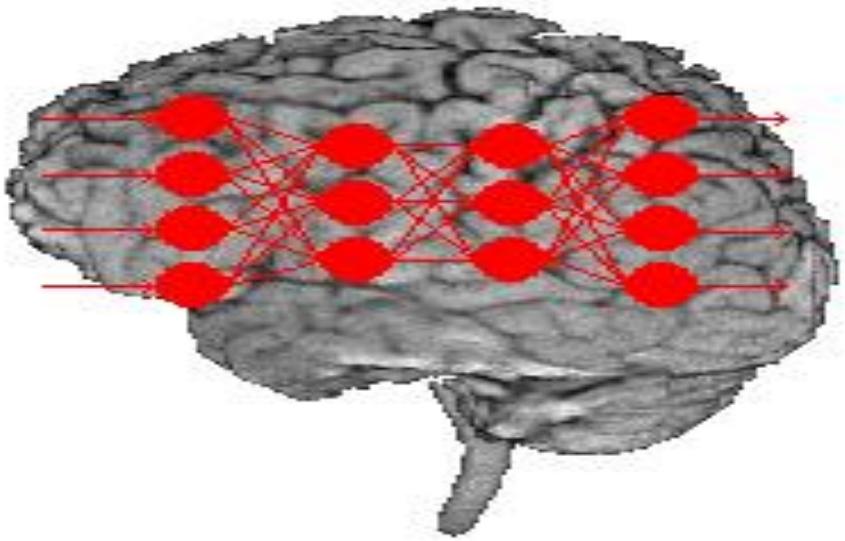
Unit II

Neural Network for Control



Backpropagation

Question: What is Backpropagation algorithm?

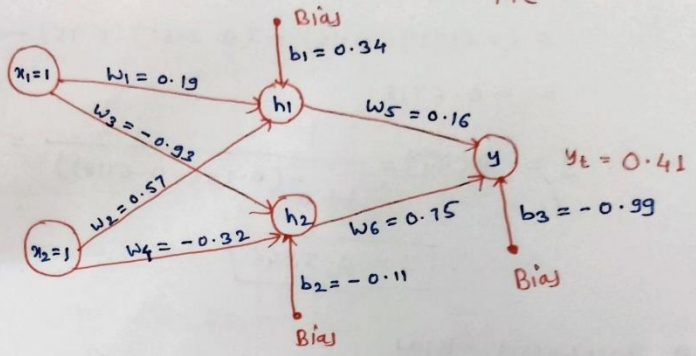
- Backpropagation is the process of updating the weights of the network in order to reduce the error in prediction.
- In this case, we keep on updating the weights and biases until the error between the targeted output and actual output is a minimum.
- We keep just adjusting each weight individually and see how it changes the output.
- Updated weights in Backpropagation are

$$w_i(new) = w_i(old) - \alpha \frac{dE}{dw_i}$$

Examples on Backpropagation

Examples on Backpropagation.

Q.1 Update weight w_5 and w_2 using Backpropagation algorithm. $\left[\phi(z) = \frac{1}{1+e^{-\alpha z}}, \alpha = 0.7\right]$



Soln: Forward Bias

(i) h_1

$$z_{h_1} = x_1 w_1 + x_2 w_2 + b_1 = (1)(0.19) + (1)(0.57) + 0.34 = 1.1$$

$$h_1 = \phi(z_{h_1}) = \frac{1}{1+e^{-\alpha z_{h_1}}} = \frac{1}{1+e^{-(0.7 \times 1.1)}}$$

$$\therefore \boxed{h_1 = 0.6835}$$

(ii) h_2

$$z_{h_2} = x_1 w_3 + x_2 w_4 + b_2 = (1)(-0.93) + (1)(-0.32) + (-0.11) = -1.36$$

$$\therefore h_2 = \phi(z_{h_2}) = \frac{1}{1+e^{-\alpha z_{h_2}}} = \frac{1}{1+e^{-(0.7 \times (-1.36))}}$$

$$\therefore \boxed{h_2 = 0.2785}$$

(iii) y

$$z_y = h_1 w_5 + h_2 w_6 + b_3 = (0.6835)(0.16) + (0.2785)(0.75) - 0.99 = -0.6718$$

$$\therefore y = \phi(z_y) = \frac{1}{1+e^{-(0.7 \times (-0.6718))}} = \frac{1}{1+e^{-\alpha z_y}}$$

$$\therefore \boxed{y = 0.3846}$$

* Backward Bias

(i) To update weight w_5

$$w_5(\text{new}) = w_5(\text{old}) - \alpha \frac{\partial E}{\partial w_5}$$

$$\therefore E \leftarrow y \leftarrow z_y \leftarrow w_5$$

$$\therefore \frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z_y} \cdot \frac{\partial z_y}{\partial w_5}$$

$$= -(y_t - y) \times \frac{\alpha e^{-\alpha z_y}}{(1+e^{-\alpha z_y})^2} \times h_1$$

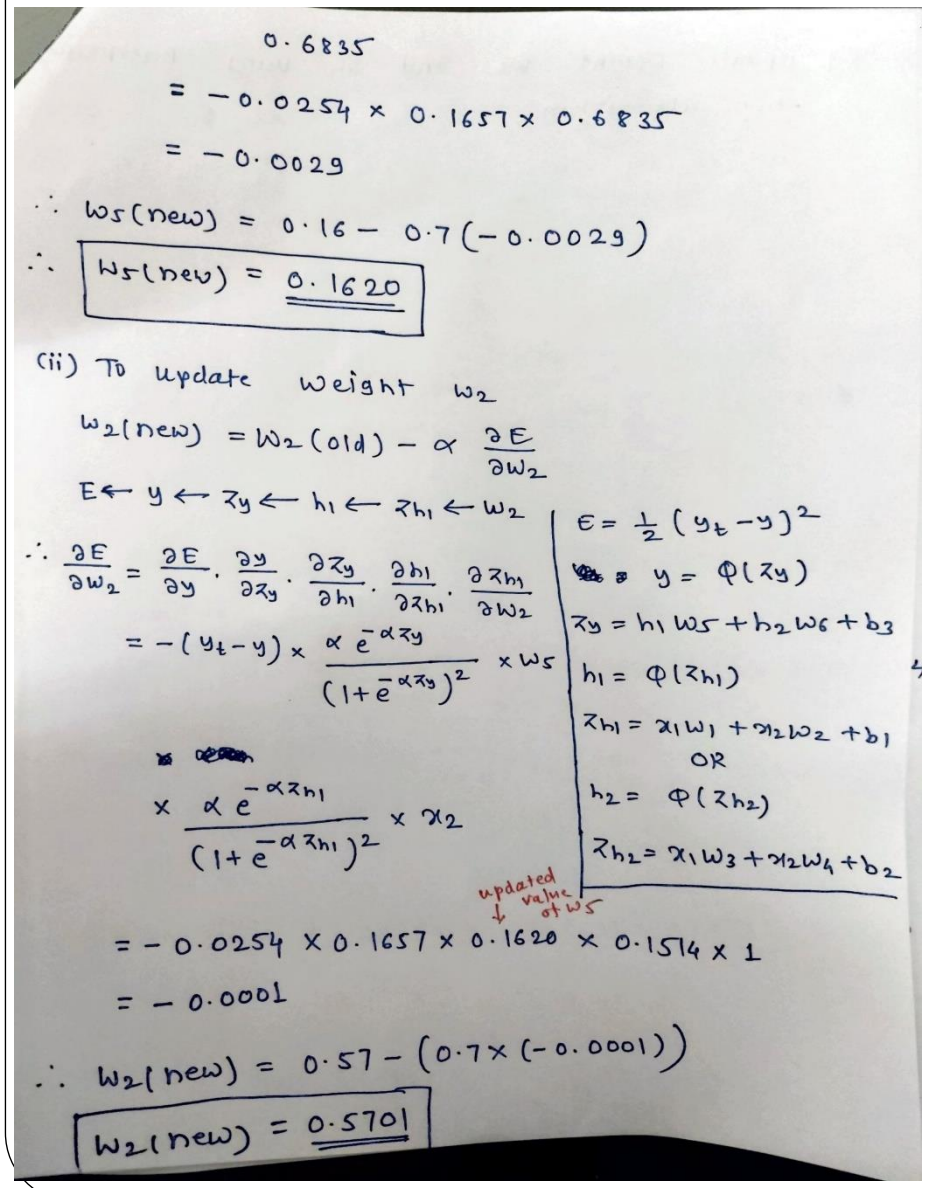
$$= -(0.41 - 0.3846) \times \frac{0.7\, e^{-(0.7 \times (-0.6718))}}{(1+e^{-(0.7 \times (-0.6718))})^2} \times$$

$E = \frac{1}{2}(y_t - y)^2$

$y = \phi(z_y)$

$z_y = h_1 w_5 + h_2 w_6 + b_3$

Examples on Backpropagation



0.6835

$= -0.0254 \times 0.1657 \times 0.6835$

$= -0.0029$

$\therefore w_5(new) = 0.16 - 0.7(-0.0029)$

$\therefore$ **$w_5(new) = 0.1620$**

(ii) To update weight w_2

$w_2(new) = w_2(old) - \alpha \frac{\partial E}{\partial w_2}$

$E \leftarrow y \leftarrow z_y \leftarrow h_1 \leftarrow z_{h_1} \leftarrow w_2$

$\therefore \frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y} \cdot \frac{\partial y}{\partial z_y} \cdot \frac{\partial z_y}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_{h_1}} \cdot \frac{\partial z_{h_1}}{\partial w_2}$

$= -(y_t - y) \times \frac{\alpha e^{-\alpha z_y}}{(1+e^{-\alpha z_y})^2} \times w_5$

$\times \frac{\alpha e^{-\alpha z_{h_1}}}{(1+e^{-\alpha z_{h_1}})^2} \times x_2$

$E = \frac{1}{2}(y_t - y)^2$

$y = \Phi(z_y)$

$z_y = h_1 w_5 + h_2 w_6 + b_3$

$h_1 = \Phi(z_{h_1})$

$z_{h_1} = x_1 w_1 + x_2 w_2 + b_1$

OR

$h_2 = \Phi(z_{h_2})$

$z_{h_2} = x_1 w_3 + x_2 w_4 + b_2$

updated value of w_5

$= -0.0254 \times 0.1657 \times 0.1620 \times 0.1514 \times 1$

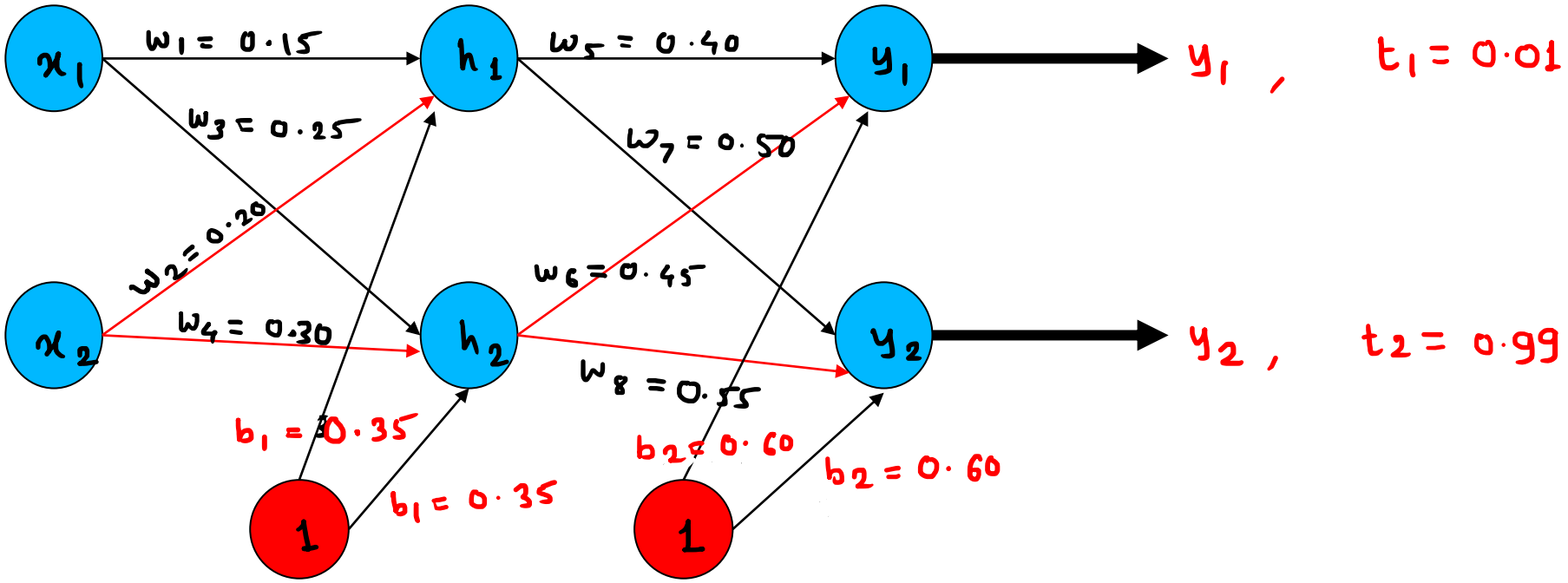
$= -0.0001$

$\therefore w_2(new) = 0.57 - (0.7 \times (-0.0001))$

$w_2(new) = 0.5701$

Backpropagation

- Q.2 Find output of the given ANN model and also update weights of the following ANN model using Backpropagation $\left[\varphi(z) = \frac{1}{1+e^{-z}}, \alpha = 0.5\right]$.



Backpropagation

Forward Bias:

(i) $z_{h_1} = x_1 w_1 + x_2 w_2 + b_1 = (0.15 \times 0.15) + (0.10 \times 0.20) + 0.35$

$= 0.39$

$\therefore h_1 = \varphi(z_{h_1}) = \frac{1}{1 + e^{-0.39}} = 0.60$

(ii) $z_{h_2} = x_1 w_3 + x_2 w_4 + b_2 = (0.15 \times 0.25) + (0.10 \times 0.30) + 0.35$

$= 0.42$

$\therefore h_2 = \varphi(z_{h_2}) = \frac{1}{1 + e^{-0.42}} = 0.60$

Backpropagation

Forward Bias:

(iii) $z_{y_1} = h_1 w_5 + h_2 w_6 + b_2 = (0.60 \times 0.40) + (0.60 \times 0.45) + 0.60$

$= 1.11$

$\therefore y_1 = \varphi(z_{y_1}) = \frac{1}{1 + e^{-1.11}} = 0.75$

(iv) $z_{y_2} = h_1 w_7 + h_2 w_8 + b_2 = (0.60 \times 0.50) + (0.60 \times 0.55) + 0.60$

$= 1.23$

$\therefore y_2 = \varphi(z_{y_2}) = \frac{1}{1 + e^{-1.23}} = 0.77$

Backpropagation

Backward Bias:

We have, total error $(E) = E_1 + E_2$

$\therefore$ Total Error $(E) = \frac{1}{2}(t_1 - y_1)^2 + \frac{1}{2}(t_2 - y_2)^2$

$= \frac{1}{2}(0.01 - 0.75)^2 + \frac{1}{2}(0.99 - 0.77)^2$

$= 0.30$

Now let's update the weights of second layer

(1) Error in $w_5 = \frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial y_1} \cdot \frac{\partial y_1}{\partial z_{y_1}} \cdot \frac{\partial z_{y_1}}{\partial w_5}$

$= -(t_1 - y_1) \cdot y_1(1 - y_1) \cdot h_1$

$= -(0.01 - 0.75) \times 0.75(1 - 0.75) \times 0.60$

Backpropagation

Backward Bias:

$\therefore$ Error in $w_5 = 0.08$

$$\therefore \quad w_5(\text{new}) = w_5(\text{old}) - \eta \cdot \frac{\partial E}{\partial w_5}$$

$$= 0.40 - (0.5 \times 0.08)$$

$$= \underline{\underline{0.36}}$$

Using similar calculations, we have

$w_6(\text{new}) = 0.408666186$

$w_7(\text{new}) = 0.511301270$

$w_8(\text{new}) = 0.561370121$

Backpropagation

Backward Bias:

* For First layer

(i) Error in $w_1 = \frac{\partial E}{\partial w_1}$

$$= \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_{h_1}} \cdot \frac{\partial z_{h_1}}{\partial w_1}$$

$$= \frac{\partial E}{\partial y_1} \cdot \frac{\partial y_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_{h_1}} \cdot \frac{\partial z_{h_1}}{\partial w_1}$$

$$= \frac{\partial E}{\partial y_1} \cdot \frac{\partial y_1}{\partial z_{y_1}} \cdot \frac{\partial z_{y_1}}{\partial h_1} \cdot \frac{\partial h_1}{\partial z_{h_1}} \cdot \frac{\partial z_{h_1}}{\partial w_1}$$

$$= -(t_1 - y_1) \cdot y_1(1-y_1) \cdot w_5 \cdot h_1(1-h_1) \cdot x_1$$

Backpropagation

Backward Bias:

$$= -(0.01 - 0.75) \times 0.75(1 - 0.75) \times 0.36 \times$$

$$0.60(1 - 0.60) \times 0.15$$

$$\frac{\partial E}{\partial w_1} = 0.0017982$$

$$\therefore \quad w_1(new) = w_1(old) - \alpha \frac{\partial E}{\partial w_1}$$

$$= 0.15 - (0.5 \times 0.0017982)$$

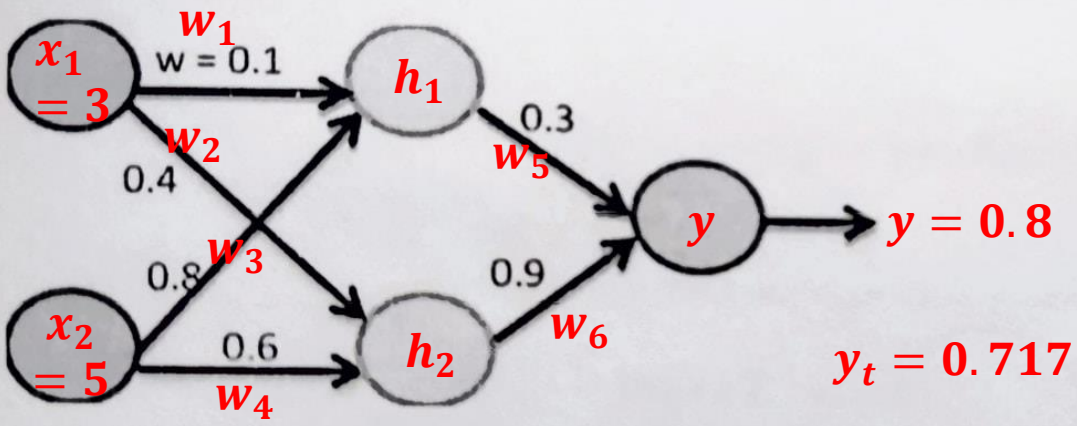
$$w_1(new) = 0.1491$$

similarly, we can find updated weights of

$w_2(new) = 0.19956143$, $w_3(new) = 0.24975114$,

$w_4(new) = 0.29950229$

Q.3 The actual outputs $(Y)_A = 0.8$ and expected output $(Y)_E = 0.717$ where the activation function is sigmoidal function with learning rate $\alpha = 0.8$ for the input set [3,5]. Update second layer weights using Backpropagation Algorithm.



$y_t = 0.717$

Step I: Forward Bias

(1) h_1

$$z_{h_1} = 3(0.1) + 5(0.8) = 4.3$$

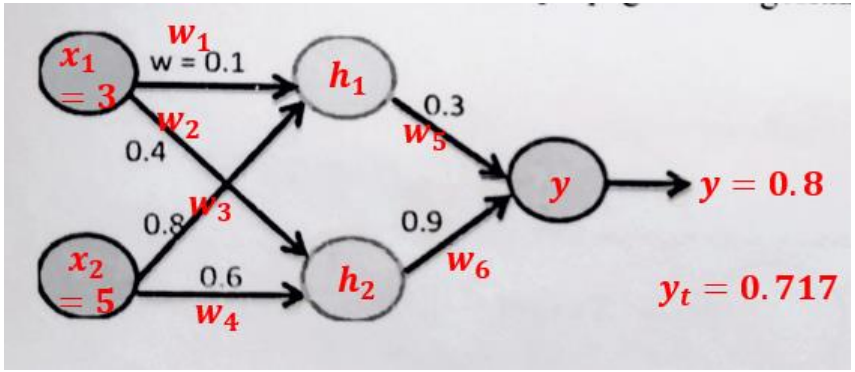
$$h_1 = \frac{1}{1 + e^{-0.8 z_{h_1}}} = 0.9689$$

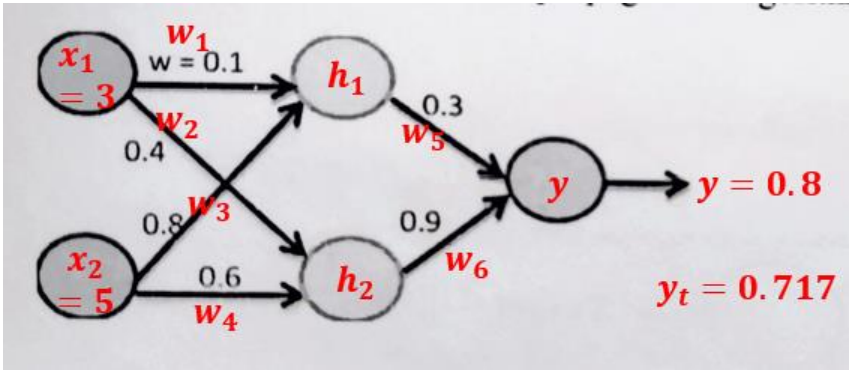
(2) h_2

$$z_{h_2} = 3(0.4) + 5(0.6) = 4.2$$

$$h_2 = \frac{1}{1 + e^{-0.8 z_{h_2}}} = 0.9664$$

(3) $y = 0.8$





Step II: Error

$$E = \frac{1}{2}(y_t - y)^2$$

Update the weight w_3

$$w_i(new) = w_i(old) - \alpha \frac{\partial E}{\partial w_i}$$

$$w_5(new) = w_5(old) - \alpha \frac{\partial E}{\partial w_5}$$

$$E \leftarrow y \leftarrow z_y \leftarrow w_5$$

$$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial z_y} \times \frac{\partial z_5}{\partial w_5}$$

$$E = \frac{1}{2}(y_t - y)^2$$

$$y = \phi(z_y)$$

$$z_y = h_1 w_5 + h_2 w_6$$

$$\frac{\partial E}{\partial w_5} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial z_y} \times \frac{\partial z_y}{\partial w_5}$$

$$\frac{\partial E}{\partial w_5} = -(y_t - y) \times \frac{0.8e^{-0.8z_y}}{\left(1+e^{-0.8z_y}\right)^2} \times h_1$$

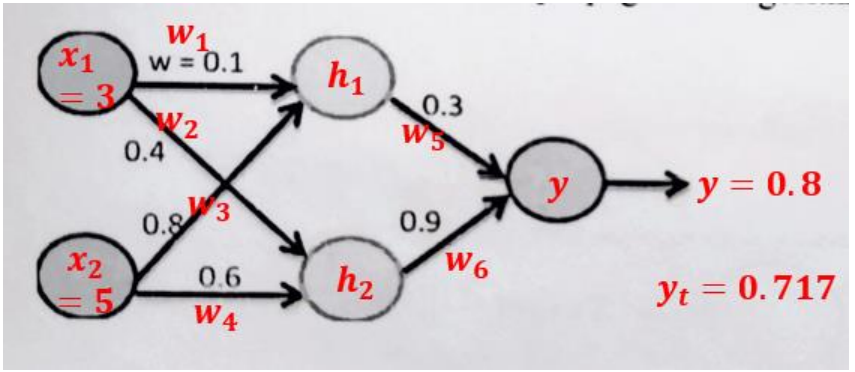
$$\frac{\partial E}{\partial w_5} = -(0.717 - 0.8) \times \frac{0.8e^{-(0.8\times 1.1604)}}{(1+e^{-0.8\times 1.1604})^2} \times 0.9689$$

$$\frac{\partial E}{\partial w_5} = 0.0830 \times (0.1624) \times 0.9689$$

$$\frac{\partial E}{\partial w_5} = 0.0131$$

$$w_5(new) = w_5(old) - \alpha \frac{\partial E}{\partial w_5}$$

$$= 0.3 - 0.8 \times (0.0131) = \mathbf{0.2895}$$



$$E = \frac{1}{2}(y_t - y)^2$$

$$y = \phi(z_y) = \frac{1}{1 + e^{-0.8z_y}}$$

$$z_y = h_1 w_5 + h_2 w_6$$

$$z_y = 0.9689(0.3) + 0.9664(0.9) = 1.1604$$

Update the weight w_3

$$w_i(new) = w_i(old) - \alpha \frac{\partial E}{\partial w_i}$$

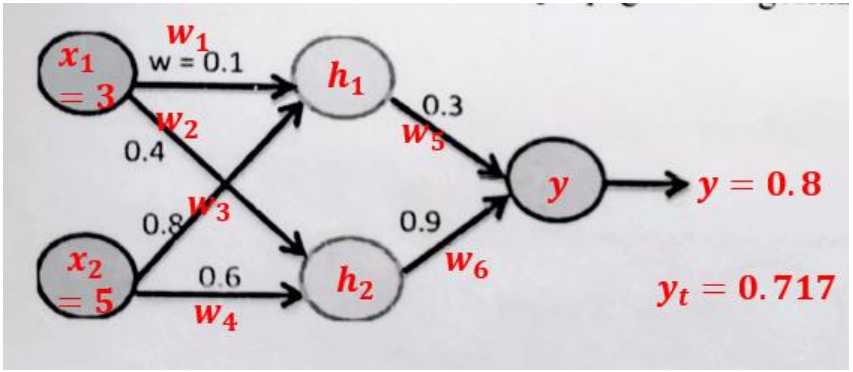
$$w_3(new) = w_3(old) - \alpha \frac{\partial E}{\partial w_3}$$

$$E \leftarrow y \leftarrow z_y \leftarrow h_1 \leftarrow z_{h_1} \leftarrow w_3$$

$$\frac{\partial E}{\partial w_3} = \frac{\partial E}{\partial y} \times \frac{\partial y}{\partial z_y} \times \frac{\partial z_y}{\partial h_1} \times \frac{\partial h_1}{\partial z_{h_1}} \times \frac{\partial z_{h_1}}{\partial w_3}$$

$$\frac{\partial E}{\partial w_3} = -(y_t - y) \times \frac{0.8e^{-0.8z_y}}{\left(1 + e^{-0.8z_y}\right)^2}$$

$$\times \mathrm{w}_5 \times \frac{0.8e^{-0.8z_{h_1}}}{\left(1+e^{-0.8z_{h_1}}\right)^2} \times x_2$$



$$E = \frac{1}{2}(y_t - y)^2$$

$$y = \phi(z_y)$$

$$z_y = h_1 w_5 + h_2 w_6$$

$$h_1 = \phi(z_{h_1})$$

$$z_{h_1} = x_1 w_1 + x_2 w_3$$

$$h_2 = \phi(z_{h_2})$$

$$z_{h_2} = x_1 w_2 + x_2 w_4$$

$$\frac{\partial E}{\partial w_3} = -(0.717 - 0.8) \times \frac{0.8e^{-(0.8\times1.1604)}}{(1+e^{-(0.8\times1.1604)})^2}$$

$$\times 0.2895 \times \frac{0.8e^{-(0.8\times4.3)}}{(1+e^{-(0.8\times4.3)})^2} \times 5$$

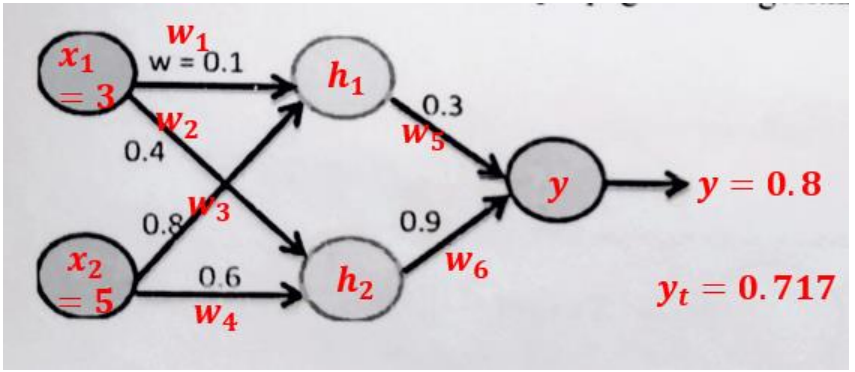
$$\frac{\partial E}{\partial w_3} = -0.083 \times 0.1624 \times 0.2895 \times 0.0241 \times 5$$

$$\frac{\partial E}{\partial w_3} = -0.0005$$

$$w_3(new) = w_3(old) - \alpha \frac{\partial E}{\partial w_3}$$

$$= 0.8 - 0.7 \times (-0.0005)$$

$$= 0.7997$$



$$E = \frac{1}{2}(y_t - y)^2$$

$$y = \phi(z_y)$$

$$z_y = h_1 w_5 + h_2 w_6$$

$$h_1 = \phi(z_{h_1})$$

$$z_{h_1} = x_1 w_1 + x_2 w_3 = 3(0.1) + 5(0.8) = 4.3$$

$$h_2 = \phi(z_{h_2})$$

$$z_{h_2} = x_1 w_2 + x_2 w_4$$

$$z_y = 0.9689(0.3) + 0.9664(0.9) = 1.1604$$

Error Calculation Methods

Question: Define Loss / Error function and explain its types.

Error calculation occur at two levels

1) **Local Error function:** It is difference between target output and actual output
2) **Global Error function:** All the local errors are aggregated together to form global error.

The global error is the measurement of how well a neural network performs to the entire training set.

Use of Error Function

a) The error function in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model

Error Calculation Methods

Question: Define Loss / Error function and explain its types with one example.

b) From the error function, we can derive the gradients which are used to update the weights

c) Error functions define what a good prediction is and isn't.

d) In short, choosing the right loss function dictates how well your estimator will be

Following are the some important global error calculations methods

a) Sum of squares error (ESS)
b) Mean Absolute Error (MAE)
c) Mean square error (MSE)
d) Root mean square (RMS)

Error Calculation Methods

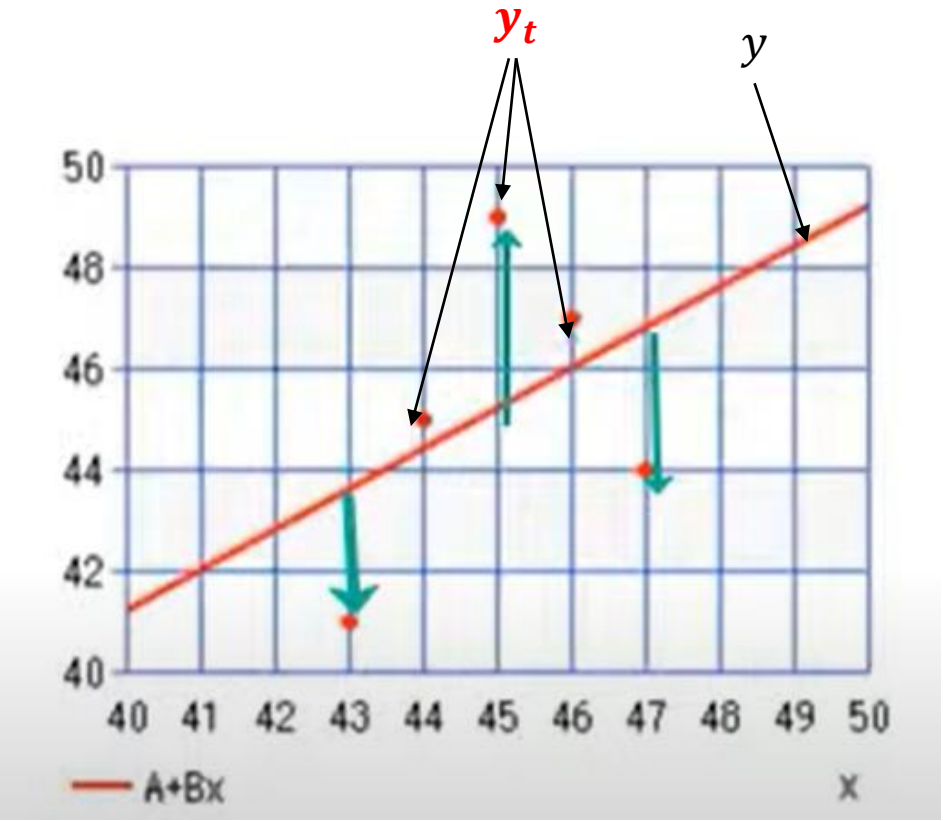
a) Sum of squares error (SSE)

It is actually squared difference of target output and actual output.

i.e. $E = \sum_{i=1}^{n} \left(y_{t_i} - y_i\right)^2$

When to use?

Use MSE when doing regression, believing that your target, conditioned on the input, is normally distributed.



Error Calculation Methods

Q.1 Find SSE error present in the following data

y	y_t
43.6	41
44.4	45
45.2	49
46	47
46.8	44

Solution

$$E = \sum_{i=1}^{n} (y_{t_i} - y_i)^2$$

y	y_t	$y_t - y$	$(y_t - y)^2$	$\sum_{i=1}^{n} (y_{t_i} - y_i)^2$
43.6	41	-2.6	6.76	30.4
44.4	45	0.6	0.36	
45.2	49	3.8	14.44	
46	47	1	1	
46.8	44	-2.8	7.84	

Error Calculation Methods

a) Mean Absolute Error

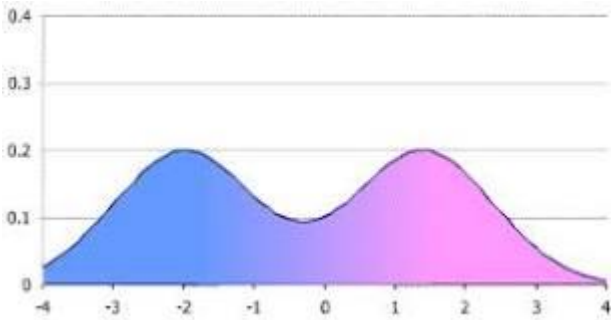
The Mean absolute error represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.

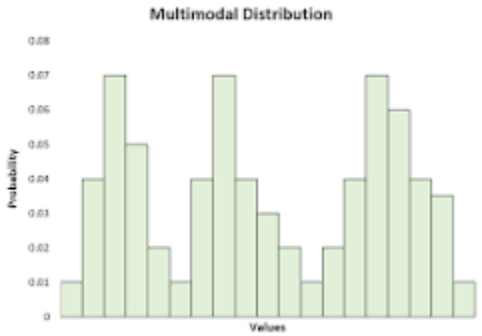
$$\text{MAE} = \frac{1}{n}\sum |y_t - y|$$

Use Mean absolute error when you are doing regression and don't want outliers to play a big role. It can also be useful if you know that your distribution is multimodal, and it's desirable to have predictions at one of the modes, rather than at the mean of them.

Error Calculation Methods

In statistics, a multimodal distribution is a probability distribution with more than one mode. These appear as distinct peaks (local maxima) in the probability density function, as shown in Figures 1 and 2.





Error Calculation Methods

Q.1 Find MAE error present in the following data

y	y_t
43.6	41
44.4	45
45.2	49
46	47
46.8	44

Solution

y	y_t	$\lvert y_t - y \rvert$	$\frac{1}{n}\lvert y_t - y \rvert$
43.6	41	2.6	$\frac{1}{5}(10.8) = \mathbf{2.16}$
44.4	45	0.6	
45.2	49	3.8	
46	47	1	
46.8	44	2.8	

Error Calculation Methods

b) Mean square error

$$E = \frac{1}{n}\sum_{i=1}^{n}\left(y_{t_i} - y_i\right)^2$$

Q.1 Find MSE error present in the following data

y	y_t
43.6	41
44.4	45
45.2	49
46	47
46.8	44

Error Calculation Methods

Solution

y	y_t	$y_t - y$	$(y_t - y)^2$	$\frac{1}{n}\sum_{i=1}^{n}(y_{t_i} - y_i)^2$
43.6	41	-2.6	6.76	6.08
44.4	45	0.6	0.36	
45.2	49	3.8	14.44	
46	47	1	1	
46.8	44	-2.8	7.84	

Q.2 Find SSE error present in the following data

Actual	Predicted
10	9
13	14
14	12
9	10
17	15

MSE=2.2

Error Calculation Methods

b) Root mean square (RMS)

$$E = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_{t_i} - y_i\right)^2}$$

Q.1 Find RMS error present in the following data

y	y_t
43.6	41
44.4	45
45.2	49
46	47
46.8	44

Error Calculation Methods

Solution

y	y_t	$y_t - y$	$(y_t - y)^2$	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{t_i} - y_i)^2}$
43.6	41	-2.6	6.76	2.47
44.4	45	0.6	0.36	
45.2	49	3.8	14.44	
46	47	1	1	
46.8	44	-2.8	7.84	

Q.2 Find RMS error present in the following data

Actual	Predicted
10	9
13	14
14	12
9	10
17	15

RMS=1.48

Weights Initialization

Key-points to remember for weights initialization

- **Weights should be small**

Large weights cause exploding gradient problem specially while using sigmoid activation function.

- **Weights should not be same.**

Same weights will prevent the neural network from learning new features.

- **Weights should have good variance.**

This will help each of the neurons to learn new features.

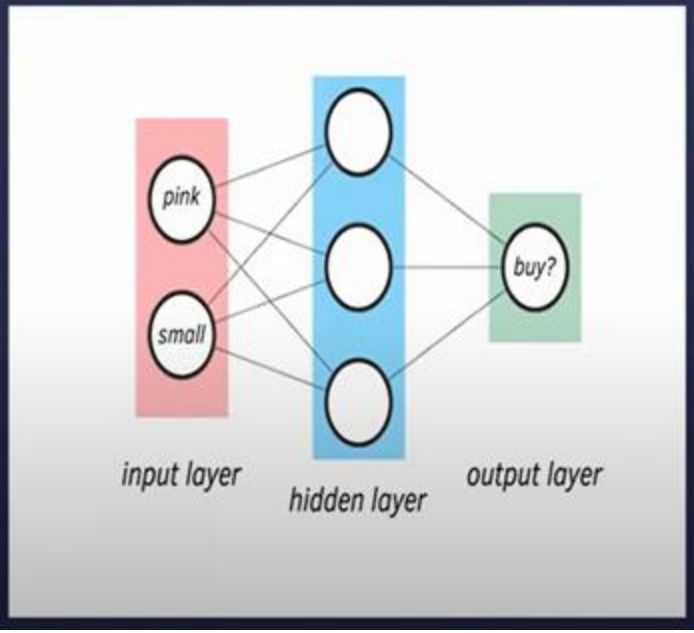
Weights Initialization

Uniform Distribution

- Here the weights are selected from a uniform distribution given by

$$W_{ij} \approx \text{Uniform}\left(\frac{-1}{\sqrt{n_i}}, \frac{1}{\sqrt{n_i}}\right)$$

- Based on this range the weights are initialized for each layer.



Weights Initialization

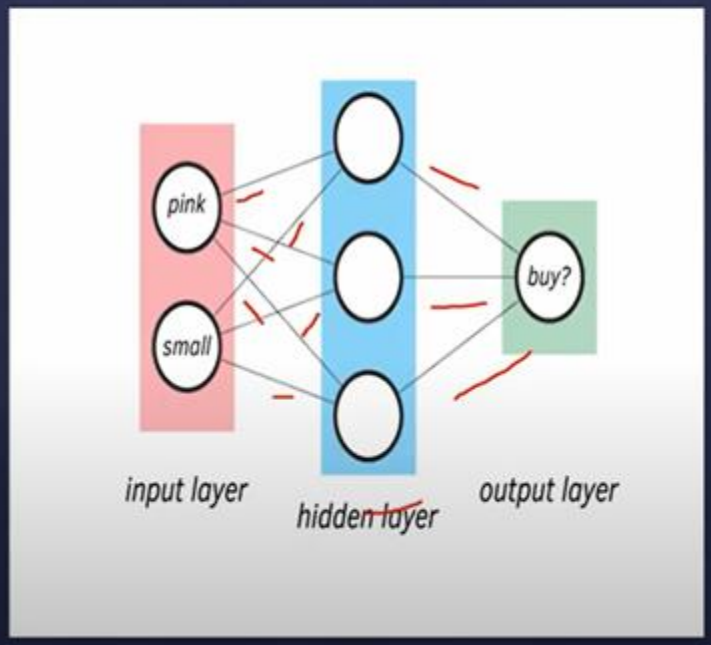
Xavier/Glorot Distribution

- Xavier/Glorot Normal

$$W_{ij} \approx \text{ND}(0, \sigma) \text{ where } \sigma = \sqrt{\frac{2}{n_i + n_o}}$$

- Xavier/Glorot Uniform

$$W_{ij} \approx \text{Uniform}\left(\frac{-\sqrt{6}}{n_i + n_o}, \frac{\sqrt{6}}{n_i + n_o}\right)$$



Weights Initialization

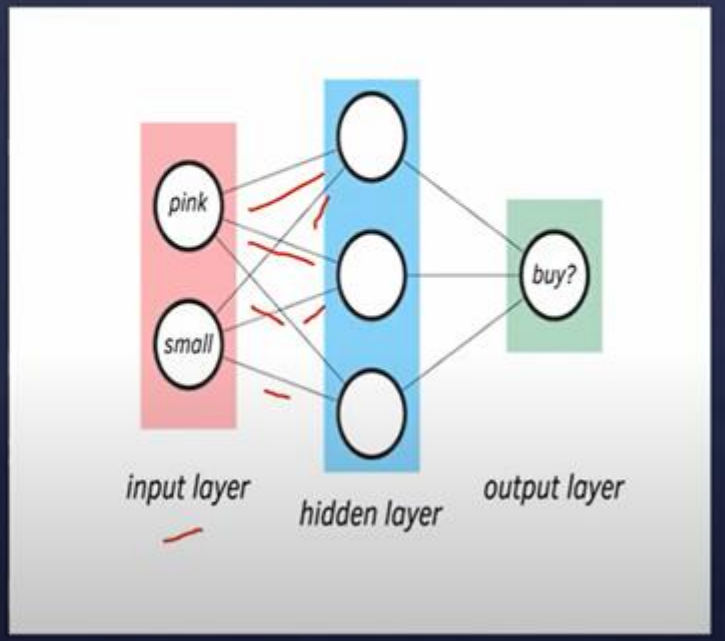
He init technique

- He init Normal

$$W_{ij} \approx \text{ND}(0, \sigma) \text{ where } \sigma = \sqrt{\frac{2}{n_i}}$$

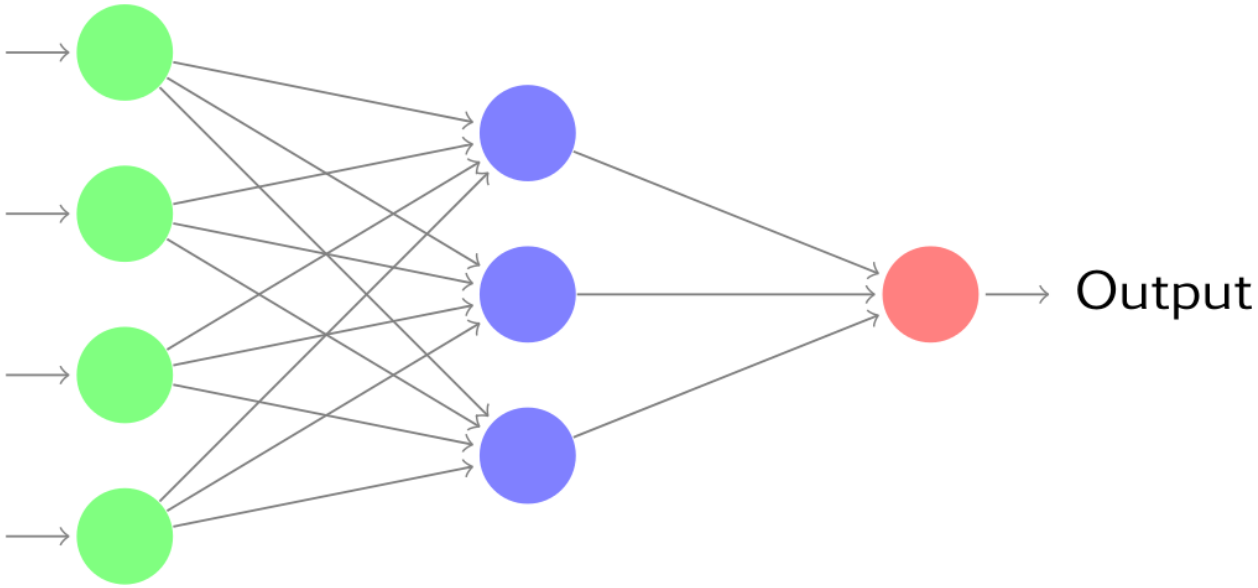
- He init Uniform

$$W_{ij} \approx \text{Uniform}\left(\frac{-\sqrt{6}}{n_i}, \frac{\sqrt{6}}{n_i}\right)$$



Weights Initialization

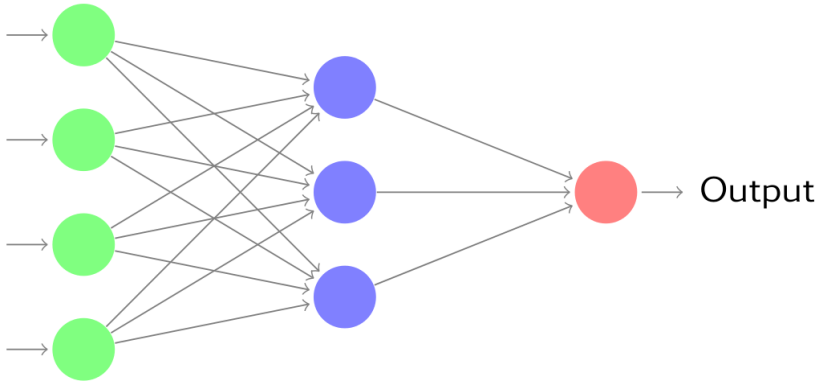
Question: Find the range of initial weights of first layer by three weight initialization techniques



Hint: Use Uniform Distribution, Xavier Normal & Uniform & He init Techniques and find range of the initial weights

Weights Initialization

Question: Find the range of initial weights of first layer by three weight initialization techniques



Solution :

① Uniform Distribution

$$= \left(-\frac{1}{\sqrt{n_i}}, \frac{1}{\sqrt{n_i}}\right)$$

$$= \left(-\frac{1}{\sqrt{12}}, \frac{1}{\sqrt{12}}\right) \; ; \; n_i = \text{no. of edges in input layer} = 4 \times 3 = 12$$

$$= (-0.2887, 0.2887)$$

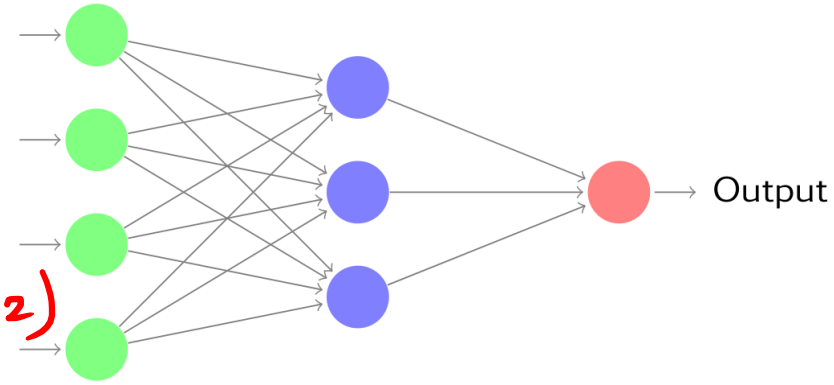
Weights Initialization

Question: Find the range of initial weights of first layer by three weight initialization techniques

(2) Xavier normal

$$= \left(0, \frac{2}{\sqrt{n_i + n_o}}\right)$$

$$= \left(0, \frac{2}{\sqrt{12+3}}\right) = (0, 0.2582)$$



n_i = no. of edges in input layer

$= 4 \times 3 = 12$

n_o = no. of edges in output layer

$= 1 \times 3 = 3$

(3) Xavier uniform

$$= \left(\frac{-\sqrt{6}}{n_i + n_o}, \frac{\sqrt{6}}{n_i + n_o}\right)$$

$$= \left(\frac{-\sqrt{6}}{15}, \frac{\sqrt{6}}{15}\right)$$

$$= (-0.1633, 0.1633)$$

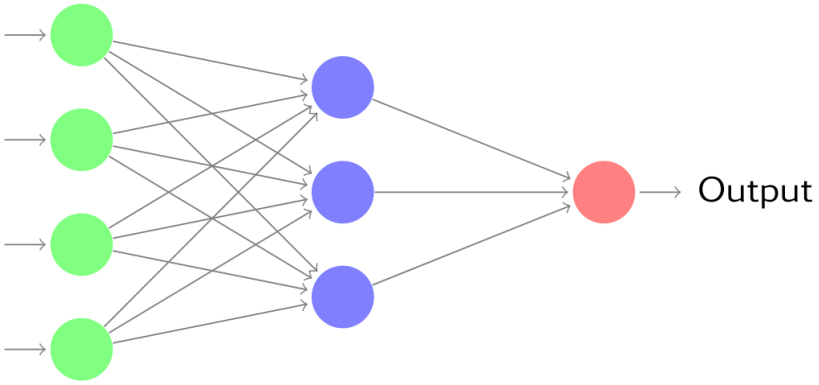
Weights Initialization

Question: Find the range of initial weights of first layer by three weight initialization techniques

(4) He init Normal

$= \left(0, \frac{2}{\sqrt{n_i}}\right)$

$= \left(0, \frac{2}{\sqrt{12}}\right) = (0, 0.5774)$



(5) He init Uniform

$= \left(-\frac{\sqrt{6}}{n_i}, \frac{\sqrt{6}}{n_i}\right)$

$= \left(-\frac{\sqrt{6}}{12}, \frac{\sqrt{6}}{12}\right)$

$= (-0.2041, 0.2041)$

n_i = no. of edges in input layer

$= 4 \times 3 = 12$

n_o = no. of edges in output layer

$= 1 \times 3 = 3$

Associative memory

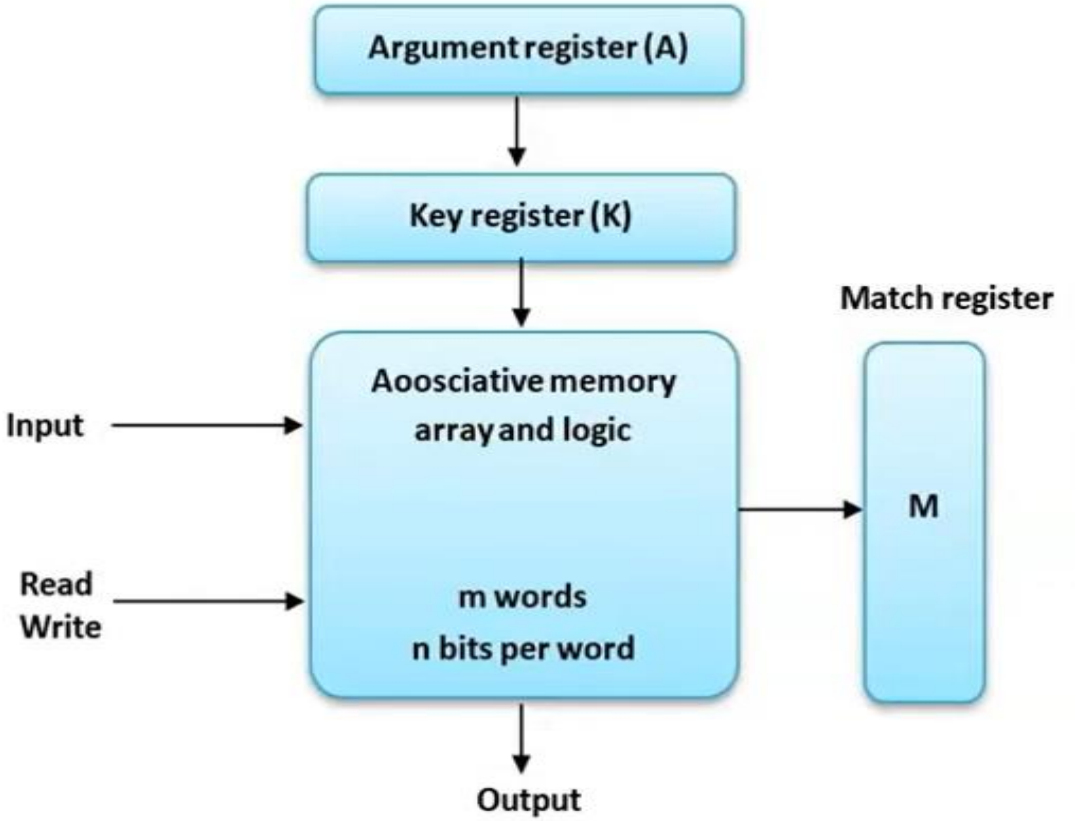
Question: What is Associative Memory? Explain its types.

- It is called as Content Addressable Memory (CAM)
- This type of memory is accessed simultaneously and in parallel on the basis of **data content** rather than by specific address or location.
- How memory write word?
  1. When a word is **written** in associative memory, no address is given.
  2. The memory is capable of finding an empty location to store the word.
- How memory read the word?
  1. When a word is to be read from an associative memory, then the content of the word or part of the word is specified.
  2. The memory locates all words that matches specified content and marks them for reading.

Associative memory

- It takes relatively less time to find an item based on content rather than by an address.

Block Diagram of Associative Memory



Associative memory

Argument Register (A):
It contains the word to be searched

Key Register (K):
This specifies which part of the argument word needs to be compared with words in memory.

If all bits in register are 1, the entire word should be compared. Else only bits having 1 will be compared.

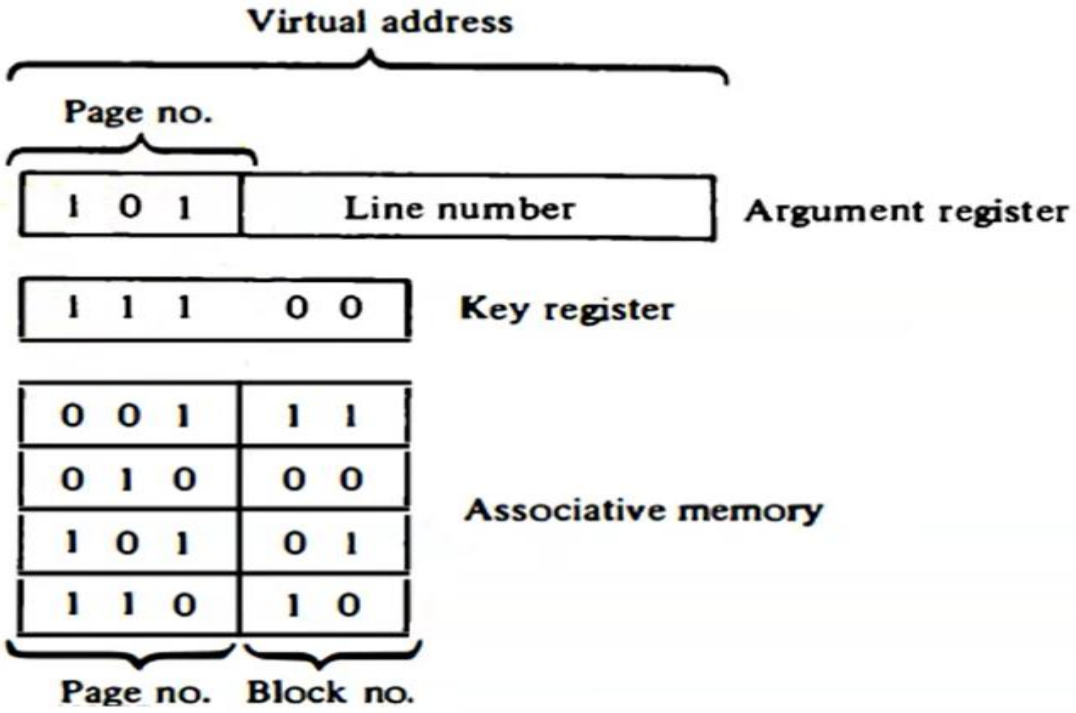
Associative memory array:
It contains the words which are to be compared with the argument word.

Associative memory

Match Register (M):

After the matching process, the bits corresponding to matching words in match register are set to be 1.

Example: Associative Memory page table



A: **101** 111100
K: 111 000000
Word 1: 100 111100 no match (0)
Word 2: **101** 000001 match (1)

Associative memory

Advantage

- Searching Process is very fast
- Suitable for parallel search

Disadvantage

- More costly than main memory

Associative memory

Following are the two types of associative memories we can observe:

1. Auto Associative Memory
2. Hetero Associative memory

Auto Associative Memory:

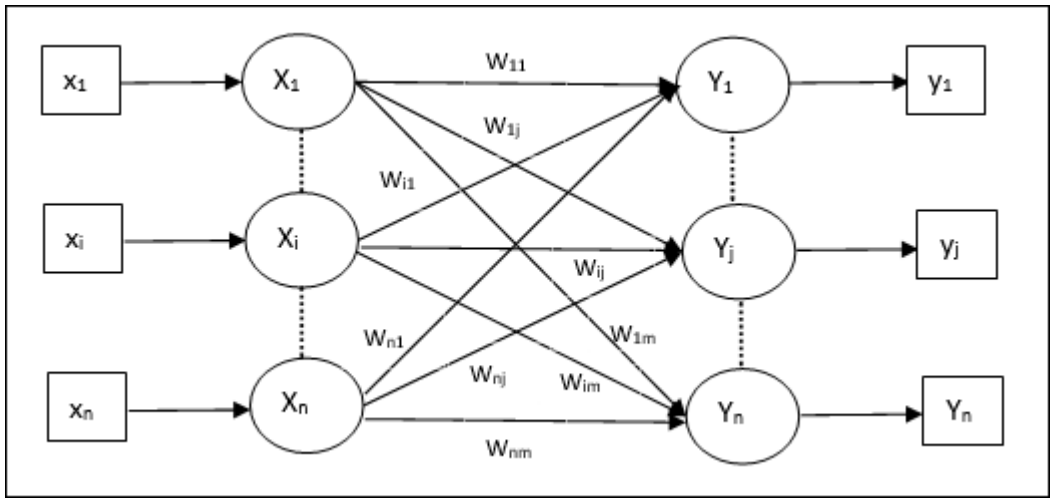
Auto associative memory is capable of retrieving a piece of data only from partial information of that of piece of data.

Neural network using auto associative memory are called **Auto Associative Memory.**

Auto Associative memory

Architecture

As shown in the following figure, the architecture of Auto Associative memory network has '*n*' number of input training vectors and similar '*n*' number of output target vectors.



Auto Associative memory

Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Algorithm:

Let X be given vector

Step 1: Consider weight matrix, $W = X^T X$

Step 2: Net input $z = XW$

Step 3: Consider activation function

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

Auto Associative memory

Example:

Train the auto associative network for input vector $[-1 \quad 1 \quad 1 \quad 1]$ and also test the network for same input vector.
Test the auto associative network with one missing, one mistake, two missing and two mistake entries in test vector.

Solution:

$$W = X^T X = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

Training network with same input vector:

$$X = \begin{bmatrix} -1 & 1 & 1 & 1 \end{bmatrix}$$

Auto Associative memory

$$z = XW = \begin{bmatrix} -1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -4 & 4 & 4 & 4 \end{bmatrix}$$

$$\therefore y = \begin{bmatrix} -1 & 1 & 1 & 1 \end{bmatrix}$$ Correct Response

Training network with one missing entry:

$$X = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}$$

$$z = XW = \begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -3 & 3 & 3 & 3 \end{bmatrix}$$

$$\therefore y = \begin{bmatrix} -1 & 1 & 1 & 1 \end{bmatrix}$$ Correct Response

Auto Associative memory

Training network with two missing entry:

$$X = \begin{bmatrix} 0 & 0 & -1 & 1 \end{bmatrix}$$

$$z = XW = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -2 & 2 & 2 & 2 \end{bmatrix}$$

$$\therefore y = \begin{bmatrix} -1 & 1 & 1 & 1 \end{bmatrix}$$ Correct Response

Training network with two mistake entry:

$$X = \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix}$$

$$z = XW = \begin{bmatrix} -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\therefore y = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$$ Not getting Correct Response

Auto Associative memory

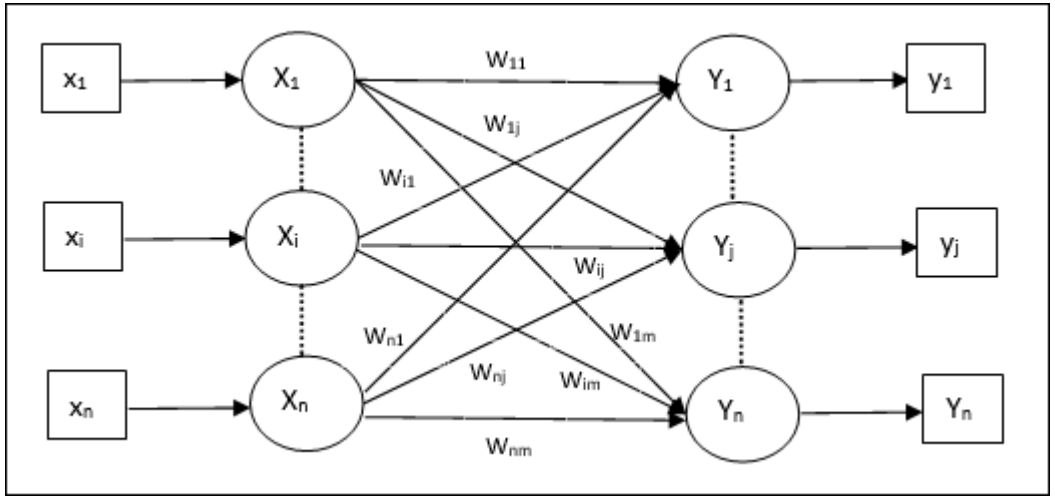
Question: Train the auto associative network for input vector [−1 -1 1 1] and [1 -1 1 -1] also test the network for two missing values.

Hetero Associative memory

Hetero Associative memory is capable of retrieving a piece of data of one category upon presentation of data from another category.

In this case

1) Training input vector X and target output vector t are different.
2) It uses Hebb's / Delta Rule to find the weight matrix



Hetero Associative memory

Algorithm:

Step 1: Consider weight matrix for each input vector as $W_i = X_i^T Y_i$

Step 2: Weight matrix for entire model $\boldsymbol{W} = \sum X_i^T Y_i$

Step 3: Calculate net input $z = x_1 w_1 + x_2 w_2 + \cdots + x_k w_k$

Step 4: Take activation function as,

$$y = \begin{cases} 1 & if\ z > 0 \\ 0 & if\ z = 0 \\ -1 & if\ z < 0 \end{cases}$$

Hetero Associative memory

Example:

Train a heteroassociative network to store the given bipolar input vector $S = (S_1, S_2, S_3, S_4)$ to the output vector $t = (t_1, t_2)$. Also test performance of network with missing and mistaken data.

S_1	S_2	S_3	S_4	t_1	t_2
1	-1	-1	-1	-1	1
1	1	-1	-1	-1	1
-1	-1	-1	1	1	-1
-1	-1	1	1	1	-1

Hetero Associative memory

Solution:

(1) For pair, $X_1 = (1,-1,-1,-1)$ and $Y_1 = (-1,1)$

$$W_1 = X_1^T Y_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}$$

(2) For pair, $X_2 = (1,1,-1,-1)$ and $Y_2 = (-1,1)$

$$W_2 = X_2^T Y_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}$$

Hetero Associative memory

(3) For pair, $X_3 = (-1,-1,-1,1)$ and $Y_3 = (1,-1)$

$$W_3 = X_3^T Y_3 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}$$

(4) For pair, $X_4 = (-1,-1,1,1)$ and $Y_4 = (1,-1)$

$$W_4 = X_4^T Y_4 = \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}$$

Hetero Associative memory

Final weight matrix,

$$W = W_1 + W_2 + W_3 + W_4 = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}$$

$$W = \begin{bmatrix} -4 & 4 \\ -2 & 2 \\ 2 & -2 \\ 4 & -4 \end{bmatrix}$$

Case I: With missing data entries

$$x = [0 \quad 1 \quad 0 \quad -1]$$

Hetero Associative memory

Net input,

$$z = xW = \begin{bmatrix} 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -4 & 4 \\ -2 & 2 \\ 2 & -2 \\ 4 & -4 \end{bmatrix} = \begin{bmatrix} -6 & 6 \end{bmatrix}$$

$$\therefore y = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

Correct Response

Hetero Associative memory

Case II: With mistaken data,

$$x = [-1 \quad 1 \quad 1 \quad -1]$$

Net input,

$$z = xW = \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -4 & 4 \\ -2 & 2 \\ 2 & -2 \\ 4 & -4 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

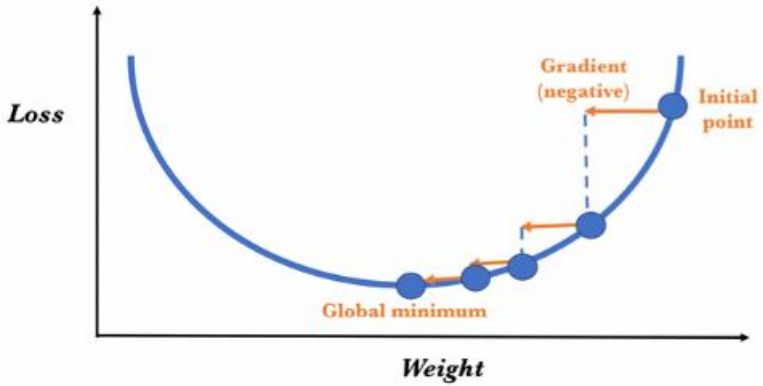
$$\therefore y = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

Not getting Correct Response

Optimization of Error Function /Cost Function

Concept of Optimization

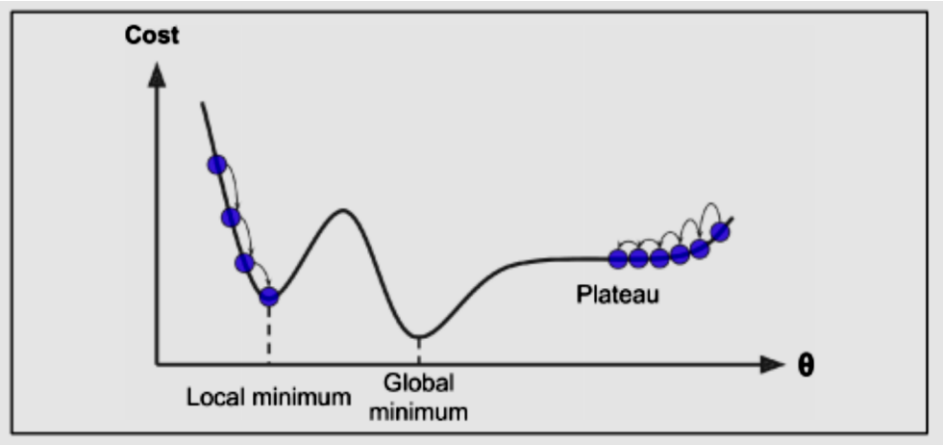
- The error function is a measure of how good our model is at making predictions. The shape of a cost function defines our goal of optimization.
- If the shape of the cost function is a convex function, our goal is to find the only minimum.
- If the shape of the cost function is not a convex function, our goal is to find the lowest possible value in the neighborhood.
- The idea behind gradient descent is converging to a solution, which could be a local minimum in the neighborhood or the global minimum.



Concept of Optimization

Non Convex Error Function

As mentioned before, our goal is to reach a global minimum. But at times when our error function has an irregular curve (mostly in the case of deep learning neural networks), with random initialization in place, one might reach a local minimum , which is not as good as the global minimum. One way to overcome this is by using the concept of momentum.



Concept of Optimization

All First-order iterative optimization algorithms are used to minimize the cost function.

Types of First Order Optimization Algorithm

Question: Explain Gradient Descent optimization algorithm. Explain its types.
Question: Explain any four optimizers in neural network and their advantage and disadvantages.

1) Gradient Descent Optimization Algorithm

It's a ***first-order optimization algorithm*** because, in every iteration, the algorithm takes the first-order derivative for updating the weights and bias.

Concept of Optimization

It is one of the most popular algorithm to perform optimization(minimize the error function) and so far the most common way to optimize neural network.

Gradient Descent is a way to minimize an objective function by updating weights in the opposite direction of the gradient of the opposite function (negative slope direction).

The learning rate α determines the size of the steps we take to reach a (local) minima. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a minima.

Gradient Descent optimization Algorithm

Algorithm:

Step 1: Find the slope of the objective function with respect to each weight

Step 2: Pick a random initial value for the weights.

Step 3: Evaluate gradient function i.e. $\frac{\partial E}{\partial W}$ at initial value

Step 4: Calculate the step sizes for each feature as

step size = gradient * learning rate.$= \alpha . \frac{\partial E}{\partial W}$

Step 5: Calculate the new weights as : new params. = old params. - step size

i.e. $w_i(new) = w_i(old) - \alpha . \frac{\partial E}{\partial W}$

Step 6: Repeat steps 3 to 5 until gradient is almost 0.

First Order Optimization Algorithms

Types of Gradient Descent optimization algorithms:

There are three variants of gradient descent, which differ in how much data we use to compute the gradient of the objective function. Depending on the amount of data, we make a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

1. Batch Gradient Algorithm (GD)
2. Stochastic Gradient Algorithm (SGD)
3. Mini batch Gradient Algorithm (MBGD)

Concept of Optimization

The Silly Training Dataset

Training examples

Student	X1 Science grade [%]	X2 Chemistry grade [%]	X3 # hours studied [hrs]	Y_{act} Final math grade[%]
1	60	80	5	82
2	70	75	7	94
3	50	55	10	45
4	40	56	7	43

Concept of Optimization

1) Stochastic Gradient Algorithm (SGD)

If we input one training example at a time

And carry out forward and back propagation to update the weights

That is called stochastic gradient descent

The Silly Training Dataset

Training examples

Student	X1 Science grade [%]	X2 Chemistry grade [%]	X3 # hours studied [hrs]	Y_{act} Final math grade[%]
1	60	80	5	82
2	70	75	7	94
3	50	55	10	45
4	40	56	7	43

Concept of Optimization

2. Stochastic Gradient Algorithm (SGD)

Advantages

- **Frequent updates of model parameter/weight**
- **Requires less Memory.**
- **Allows the use of large data sets as it has to update only one example at a time.**

Disadvantages

- **The frequent can also result in noisy gradients which may cause the error to increase instead of decreasing it.**
- **High Variance.**
- **Frequent updates are computationally expensive.**

Concept of Optimization

2) Mini batch Gradient Algorithm (MBGD)

If we input multiple training examples at a time (but less than the entire training dataset), then that is called...

mini-batch gradient descent

Batch Size=2

The Silly Training Dataset

Training examples

Student	X1 Science grade [%]	X2 Chemistry grade [%]	X3 # hours studied [hrs]	Y_{act} Final math grade[%]
1	60	80	5	82
2	70	75	7	94
3	50	55	10	45
4	40	56	7	43

The Silly Training Dataset

Training examples

Student	X1 Science grade [%]	X2 Chemistry grade [%]	X3 # hours studied [hrs]	Y_{act} Final math grade[%]
1	60	80	5	82
2	70	75	7	94
3	50	55	10	45
4	40	56	7	43

Concept of Optimization

2) Mini batch Gradient Algorithm (MBGD)

Advantages

1. **It leads to more stable convergence.**
2. **More efficient gradient calculations.**
3. **Requires less amount of memory.**

Disadvantages

1. **Mini-batch gradient descent does not guarantee good convergence**
2. **If the learning rate is too small, the convergence rate will be slow. If it is too large, the loss function will oscillate or even deviate at the minimum value.**

Concept of Optimization

3) Batch Gradient Algorithm (GD)

But if we input the entire training dataset and then carry out forward and back propagation, that is called...

batch gradient descent

The Silly Training Dataset

Training examples

Student	X1 Science grade [%]	X2 Chemistry grade [%]	X3 # hours studied [hrs]	Y_{act} Final math grade[%]
1	60	80	5	82
2	70	75	7	94
3	50	55	10	45
4	40	56	7	43

Concept of Optimization

3) Batch Gradient Algorithm (GD)

Advantages

- **Easy to understand**
- **Easy to implement**

Disadvantages

- **Because this method calculates the gradient for the entire data set in one update, the calculation is very slow.**
- **It requires large memory and it is computationally expensive.**

Concept of Optimization

Note:

Each time we use the entire training dataset, that is called an EPOCH

So in our case, for stochastic gradient descent, we would have carried out 4 passes before completing one epoch

And for mini-batch, we would have carried out 2 passes to complete an epoch.

And for batch, we would have carried out 1 pass to complete an epoch.

So why not just use the entire dataset (batch) each time we want to carry out forward and back propagation?

One major reason is that each time we want to update the weights we have to process the entire dataset... this can be computationally very expensive if the dataset is large.

Concept of Optimization

So instead, we can choose training example sizes much smaller than the entire dataset and use that to update the weights.

That is, we can use these smaller mini-batches to begin updating the weights and biases even before we have completely finished using the entire dataset, i.e. completed one epoch.

Mini-batch or stochastic gradient descent is simply faster than batch.

"Most deep learning algorithms are based on an optimization algorithm called stochastic gradient descent. "

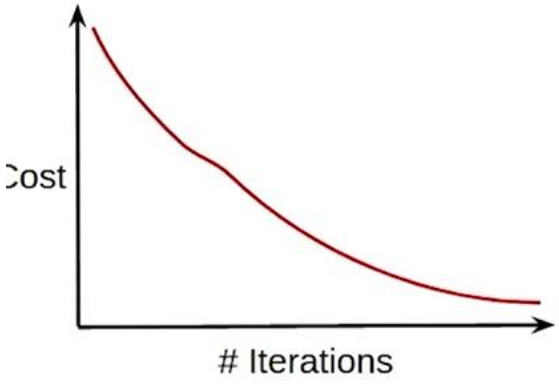
"Nearly all of deep learning is powered by one very important algorithm: stochastic gradient descent. "

Comparison: Cost Function & Time

Question: Explain the difference between Batch Gradient Descent, Stochastic Gradient Descent & Mini Batch Gradient Descent.

Batch Gradient Descent

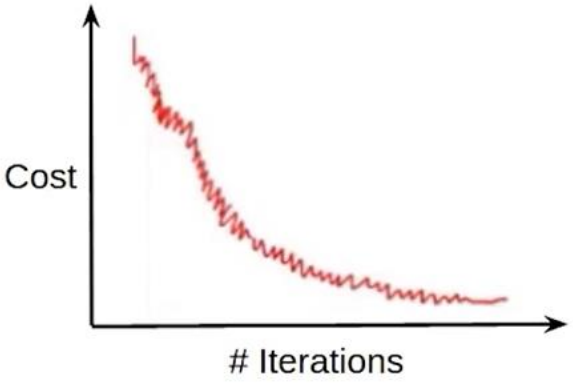
- Entire dataset for updation
- Cost function reduces smoothly



- Computation cost is very high

Stochastic Gradient Descent (SGD)

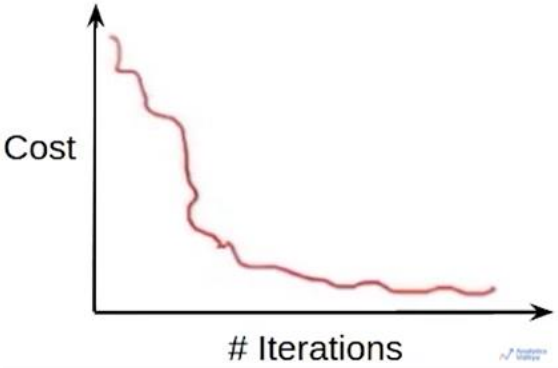
- Single observation for updation
- Lot of variations in cost function



- Computation time is more

Mini-Batch Gradient Descent

- Subset of data for updation
- Smoother cost function as compared to SGD

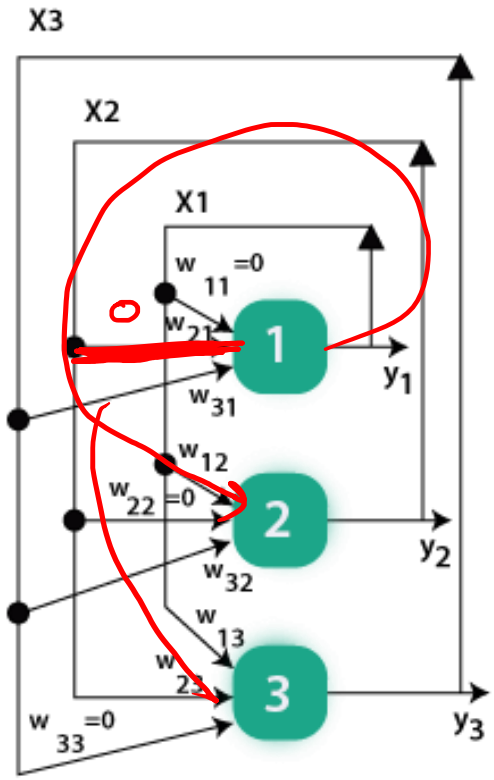


- Computation time is lesser than SGD
- Computation cost is lesser than Batch Gradient Descent

Hopfield Network

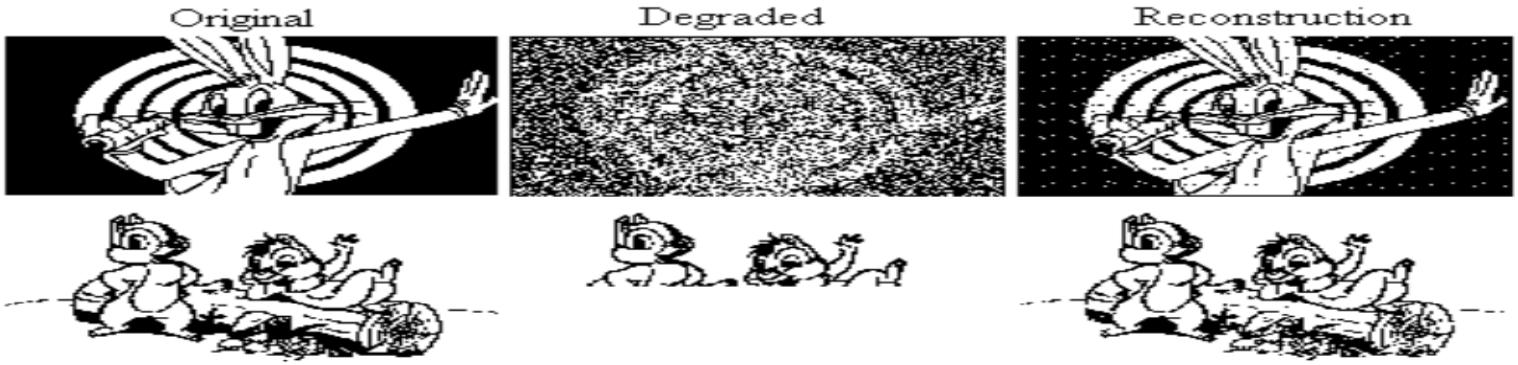
Question: What is Hopfield Network? and Explain its types

- John J. Hopfield proposed a network in 1982 which consist of a set of neurons with output of each neuron is feedback to all the other neurons.
- Hopfield networks have been used in many applications of **associative memory** & many optimization problem (Travelling- Salesman Problem).
- Hopfield network basically of 2 types,
  1. Discrete Hopfield Network
  2. Continuous Hopfield Network
- Hopfield network is an auto associative fully inter connected single layer feedback network.



Hopfield Network

- It is symmetrical weighted network. ($w_{ij} = w_{ji}$)
- It has only one layer of neurons corresponding to size of input and output which must be same.
- It is useful when given data is noisy, or partial. Since the network can store patterns, and retrieve those patterns based on partial input, due to the interconnections.



Hopfield network reconstructing degraded images from noisy (top) or partial (bottom) cues.

Discrete Hopfield Network

- Hopfield network when operates in discrete line fashion, then it is called “Discrete Hopfield Network”

- This network has two values inputs
  (a) Binary: $\{0,1\}$ (b) Bipolar : $\{1,-1\}$

- This network has symmetrical weights with no self connection

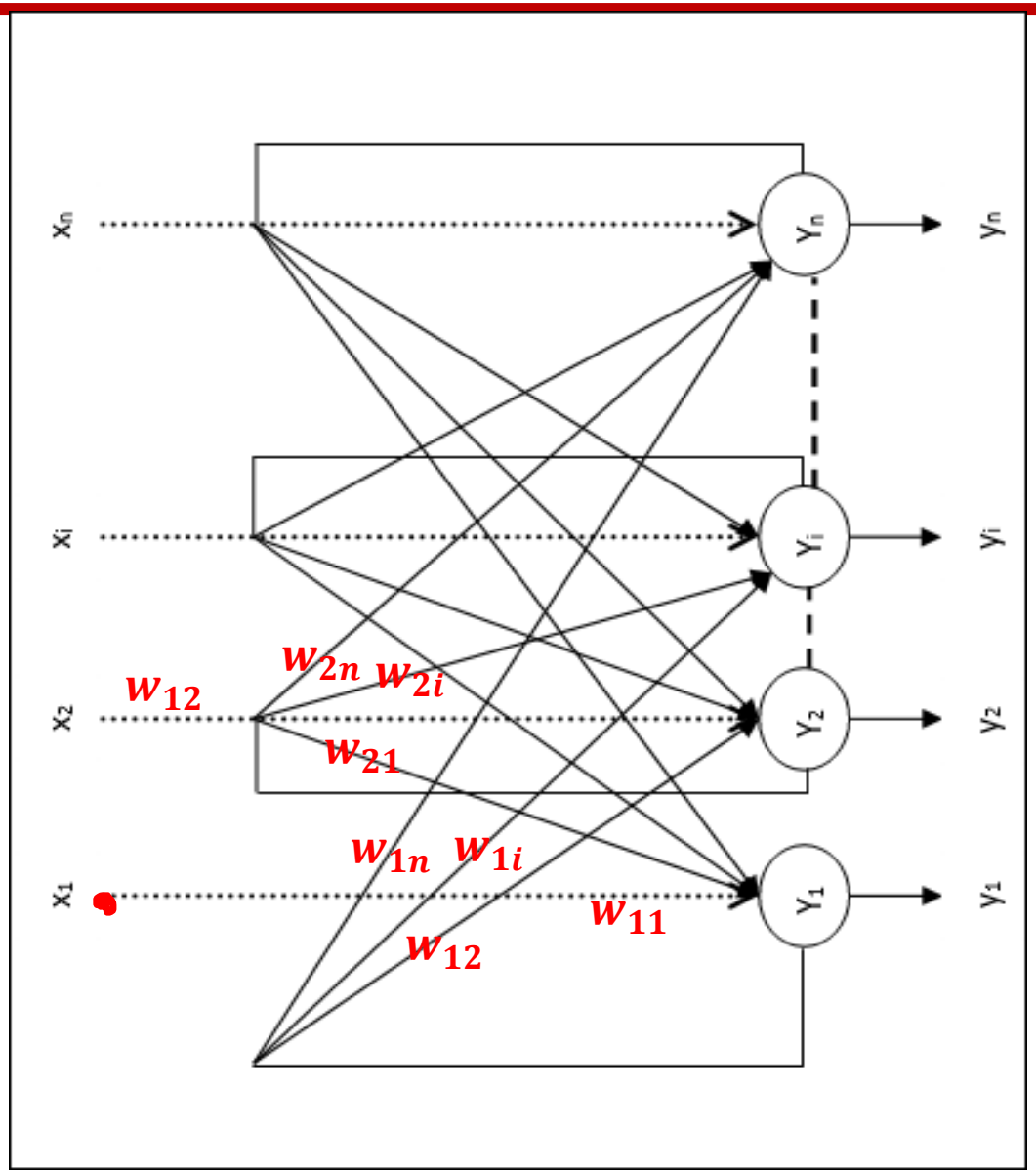
i.e. $w_{ij} = w_{ji}$ and $w_{ii} = 0$

- The architecture of discrete Hopfield network model consists of two outputs, one inverting and other non inverting.

- The output of each processing element are feedback to the input of other processing elements but not to itself.

Discrete Hopfield Network

$w_{11} = w_{22} = \cdots = w_{nn} = 0$



$w_{1n} = w_{n1}$

$w_{12} = w_{21}$

Training Algorithm- Discrete Hopfield Network

Step 1: Initialize weights (w_{ij}) to store patterns (using training algorithm).

For Binary Pattern:

$W = w_{ij} = \sum_i [2S_i - 1][2S_i - 1]^T \quad , \quad w_{ij} = 0 \text{ for } i = j$

For Bipolar Pattern:

$W = w_{ij} = \sum_i S_i S_i^T \quad , \quad w_{ij} = 0 \text{ for } i = j$

Testing Algorithm:

Step 2: Consider Binary representation of given testing vector, i.e. $x = [x_1 \quad x_2 \quad x_3 \ldots . \quad x_n]$

Step 3: Initially take, $y[y_1 \quad y_2 \quad y_3 \ldots . \; y_n] = x[x_1 \quad x_2 \quad x_3 \ldots . \; x_n]$

Step 4: To update y_i,

Consider Net input, $z_i = x_i + y(old).[W]^T_{i^{th}\ Column}$

Training Algorithm- Discrete Hopfield Network

Now, consider the activation function,

$$y_i(new) = \begin{cases} 1 & if\ z_i > 0 \\ y_i & if\ z_i = 0 \\ 0 & if\ z_i < 0 \end{cases}$$

Therefore, updated vector $y = [y_1 \quad y_2 \quad y_3 \quad y_i(new) \quad y_n]$

Repeat the iteration, for any i until we get $y = x$.

Training Algorithm- Discrete Hopfield Network

Example: *Construct an autoassociative Discrete Hopfield Network with input vector* $[1 \quad 1 \quad 1 \quad -1]$ *. Test Discrete Hopfield Network with missing entries in first & second component of stored vector.*

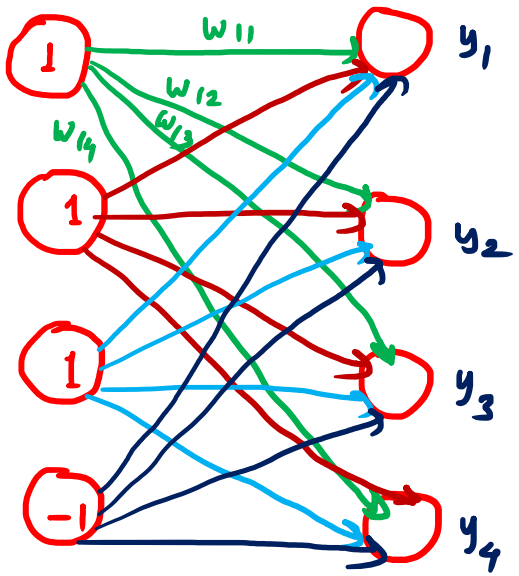
Soln :

Step 1 : Initialize the weights for "Bipolar patterns"

$$W = w_{ij} = \sum_{i} s_i^T \cdot s_i \;,$$

where s_i = Input vector

$w_{ij} = 0$ for $i = j$



Training Algorithm- Discrete Hopfield Network

We have, $s_1 = [\,1 \quad 1 \quad 1 \quad -1\,]$

$$W = s_1^T s_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}_{4\times 1} \begin{bmatrix} 1 & 1 & 1 & -1 \end{bmatrix}_{1\times 4}$$

$$= \begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

∴ Weight matrix with no self connection

Training Algorithm- Discrete Hopfield Network

$$\therefore W = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

step II: Binary representation of given input vector is $[1 \quad 1 \quad 1 \quad 0]$

We need to test Discrete Hopfield network for first and second missing entry

$\therefore$ Let $x = [0 \quad 0 \quad 1 \quad 0]$

$x_1 \quad x_2 \quad x_3 \quad x_4$

Training Algorithm- Discrete Hopfield Network

Step III : choose $y = x = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix}$

Step IV : consider y_1 for updation

Net input, $z_1 = x_1 + y(old) \cdot [W]^T_{1^{st}\ column}$

$= 0 + \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \end{bmatrix}$

$= 1$

$$y_1(new) = \begin{cases} 1 & z_1 > 0 \\ y_1 & z_1 = 0 \\ 0 & z_1 < 0 \end{cases} = \underline{\underline{1}}$$

$\therefore y = [1\ 0\ 1\ 0]$

$\therefore$ Not converged

Training Algorithm- Discrete Hopfield Network

Now consider y_2 for updation

$\therefore$ Net input $z_2 = x_2 + y(old) \cdot [W]^T_{2^{nd} \text{ column}}$

$$= 0 + [1 \quad 0 \quad 1 \quad 0] \begin{bmatrix} 1 \\ 0 \\ 1 \\ -1 \end{bmatrix}$$

$$= 2$$

$$\therefore y_2(new) = \begin{cases} 1 & z_2 > 0 \\ y_2 & z_2 = 0 \\ 0 & z_2 < 0 \end{cases} = 1$$

$\therefore$ $y = [1 \quad 1 \quad 1 \quad 0] = x$. Hence converges