1+1.5(0-1)0 = 1.$

$x_1=0, x_2=0$
$Z_{2in}\,\cancel{y_{in}} = (-0.5\times0)+(0\times0)=0$

$x_1=0, x_2=1$
$Z_{2in}\,\cancel{y_{in}} = (-0.5\times0)+(1\times1)= 1$

$x_1=1, x_2=0$
$Z_{2in}\,\cancel{y_{in}} = (-0.5\times1)+(1\times0) = -0.5+1 = -0.5$

$x_1=1, x_2=1$
$Z_2\,y_{in} = (-0.5\times1)+(1\times1) = -0.5+1 = 0.5.$

With the weight $w_{12}=-0.5$, $w_{22}=1$ we are able to classify the input at Z1 & Z2 layer.

$Y = Z_1$ or Z_2 $\quad y_{in} = Z_1V_1 + Z_2V_2$

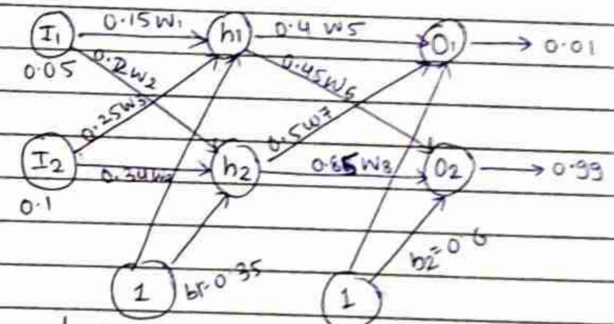
x_1	x_2	Z_1	Z_2	$Y = Z_1$ or Z_2
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

$V_1 = V_2 = 1 \qquad \alpha = 1.5$

Z_1	Z_2	$y_{in} = V_iZ_i$
0	0	$y_{in} = (1\times0)+(1\times0) = 0$
0	1	$y_{in} = (1\times0)+(1\times1) = 1$
1	0	$y_{in} = (1\times1)+(1\times0) = 1$
0	0	$y_{in} = (1\times0)+(1\times0) = 0$

∴ Perceptron learning for xor is shown below with the weights

Back Propagation :- optimize weights / update



Forward propagation

i) For h_1

$net_{h_1} = w_1\times I_1 + w_2\times I_2 + B_1\times1$
$= 0.15\times0.05 + 0.2\times0.1 + 0.35$
$= 0.3775$

Binary sigmoidal function

$$o/p\ h_1 = \frac{1}{1+e^{-net\,h_1}} = \frac{1}{1+e^{(-0.3775)}} = 0.5932$$

ii) For net h2

$$net\ h_2 = w_3 \cdot I_1 + w_4 \cdot I_2 + B_1 \cdot 1$$
$$= 0.25 \times 0.05 + 0.3 \times 0.1 + 0.35$$
$$= 0.3925$$

Binary Sigmoidal function

$$o/p\ h_2 = \frac{1}{1+e^{(-0.3925)}} = 0.59688437$$

iii) For O1

$$net\ O_1 = w_5 \times h_1 + w_6 \times h_2 + B_2 \times 1$$
$$net\ O_1 = 0.4 \times 0.5932 + 0.45 \times 0.59688 + 0.6 \times 1$$
$$= 1.10596$$

By binary sigmoidal function

$$O/p\ O_1 = \frac{1}{1+e^{(-1.10596)}} = \boxed{0.7513 \neq 0.01}$$

iv) For O2

$$net\ O_2 = w_7 \times h_1 + w_8 \times h_2 + B_2 \times 1$$
$$= 0.5 \times 0.5932 + 0.55 \times 0.59688 + 0.6 \times 1$$
$$= 1.224884$$

By binary sigmoidal function

$$o/p\ O_2 = \frac{1}{1+e^{(-1.224884)}} = \boxed{0.772928 \neq 0.99}$$

Calculate total error

$$E_{total} = \sum \frac{1}{2} (target - o/p)^2$$

$$E_{O1} = \frac{1}{2} (0.01 - 0.7513)^2$$
$$= 0.27476$$

$$E_{O2} = \frac{1}{2} (0.99 - 0.772928)^2 = 0.02356$$

$$E = E_{O1} + E_{O2}$$
$$= 0.27476 + 0.02356$$
$$= 0.29832$$

$\boxed{\text{Back Propagation}}$ w5 updation

i) $$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_1} \times \frac{\partial out_1}{\partial net_{O1}} \times \frac{\partial net_{O1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial out_1} = -(target\ O_1 - out\ O_1)^{2-1}$$
$$= -(0.01 - 0.7513)^{2-1}$$
$$= 0.7413$$

$$\frac{\partial out_{O1}}{\partial net_{O1}} = out_{O1}(1 - out_{O1})$$
$$= 0.7513(1 - 0.7513)$$
$$= 0.1868483$$

$$\frac{\partial net_{O1}}{\partial w_5} = out\ h_1$$
$$= 0.5932$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.7413 \times 0.1868483 \times 0.5932$$
$$= 0.0821645$$

To decrease the error. Subtract with learning rate ($\alpha = 0.5$)

$$w_5^{+} = w_5 - \alpha \frac{\partial E_{total}}{\partial w_5}$$

$$= 0.4 - 0.5 \times 0.0821645$$
$$w_5^{+} = 0.3589775$$

ii) $w_6^+ = w_6 - \alpha \frac{\partial E_{total}}{\partial w_6}$

$= 0.45 - 0.5 \times 0.0821645$

$= 0.4089 1775$

iii) w_7

$\frac{\partial E_{total}}{\partial out_2} = -(0.99 - 0772928)^{2-1}$

$= -0.2171$

$\frac{\partial out_{O2}}{\partial net_{O2}} = out_{O2}(1 - out_{O2})$

$= 0.17551412$

$\frac{\partial net_{O2}}{\partial w_7} = out\ h_2$

$= 0.596884$

$\frac{\partial E_{total}}{\partial w_7} = -0.2171 \times 0.17551412 \times 0.596884$

$= -0.02274$

$w_7^+ = w_7 - 0.5 \frac{\partial E_{total}}{\partial w_7}$

$= 0.5 - 0.5 (-0.02274)$

$= 0.51137$

iv) $w_8^+ = w_8 - 0.5(-0.02274)$

$= 0.55 - 0.5(-0.02274)$

$= 0.56137$

Hidden layers:-

$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h_1}} \times \frac{\partial out_{h_1}}{\partial net_{h_1}} \times \frac{\partial net_{h_1}}{\partial w_1}$

$\frac{\partial E_{total}}{\partial out_{h_1}} = \frac{\partial E_{O1}}{\partial out_{h_1}} + \frac{\partial E_{O2}}{\partial out_{h_2}}$

i) $\frac{\partial E_{O1}}{\partial out_{h_1}} = \frac{\partial E_{O1}}{\partial net_{O1}} \times \frac{\partial net_{O1}}{\partial out_{h_1}}$

$\frac{\partial E_{O1}}{\partial net_{O1}} = \frac{\partial E_{total}}{\partial out_{O1}} \times \frac{\partial out_{O1}}{\partial net_{O1}}$

$= 0.741365 \times 0.186815$

$= 0.138498$

$\frac{\partial net_{O1}}{\partial out_{h_1}} = w_5 = 0.40$

$\partial E_{O1} = \frac{\partial E_{O1}}{\partial net_{O1}} \times \frac{\partial net_{O1}}{\partial out_{h_1}}$

$= 0.138498 \times 0.40$

$= 0.553992.$

ii) $\frac{\partial E_{O2}}{\partial out_{h_2}} = \frac{\partial E_{total}}{\partial out_{O2}} = -(0.99 - 0.772928)^{2-1}$

$= -0.217079.$

$\frac{\partial out_{O2}}{\partial net_{O2}} = 0.17551412$

$= -0.217079 \times 0.17551412$

$= -0.0381004$

$\frac{\partial net_{O2}}{\partial out_{h_2}} = w_1 = 0.15$

$$\frac{\partial E_{o2}}{\partial out_{h2}} = \frac{\partial E_{o2}}{\partial net_{o2}} \times \frac{\partial net_{o1}}{\partial out_{h1}}$$

$= -0.03819 \times 0.5$

$= -0.0190502$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h2}}$$

$= 0.0553992 + (-0.01905)$

$= 0.0363492$

$$\frac{\partial out_{h1}}{\partial net_{h1}} = out_{h1}(1 - out_{h1})$$

$= 0.593269(1 - 0.593269)$

$= 0.2413007$

$$\frac{\partial net_{h1}}{\partial w_1} = i_1 = 0.05$$

$$\therefore \frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} \times \frac{\partial out_{h1}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial w_1}$$

$= 0.0363492 \times 0.2413007 \times 0.05$

$= 0.000438554$

$$w_1^+ = w_1 - \eta \frac{\partial total}{\partial w_1}$$

$= 0.15 - 0.5(0.0004385)$

$= 0.149780$

$$w_2^+ = w_2 - \eta \frac{\partial total}{\partial w_1}$$

$= 0.2 - 0.5(0.0004385)$

$= 0.199781$

Unit II -

$$Z = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \\ w_{4,1} & w_{4,2} & w_{4,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} =$$

Binary ~~classification~~ entropy

whenever yes or no

Loss function - error function is for a single training example/input

Cost function - A cost function, is average loss over the entire training dataset.

- Regression -
  - MSE - difference betn actual & predicted.
  - MAE - Mean absolute error

- Classification
  - Binary cross-entropy - used in binary classification $\left(-\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right)$
  - Categorical cross-entropy - ~~categ~~ multiclass classification $-\sum_{j=1}^{k} y_j \log(\hat{y}_j)$

Parameters :-

Few layers - underfitting

More layers - overfitting

Architecture :-

- More generally, we can calculate the activation of neuron i in layer l

$$z_i^{[l]} = w_i^{[l]T} a^{[l-1]} + b_i^{[l]}$$

$$a_i^{[l]} = g(z_i^{[l]})$$

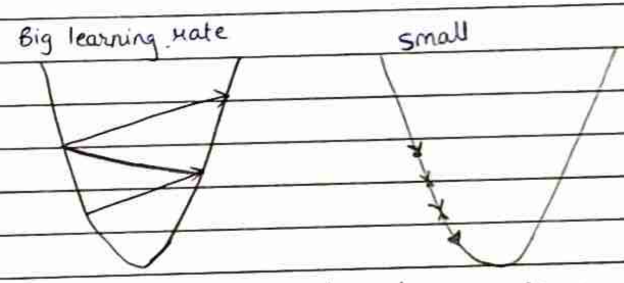
Similarly, we can calculate all of the activations for a given layer l by using our weight matrix $W^{[l]}$

Parameters and hyper parameters tuning

- Parameters : $W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$
  - No. of ephocs.
- Hyperparameters are the variables which determines the network structure (eg. Number of Hidden Units) and variables which determine how the network is trained (eg. - learning Rate).
- Hyper parameters are set before training (befor optimizing the weights and bias)
- If hyperparameters neglected and not selected properly, they can be the cause of high error, resulting in wrong predictions.
- The number of layers, neuron, iterations, choice of activation function (ReLU, Tanh, sigmoid, in the hidden layer), optimizer (Adam, Adgrad, RMS Prop), learning rate, choise of cost function, drop out rate, these are parameter control w's and b's.
- Layer size :- The second step is to tune the number of layers.
- Fewer layers may give an underfitting result while too many layers may make it overfitting
- Number of neurons - may be same or different in each layer. Depends on solution complexity
- An activation function is a parameter in each layer. The input values moving from a layer to another layer keep changing according to the activation function
- Optimizer :- One of the hyperparameters in the optimizer is the learning rate

- Learning rate :- Control the step size for a model to reach the minimum loss function (if gradient descent is working properly use small learning rate)
- A higher learning rate (alpha) makes the model learn faster, but it may miss the minimum loss function.
- A lower learning rate gives a better chance to find a minimum loss function.
- As a trade - off lower learning rate needs
  - higher epochs, or
  - more time and
  - memory capacity resources.

* Gradient Descent :-



- The gradient vector has both a direction and a magnitude
- Gradient descent algorithms multiply the gradient by a scalar known as the learning rate (also sometimes called step size) determine the next time.

  eg:- If the gradient magnitude is 2.5 and the learning rate is 0.01 then the gradient descent algorithm will pick the next point 0.025 (0.01 × 0.25) away from the previous point

Gradient Descent and its variants:

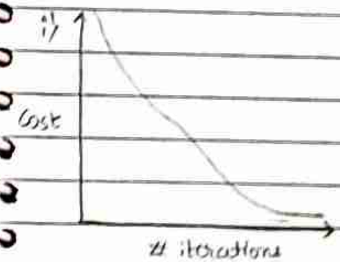
- Gradient descent algorithm updates the parameters by moving in the direction opposite to the gradient of the objective function with respect to the network parameters

  initialize w, b
  Iterate over data:
  compute $\hat{y}$
  Compute $\mathcal{L}(w, b)$
  $w_{t+1} = w_t - \eta \Delta w_t$ (Derivative loss functn w.r.t weight)
  $b_{t+1} = b_t - \eta \Delta b_t$
  till satisfied

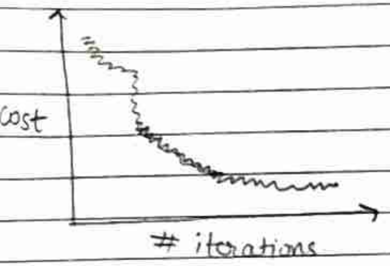
- If the observation size of the training dataset is too large, it will definitely take a longer time to build the model
- To make the model learn faster, we can assign batch size so that not all of the training data are given to the model at the same time
- Batch size is the number of training data sub-samples for the input
- eg :- If the training dataset has 77,500 observations and the batch size is 1000, the model will learn 77 times with 1000 training data sub-samples and another last ~~to~~ learning from the 500 training data sub-samples. ~~and another last learning from the 500 training~~
- The smaller batch size makes the learning process faster and bigger batch size and has a slower ~~to~~ learning process.
- Epoch :- The no. of times a whole dataset is passed through the neural network model is called an epoch.

- 1 epoch :- 1 training dataset is passed forward and backward through neural network once.
- Too small no. of epochs - underfitting because the neural network has not learned much enough
- Many epochs - Overfitting where the model can predict the data very well, but cannot predict new unseen data well enough
- Batch gradient :- Take the entire dataset - calculate the cost function - update parameter - speed, complexity & more learning rate.
  dis :- large training examples - expensive computationally.
- Stochastic gradient : Update the parameters after every single observation and every time the weights are updated it is known as an iteration
  Ad - Faster than batch gradient
  Disadvantage :- process only one example which can be additionally overhead for the system
- Mini-batch Gradient Descent :- Take a subset of data and update the parameters based on every subset

* Comparison of Gradient descent variants :-

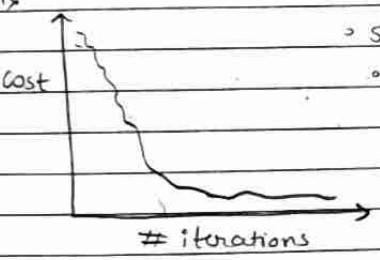
i)


- Cost function reduces smoothly
- Cost keeps on decreasing over the epochs smoothly
- Disadvantage: large training examples computationally expensive.
- Batch Gradient.



- Lot of variations in cost functions
- Updation is not smooth
- Updating the parameter based on a single observation, there are a lot of iterations
- model starts learning noise as well

SGD



- Smoother cost function as compared to SGD
- Updation of the cost function is smoother as compared to that of the cost function in SGD
- Not updating the parameters after every single observation but after every subset of the data.

Mini Batch Gradient.

$$\frac{\partial E_{total}}{out_{h1}} = 0.0363492$$

$$\frac{\partial out_{h2}}{\partial net_{h2}} = out_{h2}(1 - out_{h2})$$
$$= 0.59688(1 - 0.59688)$$
$$= 0.240614$$

$$\frac{\partial net_{h2}}{w_3} = i_2 = 0.1$$

$$\frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial out_{h2}} \times \frac{\partial out_{h2}}{\partial net_{h2}} \times \frac{\partial net_{h2}}{\partial w_3}$$

$$= 0.0363492 \times 0.240614 \times 0.1$$
$$= 0.0008746$$

$$w_3^+ = w_3 - \eta\left(\frac{\partial E_{total}}{\partial w_3}\right)$$
$$= 0.25 - 0.5(0.0008746)$$
$$= 0.2495627$$

$$w_4^+ = w_4 - \eta\left(\frac{\partial E_{total}}{\partial w_3}\right)$$
$$= 0.3 - 0.5(0.0008746)$$
$$= 0.2995627$$

~~1 Batch~~

Momentum Gradient Descent Algorithm:-
update rule, also include the history component V_t it stores all the previous gradient movements till this time t.

$$V_t = \gamma . V_{t-1} + \eta \nabla w_t = \gamma^{t-1} . \eta \nabla w_1 + \gamma^{t-2} . \eta \nabla_1 w_t \ldots \gamma^{t-n} \eta \nabla_1 w_t$$

- 1 epoch = one pass over the entire data
- 1 step = One update of the parameters
- N = Number of data points
- B = Mini Batch size

Algorithm.	# of steps in 1 epoch.
Vanilla (Batch) gradient	1
Stochastic gradient.	N
min-Batch gradient	$\frac{N}{B}$

Vainishing and Exploding gradient.

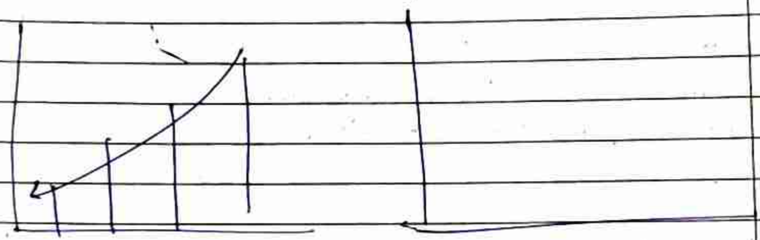
- as we are calculating gradient values from output to input ~~val~~ are vanishing so weight values are increasing. - Vanishing
- ——— // ——— decreasing. exploding

$$\boxed{\propto \frac{\partial L}{\partial w.}}$$

converge. minus - oscillating and global.

Gradient :- derivative of loss funct^n w.r.t weight.
used to update the weight to minimize the loss func^n during back propagation.

- Vanishing :- Derivative or slope will get smaller and smaller as we go backward with every layer during back propagative.
- update is very small or exponentially small, the training time takes too much longer, and in the worst case, this may completely stop the neural network training. — exploding.



- occurs when the derivatives or slope will get larger and larger as we go backward.
- because of weights, not because of activat^n function.
- Due to high weight values, the derivatives will also higher so that the new weight varies a lot to the older weight, and the gradient will never converge. So it may result in oscillating around the minima and the global minima.

Remedies :- Vanishing.

1. Choice of Activation function :- vanish occurs with sigmoid and tanh activat^n func^n because.
because the derivative are betn. 0 to 0.25 and 0-1. Therefore, updated weight values are small, & new weight values are very similar to the old weight. This leads to vanishing.
To avoid this problem using ReLU activation function because the gradient is 0 for negative and zero i/p, & 1 for positive i/p.

2. Appropriate ~~select~~ choice of weight.
weight < 1 vanishing
weight > 1 exploding. (algorithm becomes unstable or oscillates
- So choose weights randomly
- Intelligent choosing of Backpropagation learning algorithm

vanishing occurs due to sigmoid & tanh activat^n func^n.
exploding gradient due to large ~~wrong~~ weights.

- For every layer, weights are commonly sampled from a normal distribution with mean = 0, standard deviation = 1
$\mu = 0$ and $\sigma = 1$.

- "He" weight initialization.

$$\mu = 0 \text{ and } \sigma = \sqrt{\frac{1}{n_1}} \quad \text{or} \quad \sigma = \sqrt{\frac{2}{n_1}}$$

n_1 — no. of neurons in preceding layer.

- "He" initialization and "Xavier" initialization ensure that the weights ~~or~~ are close to 1

- "Xavier" weight initialization

$$\mu = 0 \text{ and } \sigma = \sqrt{\frac{2}{n_1 + n_1 + 1}}$$

* Gradient clipping :-
Give the threshold value depending on that the gradient ~~of~~ value is more or less so for that we need to perform gradient clipping method.

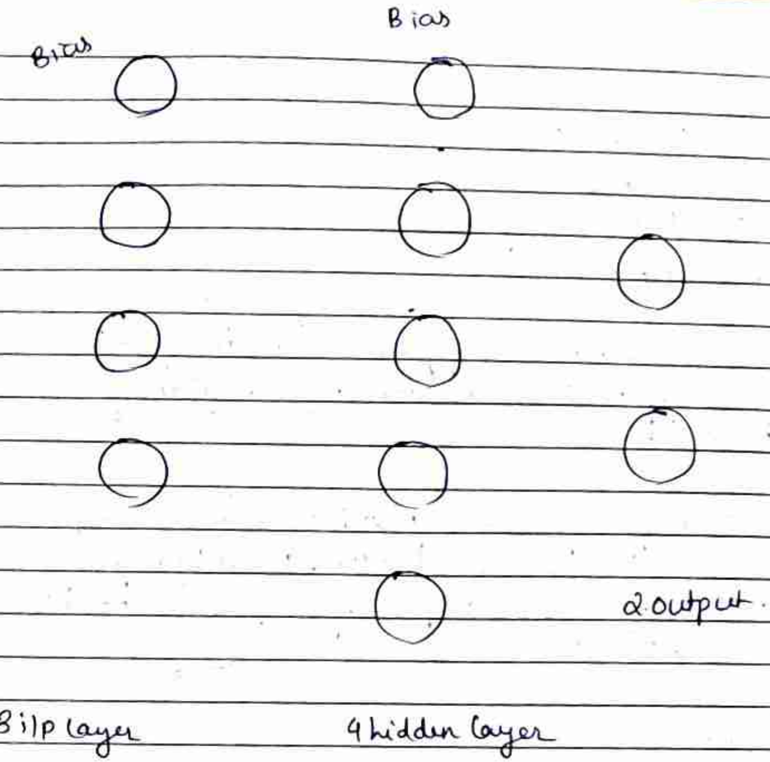
if gradient exceed, set them to max upper bound of interval
if gradient fall below set them to min lower bound of interval

Different architecture of neural network.

Bias errors and variance.
If neural n/w model is not accurate, it can make prediction errors, and these prediction errors are usually known as Bias and Variance.

- Bias is the difference betn actual & predicted values
- High Bias :- model not captured data well during training and cannot perform well on test data
- The instance where the model cannot find pattern in training set and ~~here~~ hence fails for both seen and unseen data, is called underfitting
- Neural N/W performs very well on training data, but fails as soon ~~as~~ it sees some new data from the problem domain and is very imp to take care of this in NN

No. of parameters in a feed-forward Neural Network



Assumption:-
i = no. of neuron in i/p layer
h = — " — hidden layer
o = — " — output layer

i) No. of connectn betn 1st & 2nd layer (3×4) = 12, which is nothing but the product of i & h

ii) — " — 2nd & 3rd layer (4×2) = 8 — " — h & o

iii) There are connections between layers via bias as well. No. of connections betn the bias of the first layer & neurons in second layer (except bias of the second layer): 1×4, which is nothing but h.

iv) No. of connectn betn bias of second layer and neurons of third layer: 1×2 which is nothing but o

Summing up all,
$3\times4 + 4\times2 + 1\times4 + 1\times2$
$12 + 8 + 4 + 2$
26 — trainable parameters.
$= i\times h + h\times o + h + o$

Thus, the total number of parameters in a feed-forward neural network with one hidden layer is given by
$(i\times h + h\times o) + h + o$

Scenario 2 :- A feed-forward n/w with 3 hidden layers.
No. of units in the i/p, first hidden, second hidden, third hidden & output layers are h_1, h_2, h_3, 3, 5, 6, 4 and 2

Assumption:

i = no. of neurons in input layer
h_1 = —"— first hidden layer
h_2 = —"— second —"—
h_3 = —"— third —"—
o = no. of neurons in o/p layer

i) Connect of 1 & 2 = $(3\times5) = 15$
ii) —"— 2 & 3 = $(5\times6) = 30$
iii) —"— 3 & 4 = $(6\times4) = 24$
iv) —"— 4 & o/p = $(4\times2) = 8$
v) —"— h_1 = $(1\times5) = 5$
vi) —"— h_2 = $(1\times6) = 6$
vii) —"— h_3 = $(1\times4) = 4$
viii) —"— o/p = $(1\times2) = 2$

$= (3\times5) + (5\times6) + (6\times4) + (4\times2) + (1\times5)(1\times6) + (1\times4) + (1\times2)$
$= 15 + 30 + 24 + 8 + 5 + 6 + 4 + 2$
$= 94.$
$= (i\times h_1 + h_1\times h_2 + h_2\times h_3 + h_3\times o) + h_1 + h_2 + h_3 + o.$

Formula for n hidden layers.

$$i\times h_1 + \sum_{k=1}^{n-1}(h_k\times h_{k+1}) + h_n\times o + \sum_{k=1}^{n} h_k + o.$$

Q.1] Apply