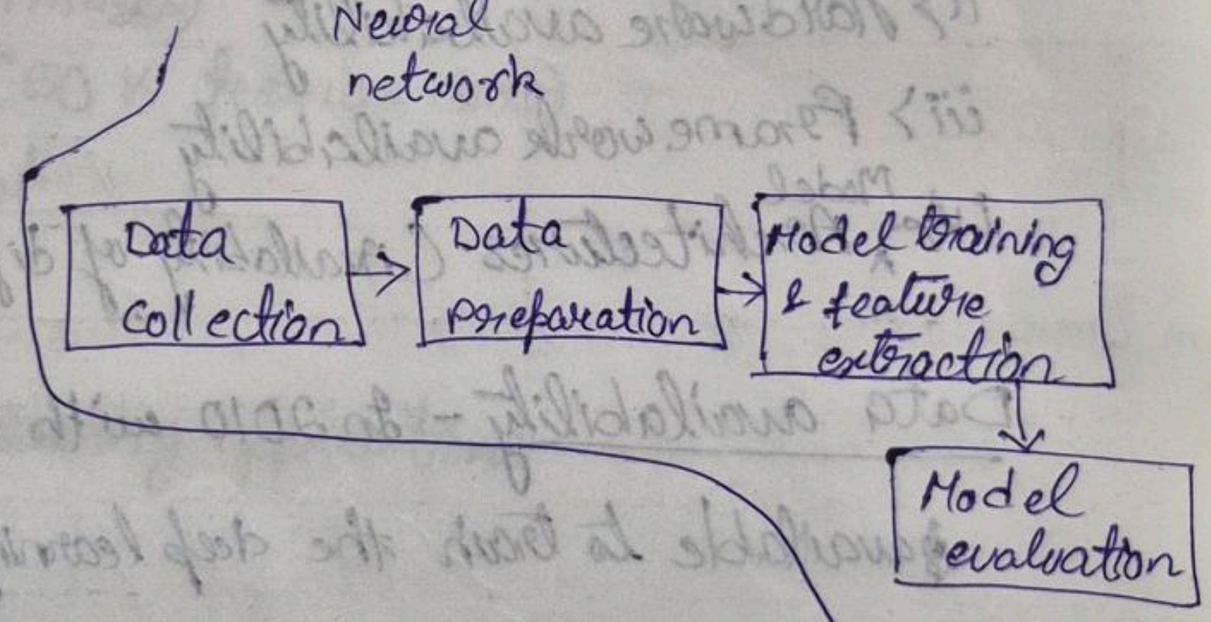
Deep Learning Using Python

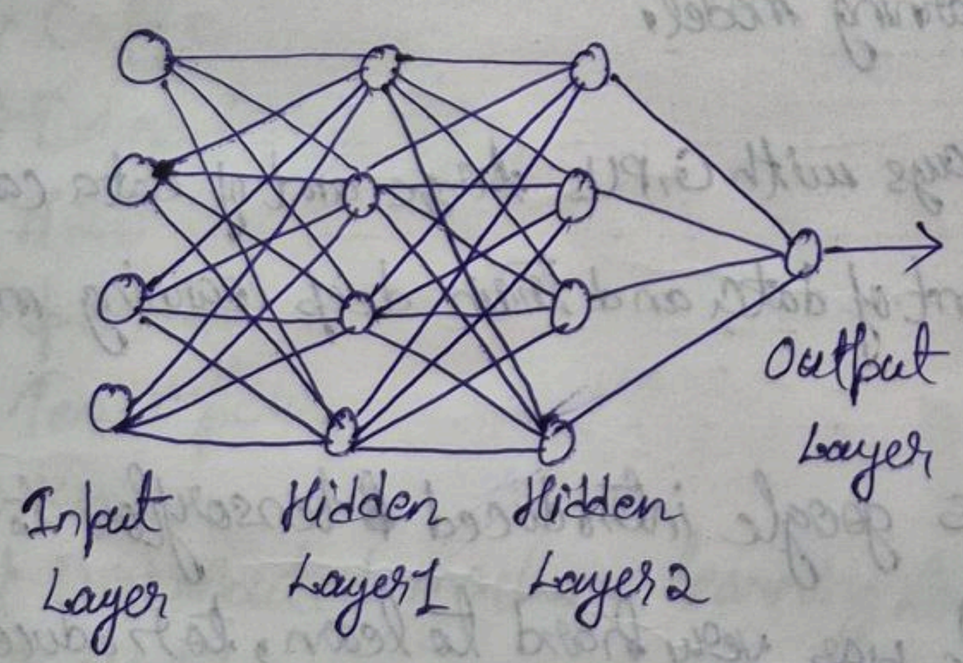
6/9

Deep learning - is a subfield of ML that uses neural networks with multiple layers to learn complex patterns and make predictions from data.
- It enables computers to automatically learn representations and features from raw data like images, video, audio or text offer with minimal human ~~interaction~~ interaction.

ML
- supervised
  - ↳ classification
  - ↳ Regression
  - Decision Tree
  - SVM
  - KNN
  - BN
  - Random forest
  - Linear Regression
  - Logistic Regression
- unsupervised
  - Kmeans
  - DBscan
  - Hirerchial
  - PCA
- Reinforcement

DL (ANN)
- supervised
  - ANN
  - CNN
  - RNN
  - LSTM
  - → Neural network
- unsupervised
  - autoencoder
  - GAN
- Reinforce-ment
  - Qlearn
  - Deep Q learning





- Each hidden layer extracts some feature of the input data.

ML vs DL -

ANN → works with numerical data
CNN → " " image data
RNN → " " sequential data
LSTM → " " " "

sequential data
i) Video

	ML	DL
i) Data Dependency	Less data (It does traing only).	More data (bcz it does training & feature selection simultaneously)
ii) Hardware Dependency	CPU (less	GPUs and more powerful hardware
iii) Training time	Less time	More time
iv) Feature selection	Manually done before training	Automatically done at the time of training.
v) Interpretability	More	Less

Q) Why DL is getting famous now?

Sol: i) Data availability

ii) Hardware availability

iii) Framework availability

iv) Model Architectures (Availability of different model architecture)

Data availability - In 2010 with mobile network large amt of data gets available to train the deep learning model.

Hardware availability - Now, a days with GPUs large amt of data can be processed to handle that amt of data and train deep learning model.

Framework availability - In 2015 google introduced tensorflow to train deep learning model which was very hard to learn, to reduce this problem "Keras" was introduced. But still it was hard so, in 2016 facebook released "pytorch" which makes deep learning a little easy.

Model architecture -

Pretrained model available for image classification is "RESNET".

" " " " " segmentation is "UNET".

For image translation pretrained model available is "Pix to Pix"
" object detection " " " " "YoLo"
" speech generation " " " " "Webnet"

Public data repositories -

i) Kaggle repository.
ii) UCI machine learning repository.
iii) Google data sets
iv) Visual Data (consisting of visual models)
v) ImageNet (for image processing)
vi) Twitter sentiment analysis dataset
vii) Youtube dataset
viii) MNIST dataset (handwritten dataset)
ix) Indian diabetics dataset
x) MSCO-CO dataset (330K images)
xi) IMDB-reviews dataset (50 K review data)
xii) Wikipedia Purpose dataset
xiii) Million Song dataset

Frameworks available - For deep learning applications

i) Caffe
ii) Deep learge 4j
iii) H20
iv) open NN
v) Tensorflow
vi) Keras
vii) Distributed machine learning toolkit
viii) MLlib
ix) Pytorch

Types of deep learning and application -

Sequential data -

i) Video iii) Speech v) Time series
ii) Audio iv) text

Application -

i) Autonomous car

ii) X ray classification in health care (CNN)

iii) Advanced facial recognisation (CNN)

iv) Visual question and answer (CNN)

v) Advanced voice assistence tools and devices (RNN & LSTM)

vi) Automatic machine translation. (RNN)

vii) Image recognition and image capturing (CNN) & (RNN)

viii) Anatroly detection & classification (ANN)

ix) Predicting earthquakes (Regression can be used & ANN) → (DL or ML algo)

x) Language translation (RNN)

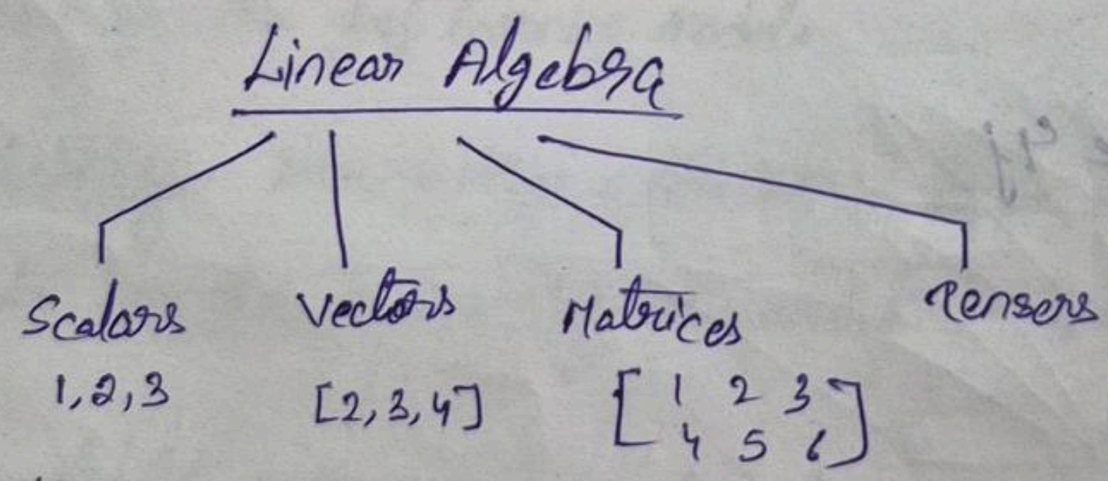
xi) Pixel ~~resolution~~ restoration (CNN)

xii) Photo descriptions (CNN)

8/9

Linear Algebra

- Linear Algebra is a branch of mathematics that deals with the study of linear systems which are sets of equations involving linear functions of variables.



Q) Why linear algebra for DL?

Sol: i) Good generalization ability.

ii) Good data representation.

Movie Recomendation System

movie — matrix → m1, m2, m3, m4, m5

movie data

movie name	summary
m1	S_1
m2	S_2
m3	S_3
m4	S_4

$S_1 \rightarrow$ The weather is good today

$S_2 \rightarrow$ My name is Ram

$S_3 \rightarrow$ How are you

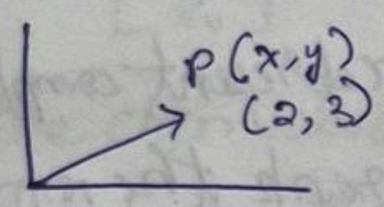
$S_4 \rightarrow$ ITER is a good college

Unique words → The weather is good today my name Ram how are you ITER a college

$S_1 \rightarrow$ 1 1 1 1 1 0 0 0 0 0 0 0 0

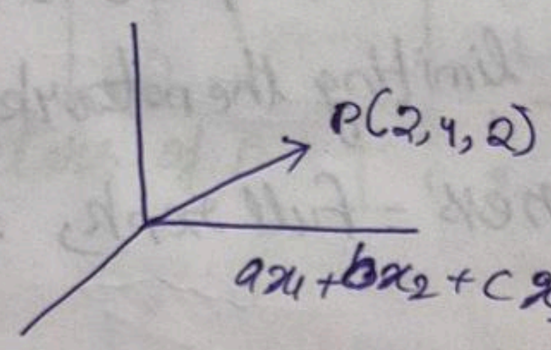
$S_2 \rightarrow$ 0 0 1 0 0 1 1 1 0 0 0 0 0 0
(is)

- In hyperplane we can represent a point of 'n' coordinates.



$P_1 = [2,3]$

$ax_1 + bx_2 = 0$



$ax_1 + bx_2 + cx_3 = 0$

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0$$

$$\underset{\text{(Row vector)}}{[a_1\ a_2\ a_3 \cdots a_n]} \underset{\text{(column vector)}}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}} = 0$$

Dot product

$a_1x_1 + a_2x_2 + \cdots + a_nx_n$

$a_1b_1 + a_2b_2 + \cdots + a_nb_n$

$A = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$, $B = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$

$$\boxed{A \cdot B = A^T B}$$

Fundamental spaces

Column space – of a matrix is the set of all linear combinations of its column.

$A = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$ $a_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $a_2 = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$

$$\boxed{a_2 = 2a_1}$$

Rank of a matrix is the dimension of its column space, the max no. of linearly independent column or rows.

Rank of $A = 1$

$B = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ rank $= 3$ $\quad C = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 2 & 6 & 9 \end{bmatrix}$ rank $= 2$

<u>Use of columnspace in neural network</u> - In a fully connected linear neural network equation $\boxed{y = wx + b}$, where $w \to$ matrix, $x \to$ is a matrix

$w \to$ weight matrix, $x \to$ input matrix

If w has low rank, its column space is small meaning the output lives in a low dimensional space limiting the networks capacity to represent complex patterns.

Model expressiveness = full rank, the more the rank the more the expressive the model is.

<u>Null space</u> - The null space of a matrix 'A' is the set of all vectors 'X' such that $\boxed{AX = 0}$

$A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \end{bmatrix}$

$AX = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 4 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0 \Rightarrow \begin{matrix} 1x_1 + 2x_2 - x_3 = 0 \\ 2x_1 + 4x_2 - 2x_3 = 0 \end{matrix}$ $\Rightarrow x_1 = -2x_2 + x_3$

$x_2 = s, x_3 = t$

$X = \begin{bmatrix} -2s + t \\ s \\ t \end{bmatrix} = s \begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Null space of $A = \left(\begin{bmatrix} -2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right)$

Q) Find out null space of $A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$

<u>Sol:</u> $AX = 0$

$\Rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = 0 \Rightarrow X$ (To find this type of ans we need row echilon form).

<u>Row Echilon form</u> -

<u>Condition</u> - i) Leading element of each row must be unity or 1.

ii) If any 0 row is present then it must be at last.

iii) In each row no.s of zeros must be more than previous row.

Condition for reduced row echilon form -

Below and above of leading element, they must be 0.

Eg: $\begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix} \xrightarrow[R_3 = R_3 - 4R_1]{R_2 = R_2 - R_1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & -1 & -2 & -3 \end{bmatrix} \xrightarrow[R_3 = R_3 + R_2]{R_1 = R_1 - R_2} \begin{bmatrix} 1 & 0 & -1 & -2 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Now, eqn is

$x_1 - x_3 - 2x_4 = 0 \Rightarrow x_1 = x_3 + 2x_4$

$x_2 + 2x_3 + 3x_4 = 0 \Rightarrow x_2 = -2x_3 - 3x_4$

Let, $x_3 = s$, $x_4 = t$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} s + 2t \\ -2s - 3t \\ s \\ t \end{bmatrix} = s \begin{bmatrix} -1 \\ 2 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 2 \\ -3 \\ 0 \\ 1 \end{bmatrix}$$

Q. Find the eigen values and eigenvectors of $A = \begin{bmatrix} 5 & 4 \\ 1 & 2 \end{bmatrix}$

Sol: $A = \begin{bmatrix} 5 & 4 \\ 1 & 2 \end{bmatrix}$

$$|A - \lambda I| = 0 \Rightarrow \begin{vmatrix} 5-\lambda & 4 \\ 1 & 2-\lambda \end{vmatrix} = 0 \Rightarrow (2-\lambda)(5-\lambda) - 4 = 0$$

$$\Rightarrow 10 - 2\lambda - 5\lambda - \lambda^2 - 4 = 0$$

$$\Rightarrow \lambda^2 - 7\lambda + 6 = 0 \Rightarrow \lambda^2 - 6\lambda - \lambda + 6 = 0$$

$$\Rightarrow \lambda(\lambda - 6) - 1(\lambda - 6) = 0$$

$$\Rightarrow \boxed{\lambda = 6, \lambda = 1}$$

For, $\lambda = 1$

$$\begin{bmatrix} 4 & 4 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$

$\Rightarrow 4x + 4y = 0$
$x + y = 0$ $\Rightarrow x = -y \therefore \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

For $\lambda = 6$

$$\begin{bmatrix} -1 & 4 \\ 1 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \begin{matrix} -x + 4y = 0 \\ x - 4y = 0 \end{matrix} \Rightarrow x = -4y \therefore \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -4 \\ 1 \end{bmatrix}$$

12/9

Entropy - The ~~order of~~ amt of is order, uncertainity, randomness associated with the event.

$$\boxed{H(P) = -\sum P_i \log(P_i)}$$

Cross entropy - Cross entropy is the ~~good~~ goodness of fit, the closeness of the predictive distributive to the actual one.

$$\boxed{H(p,q) = -\sum_i p_i \log(q_i)}$$

p → actual probability
q → predicted probability

Eg: 100 times a coin is tossed
75 heads
25 tails

case I: 72 heads, 28 tails → more near, so, low cross entropy

case II: 70 heads
30 ~~heads~~ tails

Now, $H(p,q) = -\sum_i p_i \log(q_i)$

For case I:

$$H(p,q) = -0.75 \log(0.72) - 0.25 \log(0.28) = \boxed{0.814}$$

predicted o/p is nearer to actual o/p.

For case II:

$$H(p,q) = -0.75 \log(0.70) - 0.25 \log(0.30) = \boxed{0.829}$$

Note: The cross entropy is smaller for case I than case II as it is closer to the actual distribution.

* In ML problem when we are ~~trying~~ trying to fit a model with a training data and data is already labelled as +ve and -ve examples, we would like to know how our model is generalizing itself by calculating cross entropy loss, i.e for a +ve labelled data how good the model is predicting it as positive and for e -ve labelled data how good the model is predicting it as negative.

Note - We need to minimize the cross entropy loss.

KL - Divergence - is a ~~preferen~~ difference b/w cross entropy and entropy.

$$\text{KL-Divergence}(P\|q) = H(p,q) - H(p)$$
$$= -\sum_i p_i \log(q_i) + \sum_i p_i \log(P_i)$$

$= \sum_{i} p_i \log(p_i / q_i)$

For case I: $0.75 \log(0.75/0.72) + 0.25 \log(0.25/0.28) = \cancel{0.0088}\ 0.0032$

For case II: $0.75 \log(0.75/0.70) + 0.25 \log(0.25/0.30) = \cancel{0.0032}\ 0.0088$

- Distance is less in case I than in case II as the case I is closer to the actual distribution.

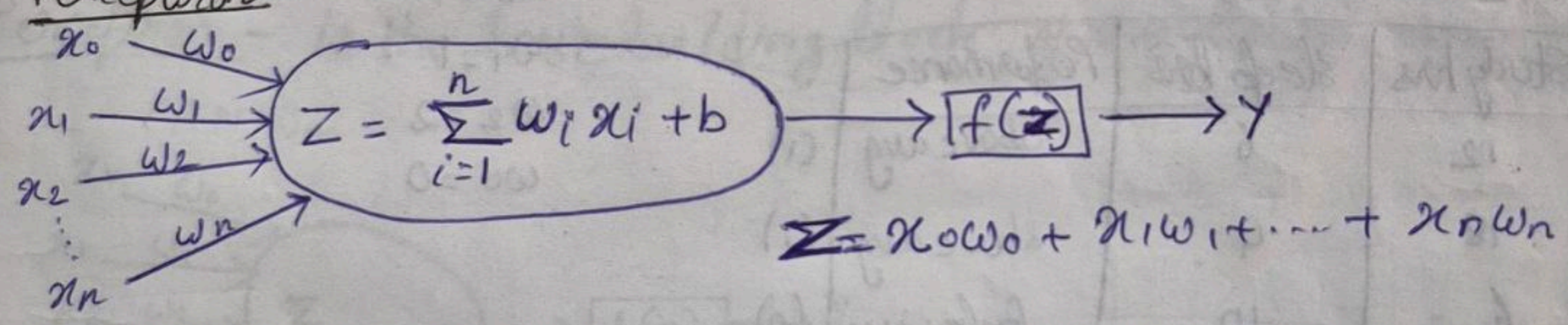
* If a neural network is regression then there will be only one ~~node~~ neuron in output layer.

* If a neural network is classification ~~then there will be min~~ and it is binary classification min of one ~~node~~ neuron and max of 2 ~~nodes~~ neuron are possible in output layer.

* If a neural network is multiclassification then no. of neural network depends on no. of different inputs.

Architecture of artificial neural network

Perceptron -



$Z = x_0 w_0 + x_1 w_1 + \dots + x_n w_n$

inputs $= x_0, \dots x_n$

weights $= w_0 \dots w_n$

Z = summation function

$f \rightarrow$ activation function

Difference b/w biological neuron and perceptron -

- Nervous system is a collection of billions of neurons and neuron is a biological cells consisting of a nucleus where all complex processing tasks takes place.
- The cells consisting of ~~inputs~~ dendrites to receive input and the input processed in the nucleus to generate output which is passed through ~~axium~~ axon.

- Perceptron -

- In perceptron there are multiple input, for each input there is an assigned weight. The weighted input is compared to dendrites. The sum and activation as a whole can be compared with nucleus and output is compared with axon.

Biological Neuron	Perceptron
i) It has very complex architecture.	i) It's simple architecture
~~ii) The process hap~~ ii) What processing task is done in nucleus is not known exactly.	ii) Here, we are performing the summation and followed by an activation function to get the output.

Q) Design a perceptron to predict the performance of a student based on no. of study hours and no. of sleep hours.

Sol: 1) Take dataset

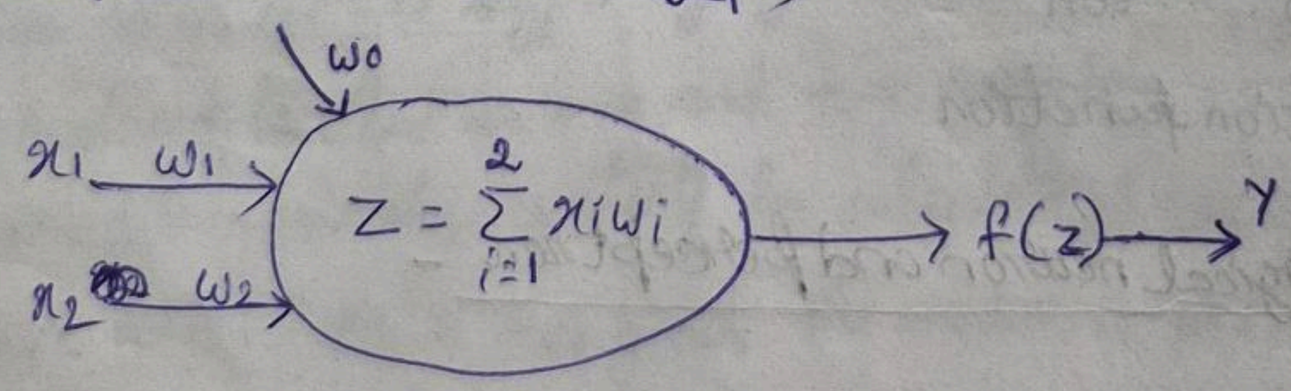
	Study hrs	sleep hrs	Performance	
S1	12	6	above avg	(1)
S2	18	4	above avg	(1)
S3	6	10	Below avg	(0)
S4	10	10	Below avg	(0)

$x_0 = 1$

$w_1 = 1$

$w_2 = 2$

$w_0 = 0$



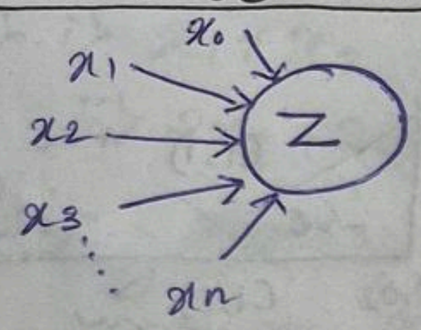
$$Z = x_0 w_0 + x_1 w_1 + x_2 w_2$$
$$= x_0 w_0 + 12 w_1 + 6 w_2 = 0 + 12 + 12 = 24$$

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

$f(24) = 1$ (as $z > 0$)

- Perceptron is a mathematical model or a mathematical function, ~~which~~ which undergoes 2 no. of processes, 1st is training and 2nd is prediction.
- In the process of training we are finding out the optimal value of the parameter i.e weight or (weight and bias).

Geometric Interpretation -

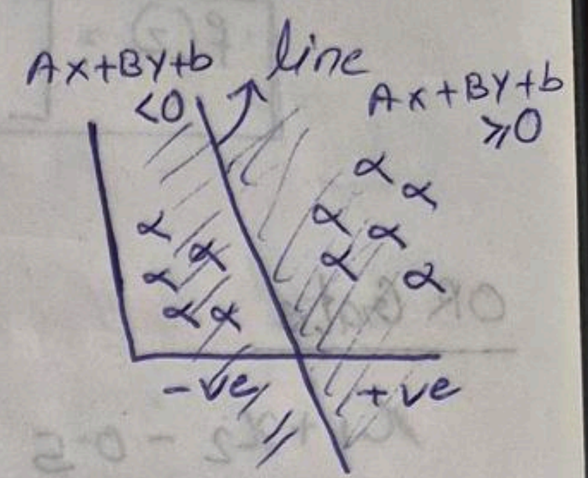


$$Z = x_0w_0 + x_1w_1 + x_2w_2 + \dots + x_nw_n$$

$$Z = x_0w_0 + \sum_{i=1}^{n} x_iw_i$$

$$\boxed{Z = \sum_{i=1}^{n} x_iw_i + b}$$ line eqn

$$\boxed{Z = XW + b}$$



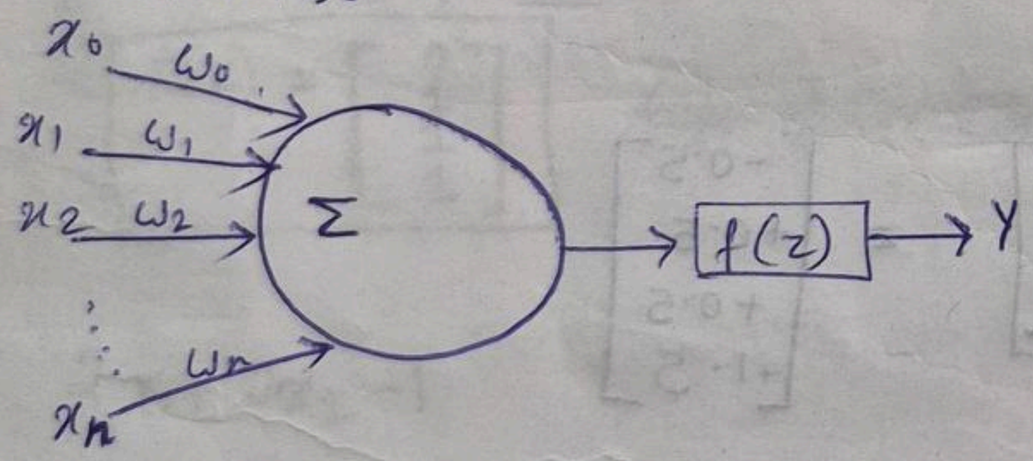
$$\underbrace{w_1}_{A}\underbrace{x_1}_{X} + \underbrace{w_2}_{B}\underbrace{x_2}_{Y} + b$$

$$= AX + BY + b$$

- Perceptron is a binary classifier because it divides the data into 2 classes by creating regions.
- A perceptron acts as a line in case of 2D, in 3D its a plane and in 4D onwards it will acts as a hyperplane.

15/9

Perceptron - is the basic building block of a artificial neural network (ANN).



- Perceptron can be used in logic gates. (AND, OR, XOR)

$$\underbrace{x_1 + x_2 - 1.5}_{b}$$

$$x_0w_0 + x_1w_1 + x_2w_2 = 0$$

$w_1 = 1, w_2 = 1$ $\quad b = -1.5 = x_0w_0$

$x_0 = 1$ (Always) $\Rightarrow w_0 = -1.5$

$$X = \begin{bmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$W = [-1.5 \quad 1 \quad 1]$$

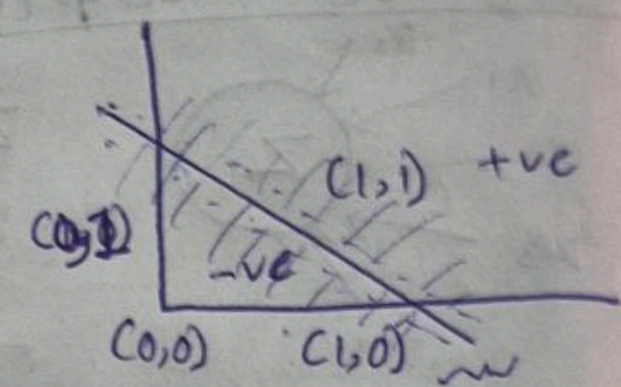
AND Gate

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

$$\omega x^T = \begin{bmatrix} -1.5 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -1.5 \\ -0.5 \\ +0.5 \end{bmatrix}$$

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

$$\boxed{f(z) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}$$ (o/p of AND Gate)



There can be ∞ no. of lines

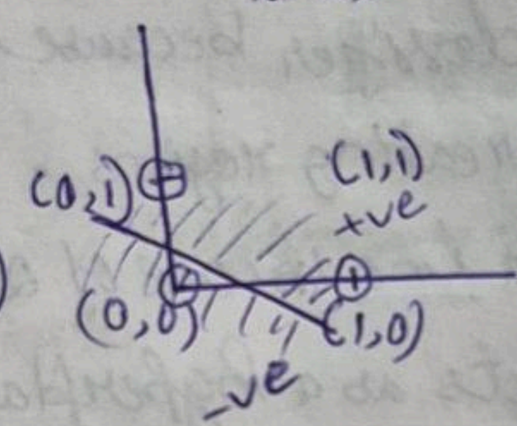
OR Gate

$x_1 + x_2 - 0.5$ (choosen) ⇒ there can be ∞ no. of lines

$\omega_1 = 1, \omega_2 = 1, \underbrace{b = -0.5}$

$x_0 = 1$ (always)

$\omega_0 = -0.5$



TRUTH Table (OR)

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

$$\omega = \begin{bmatrix} -0.5 & 1 & 1 \end{bmatrix}$$

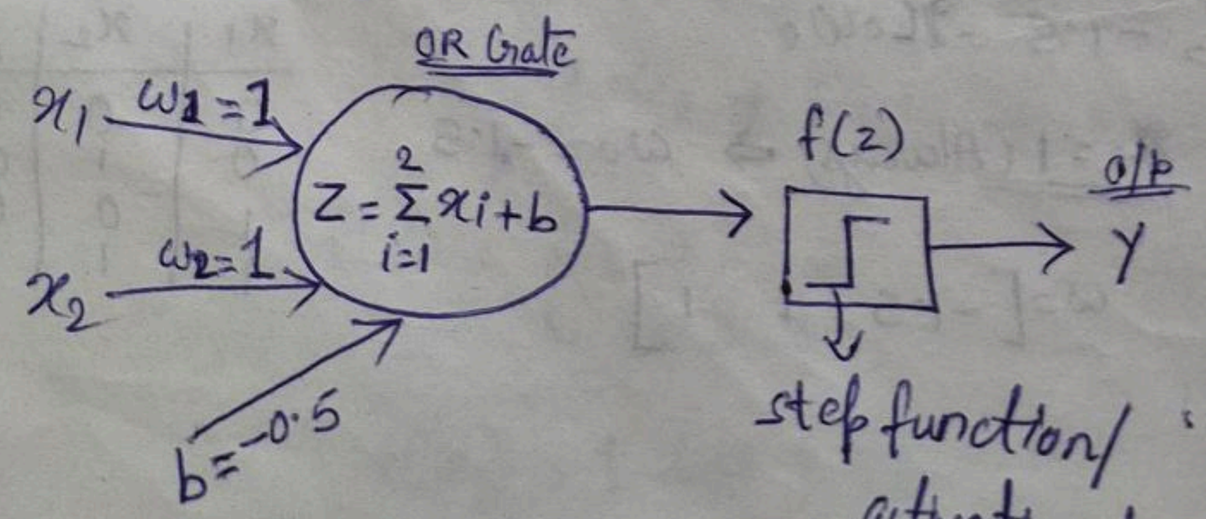
$$X = \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$\omega x^T = \begin{bmatrix} -0.5 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ +0.5 \\ +0.5 \\ +1.5 \end{bmatrix}$$

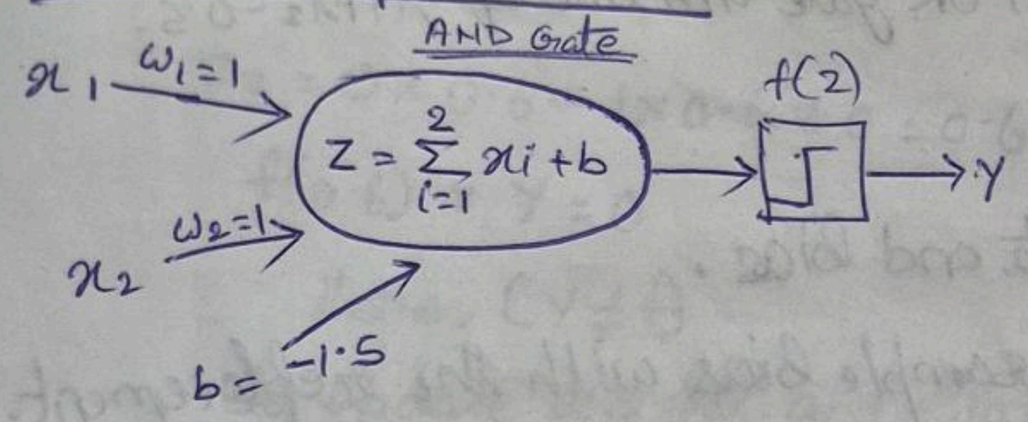
$$f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$$\boxed{f(z) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}}$$ (o/p of OR Gate)

Architecture



Architecture of AND Gate



Q). i) x_1+x_2-1 for AND Gate

ii) $2x_1+2x_2-1$ for OR Gate

Sol: i) $w_1=1, w_2=1, w_0=-1$

$$x = \begin{array}{ccc} x_0 & x_1 & x_2 \end{array}$$
$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad w = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix}$$

$$wx^T = \begin{bmatrix} -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

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

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

$$\boxed{f(z) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}} \quad \cancel{\text{(o/p of AND Gate)}} \neq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

ii) $2x_1+2x_2-1$

$w_1=1, w_2=1, w_0=-1$

$$x = \begin{array}{ccc} x_0 & x_1 & x_2 \end{array}$$
$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad w = \begin{bmatrix} -1 & 2 & 2 \end{bmatrix}$$

$$wx^T = \begin{bmatrix} -1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 3 \end{bmatrix}$$

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

$$f(z) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ (o/p of OR Gate)}$$

Q> Design a perceptron which will act as an OR gate with line eq $x_1+x_2-0.5$.

18/9

Perceptron Learning Rule

Step 1> Randomly initialize the weights and bias.

Step 2> Randomly select a training sample bias with the replacement.

Step 3> Perform the affine transformation.

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

Step 4> Pass the value of z to the step activation function to get Y.

Step 5> If Y doesn't match with your target then update the weights and bias, with the formula

If $(y \neq t)$ $\quad w_{inew} = w_{iold} + \eta(t^i - y^i)x_i$

$b_{new} = b_{old} + \eta(t^i - y^i)$

~~else~~

else

$w_{inew} = w_{iold}$, $b_{new} = b_{old}$

Step 6> Repeat this steps 2 to 5 till convergence.

Q> Find the line equation of a perceptron which will acts as an OR Gate by application of perceptron learning rule.

The initial value of $w_1 = 0.6$, $w_2 = 0.6$, $b = 0$, $\eta = 0.5$, $\theta = 1$ (threshold)

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

Sol: Truth table

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

1> $x_1 = 0$, $x_2 = 0$.

$z_1 = 0 \times 0.6 + 0 \times 0.6 = 0$

$f(z_1) = f(0) = Y = 0$

Here, $Y = t$ (no updation)

2) $x_1 = 0, x_2 = 1$

$z_2 = 0 \times 0.6 + 1 \times 0.6 + 0 = 0.6$

$f(0.6) = y = 0$

Here, $(y \neq t)$

So, <u>update weights and bias</u>

$w_{1new} = w_{1old} + 0.5 \cancel{(1-0)}^{0} 0 = 0.6 + 0 = 0.6$

$w_{2new} = w_{2old} + 0.5 (1-0) 1 = 0.6 + (+0.5) 1 = 1.1$

$b_{new} = b_{old} + 0.5 (1-0) = 0 + (0.5) = +0.5$

~~$x_1 = 0, x_2 = 0$~~

~~$z_1 = 0.6 \times 0 + 0.1 \times 0 - 0.5 = -0.5$~~

~~$f(z_1) = 0$ $(y = t) \Rightarrow$ no updation~~

~~$x_1 = 0, x_2 = 1$~~

~~$z_2 = 0 \times 0.6 + 0.1 \times 1 - 0.5 = -0.4$~~

~~$f(z_2) = f(-0.4) = 0$~~

$x_1 = 0, x_2 = 0$

$z_1 = 0.6 \times 0 + 0 \times 1.1 + 0.5$

$= 0.5$

$z_2 = 0.6 \times 0 + 0 \times 1.1 + 0.5$

$= 0.5$

$f(z_1) = f(0.5) = 0$ $(y = t)$

no updation

$x_1 = 0, x_2 = 1$

$z_2 = 0.6 \times 0 + 1.1 \times 1 + 0.5 = 1.6$

$f(1.6) = y = 1$ $(y = t)$ No updation

$x_0 = 1, x_2 = 0$

$z_3 = 1 \times 0.6 + 0 \times 1.1 + 0.5 = 1.1$

$f(1.1) = 1$ $(y = t)$ ~~update~~

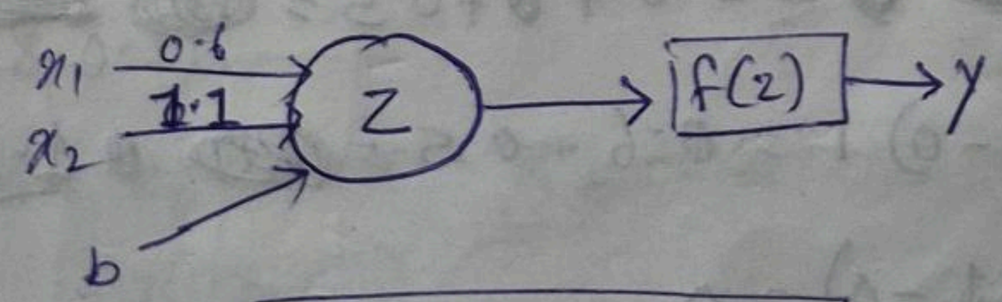
$x_1 = 1, x_2 = 1$

~~$z_4 = 1 \times 0.6 + 1 \times 0.5$~~

$z_4 = 1 \times 0.6 + 1.1 \times 1 + 0.5$

$= 2.2$

$f(2.2) = 1$ $(y = t)$



$\boxed{0.6x_1 + 1.1x_2 + 0.5}$ Ans

2019

Q) Let, $w_1 = 1.2$, $w_2 = 0.6$, $b = 0$, $\eta = 0.5$, $\theta = 1$. Find the optimal line equation for AND Gate.

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

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

Sol: $z_1 = x_1 w_1 + x_2 w_2 + \underset{(b)}{x_0 w_0}$

$w_0 = 0$, $x_0 = 1$

$z_1 = \cancel{1.2 \times 0 +} 0 \times 1.2 + 0 \times 0.6 + 0 = 0$ (no update)

$z_2 = 0 \times 1.2 + 1 \times 0.6 + 0 = 0.6 = f(0.6) = 0$ (no update)

$z_3 = 1 \times 1.2 + 0 \times 0.6 + 0 = 1.2 = f(1.2) = \underline{1}$ (updation req)

$w_{1new} = 1.2 + 0.5(0-1)1 = 1.2 - 0.5 = 0.7$

$w_{2new} = 0.6 + 0.5(0-1)0 = 0.6$

$b_{new} = 0 + 0.5(0-1) = -0.5$

~~$x_0, x_1 =$~~ $\underline{x_0 = 0, x_1 = 0}$.

$z_1 = 0 \times 0.7 + 0 \times 0.6 - 0.5 = -0.5 = f(-0.5) = 0$

$\underline{x_0 = 0, x_1 = 1}$

$z_2 = 0 \times 0.7 + 1 \times 0.6 - 0.5 = 0.6 - 0.5 = 0.1 = f(0.1) = 0$

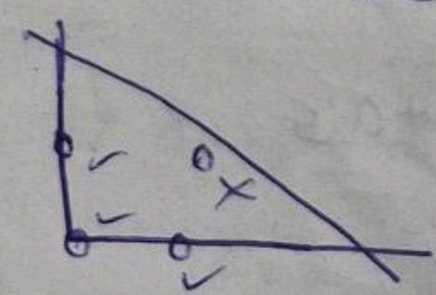
$\underline{x_0 = 1, x_1 = 0}$

$z_3 = 1 \times 0.7 + 0 \times 0.6 - 0.5 = 0.2 = f(0.2) = 0$

$\underline{x_0 = 1, x_1 = 1}$

$z_4 = 1 \times 0.7 + 1 \times 0.6 - 0.5 = 0.7 + 0.6 - 0.5 = 0.8$

$= f(0.8) = \underline{0}$



~~$w_{1new} =$~~

updation

$w_{1new} = 0.7 + 0.5(1-0)1 = 0.7 + 0.5 = \underline{1.2}$

$w_{2new} = 0.6 + 0.5(1-0)1 = 0.6 + 0.5 = \underline{1.1}$

$b_{new} = -0.5 + 0.5(1-0) = 0$

$x_0 = 0, x_1 = 0$

$z_1 = 0 \times 1.2 + 0 \times 1.1 + 0 = 0 = f(0) = 0$

$x_0 = 0, x_1 = 1$

$z_2 = 0 \times 1.2 + 1 \times 1.1 + 0 = 1.1 = f(1.1) = 1$ (updation required)

$w_{1new} = 1.2 + 0.5(0 - 1)0 = 1.2$

$w_{2new} = 1.1 + 0.5(0-1)1 = 1.1 - 0.5 = 0.6$

$b_{new} = 0 + 0.5(0-1) = -0.5$

$x_0 = 0, x_1 = 0$

$z_1 = 0 \times 1.2 + 0 \times 0.6 - 0.5 = f(-0.5) = 0$

$x_0 = 0, x_1 = 1$

$z_2 = 0 \times 1.2 + 1 \times 0.6 - 0.5 = f(0.1) = 0$

$x_0 = 1, x_1 = 0$

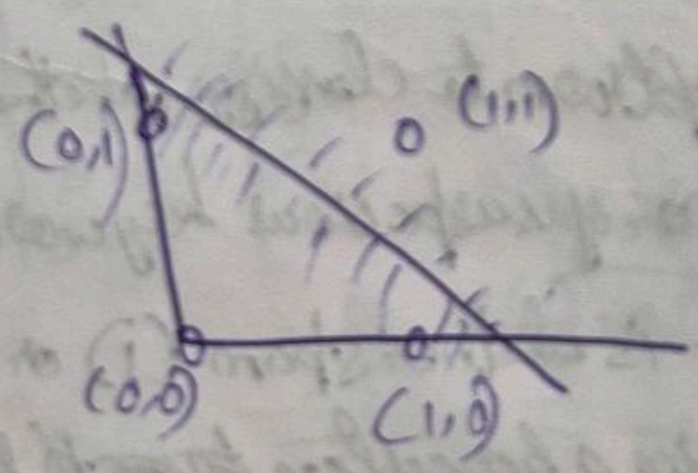
$z = 1 \times 1.2 + 0 \times 0.6 - 0.5 = 1.2 - 0.5 = 0.7 = f(0.7) = 0$

$x_0 = 1, x_1 = 1$

$z = 1 \times 1.2 + 1 \times 0.6 - 0.5 = 1.2 + 0.6 - 0.5 = 1.2 + 0.1 = 1.3$

$= f(1.3) = 1$

$\therefore \boxed{1.2x_1 + 0.6x_2 - 0.5 = 0}$ Ans



Q) Given perceptron weights $w_1 = 1, w_2 = 1, b = -0.2$. Write the equation of decision boundary and classify the points $(0.5, 0.5)$ and $(0.8, 0.2)$.

Sol: Line eqn $= x_1 + x_2 - 0.2$

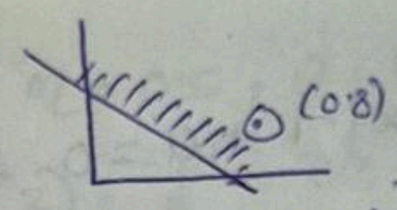
for $(0.5, 0.5)$

$f(x) = x_1 + x_2 - 0.2 = 0.8$

for $(0.8, 0.2)$

$f(x) = 0.8$

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



$f(0.8) = 1$ (∵ it will lie in the +ve region). Ans

Q> Train a perceptron for NAND Gate for the input $(x_1, x_2) = (0,1)$.
* Target (Y) = NOT $(x_1 \cdot x_2)$. Initial weights are $w_1 = 0$, $w_2 = 0$, $b = 0$ and $\eta = 1$. Perform weight update step by step untill convergence.

Sol: $\theta = 0$ (if not given)

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

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

$x_0 = 0, x_1 = 0$

$z = 0 \times 0 + 1 \times 0 + 0 = 0$ $f(z) = 1$

$w_{1new} = w_{1old} + \eta (t^i - y^i) x_1$

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

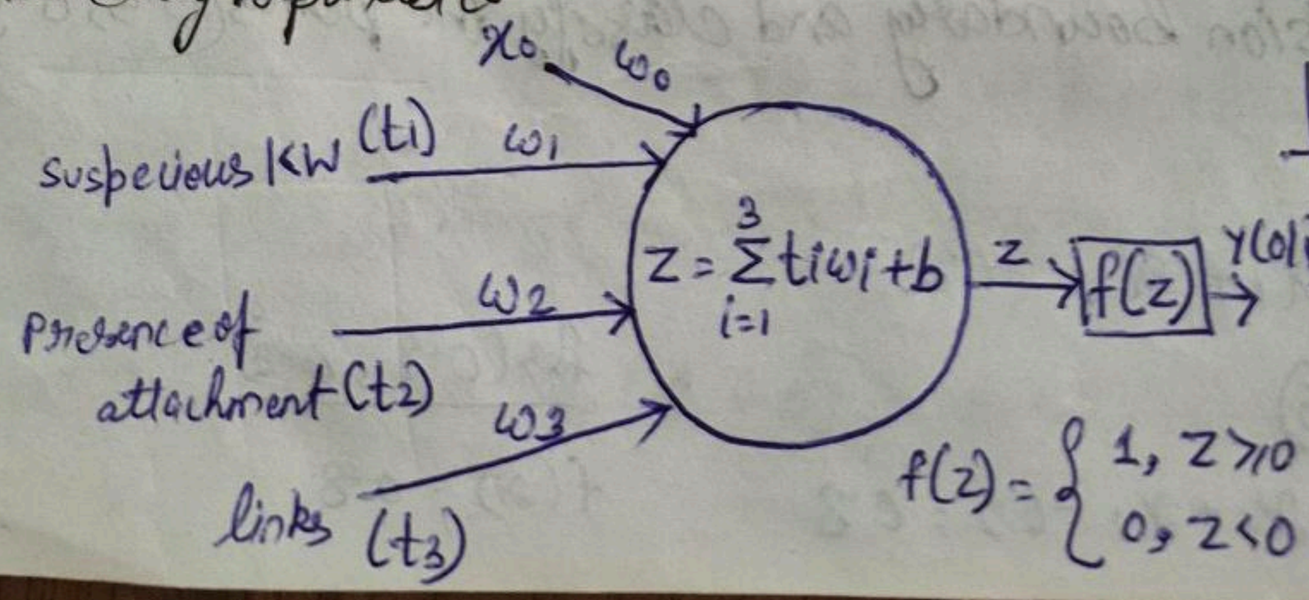
$w_{2new} = 0 + 1(1-0)0 = 0$

$b_{new} = b_{old} + \eta (t^i - y^i) = 0 + 1(1-0) = 1$

$x_0 = 0, x_1 = 0$

Q> You are using a perceptron to classify whether an email is spam or not spam. Input features are no. of suspecious keywords, presence of attachments, no. of links. Target is whether spam (1) or not spam (0). What are the advantages of using a perceptron for this task? What are the limitations and suggest an improvement if the data is not linearly separable?

Sol:



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

t_1	t_2	t_3	y

Q1) Define a perceptron and explain its role as a binary classifier.
Q2) Explain the purpose of weights or biases in perceptron.
Q3) Why is a step activation function commonly used in a perceptron?
Q4) What is linear separability and why it is important for a perceptron?
Q5) Explain why a single layer perceptron can't solve the XOR problem?
Q6) How does the choice of activation funcn affect the learning process of perceptron?

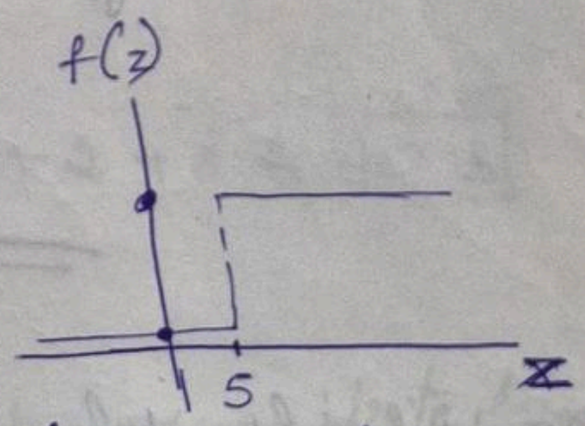
<u>Activation function</u> -
An activation function decides whether a neuron should be activated or not.

<u>3 types</u> -
i) Binary step function
ii) Linear activation function
iii) Non linear activation function.

<u>Binary step function</u> - Depends on a threshold value that decides whether a neuron should be activated or not.

$$f(z) = \begin{cases} 1, & z \geq \theta = 5 \\ 0, & z < \theta \end{cases} \qquad \theta \rightarrow \text{threshold}$$

Z	f(z)	θ=5	Y
0	f(0)	θ=5	0
1	f(1)	θ=5	0
4	f(4)	θ=5	0
14	f(14)	θ=5	1



→ Binary activation function is not differentiable
$\frac{d(1)}{dz} = 0$ (always constant).

<u>Linear activation function</u> -
- Also known as <u>no activation</u> or <u>identity function</u> is where the activation is proportional to input.

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

- If the value of activation function is 'x' the o/p value is also 'x'.

<u>Limitation</u> - The derivative of the funcn is constant and has no relation to input "x". Therefore, we can't use the

linear activation function in backpropagation.

- If we apply the linear activation ~~function~~ function, all layer will ~~collapse~~ collapse to a single layer in multi layer perceptron.
- So essentially the linear activation funch turns the neural network ~~in~~ into just one layer and due to its limited power this doesn't allow the model to create ~~complete~~ complex mapping b/w input and output.

Non-linear activation function -

i) Sigmoid activation function ii) tanh $\boxed{\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}$ iii) Relu $\rightarrow \max(0,x)$ iv) Softmax

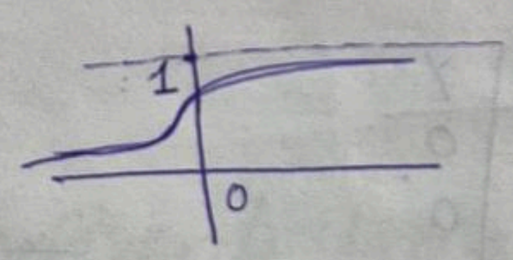
v) Leaky Relu vi) Parameter ReLU vii) Swish viii) GELU ix) SELU

Sigmoid activation function -

$$f(z) = \frac{1}{1+e^{-z}} = y \quad , z \rightarrow o/p \text{ of affine tranformation}$$

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

$$f'(z) = y(1-y) = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = \boxed{\frac{e^{-z}}{(1+e^{-z})^2}}$$



Sigmoid function Value is from (0 to 1) for $(-\infty, \infty)$

Q) A perceptron with i/p vector $(x_1, x_2) = (1,1)$ has weights $(w_1, w_2) = (0.5, 0.5)$ and bias $= -0.6$. Check whether it is correctly classify the AND function for input (1,1).

Sol: Truth Table

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

$w_1 = 0.5,\ w_2 = 0.5$

$x_1 = 1,\ x_2 = 1$

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

$z = w_1 x_1 + w_2 x_2 + b = 0.5 \times 1 + 0.5 \times 1 - 0.6 = 0.5 + 0.5 - 0.6 = 0.4$

$f(0.4) = 1$ (Yes) Ans

Q) A perceptron with learning rate $\eta = 0.1$ and current weights are $w_1 = 0.2,\ w_2 = -0.3,\ b = 0.1$ for input $(1, 2)$ and target output $= 1$. If perceptron output is $y = 0$, update the weights and bias.

Sol: $w_1 = 0.2,\ w_2 = 0.3,\ b = 0.1,\ \eta = 0.1$

$x_1 = 1,\ x_2 = 2$

$z = w_1 x_1 + w_2 x_2 + b$

$= 0.2 \times 1 + (-0.3) \times 2 + 0.1 = 0.2 - 0.6 + 0.1 = -0.3$

$f(-0.3) = 0$ (updation required)

$w_{1new} = 0.2 + 0.1(1-0)1 = 0.2 + 0.1 = 0.3$

$w_{2new} = -0.3 + 0.1(1-0)2 = -0.3 + 0.2 = -0.1$

$b = 0.1 + 0.1(1-0) = 0.1 + 0.1 = 0.2$

$z_{new} = 0.3 \times 1 + (-0.1) \times 2 + 0.2 = 0.3 - 0.2 + 0.2 = \boxed{0.3}$

$f(0.3) = 1$ (Ans)

Q) Give perceptron weights $w_1 = 1,\ w_2 = -1,\ b = -0.2$. Write the eqn of decision boundary.

b) Classify the points $(0.5, 0.5)$ and $(0.8, 0.2)$.

Sol: $w_1 = 1,\ w_2 = -1,\ b = -0.2$

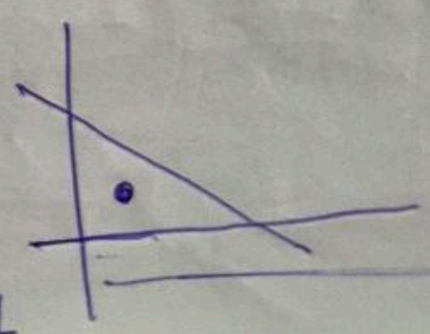
a) $z = w_1 x_1 + w_2 x_2 + b = 1 \cdot x_1 + (-1) x_2 + (-0.2)$

$= \boxed{x_1 - x_2 - 0.2}$ Ans

b) $x_1 = 0.5,\ x_2 = 0.5$

$z = 0.5 - 0.5 - 0.2 = -0.2$

$f(-0.2) = 0$



$x_1 = 0.8,\ x_2 = 0.2$

$z = 0.8 - 0.2 - 0.2 = 0.8,\ f(0.4) = 1$

Q) ~~What~~ For a 3 class perceptron the raw output are $z = [1.2, 0.8, -0.5]$. compute the softmax probabilities for each class.

Sol: Softmax can be used only for last layer of classification.

$z_1 = 1.2$ $\quad \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$

$z_2 = 0.8$ $\quad \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$

$z_3 = -0.5$ $\quad \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$

$$\boxed{\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_i e^{z_i}}}$$

$$z_i = e^{1.2} + e^{0.8} + e^{-0.5} = \cancel{6.152}\ 6.132$$

$$\underbrace{z_1 = 0.54,\ z_2 = 0.36,\ z_3 = 0.09}_{\substack{0.999 \approx 1 \\ \downarrow \\ \text{upon adding}}} \quad \text{(Probabilities)}$$

As $z_1 = 0.54$ (highest). Sample belongs to class 1.

Q) $z = [1000, 1001, 1002]$. Calculate the softmax probabilities for each class.

Sol: Softmax can be used only for last layer of classification.

$$\boxed{z_i = \frac{e^{z_i}}{\sum_i e^{z_i}}}$$

$$z_i = e^{1000} + e^{1001} + e^{1002} = 2.1882 \ldots \times 10^{435}$$

$$z_1 = \frac{e^{1000}}{2.188 \times 10^{435}} = 7.48 \times 10^{622}$$

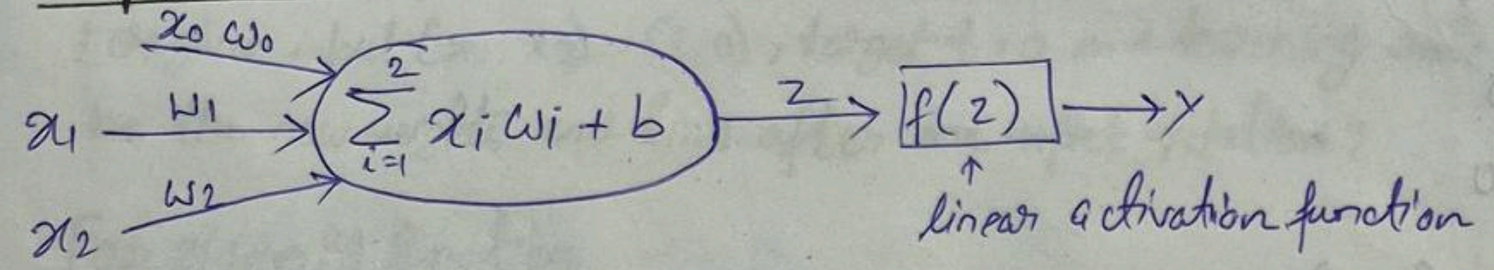
$$z_2 = \frac{e^{1001}}{2.188 \times 10^{435}} = \cancel{2.90 \times 10^{1}}\ 2.955 \times 10^{245}$$

$$z_3 = \frac{e^{1002}}{2.188 \times 10^{435}} = 1.503 \times 10^{-186}$$

Highest value is z_1. So sample belongs to class 1

6/10/25

Perceptron as a Regression -



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

$$\boxed{f(z) = z}$$

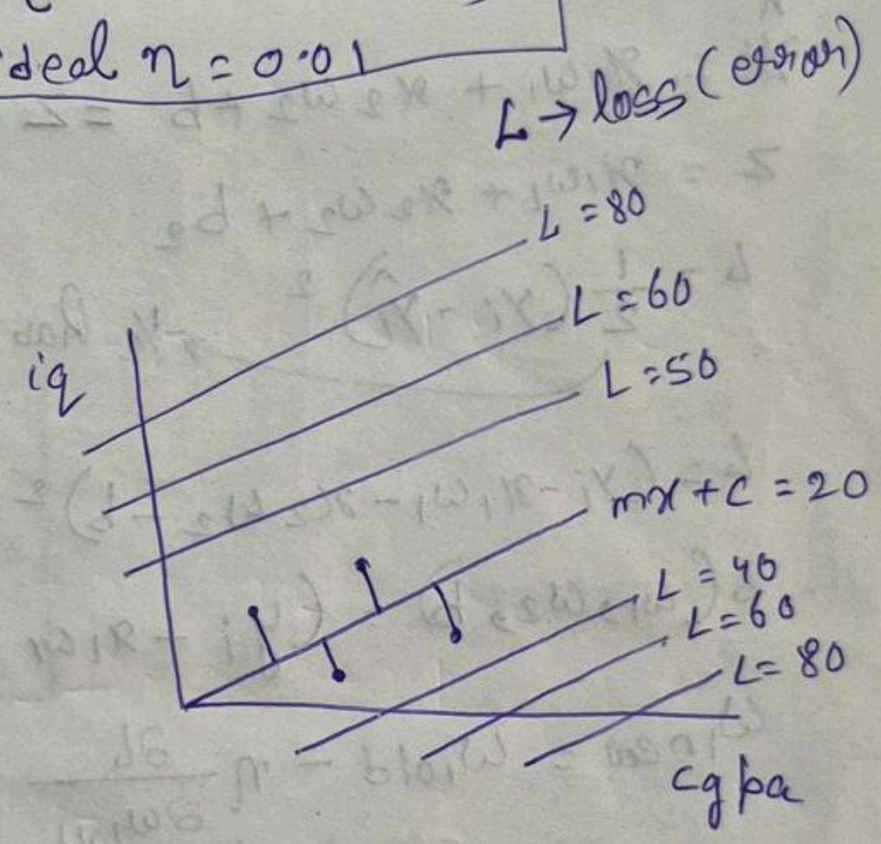
$\boxed{f(x) = x}$ → linear activation function

Perceptron training (Linear)

$$w_{1new} = w_{1old} + \eta (t-y) x_1$$
$$w_{2new} = w_{2old} + \eta (t-y) x_2$$
$$b_{new} = b_{old} + \eta (t-y)$$

$\eta = 0$ to 1 (always)
ideal $\eta = 0.01$

cgpa	iq	performance
82	8	1
90	9	1
50	5	0
60	6	0



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

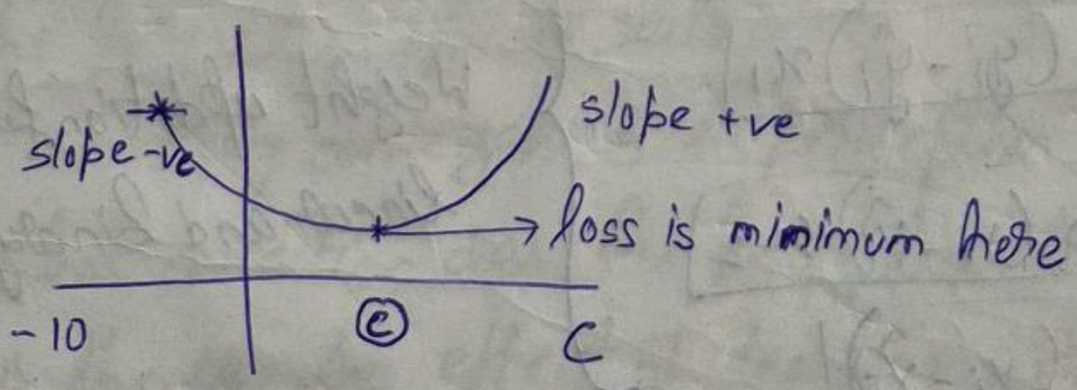
y_i → Target, $\hat{y}_i$ → Predicted

$$\hat{y}_i = m x_i + c$$

$$L(m, c) = (y_i - m x_i - c)^2$$

$$L(c) = (y_i - 78.5 x_i - c)^2, \quad m = 78.5 \text{ (alvan)}$$

$L = c^2$ (as L depends on c only).



slope in -ve (value is increasing)
slope in +ve (value is decreasing)

$$\text{Slope} = \frac{dL}{dt}$$

$$f(x) = x^2$$

$$\frac{d}{dx} f(x) \Big|_{x=2} = \frac{d}{dx} x^2$$
$$= 2x$$
$$= 4$$

$C_{new} = C_{old} - slope$

when,

$C_{old} = -10$

$slope = -40$

$C_{new} = -10 - (-40) = -10 + 40 = 30$

$\boxed{C_{new} = C_{old} - \eta\, slope}$

$C_{old} = -10$

$slope = -40$

$C_{new} = -10 - 0.01(-40) = -10 + 0.40 = -9.6$

$\hat{y}_i = x_1 w_1 + x_2 w_2 + b = z$

$z = x_1 w_1 + x_2 w_2 + b_2$

$L = \frac{1}{2}(y_i - \hat{y}_i)^2$ → ½ has introduced to make calculation easy (given in book)

$L = (y_i - x_1 w_1 - x_2 w_2 - b)^2$

$L(w_1, w_2, b) = (y_i - x_1 w_1 - x_2 w_2 - b)^2$

$w_{1new} = w_{1old} - \eta \frac{\partial L}{\partial w_{1old}}$

$w_{2new} = w_{2old} - \eta \frac{\partial L}{\partial w_{2old}}$

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

$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}_i} \times \frac{\partial \hat{y}_i}{\partial z} \times \frac{\partial z}{\partial w_1} = -\frac{2}{2}(y_i - \hat{y}_i) \times 1 \times x_1$

$w_{1new} = w_{1old} - \eta(-(y_i - \hat{y}_i)x_1)$

$= \boxed{w_{1old} + \eta(y_i - \hat{y}_i)x_1}$

$w_{2new} = \boxed{w_{2old} + \eta(y_i - \hat{y}_i)x_2}$

$b_{new} = \boxed{b_{old} + \eta(y_i - \hat{y}_i)}$

Weight updation formula for linear and binary neuron.

9/10

Q) A sigmoid perceptron is initialized with weights $(w_1, w_2) = (0.7, 0.8)$ and bias $b = 0$. For given input $(x_1, x_2) = (1, 0)$, target $t = 0$ and learning rate $\eta = 0.2$. What will be the new weights and bias after one ~~repet~~ updation?

For sigmoid function

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

$$\frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}}\left(1 - \frac{1}{1+e^{-z}}\right) = \hat{y}(1-\hat{y})$$

$$\therefore \frac{\partial L}{\partial w_1} = -(y-\hat{y})\hat{y}(1-\hat{y})x_1$$

$$\frac{\partial L}{\partial w_2} = -(y-\hat{y})\hat{y}(1-\hat{y})x_2$$

$$\frac{\partial L}{\partial b} = -(y-\hat{y})\hat{y}(1-\hat{y})$$

$$\boxed{\begin{aligned} w_{1new} &= w_{1old} + \eta(y-\hat{y})\hat{y}(1-\hat{y})x_1 \\ w_{2new} &= w_{2old} + \eta(y-\hat{y})\hat{y}(1-\hat{y})x_2 \\ b_{new} &= b_{old} + \eta(y-\hat{y})\hat{y}(1-\hat{y}) \end{aligned}}$$

11/10

Q1) What's a perceptron? What are the main components of a perceptron?

Q2) Explain the working principle of a perceptron. What's the role of weights and bias in a perceptron?

Q3) Why is an activation function used in a perceptron?

Q4) What type of problem can a single layer perceptron solve?

Q5) What's the different b/w perceptron and neuron in a biological brain?

Q6) How does the perceptron learning rule work?

Q7) Explain the perceptron update rule.

Q8) What is meant by convergence in perceptron learning?

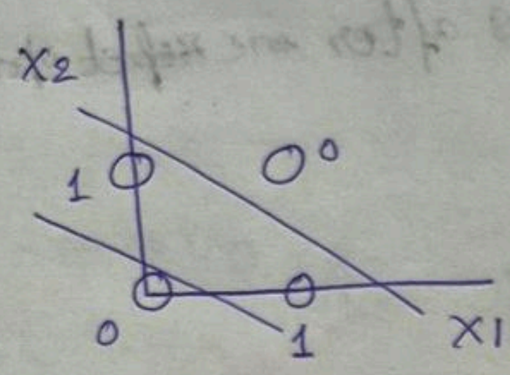
Q9) Under what condition does the perceptron learning algorithm converge?

Q10) What is the ~~effecte~~ effect of learning rate in perceptron training? What happens if the learning rate is too high or too low?

Q11) What is the stopping condn for perceptron learning?

XOR

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0



$$XOR = \underbrace{(x_1+x_2)}_{OR} \cdot \underbrace{(\overline{x_1}+\overline{x_2})}_{NAND}$$

AND

X_1	X_2	Y	(x_1+x_2)	$(\overline{x_1}+\overline{x_2})$	$h_1 \cdot h_2$
0	0	0	0	1	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	1	0	0

$OR = x_1 + x_2 - 0.5$

$AND = x_1 + x_2 - 1.5$

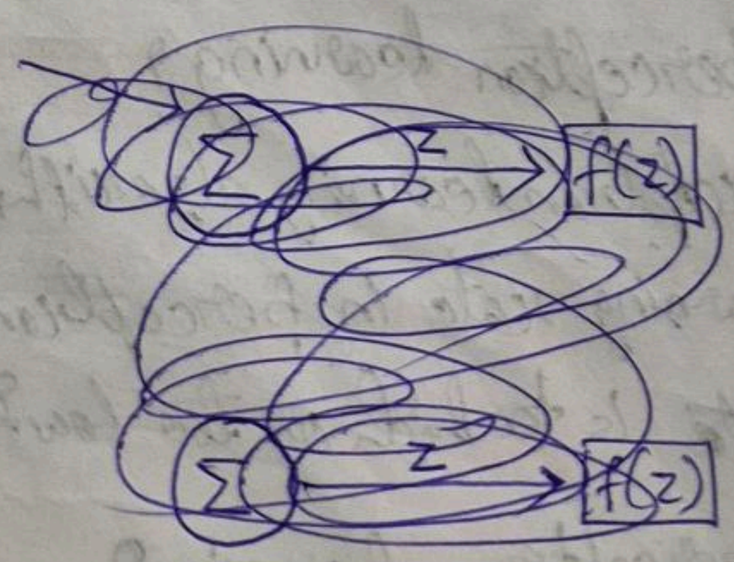
$NAND = -x_1 - x_2 + 1.5$ (NOT AND)

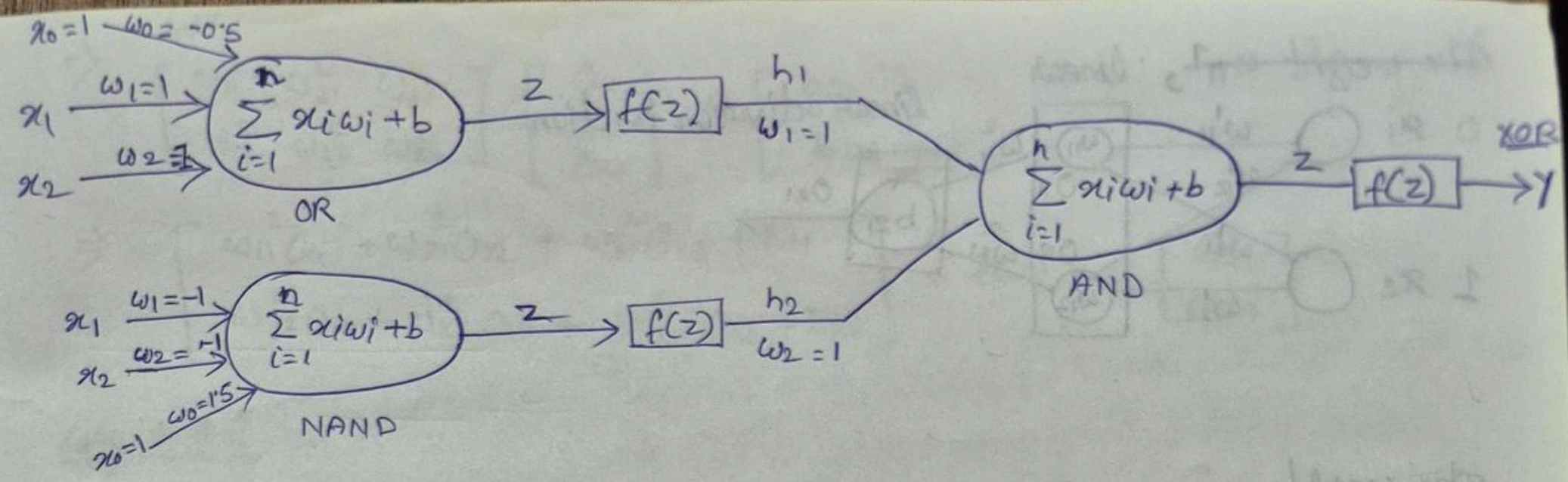
$$f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

$$\underset{X}{\begin{bmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}} \underset{W}{\begin{bmatrix} -0.5 & 1.5 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}} = \underset{Z}{\begin{bmatrix} -0.5 & 1.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \\ 1.5 & -0.5 \end{bmatrix}} \quad f(z) = \begin{array}{cc} h_1 & h_2 \\ \end{array} \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

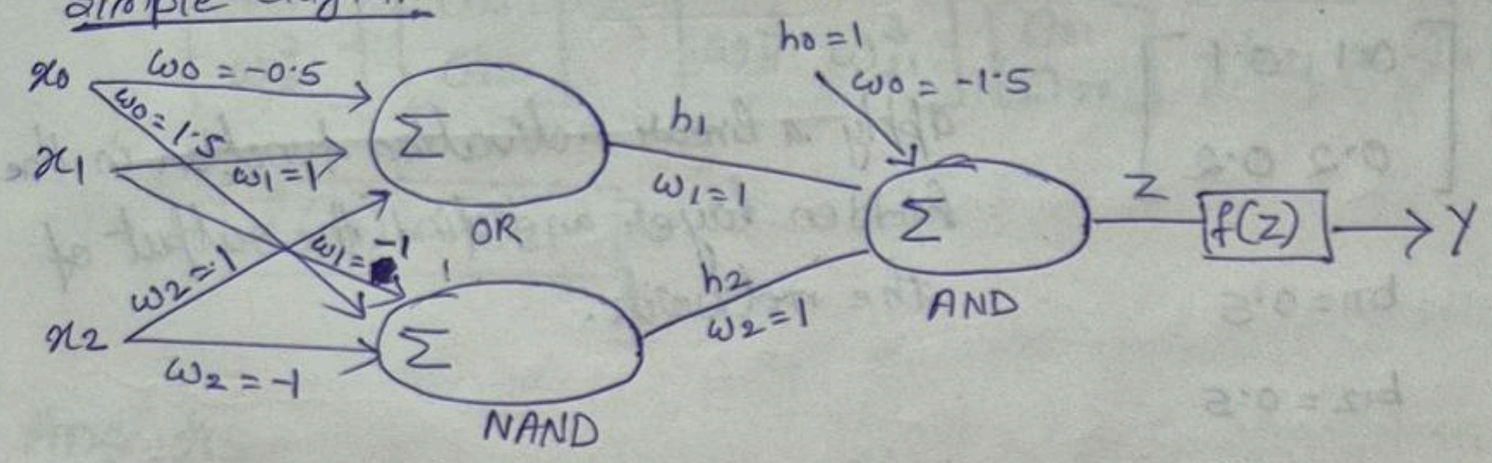
Now,

$$\begin{array}{ccc} x_0 & h_1 & h_2 \end{array} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \quad \boxed{f(z) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \underline{\underline{XOR}}}$$



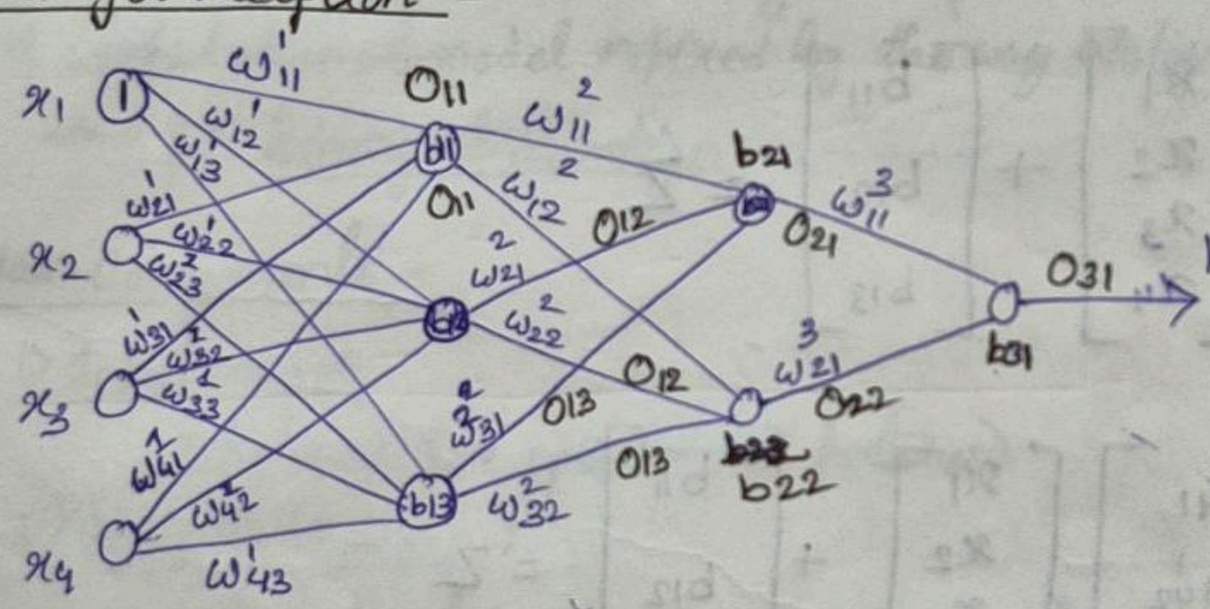


Simple diagram

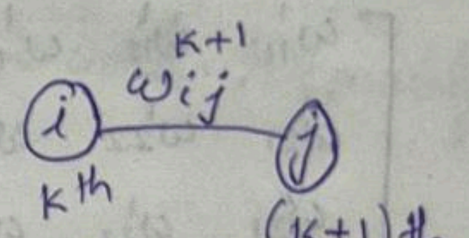


MLP
(Multilayered Perceptron)

Multilayer Perceptron -



L^0 $4\times3=12w$ + $3b$ = 15 L^1 $3\times2=6w$ + $2b$ = 8 L^2 $2\times1=2w$ + $1b$ = 3 L^3

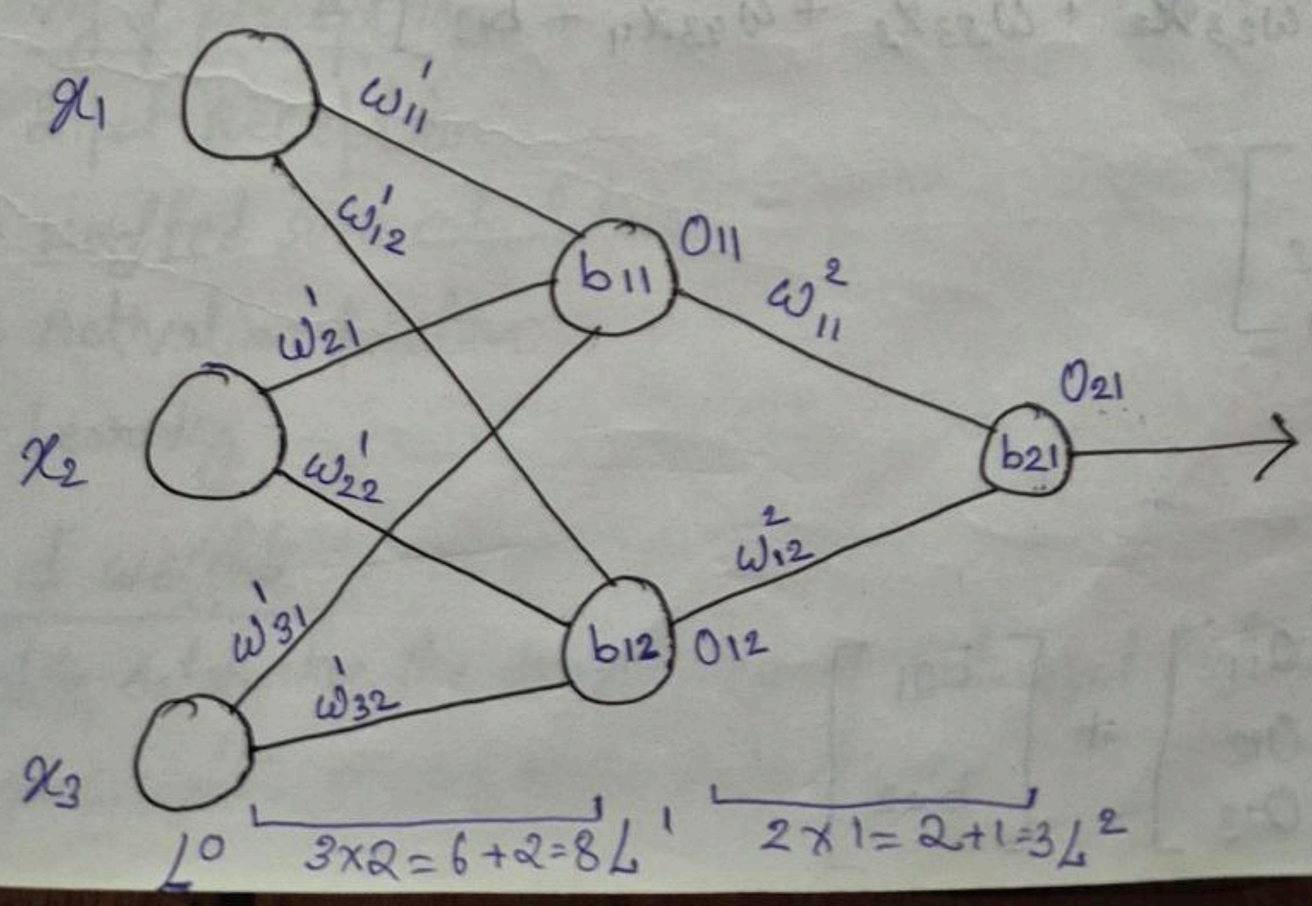


h_{ij} → i layer #
j node #

O_{ij} i → layer #
j → node #

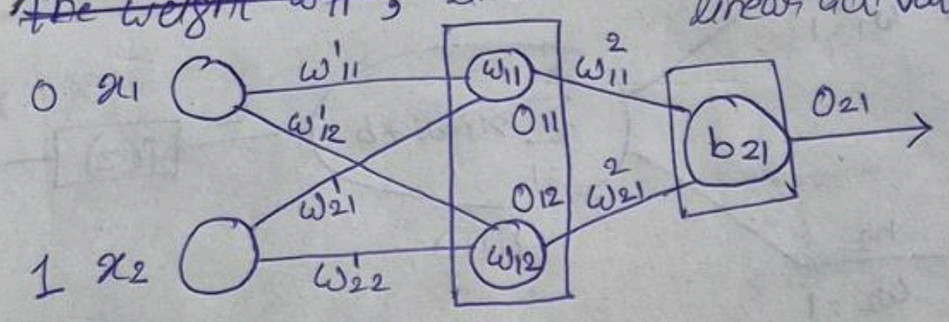
Total learnable parameters
$= 15+8+3 = 26$

Q> Annote the MLP with weight, Bias, output.



Total learnable parameters
$= 8+3 = \boxed{11}$

~~The weight w_{11}^1~~, linear linear activation function



The weight

$$\begin{bmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ 0.2 & 0.2 \end{bmatrix}$$

apply a linear activation function in the hidden layer and find the output of the network.

$w_{11}^2 = 0.1$ $\quad b_{11} = 0.5$

$w_{21}^1 = 0.2$ $\quad b_{12} = 0.5$

Layer #1

$$\underbrace{\begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \\ w_{41}^1 & w_{42}^1 & w_{43}^1 \end{bmatrix}}_{W} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = Z$$

$$\Rightarrow \underbrace{\begin{bmatrix} w_{11}^1 & w_{21}^1 & w_{31}^1 & w_{41}^1 \\ w_{12}^1 & w_{22}^1 & w_{32}^1 & w_{42}^1 \\ w_{13}^1 & w_{23}^1 & w_{33}^1 & w_{43}^1 \end{bmatrix}}_{W^T} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = Z$$

$$\Rightarrow \begin{bmatrix} w_{11}^1 x_1 + w_{21}^1 x_2 + w_{31}^1 x_3 + w_{41} x_4 + b_{11} \\ w_{12}^1 x_1 + w_{22}^1 x_2 + w_{32}^1 x_3 + w_{42}^1 x_4 + b_{12} \\ w_{13}^1 x_1 + w_{23}^1 x_2 + w_{33}^1 x_3 + w_{43}^1 x_4 + b_{12} \end{bmatrix} = Z$$

$$\Rightarrow f(z) = \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix}$$

Layer #2

$$\underbrace{\begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \\ w_{31}^2 & w_{32}^2 \end{bmatrix}}_{W} \begin{bmatrix} O_{11} \\ O_{12} \\ O_{13} \end{bmatrix} + \begin{bmatrix} b_{21} \\ b_{22} \end{bmatrix}$$

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

$$\Rightarrow \begin{bmatrix} w_{11}^2 O_{11} + w_{21}^2 O_{12} + w_{31}^2 O_{13} + b_{21} \\ w_{12}^2 O_{11} + w_{22}^2 O_{12} + w_{32}^2 O_{13} + b_{22} \end{bmatrix} = f(z) = \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix}$$

Layer #3

$$\begin{bmatrix} w_{11}^3 \\ w_{21}^3 \end{bmatrix} \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} = \begin{bmatrix} w_{11}^3 & w_{21}^3 \end{bmatrix} \begin{bmatrix} O_{21} \\ O_{22} \end{bmatrix} + b_{31} = w_{11}^3 O_{21} + w_{21}^3 O_{22} + b_{31}$$

$$\boxed{f(z) = O_{31}}$$

Answers

1) A perceptron is the simplest ~~way~~ type of ~~AI neu~~ Artificial Neutral Network a computational model inspired by the way biological neurons work. It is mainly used for binary classification.

Main Components -

i) Input Layer

Take multiple input values (features)

ii) Weights ($w_1, w_2, \ldots$)

Each function is assigned a weight showing its importance in decision.

iii) Summation function

iv) Activation function

v) Output

2) Working of perceptron

i) Input Reception

ii) Weighted sum calculation

iii) Activation function

iv) Learning

Role of weights

- Weights determine the importance of each input feature on the output.
- Larger weights means more importance.

Role of perceptron

- It acts as a decision making unit.
- It classifies data into one of two categories based on weighted combination of inputs.

3) The activation function is used in a perceptron to introduce non-linearity and decide whether the ~~neuron~~ neuron should fire (activate) or not.

~~It also used to~~.

4) A single layer perceptron can only solve linearly separable problems.

Eg: AND Gate, OR Gate can be solved but XOR can't.

5) The perceptron learning rule is a method to adjust the weights and bias of a perceptron so it can correctly classify training examples.

1) Initialise weights and bias
2) Compute output
3) Calculate error $(t-y)$
4) Update weights and bias
5) Repeate untill max no. of iterations is reached.

7) It's a simple error correction mechanism that allows the perceptron to learn linearly separable patterns by gradually adjusting weights and bias towards correct classification.

8) In perceptron learning, convergence refers to the point at which the ~~perceptr~~ perceptron stops updating its weights and bias b/c it has correctly classified in all training examples.

10) Learning rate is a small +ve constant that controls how much weight and bias are adjusted during each update.

Too high

- The perceptron may overshoot the correct solution.
- Can lead to ~~oss~~ oscillation and failure to converge

Too low

- Learning becomes slow.
- May take long time to converge.

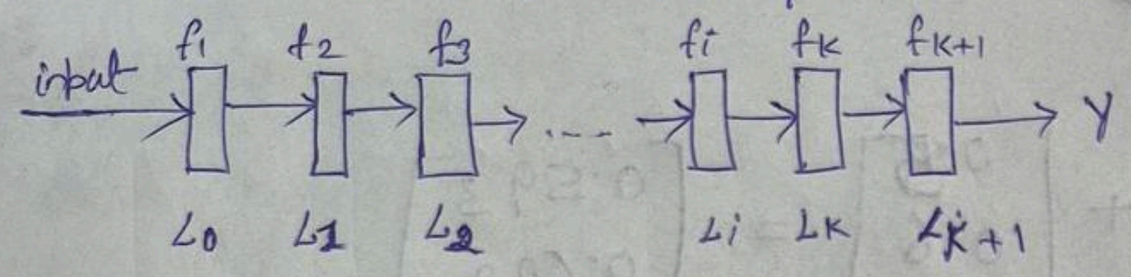
11) Stopping condition for perceptron learning are -

i) All training examples are correctly classified

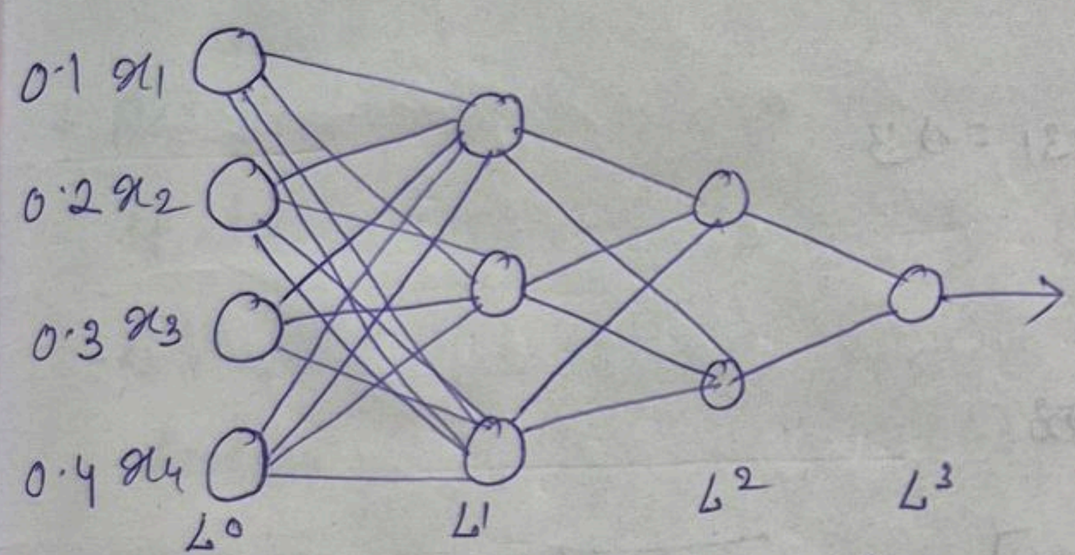
ii) Max no. of iterations (epochs) reached.

13/10

Neural network is a cascade of non-linear function



$f_K(f_i(\ldots f_3(f_2(f_1(x)))\ldots)) \rightarrow$ cascade of non-linear function



$$\left(\sigma\left(\sigma\left(\sigma\left(\underbrace{a[0]\,\omega^{[1]}+b^{[1]}}_{a[1]}\right)\omega^{[2]}+b^{[2]}\right)\omega^{[3]}+b^{[3]}\right)\right)$$

$a[2]$

$a[3]$

$a[1] = \sigma(a[0]\omega^{[1]} + b^{[1]})$

$a[2] = \sigma(a[1]\omega^{[2]} + b^{[1]})$

$a[3] = \sigma(a[2]\omega^{[3]} + b^{[1]})$

For 1st layer

12 weights, 3 bias

$$\begin{bmatrix} \omega_{11}^1 & \omega_{12}^1 & \omega_{13}^1 \\ \omega_{21}^1 & \omega_{22}^1 & \omega_{23}^1 \\ \omega_{31}^1 & \omega_{32}^1 & \omega_{33}^1 \\ \omega_{41}^1 & \omega_{42}^1 & \omega_{43}^1 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 \\ 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 \end{bmatrix}, \quad b = \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.7 \\ 0.5 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \omega_{11}^1 & \omega_{21}^1 & \omega_{31}^1 & \omega_{41}^1 \\ \omega_{12}^1 & \omega_{22}^1 & \omega_{32}^1 & \omega_{42}^1 \\ \omega_{13}^1 & \omega_{23}^1 & \omega_{33}^1 & \omega_{43}^1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0.6 \\ 0.7 \\ 0.5 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0.1 & 0.2 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.2 \\ 0.1 & 0.2 & 0.1 & 0.2 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} + \begin{bmatrix} 0.6 \\ 0.7 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.22 \\ 0.23 \\ 0.24 \end{bmatrix}$$

For layer 2

$$\begin{bmatrix} w_{11}^2 & w_{12}^2 \\ w_{21}^2 & w_{22}^2 \\ w_{31}^2 & w_{32}^2 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.2 & 0.2 \end{bmatrix}$$

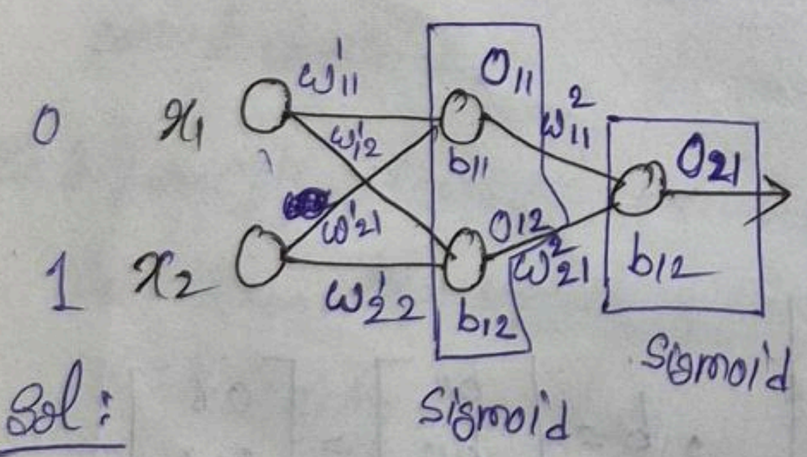
$$\Rightarrow \overset{W}{\begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.2 & 0.2 \end{bmatrix}} \begin{bmatrix} 0.22 \\ 0.23 \\ 0.24 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.593 \\ 0.693 \end{bmatrix}$$

For layer 3

$$\overset{W^T}{\begin{bmatrix} w_{11}^3 & w_{12}^3 \end{bmatrix}} = \begin{bmatrix} 0.5 & 0.6 \end{bmatrix}, \quad b_{31} = 0.3$$

$$\begin{bmatrix} 0.5 & 0.6 \end{bmatrix} \begin{bmatrix} 0.593 \\ 0.693 \end{bmatrix} + \begin{bmatrix} 0.3 \end{bmatrix} = \boxed{1.0123} \text{ Ans}$$

Q) Annotate the given network and output generated



Sol:

$$\begin{bmatrix} w_{11}^1 & w_{12}^1 \\ w_{21}^1 & w_{22}^1 \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}, \quad \begin{bmatrix} b_{11} \\ b_2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix}$$

$$\sigma\left(\begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix} \right) = \sigma \begin{bmatrix} 0.6 \\ 0.7 \end{bmatrix} = \begin{bmatrix} \frac{1}{1+e^{-0.6}} \\ \frac{1}{1+e^{-0.7}} \end{bmatrix}$$

$$= \begin{bmatrix} 0.6456 \\ 0.6681 \end{bmatrix}$$

Layer 2 ~~Now~~,

$$\sigma\left(\begin{bmatrix}0.6456\\0.6681\end{bmatrix}\overset{W}{\begin{bmatrix}0.2\\0.4\end{bmatrix}}+0.6\right) = \begin{bmatrix}0.6456 & 0.6681\end{bmatrix}\begin{bmatrix}0.2\\0.4\end{bmatrix}+0.6$$

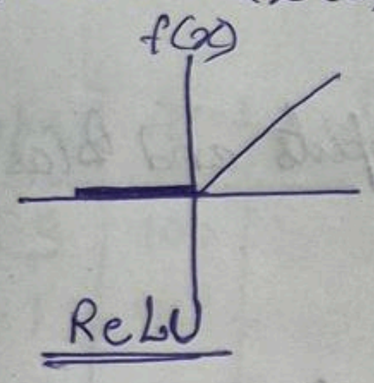
$$= 0.3963 + 0.6 = \boxed{0.99636}$$

$$\sigma(0.99636) = \frac{1}{1+e^{-0.99636}} = \boxed{0.7303 = O_{21}}$$ Ans

16/10

ReLU activation -

$f(x) = \max(0, x)$



x	$f(x)$
100	100
-80	0
5	5
-4	0

- It is differentiable

$$f'(x) = \begin{cases} 0, & x \le 0 \\ 1, & x > 0 \end{cases}$$

- ReLU stands for Rectified Linear Unit. If the input 'x' is (-ve) then $f(x) = 0$ if, input is (+ve) then $f(x) = x$.

- Although it gives an impression of linear function, ReLU has a derivative function and allows for back propagation making it computationally efficient.

- ReLU doesn't make all the neurons active at the same time. The neuron will only be deactivated if the output of the linear transformation is less than 0.

- The limitation is, ReLU faces a Dying ReLU problem.

~~Training process of MLP~~ -

General Training process of MLP -

Training process of MLP involves the following steps -

i) Forward propagation - Here, input data passes ~~through~~ through the network layer by layer producing an output.

ii) Loss calculation - Compare the output to the true levels using a loss function (MSE, cross entropy).

iii) Back propagation - calculate gradients of the loss with respect to each weight by propagating error backward through the network.

iv) Weight updates - Use an optimization algorithm (like gradient descent) to adjust weights and biases to minimize the loss.

v) Repeat the above steps ~~repe~~ iteratively over many epochs untill the network converges to a good solution.

Forward propagation in MLP - Forward propagation is the process of passing input through the network to get output.

Step 1: For each neuron compute a weighted sum of inputs and bias.

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

Step 2: Apply an activation function to get the neuron output, ~~ou~~ output of 1 layer becomes input to the next layer.

Continue layer by layer untill final output layer produces predictions.

Loss function - is a method of evaluating how well your algorithm is modelling your data. If the loss function value is high your algorithm is performing poorly and if the loss function value is small your algorithm is performing great.

- Loss function is a mathematical funcn of the parameters of Machine learning parameter. If we adjust the parameters the loss function value changes.

Categories of loss function -

~~i) For regression~~

Regression	classification	Autoencoder	GAN
MSE	Primary cross entropy	KL Divergence	Discriminator loss
MAE	Categorical cross entropy		min max GAN loss.
huber loss	hinge loss		

23/10

Regression

① Mean Squared Error (MSE) | L_2 loss | squared loss -

Difference b/w loss function and cost function

Loss function	Cost function
is calculated for a single data sample	is calculated for a batch (entire data)
$loss = (y_i - \hat{y}_i)^2$	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

cgpa	iq	Package (lpa)	Predicted Value ($\hat{y}$)	error	error^2
6.3	100	6.3	6.1	0.2	0.04
7.1	91	4.1	4	0.1	0.01
8.5	83	3.5	3.7	-0.2	0.04
9.2	102	7.2	7	0.2	0.04
		40	8	32	1024

Advantages and limitation of MSE -

Advantage

1. It is easy to interpret
2. MSE is differentiable $\quad \frac{\partial L}{\partial y_i} = -2(y_i - \hat{y}_i)$
3. It has only one local minima

Disadvantage

1. error unit is different
2. It is not robust to outliers

MLP set up for regression problem where MSE is used or loss function is used.

- Output layer should contain one neuron and for output neuron the linear activation function is required.
- We can use MSE as a loss function ~~of~~ if the dataset is free of outliers.

② Mean Absolute Error (MAE) -

Loss function	Cost function
$L = \lvert y_i - \hat{y}_i \rvert$	$Cost = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i \rvert$

cgpa	iq	package	lpa	predicted value	error	|error|
					0.2	0.2
					0.1	0.1
					-0.2	0.2
					0.2	0.2
					3.2	

Advantages

1. It is intuitive to understand
2. The error unit is same
3. It's robust to outliers.

- In MAE each error contributes linearly to the total loss.

Disadvantages

1. It's not differentiable

③ Huber loss

$$L = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{for } |y-\hat{y}| \le \delta \\ \delta(|y-\hat{y}| - \frac{1}{2}\delta) & \text{for } |y-\hat{y}| > \delta \end{cases}$$

→ acts as MSE

→ quadratic branch

→ linear branch

→ acts as MAE

Q) Given $y=5$, $\hat{y}=3$, $\delta=1$. Find the huber loss

Sol: $e = |y-\hat{y}| = |5-3| = 2$

$e > \delta$, so linear branch will be choosen

$L = 1(2 - \frac{1}{2}(1)) = 2 - 0.5 = \boxed{1.5}$ Ans

Q) $y=1$, $\hat{y}=0.5$, $\delta=1$. Find the ~~huber loss~~ huber loss

Sol: $e = |y-\hat{y}| = |1-0.5| = 0.5$

$\delta > e$, so, quadratic ~~branch will~~ branch will be choosen

$L = \frac{1}{2}(0.5)^2 = \frac{1}{2} \times 0.25 = \boxed{0.125}$ Ans

Advantage

① It is also differentiable (when 25% data is outliers)

Classification -

1) Binary Cross entropy | log loss -

- It is used for Binary classification problem

Loss $L = -y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)$

$$\text{cost} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

cgpa	iq	performance (y_i)	$\hat{y}_i$
80	8	1	0.73
60	6	0	
50	5	0	

25/10

Q1) What is role of hidden layers in MLP?

Q2) How does an MLP learn from data explain the general training process?

Q3) What are common activation functions used in MLPs and what are ~~them~~ their pros and cons?

Q4) How does the no. of hidden layers and neurons affect model capacity and performance?

Q5) What is the loss function in the MLP training?

Q6) How can MLP be used for classification and regression?

Q7) MLP best suited for what kind of problems?

2) Categorical cross entropy - (used in softmax regression)

cgpa	iq	placed	one hot encoding
8	80	Yes	1 0 0
6	60	NO	0 1 0
7	70	may be	0 0 1

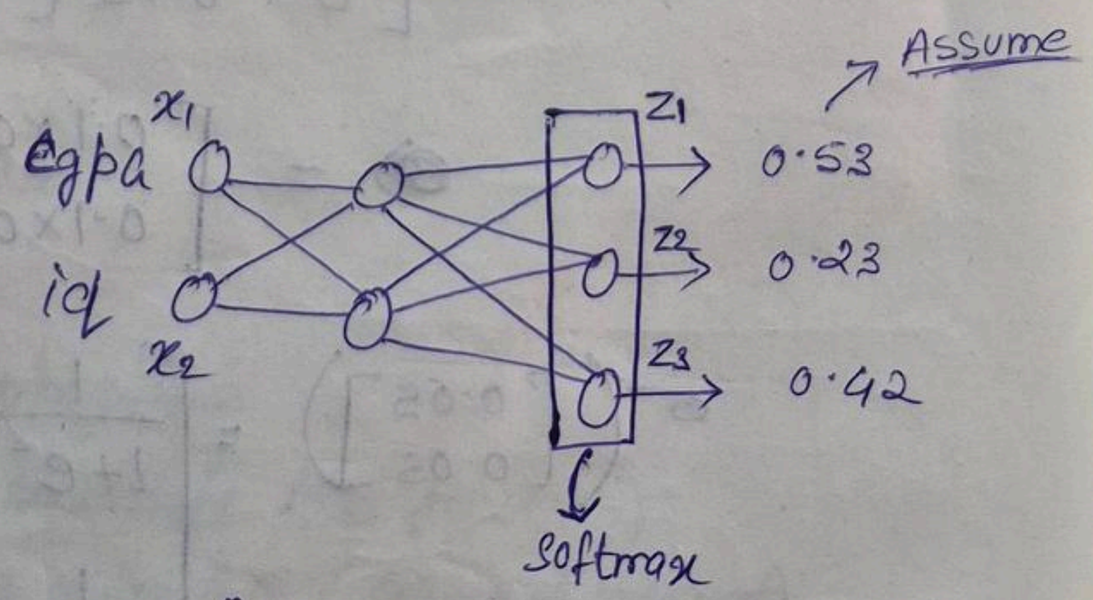
- It is used in a loss function in multiclass classification problem.

$$\boxed{L = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)}$$

K = no. of classes

$y_j \rightarrow$ true class

$\hat{y}_j \rightarrow$ predicted class



$\hat{y} = [0.53, 0.23, 0.42]$

$y_1 = [1 \quad 0 \quad 0]$

$L = -y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3)$

$$\Rightarrow L = -1\log(0.53) - 0\,\cancel{\log(0.23)}^{\,0} - 0\,\cancel{\log(0.42)}^{\,0}$$

$$= -\log(0.53) = \boxed{0.275}$$

Q) Design a MLP with the following information.

input(2) $\rightarrow$ hidden(2) $\rightarrow$ output(3)
(sigmoid) (softmax)

a) Find the predicted value ($\hat{y}$) using forward propagation

$$w^{[1]} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} \quad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

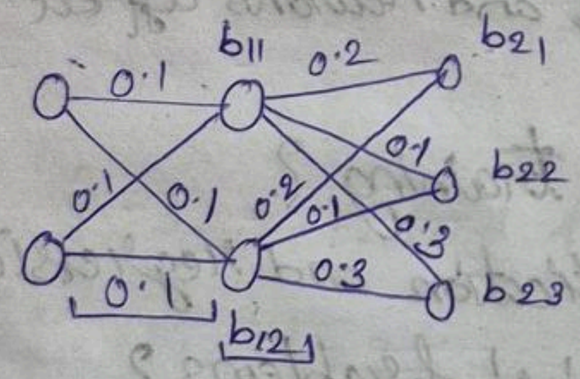
$$w^{[2]} = \begin{bmatrix} 0.2 & 0.2 \\ 0.1 & 0.1 \\ 0.3 & 0.3 \end{bmatrix} \quad b^{[2]} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.4 \end{bmatrix}$$

$$x = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

b) Find the categorical cross entropy loss for this sample having ~~trace~~ true value

$$Y = [1 \quad 0 \quad 0]$$

Sol:



a) Layer #1

$$\sigma\left(\begin{bmatrix} w^1_{11} & w^1_{12} \\ w^1_{21} & w^1_{22} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \end{bmatrix}\right) = \begin{bmatrix} O_{11} \\ O_{12} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \cancel{0.1\times0.2 + 0.1\times0.3 + 0.1\times0.2 + 0.1\times0.3}$$

$$= \begin{bmatrix} 0.1\times0.2 + 0.1\times0.3 \\ 0.1\times0.2 + 0.1\times0.3 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix} \begin{matrix} z_0 \\ z_1 \end{matrix}$$

$$\sigma\left(\begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix}\right) = \begin{bmatrix} \dfrac{1}{1+e^{-0.05}} \\ \dfrac{1}{1+e^{-0.05}} \end{bmatrix} = \begin{bmatrix} 0.5125 \\ 0.5125 \end{bmatrix} \begin{matrix} O_{11} \\ O_{12} \end{matrix}$$

Layer#2

$$\underset{\omega}{\begin{bmatrix} 0.2 & 0.2 \\ 0.1 & 0.1 \\ 0.3 & 0.3 \end{bmatrix}} \underset{o}{\begin{bmatrix} 0.512 \\ 0.512 \end{bmatrix}} + \underset{b}{\begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}} = \begin{bmatrix} 0.2048 \\ 0.1024 \\ 0.3072 \end{bmatrix} + \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.4048 \\ 0.4024 \\ 0.7072 \end{bmatrix}$$

Softmax

$$Z_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}, \quad Z_2 = \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}, \quad Z_3 = \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$$

$$= 0.298 \qquad = 0.297 \qquad = 0.403$$

class of the sample is 3.

b) $\hat{y} = [0.298 \quad 0.297 \quad 0.403]$

$$L = -y_1 \log(\hat{y}_1) - \cancel{y_2 \log(\hat{y}_2)}^{0} - \cancel{y_3 \log(\hat{y}_2)}^{0}$$

$$\cancel{= -0.298 \log(}$$

$$= -1 \log(0.298) = \boxed{0.525} \text{ Ans}$$

Backpropagation learning - is an algorithm for supervised learning of artificial neural networks using gradient descent.

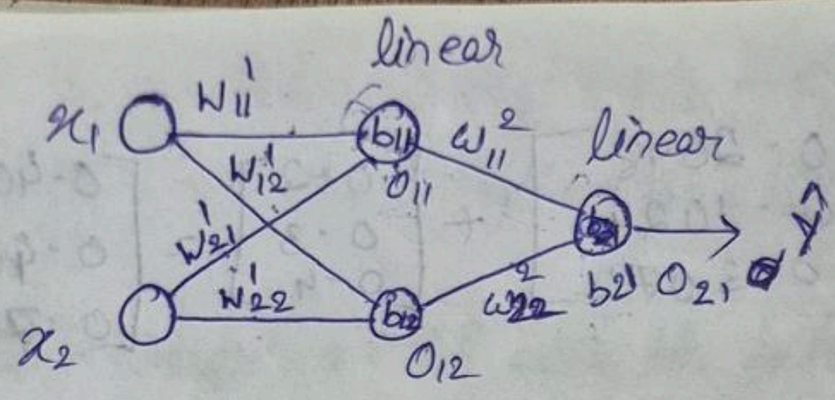
- Given an artificial neural network and an error function the methods calculate the gradient of the error function with respect to neural network, weights and biases.

- Backpropagation is an algorithm to train a neural network and training a neural network means for a given data it finds the optimum values of weights and biases.

```
for i in range(epochs):
    for j in range(x.shape[0])
        select row1 (random)
        predict using forward propagation
        calculate loss (using a suitable loss function)
        update weights and bias using GD (Gradient descent)
```

$$W_n = W_0 - \eta \frac{\partial L}{\partial W_0}$$

Calculate the



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

9 learnable parameters are there in the above diagram

<u>Minimize the loss</u>

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

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

$$z = \hat{y} = O_{11}\omega_{11}^2 + O_{12}\omega_{21}^2 + b_{21}$$

$$\hat{y} = \underbrace{\left[x_1 \omega_{11}^1 + x_2 \omega_{21}^1 + b_{11}\right]}_{O_{11}} \omega_{11}^2 + \left[x_1 \omega_{12}^1 + x_2 \omega_{22}^1 + b_{12}\right] \omega_{21}^2 + b_{21}$$

<u>1st</u>

$$\omega_{11\,new}^2 = \omega_{11\,old}^2 - \eta \frac{\partial L}{\partial \omega_{11}^2} = 2(\hat{y} - y) O_{11}$$

$$\omega_{21\,new}^2 = \omega_{21\,old}^2 - \eta \frac{\partial L}{\partial \omega_{21}^2} = 2(\hat{y} - y) O_{12}$$

$$b_{21\,new} = b_{21\,old} - \eta \frac{\partial L}{\partial b_{21}} = 2(\hat{y} - y)$$

$$\omega_{11\,new}^1 = \omega_{11\,old}^1 - \eta \frac{\partial L}{\partial \omega_{11}^1}, \quad \frac{\partial L}{\partial \omega_{11}^2} = \underbrace{\frac{\partial L}{\partial \hat{y}}}_{2(\hat{y}-y)} \cdot \underbrace{\frac{\partial \hat{y}}{\partial z}}_{1} \cdot \underbrace{\frac{\partial z}{\partial \omega_{11}^2}}_{O_{11}}$$

$$\omega_{12\,new}^1 = \omega_{12\,old}^1 - \eta \frac{\partial L}{\partial \omega_{12}^1}$$

$$\omega_{21\,new}^1 = \omega_{21\,old}^1 - \eta \frac{\partial L}{\partial \omega_{12}^1}, \quad \frac{\partial L}{\partial \omega_{21}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \underbrace{\frac{\partial z}{\partial \omega_{21}^2}}_{O_{12}}$$

$$\omega_{22\,new}^1 = \omega_{22\,old}^1 - \eta \frac{\partial L}{\partial \omega_{22}^1}$$

$$b_{11\,new} = b_{11\,old} - \eta \frac{\partial L}{\partial b_{11}}$$

$$b_{12\,new} = b_{12\,old} - \eta \frac{\partial L}{\partial b_{12}}, \quad \frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \underbrace{\frac{\partial z}{\partial b_{21}}}_{1}$$

$$\frac{\partial}{\partial \hat{y}} (y - \hat{y})^2 = -2(y - \hat{y}) = 2(\hat{y} - y)$$

$$\frac{\partial \hat{y}}{\partial z} = \frac{\partial f(z)}{\partial z} = \frac{\partial z}{\partial z} = 1$$

$$\frac{\partial z}{\partial \omega_{11}^2} = O_{11}$$

2nd

$L \to \hat{y} \to z_f \to O_{11} \to z_{11} \to w^1_{11}$

$$\frac{\partial L}{\partial w^1_{11}} = \boxed{\overset{2(\hat{y}-y)}{\frac{\partial L}{\partial \hat{y}}} \; \overset{1}{\frac{\partial \hat{y}}{\partial z_f}} \; \overset{w^2_{11}}{\frac{\partial z_f}{\partial O_{11}}} \; \overset{1 \text{ (as linear activation fon}^n)}{\frac{\partial O_{11}}{\partial z_{11}}}} \; \overset{\to x_1}{\frac{\partial z_{11}}{\partial w^1_{11}}} = 2(\hat{y}-y)\, w^2_{11}\, x_1$$

$$\frac{\partial L}{\partial w^1_{21}} = \boxed{\frac{\partial L}{\partial \hat{y}} \; \frac{\partial \hat{y}}{\partial z_f} \; \frac{\partial z_f}{\partial O_{11}} \; \frac{\partial O_{11}}{\partial z_{11}}} \; \overset{\to x_2}{\frac{\partial z_{11}}{\partial w^1_{21}}} = 2(\hat{y}-y)\, w^2_{21}\, x_2$$

$$\frac{\partial L}{\partial b_{11}} = \boxed{\frac{\partial L}{\partial \hat{y}} \; \frac{\partial \hat{y}}{\partial z_f} \; \frac{\partial z_f}{\partial O_{11}} \; \frac{\partial O_{11}}{\partial z_{11}}} \; \overset{\to 1}{\frac{\partial z_{11}}{\partial b_{11}}} = 2(\hat{y}-y)\, 1 = 2(\hat{y}-y)\, w^2_{11}$$

27/10

Q) Design an MLP with the following setup

input (2) → hidden (2) → output (1)

(sigmoid) (linear)

a) Find the $\hat{y}$ of the model for the following information

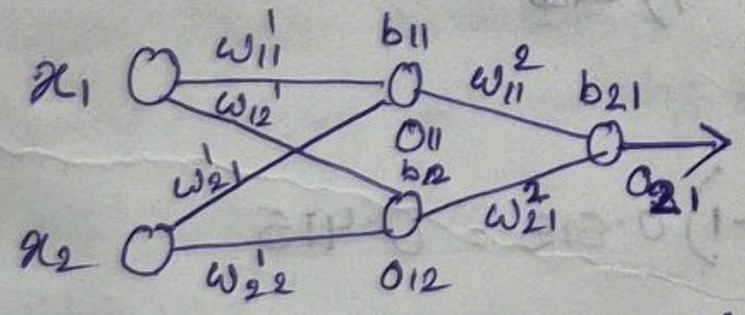
$x = \begin{bmatrix} 0.6 \\ 0.9 \end{bmatrix}$, $y_{true} = 1$, $\eta = 0.1$

$w^{[1]} = \begin{bmatrix} 0.2 & -0.4 \\ 0.7 & 0.1 \end{bmatrix}$, $b^{[1]} = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$

$w^{[2]} = \begin{bmatrix} 0.3 & -0.5 \end{bmatrix}$, $b^{[2]} = 0.05$

b) Find the updated weights, and bias for hidden to output layer. by choosing a suitable loss function for the given setup.

Sol:



Loss function = MSE

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

* If a MLP solves a <u>regression problem</u> the output layer should contain only 1 neuron and on that neuron <u>linear activation function</u> need to be applied.

* If an MLP acts as binary classifier the output layer should contain one neuron and sigmoid activation function need to be applied on that neuron.

* If an MLP acts as a multiclass classifier the output layer should contain 'K' no. of neurons where, K = no. of classes and on the output layer softmax activation function need to be applied.

* Now, for regression function loss function applied <u>MSE, MAE, huber loss</u>.

* " " Binary function " " " <u>binary cross entropy</u>.

* Now for the multiclass classifier loss function applied categorical cross entropy

Sol: a) Layer #1

$$\sigma\left(\underbrace{\begin{bmatrix} 0.2 & -0.4 \\ 0.7 & 0.1 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}}_{X} + \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}\right) = \sigma\left(\underbrace{\begin{bmatrix} 0.06 \\ 0.66 \end{bmatrix}}_{Z}\right) = \begin{bmatrix} \frac{1}{1+e^{-0.06}} \\ \frac{1}{1+e^{-0.66}} \end{bmatrix}$$

$$= \begin{bmatrix} 0.515 \\ 0.66 \end{bmatrix} = \begin{bmatrix} O_{11} \\ O_{12} \end{bmatrix}$$

Layer #2

$$\begin{bmatrix} 0.3 & -0.5 \end{bmatrix} \begin{bmatrix} 0.515 \\ 0.66 \end{bmatrix} + \begin{bmatrix} 0.05 \end{bmatrix} = \begin{bmatrix} -0.125 \end{bmatrix} = \hat{y}$$

b) w_{11}^2, w_{21}^2, b_{21}, loss function is squared loss $\Rightarrow L = (y - \hat{y})^2$

$$w_{11\,new}^2 = w_{11}^2 - \eta \frac{\partial L}{\partial w_{11}^2} \qquad = 2(\hat{y} - y)\, O_{11}$$

$$w_{21\,new}^2 = w_{21}^2 - \eta \frac{\partial L}{\partial w_{21}^2} \qquad = 2(\hat{y} - y)\, O_{12}$$

$$b_{21\,new} = b_{21\,old} - \eta \frac{\partial L}{\partial b_{21}} \qquad = 2(\hat{y} - y)$$

$$w_{11\,new}^2 = 0.3 - 0.1 \times 2(-0.125 - 1)\, 0.515 = 0.415$$

$$w_{21\,new}^2 = -0.5 - 0.1 \times 2(-0.125 - 1)\, 0.66 = -0.3515$$

$$b_{21\,new} = 0.05 - 0.1 \times 2(-0.125 - 1) = 0.275$$

c) Find the ~~input~~ updated weights and biases for the input to hidden layer.

hidden to output layer -

w_{11}^2, w_{21}^2, b_{21}

$$Z_f = O_{11} w_{11}^2 + O_{12} w_{21} + b_{21}$$

$$w_{11\,new}^2 = w_{11\,old}^2 - \eta \frac{\partial L}{\partial w_{11}^2}, \quad \frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial Z_f} \cdot \frac{\partial Z_f}{\partial w_{11}^2} = 2(\hat{y} - y) \times 1 \times O_{11}$$

$$w_{21\,new}^2 = w_{21\,old}^2 - \eta \frac{\partial L}{\partial w_{21}^2}, \quad \frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial Z_f} \times \frac{\partial Z_f}{\partial w_{21}^2} = 2(\hat{y} - y) \times 1 \times O_{12}$$

$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b_{21}}, \quad \frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial Z_f} \times \frac{\partial Z_f}{\partial b_{21}} = 2(\hat{y} - y) \times 1 \times 1$$

hidden to input

$w'_{11}, w'_{12}, w'_{21}, w'_{22}, b_{11}, b_{12}$

$$w'_{11new} = w'_{11old} - \eta \frac{\partial L}{\partial w'_{11}} \quad \text{—— (1)}$$

write the updation rule for all weights and biases as eq (1)

$L \to \hat{y} \to z_f \to o_{11} \to z_{11} \to w'_{11}$

$L \to \hat{y} \to z_f \to o_{12} \to z_{12} \to w'_{12}$

$L \to \hat{y} \to z_f \to o_{11} \to z_{11} \to w'_{21}$

$L \to \hat{y} \to z_f \to o_{12} \to z_{12} \to w'_{22}$

$L \to \hat{y} \to z_f \to o_{11} \to z_{11} \to b_{11}$

$L \to \hat{y} \to z_f \to o_{12} \to z_{12} \to b_{12}$

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w'_{11}}$$

$2(\hat{y}-y)$; 1 ; w''_{11} ; $o_{11}(1-o_{11})$; x_1

$$\frac{\partial L}{\partial w'_{12}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w_{12}}$$

w''_{21} ; $o_{12}(1-o_{12})$; x_1

$$\frac{\partial L}{\partial w'_{21}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w'_{21}}$$

w''_{11} ; $o_{11}(1-o_{11})$; x_2

$$\frac{\partial L}{\partial w'_{22}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{12}} \times \frac{\partial o_{11}}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial w'_{22}}$$

w''_{12} ; x_2

$$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial b_{11}}$$

w''_{11} ; 1

$$\frac{\partial L}{\partial b_{12}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial o_{12}} \times \frac{\partial o_{12}}{\partial z_{12}} \times \frac{\partial z_{12}}{\partial b_{12}}$$

w''_{21} ; 1

$$\frac{\partial o_{11}}{\partial z_{11}} = \frac{\partial}{\partial z_{11}} \sigma(z_{11})$$

$$= \frac{\partial}{\partial z_{11}} \left(\frac{1}{1+e^{-z_{11}}}\right)$$

$$= \frac{e^{-z_{11}}}{(1+e^{-z_{11}})^2} = o_{11}$$

$$\frac{\partial}{\partial z_{11}} \frac{e^{-z_{11}}}{(1+e^{-z_{11}})^2}$$

$$= \frac{1}{1+e^{-z_{11}}}\left(1 - \frac{1}{1+e^{-z_{11}}}\right)$$

$$= o_{11}(1-o_{11})$$

Training the MLP -

- Forward propagation ✓
- Back propagation ✓

Neural network performance enhancement -

i) Tunning the hyper parameters

a) hidden layer b) Batch

c) No. of neurons d) Learning rate

e) Activation function

f) optimizers

g) epochs

ii) Solving neural network problems

a) vanishing gradient

~~b) verifying problem~~

b) overfitting problem.

Variants of Gradient descent -

i) Batch GD

ii) stochastic GD

iii) Mini Batch GD

Batch GD

epoch (1) Training Dataset = $\underbrace{100 \text{ samples}}_{\text{Batch}}$

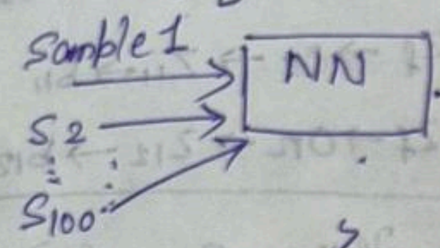
Batch $\to$ $\boxed{NN}$ $\to$ $\hat{y}_{Batch}$ - y - L, then update weights and biases

So, $\boxed{\text{no. of updates} = \text{no. of epochs}}$ (Batch GD)

Stochastic Gradient Descent –

100 training Data

~~Sample 1 → NN → $\hat{Y}_1$~~
~~epoch (200)~~ S_2 epoch(200){
size of training of sample (100){



$\hat{Y}_1$ Y → L update parameters

$\hat{Y}_2$ Y → L " "

$\hat{Y}_3$ Y → L " "

∴ In 1 epoch → 100 updates happens in (SGD)

So, in 200 epochs, 200×100 = 20,000 updates

Batch GD	Stochastic GD
i) Computational speed is ~~high~~ low.	i) ~~Computational~~ convergence speed is high
ii) Convergence speed is low.	ii) Computational speed is high.
iii) Stable	iii) Noise and unstable.

Mini Batch Gradient Descent –

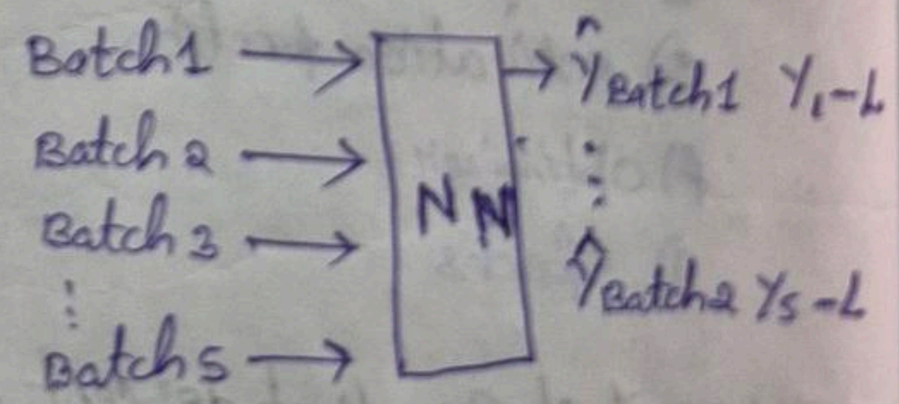
100 training samples

into no. of mini Batches

batch size = 20

no. of Batches = 5

epoch (200){
Batch (5){



∴ 1 epoch = 5 updates

200 epochs = 200×5 = 1000 updates.

Overfitting data – For training data the loss is decreasing.
For test data the loss is increasing.

* Training accuracy is high but test accuracy is low.

Solutions to combat overfitting -

i) Early stopping
ii) Dropout
iii) Regularization
iv) Data augmentation

Early stopping -

Before testing the model splitting ratio is -

Training set = 50% validation set = 20% Test set = 30%

1000{

training set → NN → L training L validation

L training L validation

500 ↓ ↓

501 ↓ ↑

→ Comes out of loop when loss of training decreasing & loss of validation increasing.

(i.e) Early stopping.

* no. of hidden layer increases with increasing complexity of the model.

No. of neurons

$$n_h = \frac{n_{input} + n_{out}}{2} \quad \text{(linear)}$$

n_h → no. of hidden layer

* hidden layer activation function → sigmoid / tanh / ReLU

Vanishing gradient problem -

If we have a lot of hidden layer (eg: 30, 40) it is known as deep neural ~~network~~ network.

- $0.1 \times 0.1 \times 0.1 \times 0.1 = 0.00001 \approx 0$ (vanishing)
- ReLU activation fun^n is used to avoid vanishing ~~gradient~~ gradient problem.
- If η is high it will keep oscillating in a the range

$$w'_{11}\,new = w'_{11}\,old - \cancel{\eta \frac{\partial L}{\partial w'_{11}}} \rightarrow 0$$ minimize the loss

$$\frac{\partial L}{\partial w'_{11}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_8} \times \frac{\partial z_8}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{11}} \rightarrow 0 \times \frac{\partial z_{11}}{\partial w_{11}}$$

(derivation of sigmoid activation function, tanh activation function).

- To solve this we need to use another activation function ReLU.

$\eta = 0.5$

$\eta \left\{ \begin{array}{l} BGD \\ MGD \\ SGD \end{array} \right\}$

3/11

Chapter - 6

Challenges in Gradient Descent -

Step 1) Randomly initialize the parameter of neural network (w, b).

Step 2) Perform forward propagation

Step 3) Calculate the cost

Step 4) Update the parameters through back propagation taking gradient w.r.t the parameters.

Challenges in GD -

The problems in the current optimizers are

i) Deciding the correct value for the learning rate.

(η value will be same across all epochs).

$$\boxed{w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_t}}$$

ii) So, to overcome this particular problem, we can use learning rate scheduling

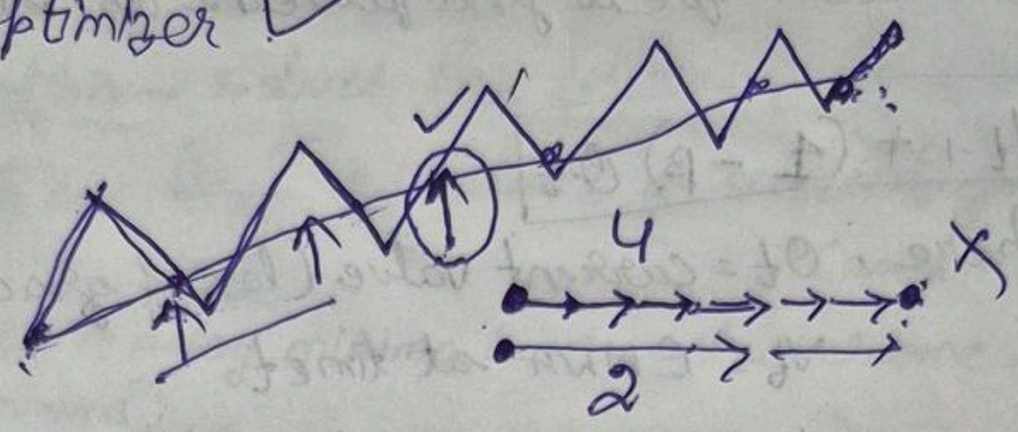
- Can't be applied to all neural network

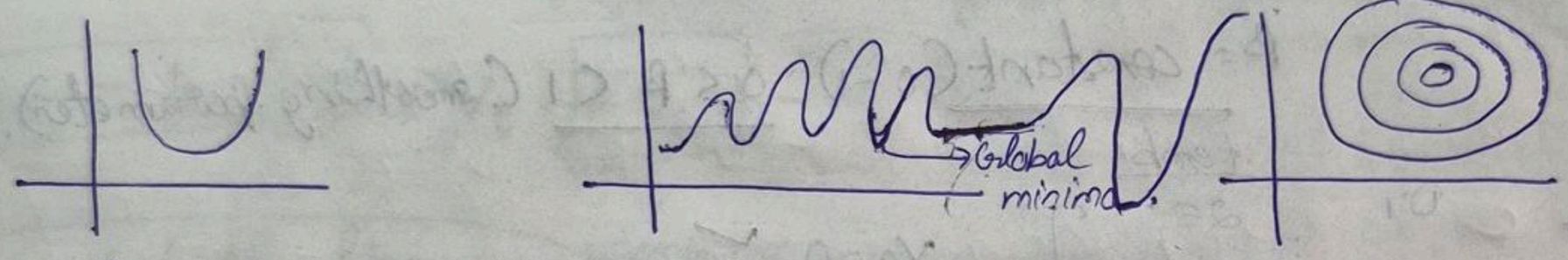
iii) We have to find the optimal weight and bias (for several parameters if there are 2 parameters w_1 & w_2 for both we can't ~~kep~~ keep separate learning rates in the existing gradient descent).

iv) Local minima problem - In non convex surface we have multiple minimas but our solution is the global minima, while at the time of convergence mostly we stuck in local minima.

v) <u>Saddle Point</u> - It is a point on the surface of a function where slopes are 0 in all orthogonal directions but it is neither a local minima ~~or~~ nor a local maxima.

- To remove this problem we can use some other optimizers -
  i) Momentum based optimizer
  ii) Adagrad
  iii) RMS prop
  iv) Adam





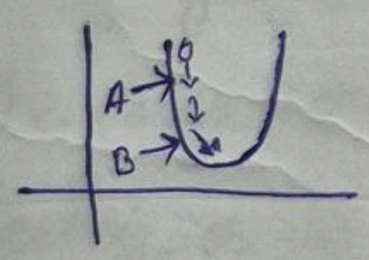
convex

non convex

Contour plot

↓

is a 2D curve where the loss function is a function of single parameter.



Suppose at point A if we set to know some how that my global minima is at lower direction.
Then at point B, when I got to know confirmly that my global minima is there then I should take long ~~step~~ step means have to increase the learning rate but this is not possible in our all three Gradient Descents.
(Batch, stocastic, mini Batch). based GD.
Therefore, n should be adaptive.

~~Therefore, learning~~

<u>Exponentially weighted moving average (EWMA)</u> -

EWMA is a technique based on which we are capable of finding trends in a time series based data. It is basically used in time series forecasting, financial forecasting, signal processing and in deep learning to ~~cool op~~ build cool optimizers.

- By calulating ~~expont~~ exponentially weighted moving average we are giving more importance to the current data than the previous data.
- So, we have to focus on 2 points in EWMA, ~~the first point the is the~~
  i) In due time the weightage of the data will reduce,
  ii) It is a technique to find patterns or hidden trends in time series data.

$$\boxed{V_t = \beta V_{t-1} + (1-\beta)\theta_t}$$

where, θ_t = current value (loss of gradient)

V_t = EMWA at time t.

V_{t-1} = EMWA at time t-1.

β = constant (0-1) $0 < \beta < 1$ (smoothing parameter). 0.99

Eg:

	temp
D1	25
D2	13
D3	17
D4	23
D5	5

$V_0 = 0$ ✓

$V_1 = 0.99 \times 0 + 0.1 \times 25 = 25$

$V_2 = 0.99 \times 25 + 0.1 \times 13$

$V_3 = 0.99 \times 13 + 0.1 \times 17$

$\beta = 0.99$

$V_t = \beta V_{t-1} + (1-\beta)\theta_t$

$V_0 = 0$

$V_1 = \beta \cancel{V_0}^{0} + (1-\beta)\theta_1 = (1-\beta)\theta_1$

$V_2 = \beta V_1 + (1-\beta)\theta_2 = \beta(1-\beta)\theta_1) + (1-\beta)\theta_2$

$V_3 = \beta V_2 + (1-\beta)\theta_3 = \beta[\beta(1-\beta)\theta_1) + (1-\beta)\theta_2)] + (1-\beta)\theta_3$

$= \beta^2(1-\beta)\theta_1 + \beta(1-\beta)\theta_2 + (1-\beta)\theta_3$

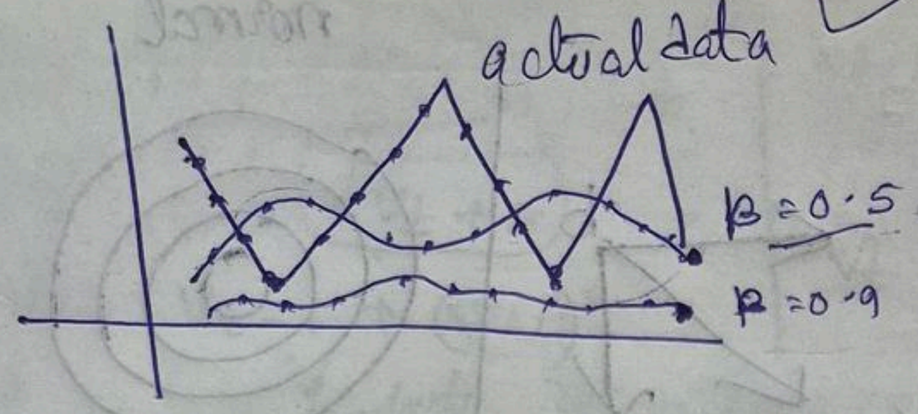
$V_4 = \beta V_3 + (1-\beta)\theta_4$

$= \beta[\beta^2(1-\beta)\theta_1 + \beta(1-\beta)\theta_2 + (1-\beta)\theta_3] + (1-\beta)\theta_4$

$= \beta^3(1-\beta)\theta_1 + \beta^2(1-\beta)\theta_2 + \beta(1-\beta)\theta_3 + (1-\beta)\theta_4$

$= (1-\beta)[\beta^3\theta_1 + \beta^2\theta_2 + \beta\theta_3 + \theta_4]$

$(0.99)^3 \quad (0.99)^2 \quad 0.99$



} Graph becoming more smooth.

<u>Momentum based optimization</u> – solves the following non convex optimization problems such as high curvature, constant gradients (saddle point) & noisy gradient (where there is local minima & we can't overcome the local minima).

- Momentum based optimization can navigate all these 3 problems

$$\boxed{W_{t+1} = W_t + V_t}, \quad V_t = \boxed{\beta V_{t-1}} - \eta \nabla w_t \rightarrow \frac{\partial L}{\partial w_t}$$

$$\boxed{W_{t+1} = W_t - \eta \nabla w_t}$$

momentum

* To speed up the process and optimization ~~prob~~

<u>Adagrad optimization</u>

<u>Adaptive gradient optimization</u> –

- In this optimization technique we don't fix the learning rate rather we change it or we adopt the learning rate acc. to the situation.
- The cases were <u>adagrad works</u> well in comparision to other optimization techniques

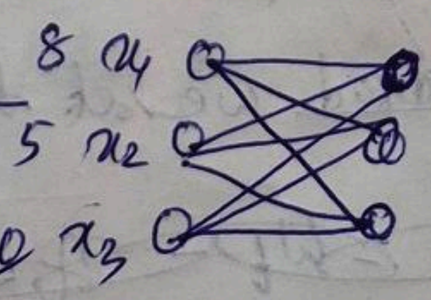
<u>Case I : The ip feature scales are different.</u>

X_1	X_2	$Y = lo$
CGPA	Package	
1-10	1K-L	

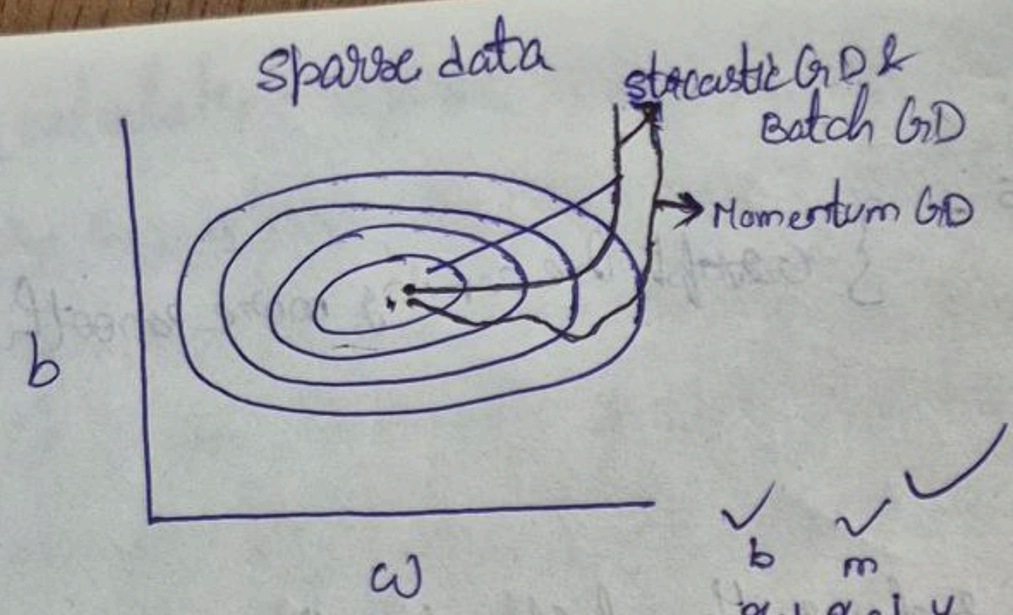
normalization

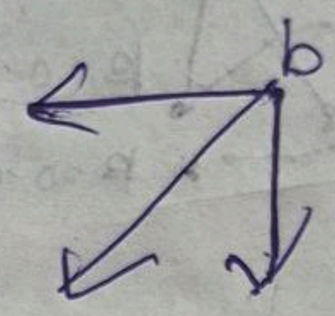
<u>Case II : If the data is sparse then we can use adagrad optimization b/c the momentum and SGD can't sparse the data</u>

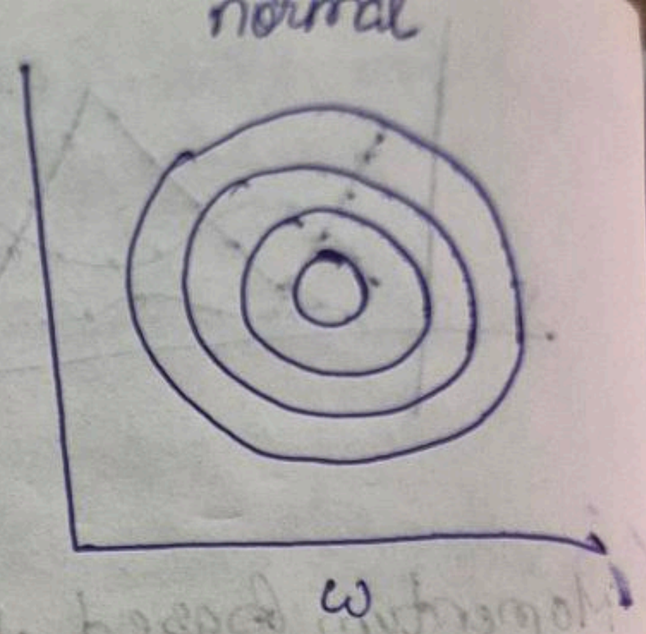
cgpa	iq	IIT	package
–	–	1	
		0	



The question is if there is sparsity ~~wh~~ in data what will happen.



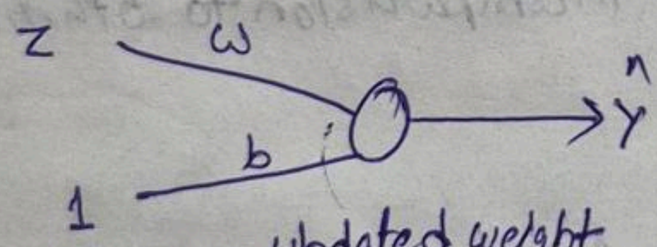




b	m	
x_1	x_2	y
1	0	
1	0	
1	0	
1	0	→ sparse column

taking b side (as it's non zero)

- In elongated ~~bott~~ ball problem the slope changes in one axis but is fixed in one axis, but is fixed in another axis. If the data is sparse the graph plotted for loss w.r.t parameters will be elongated otherwise it is circular.
- In elongated graph the slope changes in 1 axis and ~~fixed~~ fixed in ~~another axis~~ other axis. In this case our normal optimization algo doesn't solve well, to solve this problem the <u>adagraph algo</u> is used.



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

<u>updated weight</u>

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}} \qquad \frac{\partial L}{\partial w_{t-1}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_{t-1}} = -2(y-\hat{y})$$

$$b_t = b_{t-1} - \eta \frac{\partial L}{\partial b_{t-1}} \qquad \frac{\partial L}{\partial b_{t-1}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_{t-1}} = -2(y-\hat{y})$$

- To overcome this disadvantage <u>adagrad</u> is being introduced.
- In <u>adagrad</u> we set different learning rates for different parameters.

$$\cancel{w_{t+1} = w_t - \eta \frac{\partial w_t}{}}$$

- $\cancel{w_{t+1} = w_t - \eta \, \nabla w_t}$ $\quad \cancel{v_t = v_{t-1} + (\nabla w_t)^2}$

- $w_{t+1} = w_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \nabla w_t$ → learning rate factor

$V_t = V_{t-1} + (\nabla w_t)^2$

V_t → current velocity, V_{t-1} → old velocity, $(\nabla w_t)^2$ → (Gradient)^2

$b_{t+1} = b_t - \frac{\eta}{\sqrt{V_t + \epsilon}}$ → it will become small → if it's very large

$V_t = V_{t-1} + (\nabla b_t)^2$ → as it's not square its value will increase so, V_t will also increase.

- We generally don't use adagrad in NN, we can use adagrad in linear regression but not for complex neural network.
- Adagrad able to rich near to the solution but doesn't converge to the solution the reason is that we are decreasing the learning rate.

for, w → it is increasing the learning rate
b → it is decreasing the learning rate

- Learning rate is divided by $\sqrt{V_t + \epsilon}$ and where epochs increases the learning rate becomes smaller due to which the updates become small and when no updates happens it stops at a point and can't reach the minima or we can't reach to the global minima.

8/11/25

RMS Prop (Root mean square propagation) – this optimization technique is an improvement over ~~adag~~ adagrad.

Obj – In RMS block is to stop V_t to become very large to overcome the limitation of adagrad.

$$V_t = \beta V_{t-1} + (1-\beta)(\nabla w_t)^2$$

$$\boxed{w_{t+1} = w_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \nabla w_t}$$ → weight updation formula

where, $\beta = 0.95$

$$\boxed{b_{t+1} = b_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \nabla b_t}$$ → Bias updation formula

Eg: $V_1 = 0.95 \times 0 + 0.05(\nabla w_t)^2 = 0.05(\nabla w_t)^2$

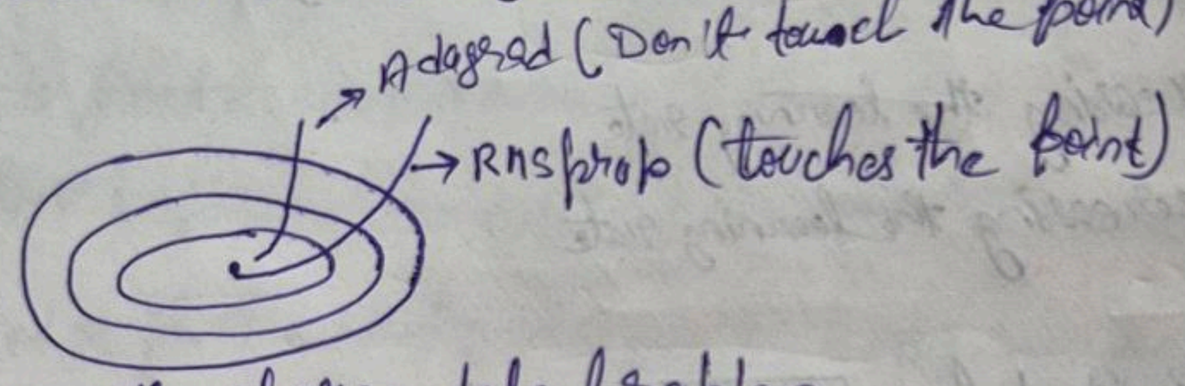
$V_2 = 0.95\left[0.05(\nabla w_1)^2\right] + 0.05(\nabla w_2)^2$

$V_3 = 0.95\left[0.95 \times 0.05 \times (\nabla w_1)^2 + 0.05 \times (\nabla w_2)^2\right] + 0.05(\nabla w_3)^2$

$= \underline{0.95 \times 0.95 \times} 0.05(\nabla w_1)^2 + \underline{0.95 \times} 0.05(\nabla w_2)^2 + \underline{0.05} \times (\nabla w_3)^2$

~~Adam optimization~~

RMS prop doesn't have any limitation & we can have a smooth ~~converge~~ convergence w.r.t to adagrad.



* To overcome the sparse data problem.

Adam Optimization-

The full form is Adaptive Moment Estimation. This is the most powerful optimizer.

- It borrows the concept from the previous optimizers like adagrad, momentum based and RMS prop.

- In adam optimization mathematical formulation is-

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{V_t + \epsilon}} \times m_t, \quad \text{where } \boxed{m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla w_t}$$

momentum based

$$\boxed{V_t = \beta_2 V_{t-1} + (1-\beta_2)(\nabla w_t)^2}$$

~~adagrad~~ RMS prop

bias correction

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{V}_t = \frac{V_t}{1-\beta_2^t}$$

no. of epoch

$\beta_1 = 0.9$

$\beta_2 = 0.99$

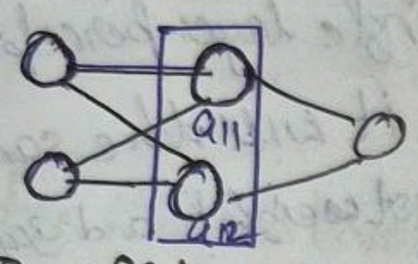
time stance

at what instance it is taking the input

Q) Difference b/w adagrad & RMS prop.

* Gradient descent is basically used to overcome the convex problem.

Q1) ~~If~~ Why we should go for random initialization of weight and biases? What will happen if the weight and biases are 0 at initialization?



$\boxed{w=0,\ b=0}$

① Apply ReLU in the hidden layer.

② Apply sigmoid in the hidden layer.

③ Apply tanh in the hidden layer.

Sol 1: 1) $ReLU = max(0, z)$

$a_{11} = m(0, z_{11}) = 0$

$a_{12} = m(0, z_{12}) = 0$

$z_{11} = x_1 w^1_{11} + x_2 w^1_{21} + b_{11} = 0$

$z_{12} = x_1 w^1_{12} + x_2 + w^1_{22} + b_{12} = 0$

$a_{11} = a_{12},\ z_{11} = z_{12}$

$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \cancel{\frac{\partial \hat{y}}{\partial a_{11}}}^{\,0} \cdot \frac{\partial z_{11}}{\partial w^1_{11}} = 0 \quad \text{(No updation)}$$

$\frac{e^{-z}}{1+e^{-z}}$

$\frac{1}{1+1} = \frac{1}{2}$

2) $w=0,\ b=0$

$a_{11} = \sigma(z_{11}) = 0.5$

$a_{12} = \sigma(z_{12}) = 0.5$

~~$z^1_{11} x_1 + w^1_{21} x_2 + b_{11}$~~

$z_{11} = w^1_{11} x_1 + w^1_{21} x_2 + b_{11} = 0$

$z_{12} = 0$

$\therefore z_{11} = z_{12},\ a_{11} = a_{12}$

$$\cancel{\frac{\partial L}{\partial w^1_{11}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{11}} \frac{\partial z_{11}}{\partial w^1_{11}}}$$

$$\cancel{\frac{\partial L}{\partial w^1_{21}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{11}} \frac{\partial a_{11}}{\partial z_{12}} \frac{\partial z_{12}}{\partial w^1_{12}}}$$

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{11}^1} \quad x_1$$

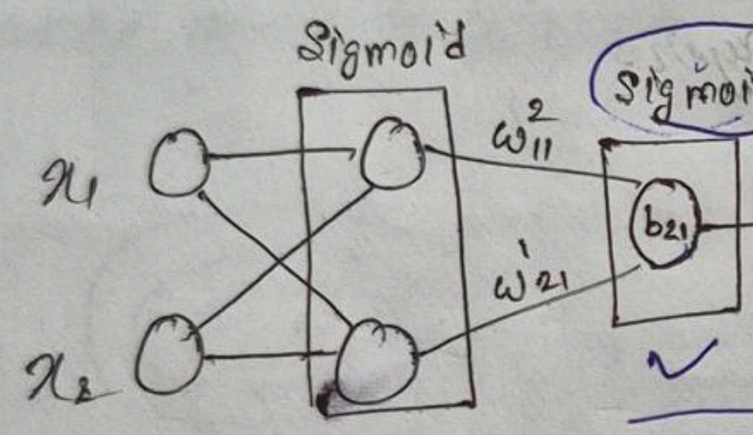
$$\frac{\partial L}{\partial w_{21}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{11}} \cdot \frac{\partial a_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{21}^1} \quad x_1$$

$$\frac{\partial L}{\partial w_{12}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{12}} \cdot \frac{\partial a_{12}}{\partial z_{12}} \cdot \frac{\partial z_{12}}{\partial w_{12}^1} \quad x_2$$

$$\frac{\partial L}{\partial w_{22}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a_{12}} \cdot \frac{\partial a_{12}}{\partial z_{12}} \cdot \frac{\partial z_{12}}{\partial w_{22}^1} \quad x_2$$

$\therefore$ It will behave as a single layer perceptron. b/o it will take same const weight and same bias. then the same problem will occur as previous.

10/11



Sigmoid $\rightarrow$ always for Binary classification. $0-1$

L = binary cross entropy

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

$$\frac{\partial}{\partial \hat{y}}\left[-y\log(\hat{y}) - (1-y)\log(1-\hat{y})\right] = \frac{-y}{\hat{y}} - \frac{(1-y)}{(1-\hat{y})}(-1)$$

$$= \frac{-y}{\hat{y}} + \left(\frac{1-y}{1-\hat{y}}\right) = \frac{-y(1-\hat{y}) + \hat{y}(1-y)}{\hat{y}(1-\hat{y})}$$

$$= \frac{-y + \cancel{y\hat{y}} + \hat{y} - \cancel{y\hat{y}}}{\hat{y}(1-\hat{y})} = \boxed{\frac{(\hat{y}-y)}{\hat{y}(1-\hat{y})}}$$

$$= \boxed{(\hat{y}-y)\,o_{11}}$$

$$w_{11}^2\,new = w_{11}^2\,old - \eta\frac{\partial L}{\partial w_{11}^2}, \quad \frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_f} \times \frac{\partial z_f}{\partial w_{11}^2} = \frac{\hat{y}-y}{\cancel{\hat{y}(1-\hat{y})}} \times \cancel{\hat{y}(1-\hat{y})}\cdot o_{11}$$

$$w_{21}^2\,new = w_{21}^2\,old - \eta\frac{\partial L}{\partial w_{21}^2}, \quad \frac{\partial L}{\partial w_{21}^2} = \boxed{(\hat{y}-y)\,o_{12}}$$

$$b_{21}\,new = b_{21}\,old - \eta\frac{\partial L}{\partial b_{21}}, \quad \frac{\partial L}{\partial b_{21}} = \boxed{(\hat{y}-y)}$$

hidden to input layer

$$w'_{11}\,new = w'_{11}\,old - \eta \frac{\partial L}{\partial w'_{11}}$$

$$\frac{\partial L}{\partial w'_{11}} = \underbrace{\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f}}_{(\hat{y}-y)} \cdot \underbrace{\frac{\partial z_f}{\partial o_{11}}}_{w_{11}^2} \cdot \underbrace{\frac{\partial o_{11}}{\partial z_{11}}}_{o_{11}(1-o_{11})} \cdot \frac{\partial z_{11}}{\partial w'_{11}} \rightarrow x_i$$

Q1) What is the motivation behind introducing momentum ~~bas~~ in GD? Explain using the notion using the optimization landscape.

Q2) Explain the hyperparameters used in momentum based optimization. Why is momentum coefficient, β usually choosen close to 1

Q3) Compare ~~batch~~ batch GD v/s GD with momentum.

Q4) What happens when momentum coefficient is set to

a) 0 , b) 0.5

Q5) Explain the role of exponential moving average in momentum based ~~optimization~~ optimizers.

Q6) What is the main motivation behind introducing adagrad. Explain what problems it was designed to solve?

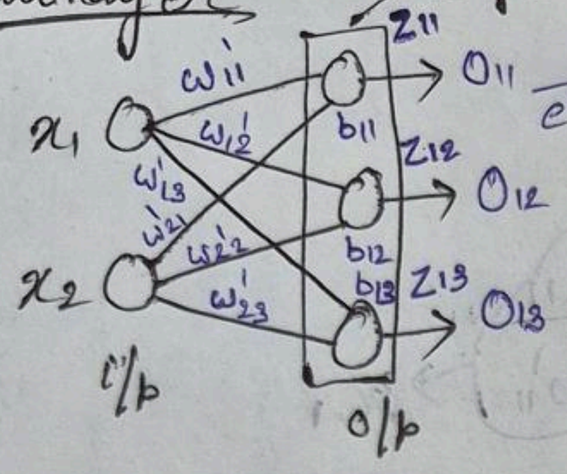
Q7) Write parameter update rule for adagrad and define each term clearly.

Q8) Why does adagrad has different learning rate for each parameters.

Q9) What is the motivation behind the RMS prop and how does it improve upper adagrad.

Q10) Write the RMS prop update eqn and explain the purpose of decay rate.

Multilayer



→ softmax activation function

$$\frac{e^{z_{11}}}{e^{z_{11}}+e^{z_{12}}+e^{z_{13}}} = \text{categorical cross entropy}$$

$$\frac{e^{z_{12}}}{e^{z_{11}}+e^{z_{12}}+e^{z_{13}}} =$$

$$\frac{e^{z_{13}}}{e^{z_{11}}+e^{z_{12}}+e^{z_{13}}}$$

$$\underset{W}{\begin{bmatrix} w'_{11} & w'_{12} & w'_{13} \\ w'_{21} & w'_{22} & w'_{23} \end{bmatrix}} \underset{X}{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}} + \underset{b}{\begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}}$$

$$W^T X + b$$

$$\begin{bmatrix} w'_{11} & w'_{21} \\ w'_{12} & w'_{22} \\ w'_{13} & w'_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_{11} \\ b_{12} \\ b_{13} \end{bmatrix}$$

$$= \begin{bmatrix} w'_{11} x_1 + w'_{21} x_2 + b_{11} \\ w'_{12} x_1 + w'_{22} x_2 + b_{12} \\ w'_{13} x_1 + w'_{23} x_2 + b_{13} \end{bmatrix} = \begin{bmatrix} z_{11} \\ z_{12} \\ z_{13} \end{bmatrix}$$

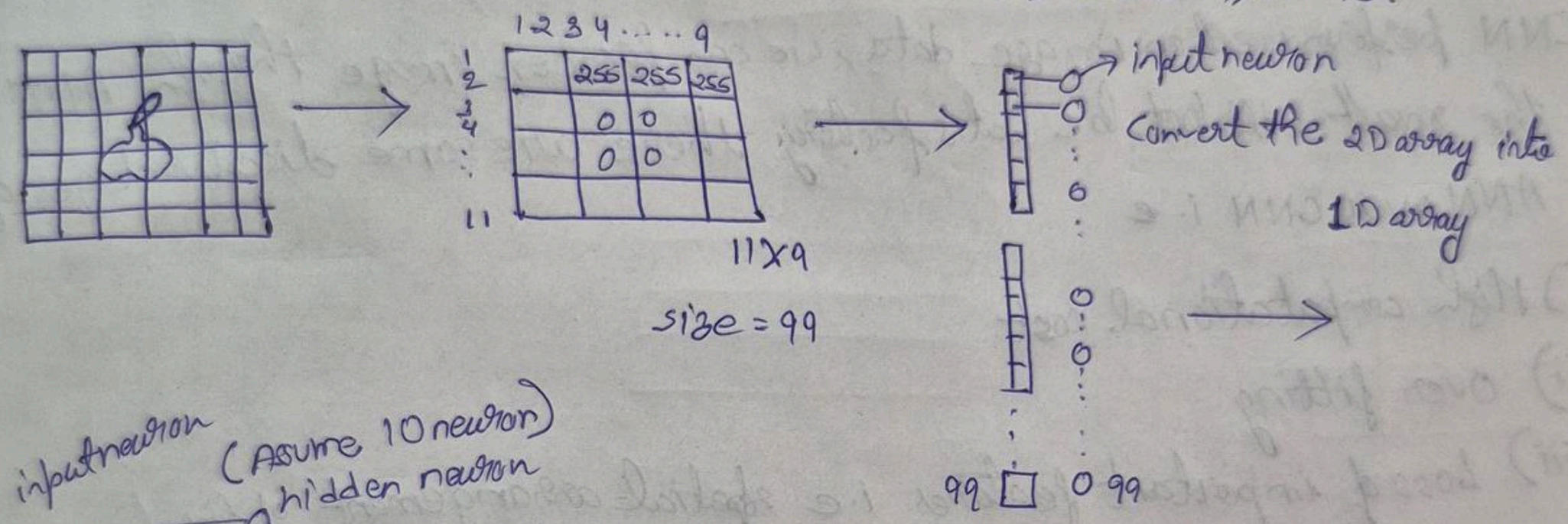
24/11

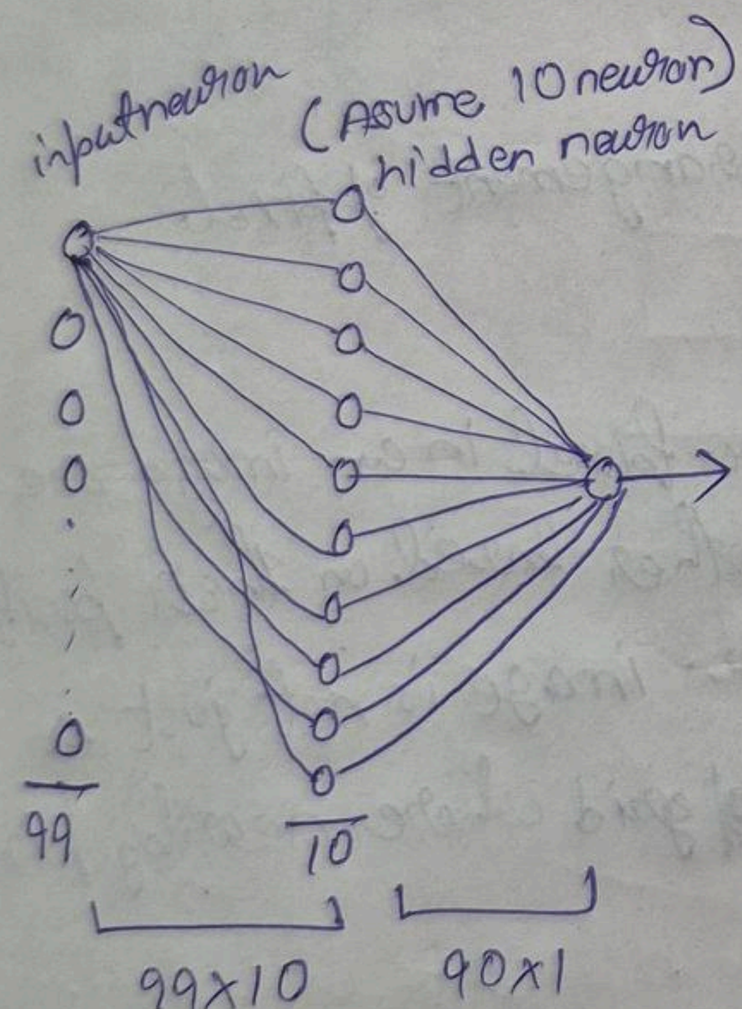
Convolutional Neural Network

Q) Why CNN?

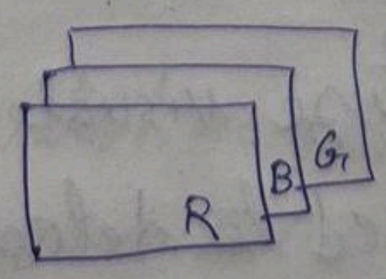
Q) What is the limitations of ANN?

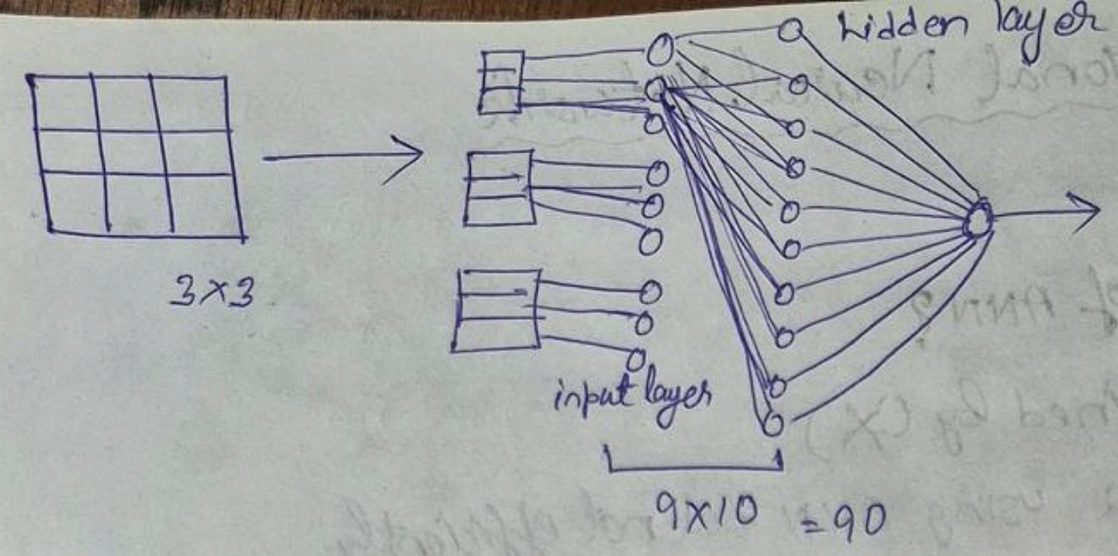
- Convolutional operation defined by (X).
- We can handle image data using ANN but not efficiently.
- Image data of 2 types → RGB (coloured image)
  → Grey scale image (Black & white image)
- Image is a collection of pixels.
- Range for Grey scale (0-255) → 0 → Black, 255 → White.





- Grey scale - one channel
- RGB - 3 channels





- If an image having 64x64 size than 4096 is the input layer and h_1 layer will have 1000 neuron & $h_2 = 500$. $\boxed{h_1 \times h_2 = 5,00,000}$

Q) Why CNN is used to process image data?

→ CNN perform well on image data, we can process image through ANN but the result will not be satisfactory there are some disadvantages of ANN over CNN i.e

i) High computational cost
ii) Over fitting
iii) Loss of important features i.e spatial arrangement of pixel.

Spatial -

spatial relationship among pixel describe how pixel in an image are connected, arranged and related to each other based on their position. these relationship are crucial because an image is not just a collection of pixel value it is a structure grid where nearby pixel often share meaningful information.

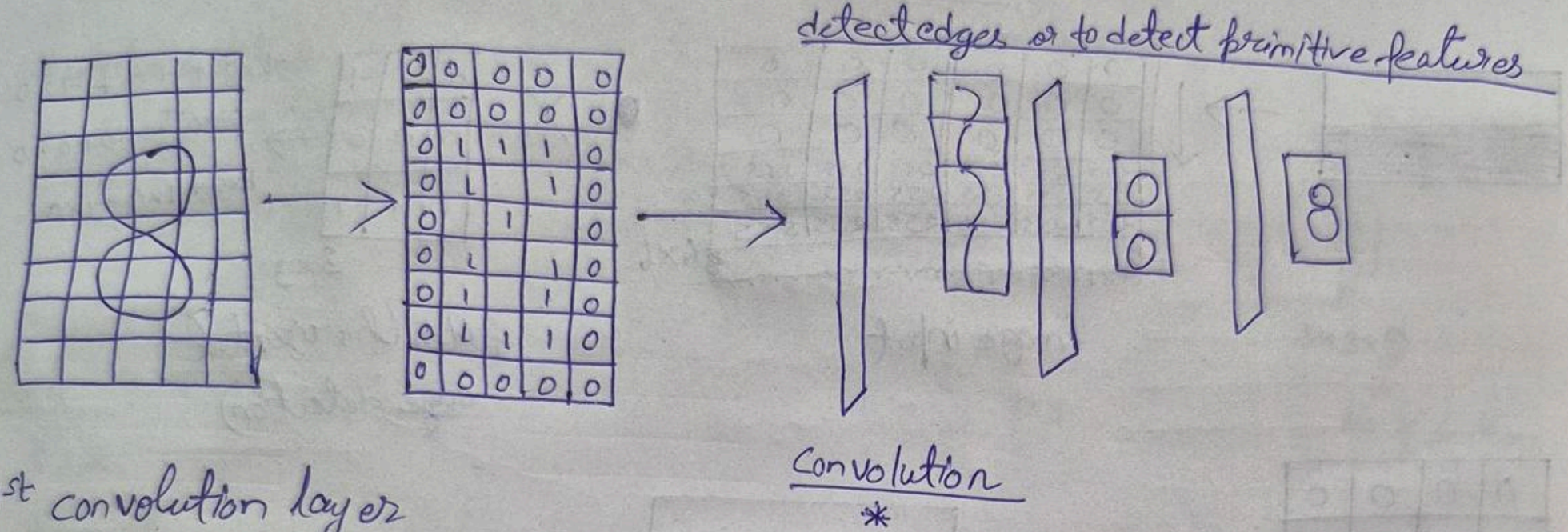
Q) What is CNN?

Sol: A convolution Neural Network is a feed forward neural network that is generally used to analyze visual images by processing data with grid like topology. CNN used to detect and classify object in an image.

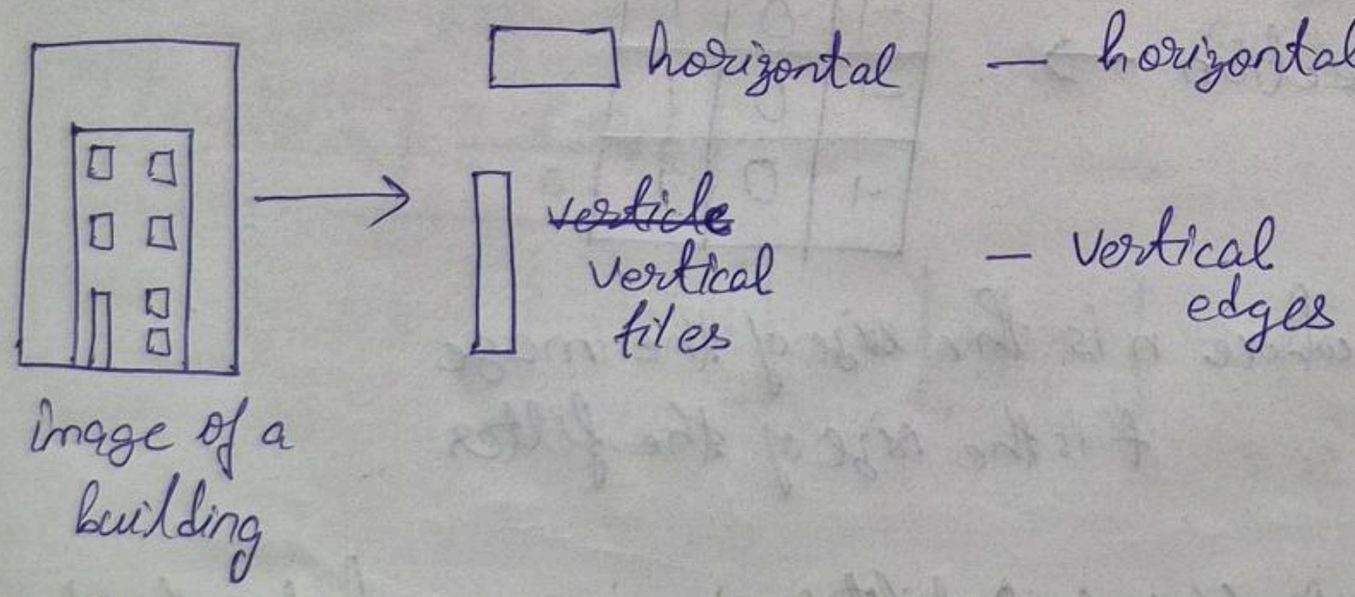
Applications of CNN -

- CNN is used for image classification.

- object ~~tokenization~~, localization.
- object detection
- face detection and recognition
- image segmentation
- improve resolution
- converting black & white image to a coloured image
- Pose estimation.



1st convolution layer
2nd ReLU layer
3rd Pooling layer
4th Fully connected layer



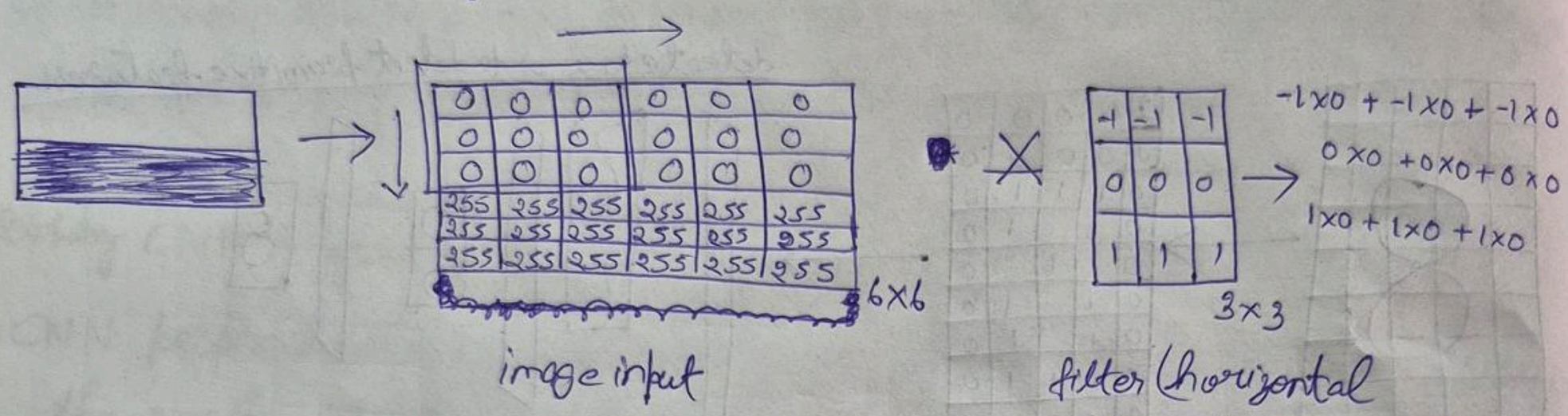
Convolution operation -

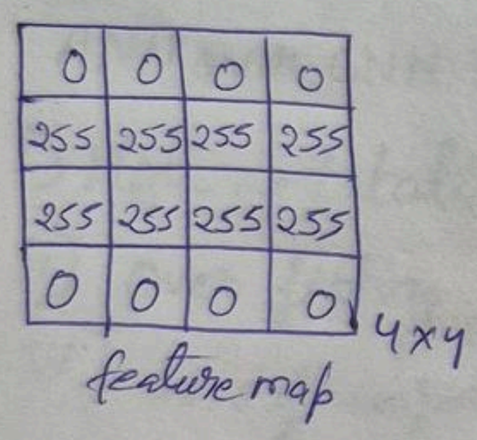
- It's the heart of CNN which is responsible for detecting the edges & features of images.
- In case of convolution operation we need to take a filter grid which is convolved on the pixel grid and the output is known as a feature map.
- The filter is placed on the left top and perform item wise multiplication.
- The filters are initialized with some random values and through back

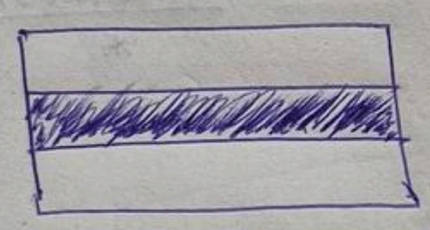
propagation the CNN decides the resultant values of the filters.

- The value of filters behave similar to the weights of ANN.
- CNN creates filters as per its requirements instead of relying on the created filter so, we have to just decide the filter matrix.

* edge - change in intensity (means change in the value of pixels)







For vertical edge detection →

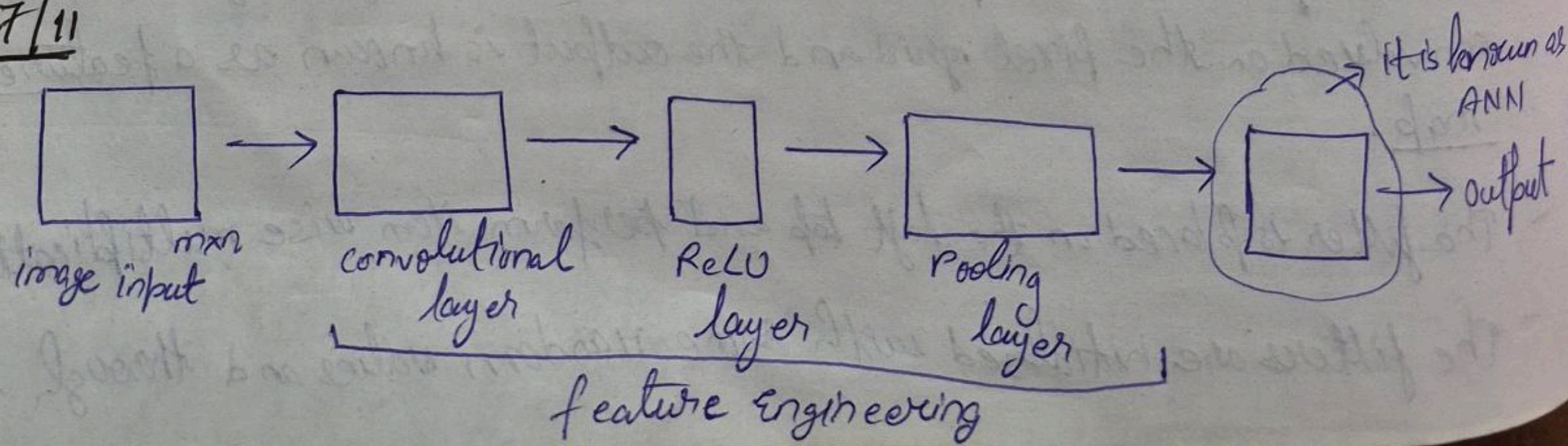
-1	0	1
-1	0	1
-1	0	1

$(n-f+1) \times (n-f+1)$, where n is the size of the image
f is the size of the filter

Q: If the size of image is 64x64 & filter size is 3x3 what is the feature map size.

Sol: $\cancel{(64-f+1) \times (64-f+1)} = 64-3+1 = 62$
$\cancel{= (65-f) \times (65-f) =}$

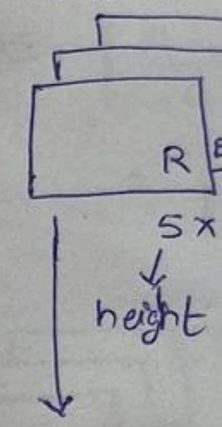
27/11



Convolutional

Perform

Convolution

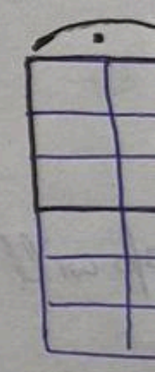


It will be sha

(colourc

Stride

S = 2



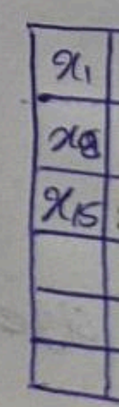
30/11

Filter

orign

Fix

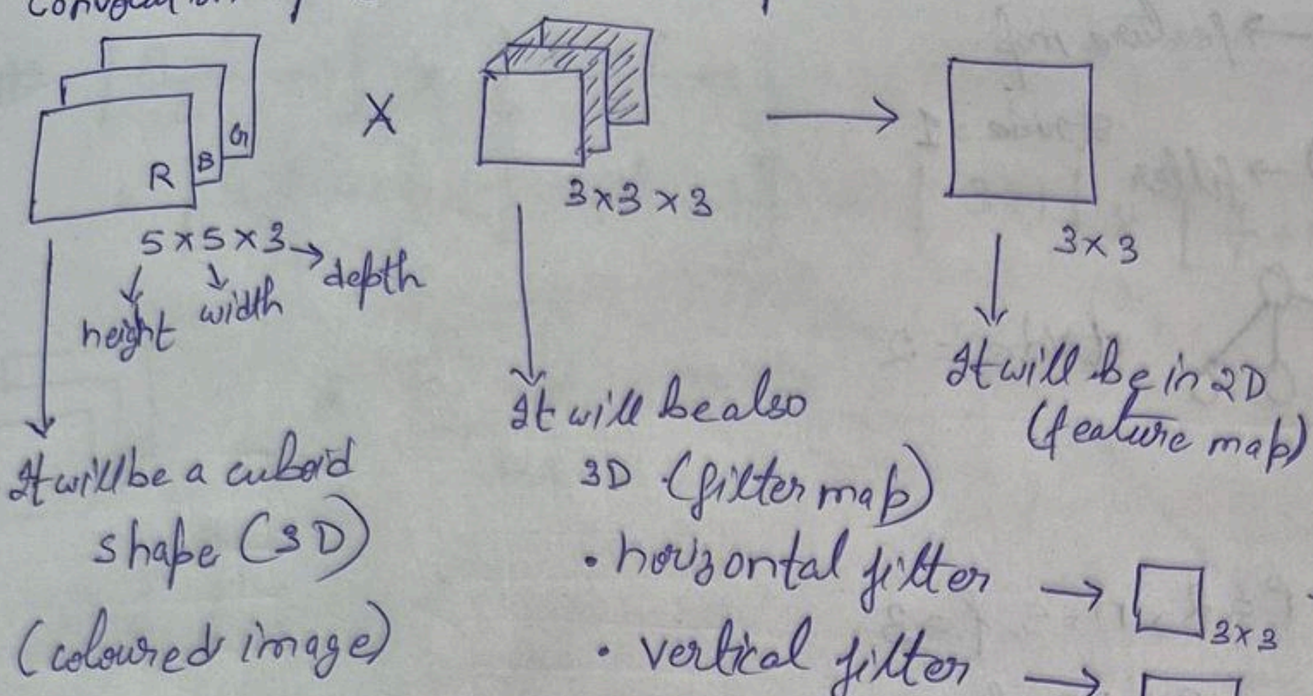
AD



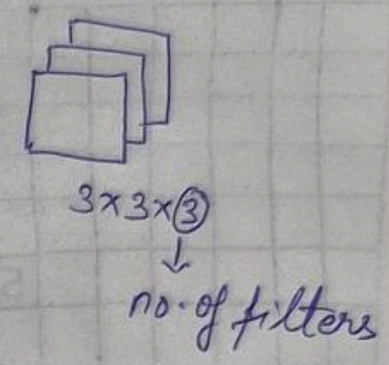
Convolutional layer

Perform convolution operation (X)

Convolution operation is a linear operation

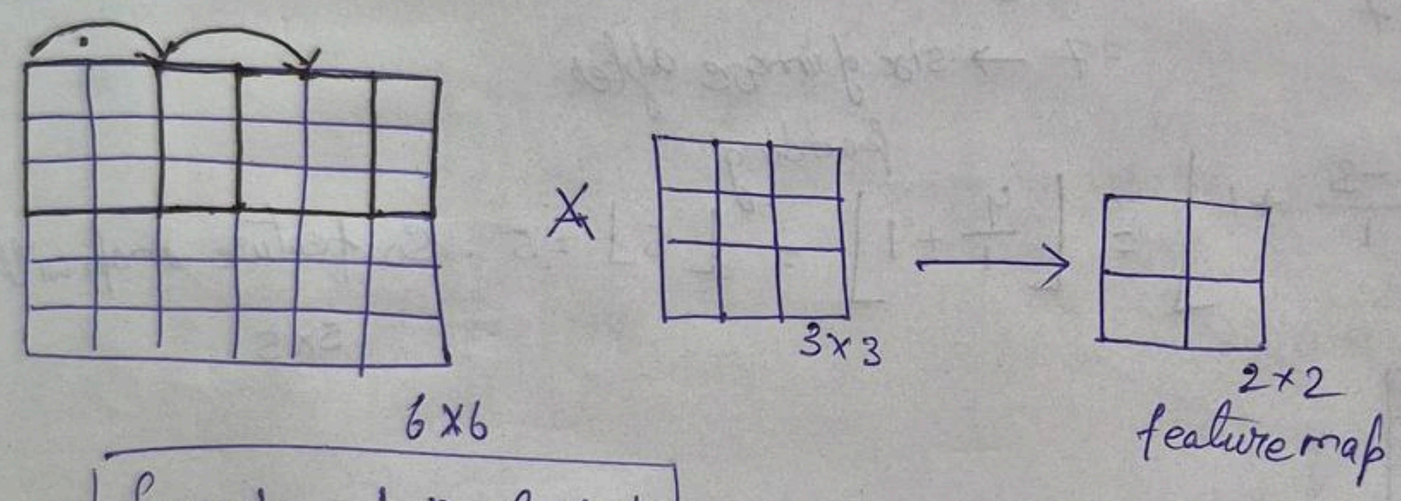


- horizontal filter → 3x3
- vertical filter → 3x3
- slanting filter → 3x3



Stride

S = 2



$$\boxed{\text{formula} = \left\lfloor \frac{n-f}{s}+1 \right\rfloor}$$

$$= \left\lfloor \frac{6-3}{2}+1 \right\rfloor = \left\lfloor \frac{3}{2}+1 \right\rfloor = \lfloor 2.5 \rfloor = 2$$

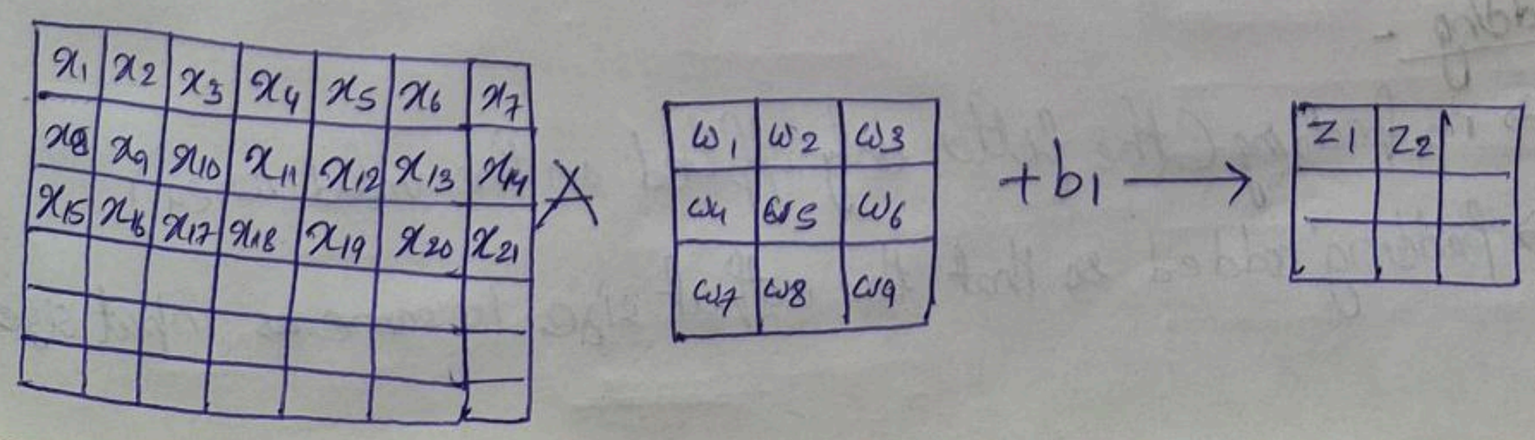
* We are ignoring some feature
* Low values are ignored & high values are not ignored.

30/11

Filter - It is 2D grid consisting of weight parameter which convolve on the original image to extract the relevant features from an image.

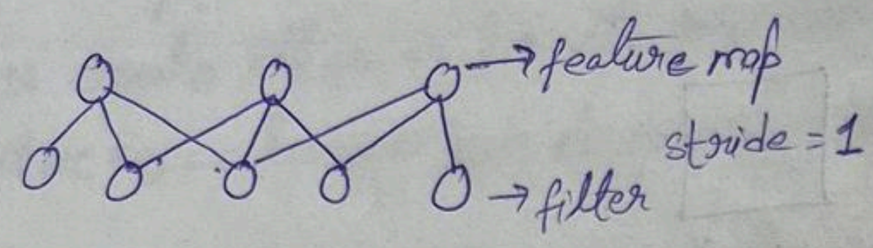
Filter size → (3x3), (5x5), (7x7), (11x11)

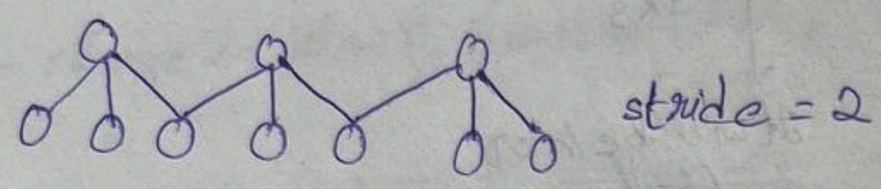
ADD filter size → to get a center pixel so extraction of the feature is efficient than even filter size.



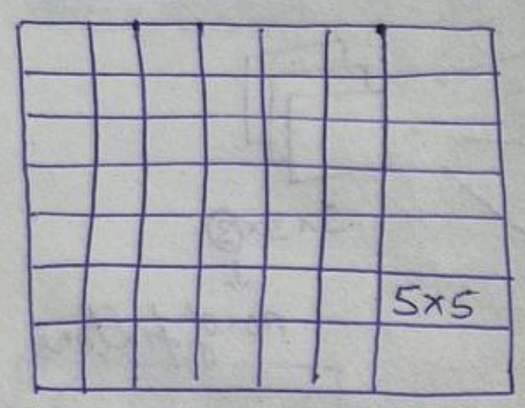
$Z = x_1\omega_1 + x_2\omega_2 + x_3\omega_3 + x_8\omega_4 + x_9\omega_5 + x_{10}\omega_6 + x_{15}\omega_7 + x_{16}\omega_8 + x_{17}\omega_9$

* We use 3x3 when we want to extract minute feature.





Padding -



$P = 2, n = 5, f = 3$

$n + 2p - f + 1$

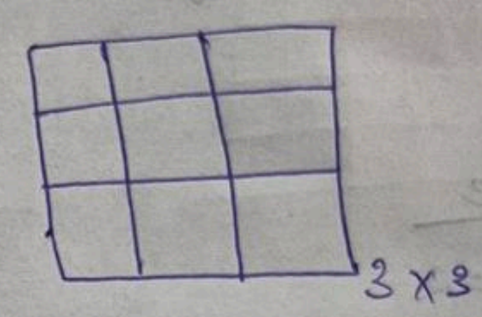
$= 5 + 2\times2 - 3 + 1$

$= 5 + 4 - 3 + 1$

$= 10 - 3$

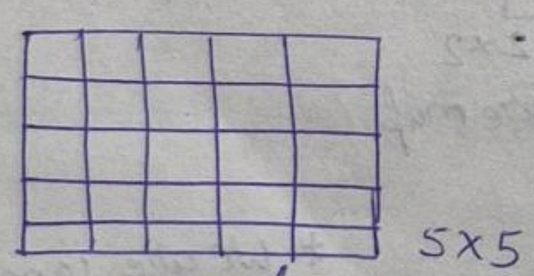
$= 7 \rightarrow$ size of image after padding

X



$\left\lfloor \frac{n-f+1}{s} \right\rfloor = \left\lfloor \frac{7-3}{1} + 1 \right\rfloor = \left\lfloor \frac{4}{1} + 1 \right\rfloor = \lfloor 5 \rfloor = 5$. So feature map will be 5x5

→



feature map

Q) Why padding in needed?

Sol: Padding means extra pixels added around the boarders of input image.

Q) What is use of padding?

Sol: . Padding preserves spatial dimension.

- It preserves shrinking of feature map.
- ensure ~~bor~~ border information is captured.
- Maintain symmetry in the output.

Classification of padding -

① Valid padding → no padding (the filter only applied on the valid area)

② Same padding → padding added so that the output size is same as input size.

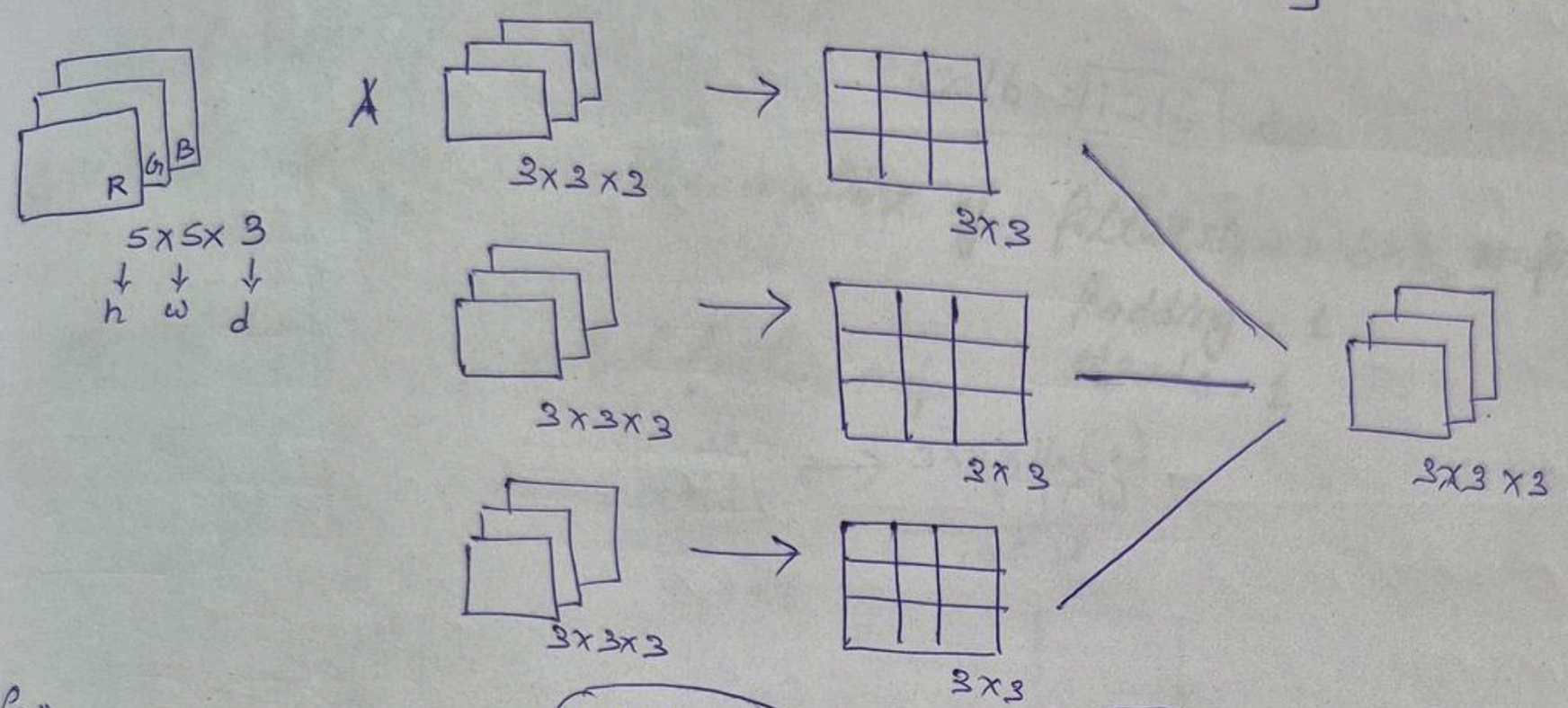
Special case

$n \times m$ filter what is the size of the feature map?

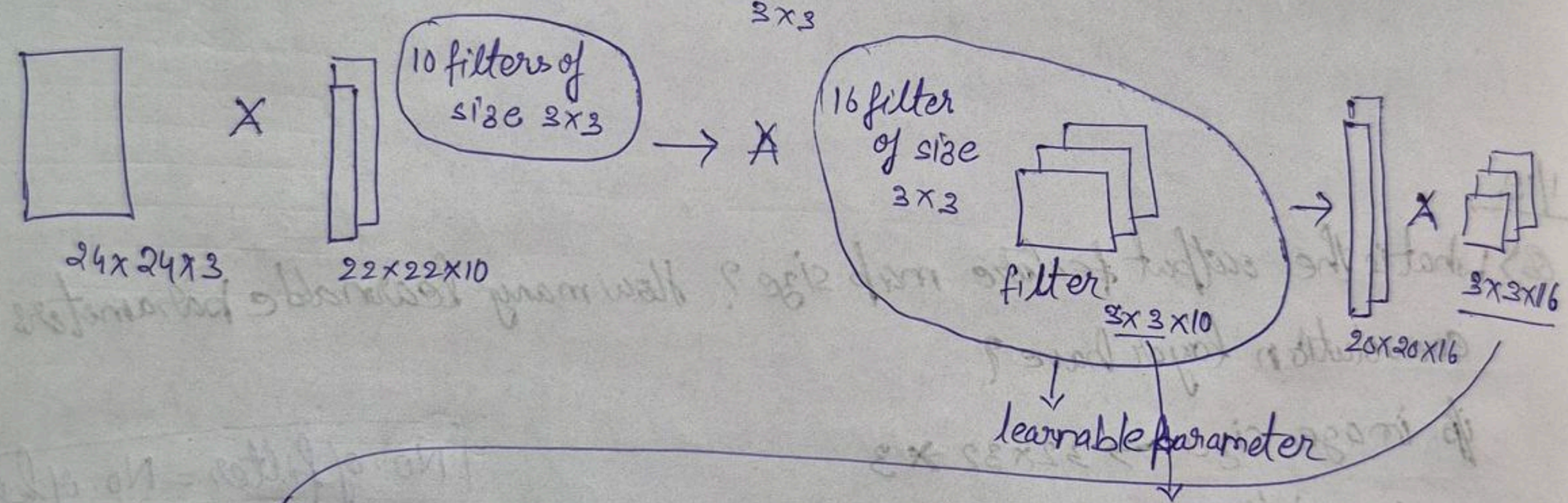
6×7 $f \times f$

Ans $\left\lfloor \frac{n-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{m-f}{s} + 1 \right\rfloor$

$= \left\lfloor \frac{6-3}{1} + 1 \right\rfloor \times \left\lfloor \frac{7-3}{1} + 1 \right\rfloor = \left\lfloor 3+1 \right\rfloor \times \left\lfloor \frac{4}{1} + 1 \right\rfloor = 4 \times 5$



Eg:



learnable parameters $= 3 \times 3 \times 16$

$= [(3 \times 3) + 1] \times 16$

$= (9+1) \times 16 = 10 \times 16 = 160$

$= ((3 \times 3) + 1) \rightarrow$ bias $\times 10$

$= [9+1] \times 10 = 10 \times 10 = 100$

Q) What is the output feature map size? How many learnable parameter does the convolution layer have?

input image size $32 \times 32 \times 3$

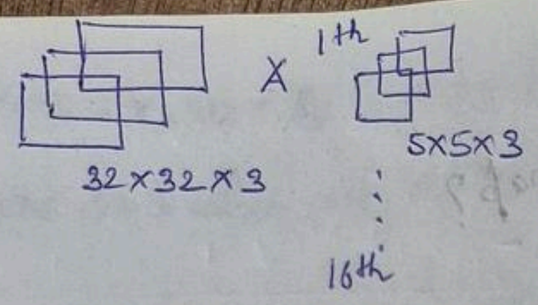
convolution kernal 5×5

padding 0

filter 16

stride 1

Sol:



1/12.

Q) What is the output feature map size? How many learnable parameters does the convolution layer have?

image size → 32x32x3

convolution → kernel 5x5

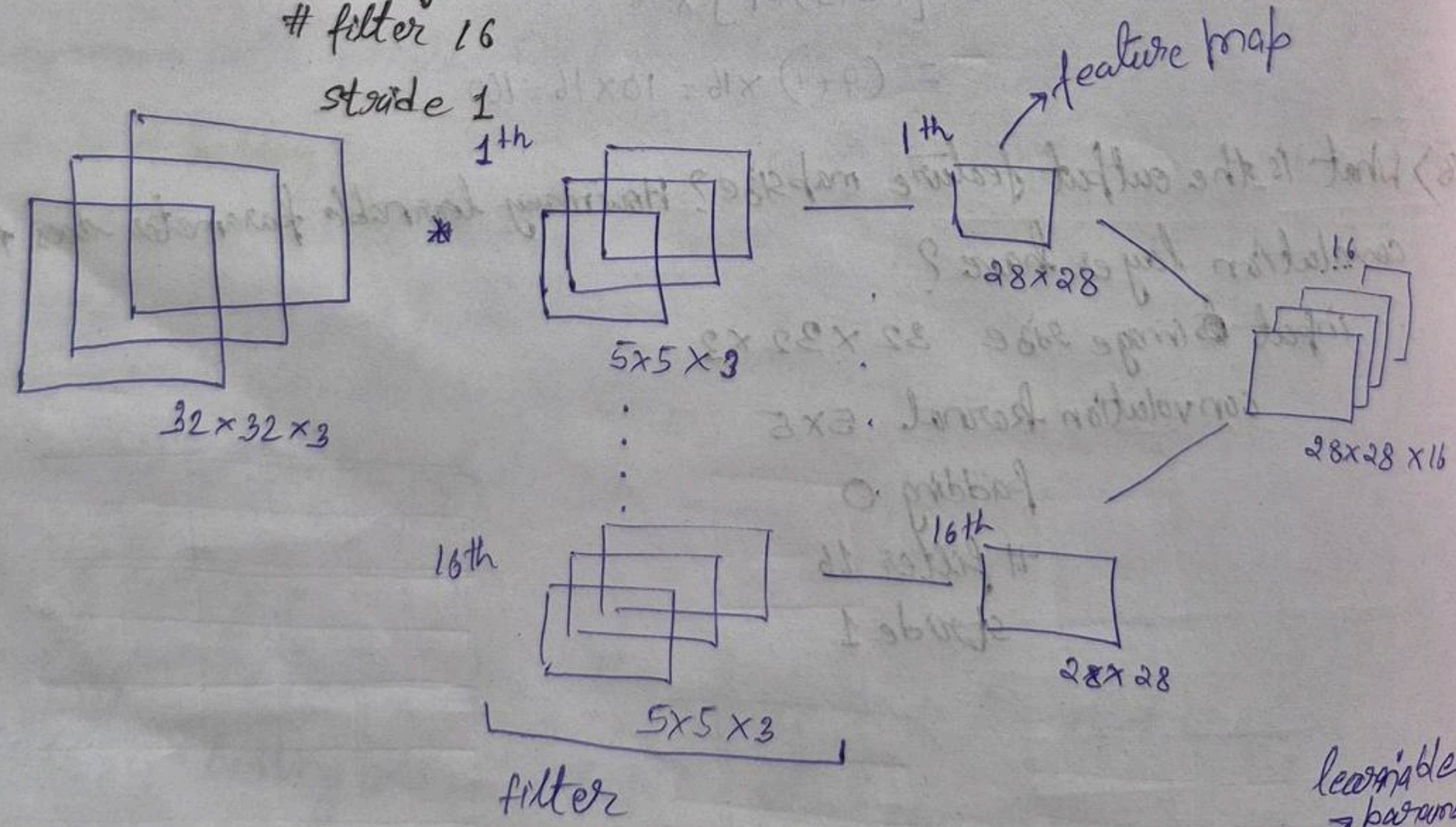
padding 0

filter 16

stride 1

$\boxed{\text{No. of filter} = \text{No. of biases}}$

Sol:



5x5x3 = 75 ~~filters~~ (weight) + 1 (bias) = $\boxed{76}$ → for 1 filter

learnable parameter

$$H_{out} = \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor = \left\lfloor \frac{32 + 2\times 0 - 5}{1} + 1 \right\rfloor = 28$$ $\boxed{n \rightarrow \text{is the size of input image}}$

$$W_{out} = \left\lfloor \frac{n + 2p - f}{s} + 1 \right\rfloor = \left\lfloor \frac{32 + 2\times 0 - 5}{1} + 1 \right\rfloor = 28$$

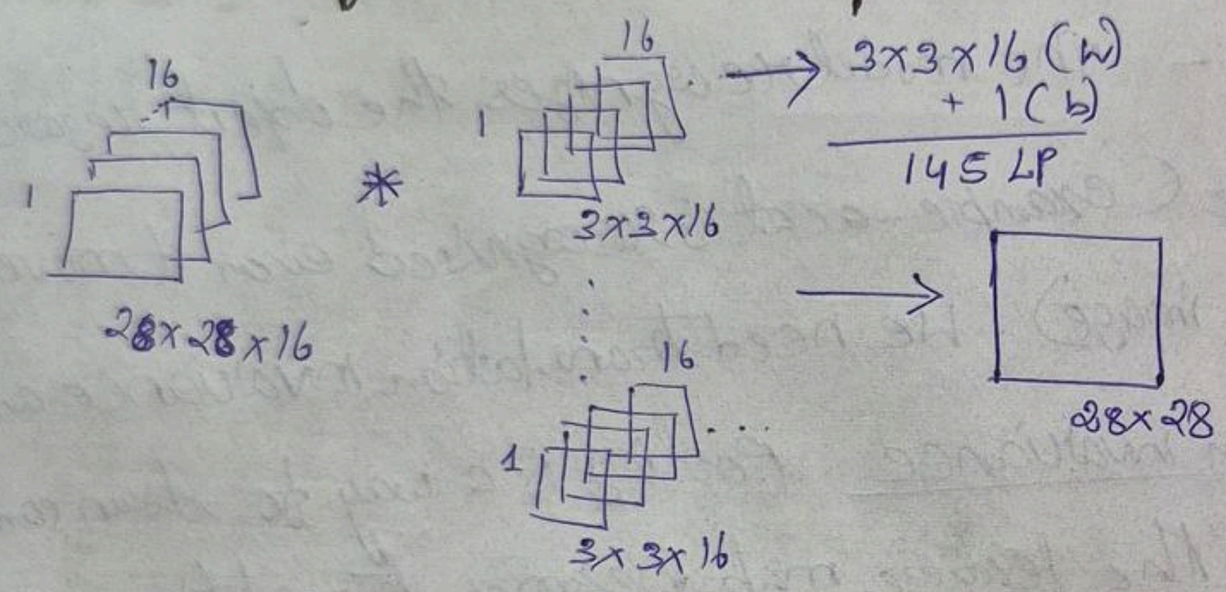
$\therefore$ Size of feature map is $\underline{28 \times 28 \times 16}$

Total learnable parameters $= 76 \times 16 = \boxed{1216}$ Ans

Q) Add another convolution matrix of filter size $= 3\times 3$, # filters $= 10$
padding $= 1$
stride $= 1$

What is the size of output feature map.

Sol:

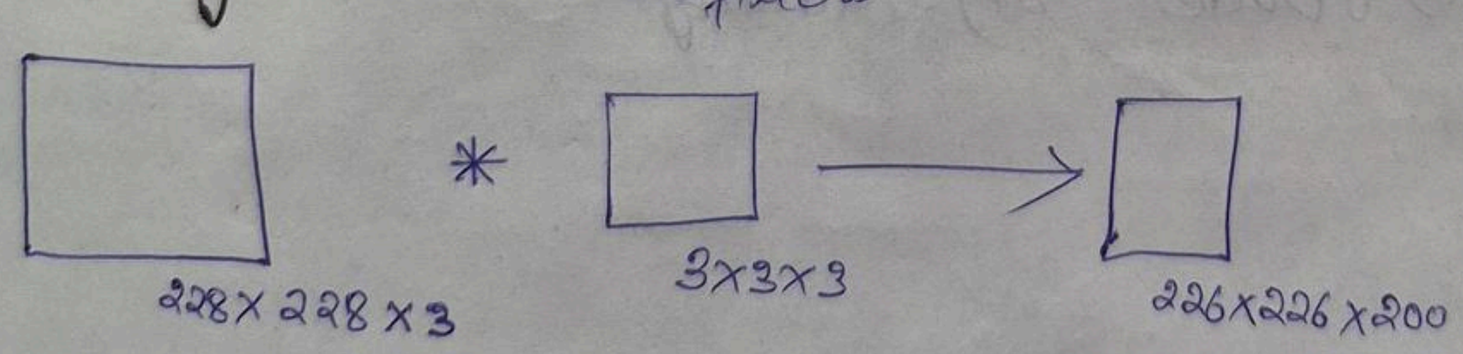


$$H_{out} = \left\lfloor \frac{H_{in} + 2p - f}{s} + 1 \right\rfloor = \left\lfloor \frac{28 + 2 - 3}{1} + 1 \right\rfloor = 28$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2p - f}{s} + 1 \right\rfloor = \left\lfloor \frac{28 + 2 - 3}{1} + 1 \right\rfloor = 28$$

Total learnable parameters (LP) $= 145 \times 10 = \boxed{1450}$ Ans
10 filters

Pooling layer



- The pooling layer in CNN is used to reduce the size of the feature maps to speed up the computation and also makes the feature detection more robust.

- There are 2 problems in convolution operation.
  i) Memory issue ii) Translation varience issue

- To overcome these issues pooling layer is applied.

Translation variance - It describes a situation where a model is sensitive to the position of objects or features in the i/p images in otherwords a model that exibits translation variance would produce different o/p or predictions based on the exact location of object within the image. small shift or translation of the object would significantly affect the model's performance or recognition capacity.

Translation invariance - The model recognizes the object regardless of its position in the image (example - a cat is recognised even it moves to a different part of the image). We need translation invariance and pooling provides translation invariance. Pooling is a way to down sample the feature map by this the feature map becomes translation invariant.

Types of pooling -

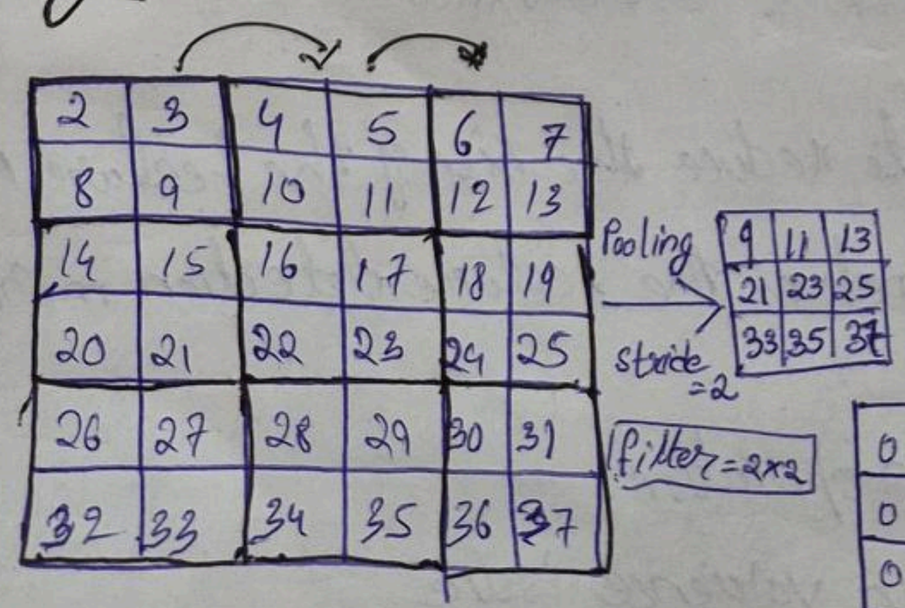
i) Max pooling. ii) Average pooling iii) Min pooling

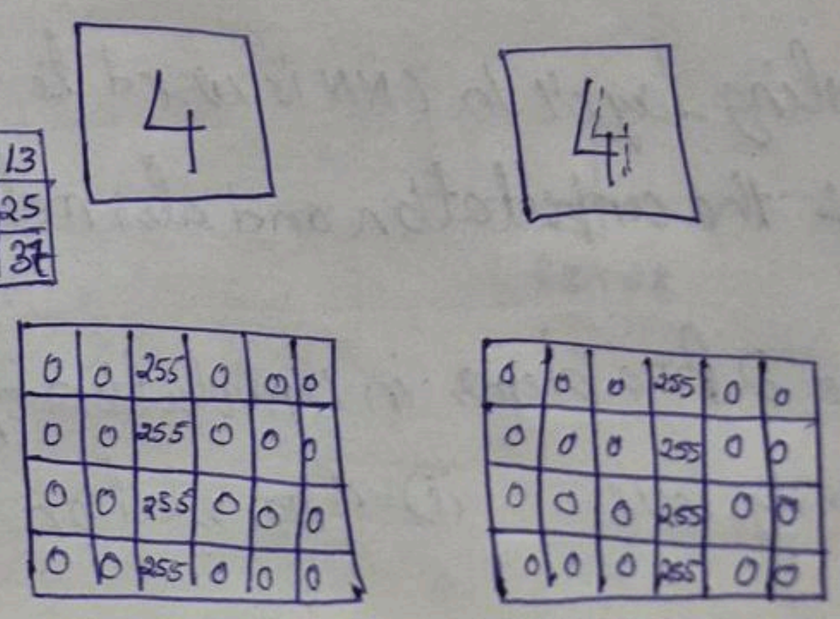
iv) Global pooling

The common hyperparameters of pooling layers -

i) Poolsize ii) Stride iii) Padding

Max pooling





steps of max pooling

i) pick a window size of two

ii) pick a stride of two

iii) move the window across activation maps

iv) from each window pick the maximum value

* Deep lizard pooling demo -

Q1) Compute total trainable parameters for the CNN.

Input = 32×32×3

Conv 1 = 5×5, s = 1, p = 0, f = 6

pool 1 = 2×2, s = 2

conv 2 = 5×5, s = 1, p = 0, f = 16

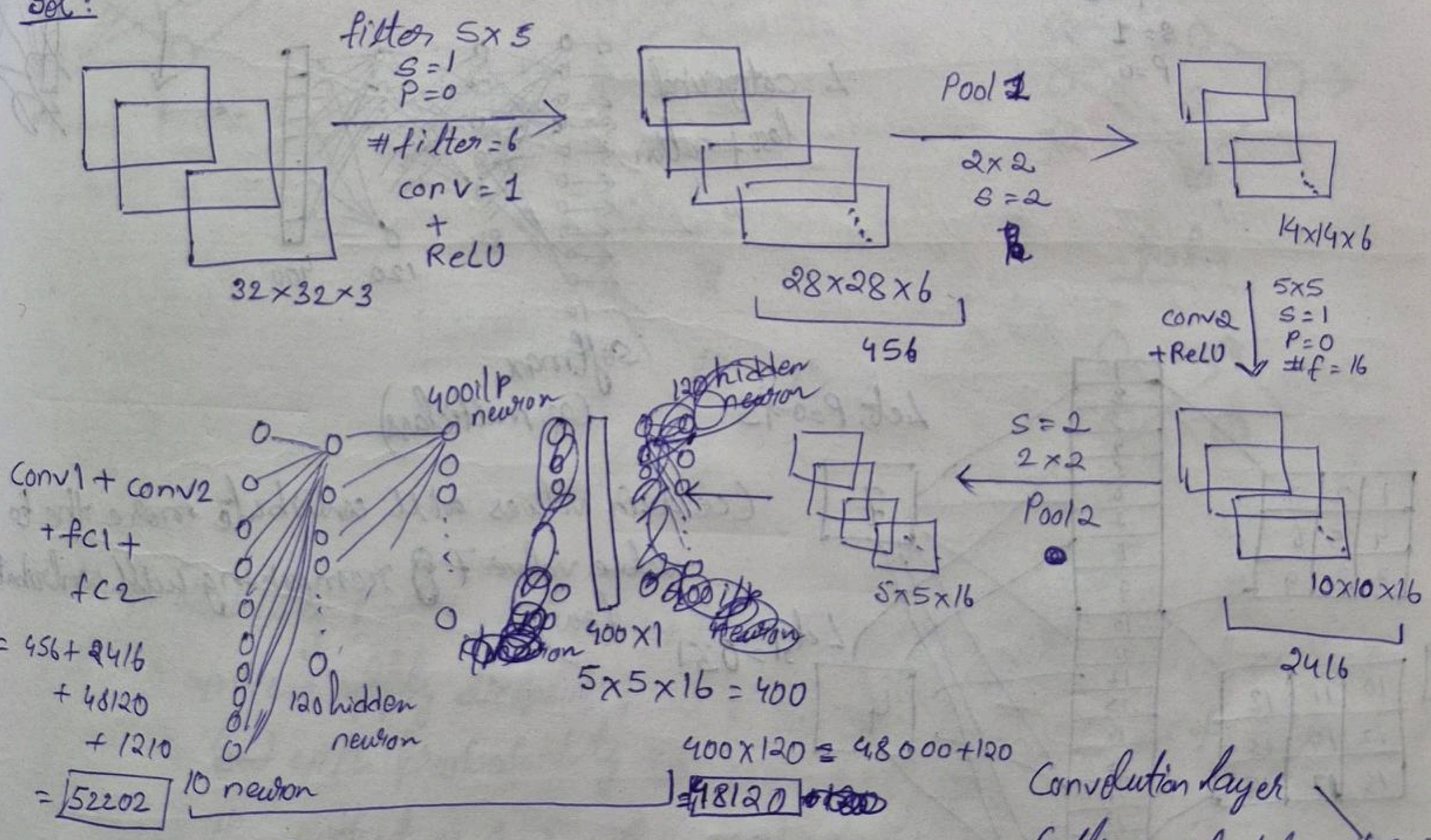
pool 2 = 2×2, s = 2

Fully connected to 120 neurons, the

Fully connected to 10 neurons (softmax)

Include bias for every filter and FC neurons.

Sol:



Conv1 + conv2 + fc1 + fc2

= 456 + 2416 + 48120 + 1210

= 52202

Convolution layer

Fully connected layer → contribute to LP.

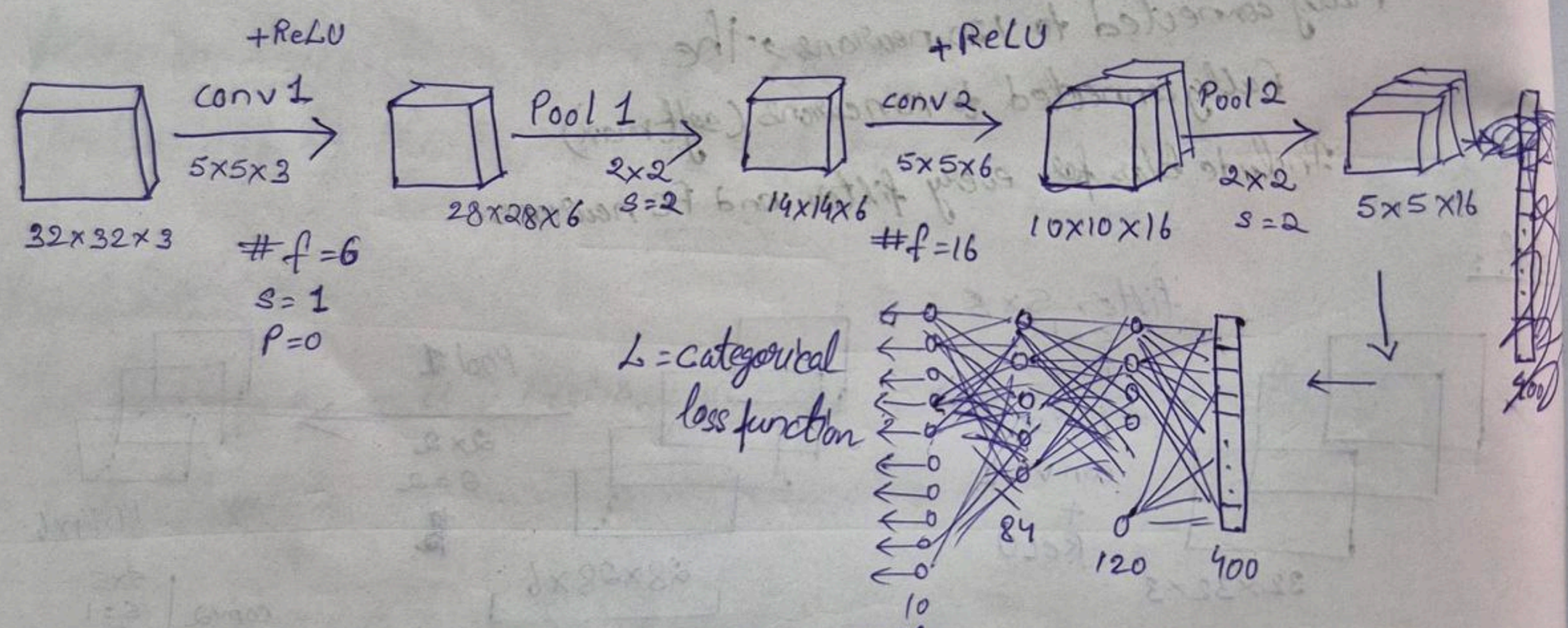
Total parameter = 456 (conv1)

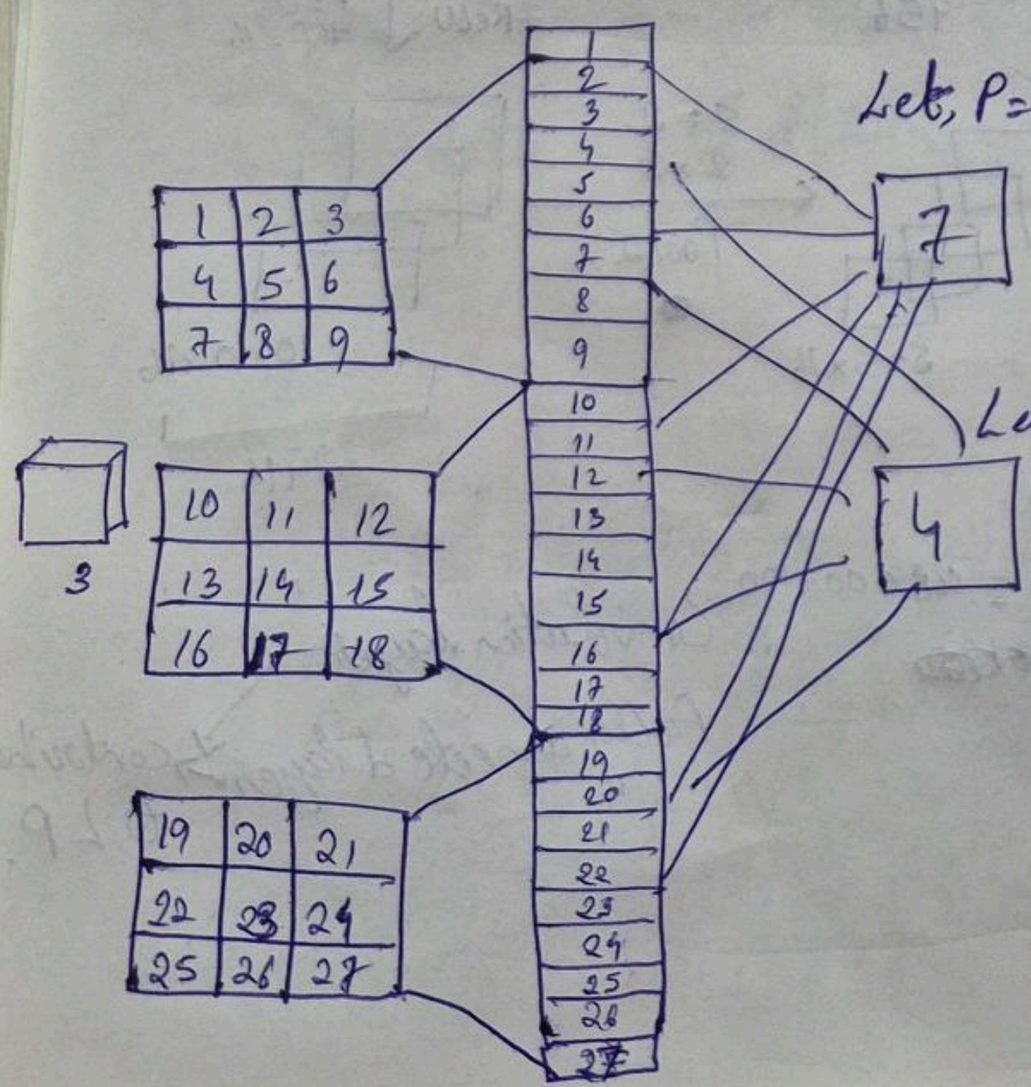
Total parameter = 2416 (conv2)

6/12

Q1) Draw a CNN architecture within given table.

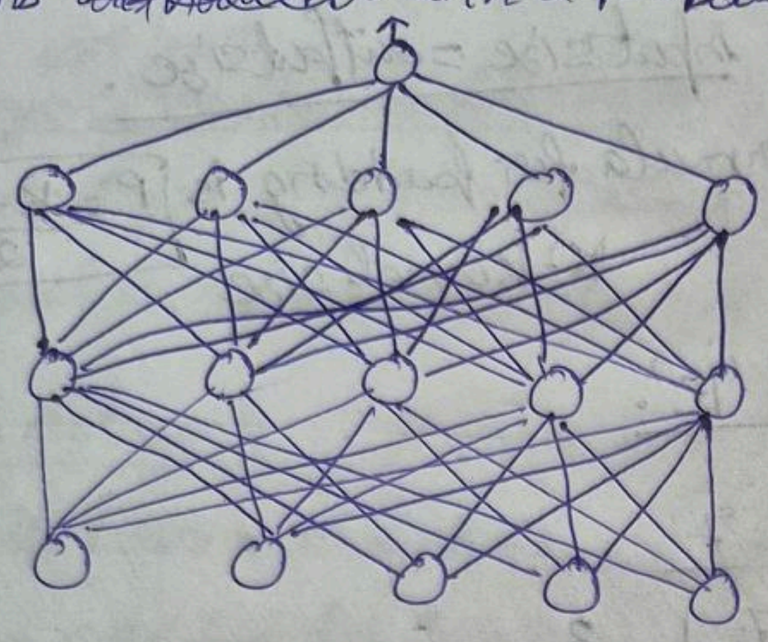
layer	Activation shape	Activation size
input	(32 x 32 x 3)	3072
conv. layer 1	(28 x 28 x 6)	
Pooling layer 1	(14 x 14 x 6)	
Conv. layer 2	(10 x 10 x 16)	
Pooling layer 2	(5 x 5 x 16)	
Fully connected layer 2	(120 x 1)	
Fully connected layer 2	(84 x 1)	
output	(10 x 1)	

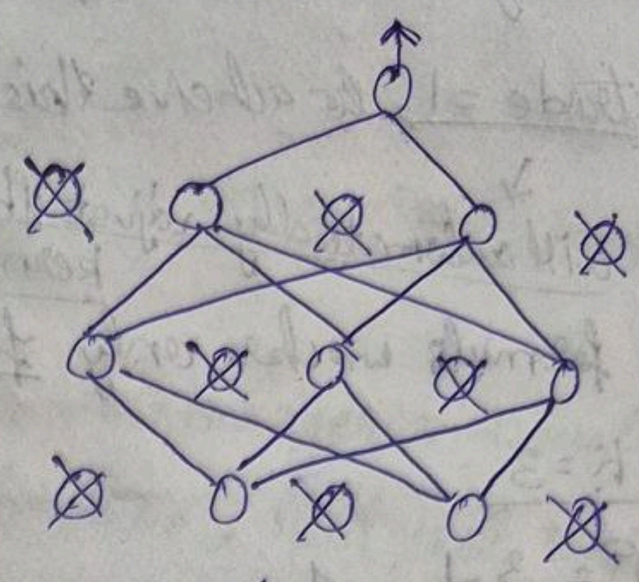




(certain values will contribute more the to ~~value~~ value 7 & remaining will contribute low)

Dropout - is a technique used in deep learning to prevent overfitting. During training dropout works ~~out~~ by randomly turning of (dropping) some neurons ~~without loss~~ with a probability P.



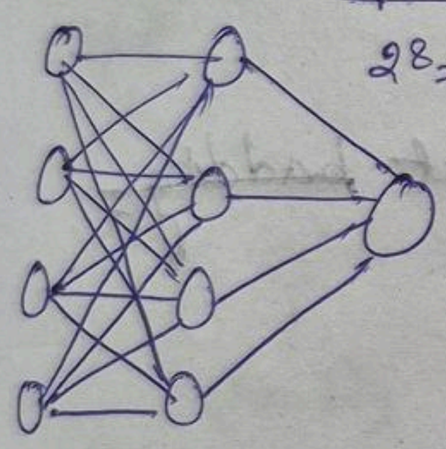


after dropout

- Dropout is applied at the time of training not at the time of testing.
- We take average of all the calculation of epochs.

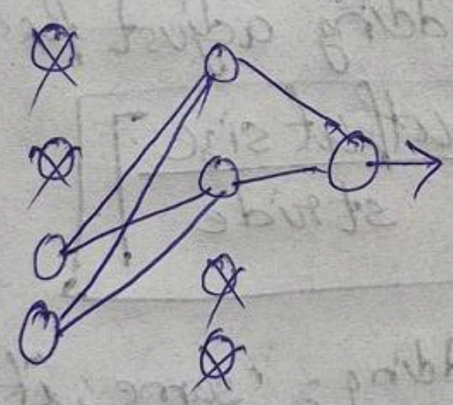
epoch = 4

$2^8 = 256$

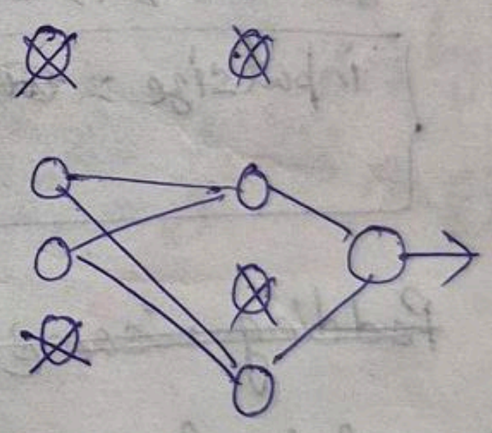


P=0.5 P=0.5

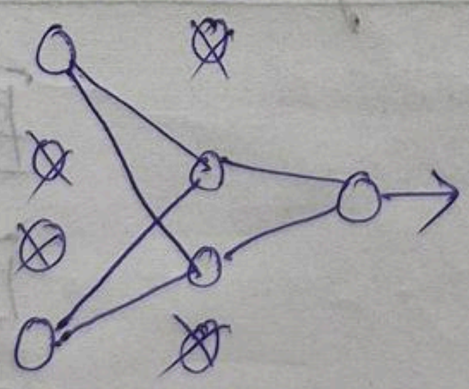
epoch 1



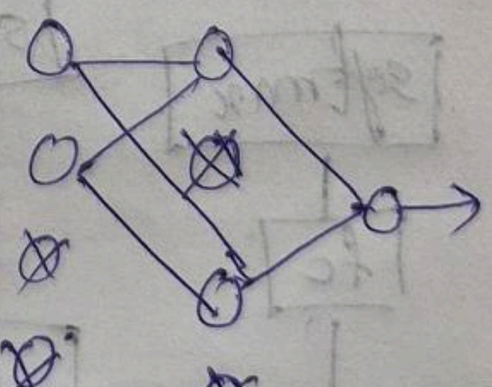
epoch 3



epoch 2



epoch 4



- With probability P=0.6, suppose we have a neuron with original output $x=10$ and we apply dropout P=0.6 the neuron keeps its output as 10 with probability 1-P=0.4, the neuron is dropped and keep the output as 0. So, the expected output is $(P \cdot x) + (1-P)(0)$

$= 6.$

So, on average the neuron output become 6 instead of 10.

8/12

Padding P in same padding -

A padding value is choosen so that input size = output size.

For stride = 1, to acheive this the formula for padding is $\boxed{P = \frac{K-1}{2}}$,

will automatically adjust the kernel size

K = kernel size

This formula works only for odd 'K'.

For, K = 3

$P = \frac{3-1}{2} = 1$

For, K = 7

$P = \frac{7-1}{2} = 3$

For, K = 5

$P = \frac{5-1}{2} = 2$

For, K = 11

$P = \frac{11-1}{2} = 5$

~~For, stri~~

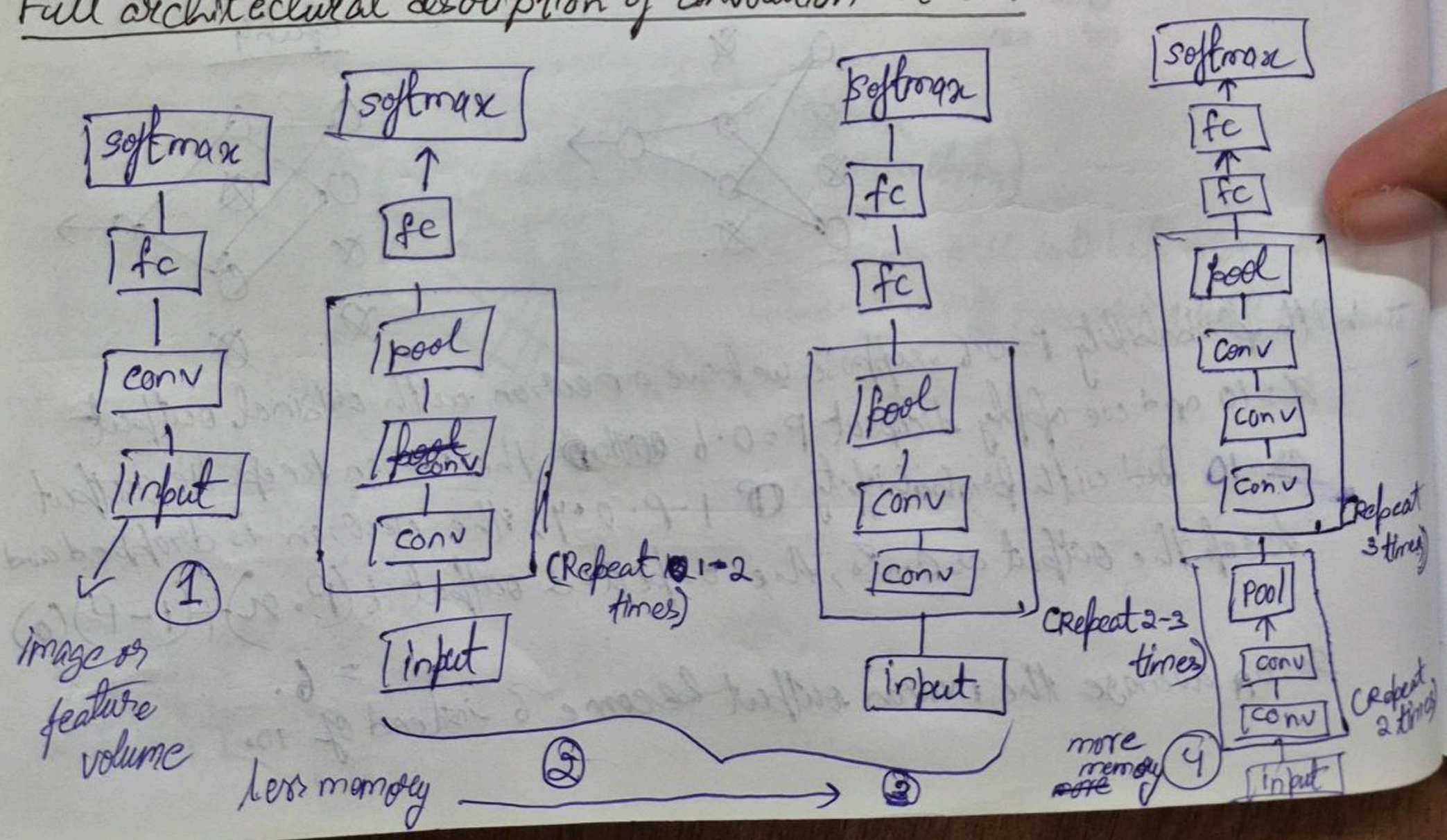
When stride ≠ 1, same padding always uses

When stride > 1, same padding adjust padding so that ~~padding~~

$$\boxed{\text{input size} = \left\lceil \frac{\text{output size}}{\text{stride}} \right\rceil}$$

~~Padding = same~~ If padding = 'same', it will automatically adjust i/p = o/p

Full architectural description of convolution network -

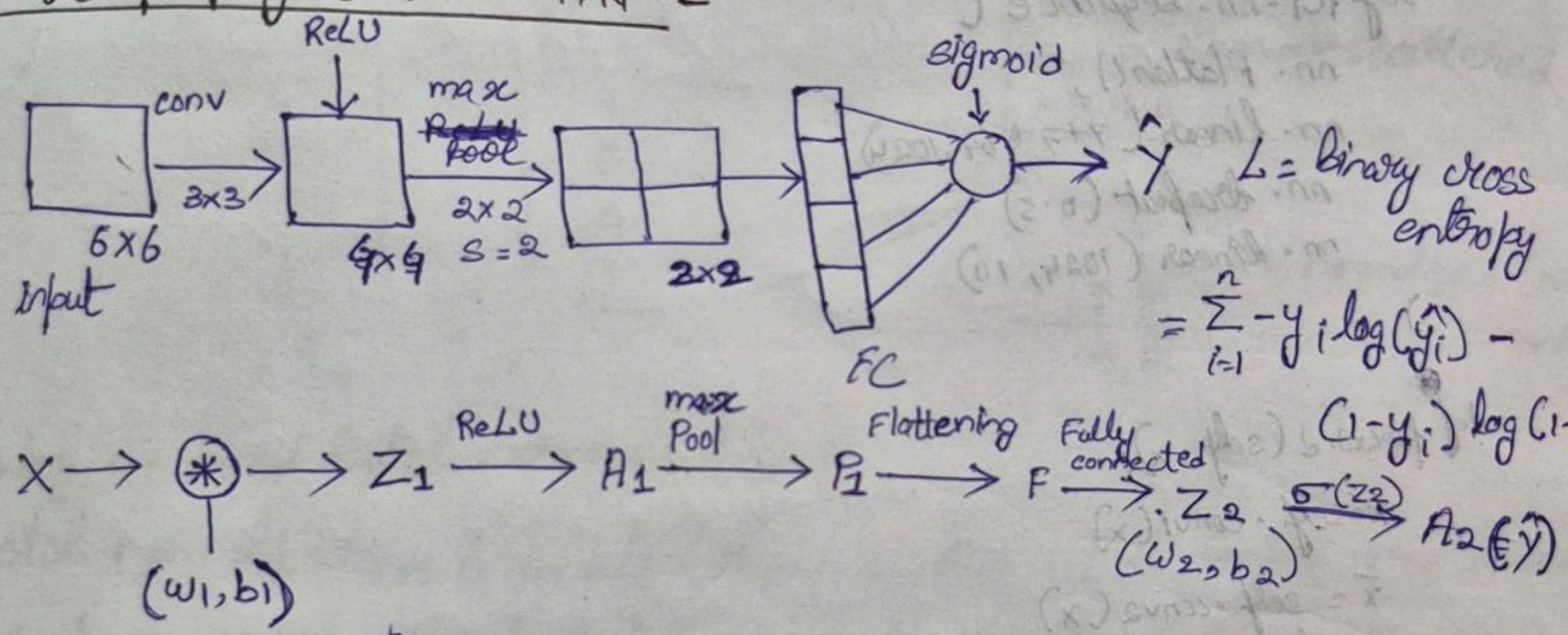


- shallow network which is the 1st n/w has few convolutional layers b/c it has fewer parameters therefore it needs less computation and can be used for simple task and when the data is limited.

- 2nd layer & 3rd layer - medium depth layer - This n/w uses few convolution layers repeated per stage and these networks captures more complex features in comparision to 1st network.

- Deep network - It consists of many convolution layer and multistages and it learns very hierarchical architecture, abstract representation needed for hard region task.

Forward propagation in CNN -



$Z_1 = conv(x w_1) + b_1$

$A_1 = ReLU(Z_1)$

$P_1 = maxpool(A_1)$

$F = Flatten(P_1)$

$Z_2 = (w_2 F) + b_2$

$A_2 = \sigma(Z_2)$

Backpropagation -

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

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

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial Z_2} \frac{\partial Z_2}{\partial w_2} \quad ; \quad \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial Z_2} \frac{\partial Z_2}{\partial F} \frac{\partial F}{\partial P_1} \frac{\partial P_1}{\partial A_1} \frac{\partial A_1}{\partial Z_1} \frac{\partial Z_1}{\partial w_1}$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial Z_2} \frac{\partial Z_2}{\partial b_2} \quad ; \quad \frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial Z_2} \frac{\partial Z_2}{\partial F} \frac{\partial F}{\partial P_1} \frac{\partial P_1}{\partial A_1} \frac{\partial A_1}{\partial Z_1} \frac{\partial Z_1}{\partial b_1}$$

```
class MNIST convNet (nn.module)
    def_init_(self):
        super (MNISTConvNet, self) _init_ ()
        self. conv1 = nn. sequential ()
          nn.conv2d (1, 32, 5, padding = 'same'),
          nn.ReLU (),
          nn.Maxpool2d (2)
        )
        self. conv2 = nn. sequential (
          nn. conv2d (32, 64, 5, padding = 'same')
          nn. ReLU ()
          nn. Maxpool2d (2)
        -)
        self fc1 = nn. sequence (
          nn. flatten (),
          nn. linear (7*7*64, 1024)
          nn. dropout (0.5)
          nn. linear (1024, 10)
        )
    def forward (self, x):
        x = self. conv1 (x)
        x̄ = self. conv2 (x)
        return self. fc (x)
        lr = 0.0001
        num_epoch = 40
        model = MNIST convNet()
        loss_fn = nn. Cross entropy Loss ()
        optimizer = optim. SGD (model parameters (),
```

$$\frac{\partial L}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left[-y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)\right]$$

$$= \frac{-y}{\hat{y}_i} + \frac{(1-y_i)}{(1-\hat{y}_i)} = \frac{-y(1-\hat{y}_i) + \hat{y}_i(1-y_i)}{\hat{y}(1-\hat{y}_i)} = \frac{\hat{y}_i - y_i}{\hat{y}(1-\hat{y}_i)}$$

$$\frac{\partial \hat{y}}{\partial z_2} = \frac{\partial \sigma(z_2)}{\partial z_2} = \hat{y}_i(1-\hat{y}_i)$$

$$\frac{\partial z_2}{\partial w_2} = \frac{\partial}{\partial w_2}\left[w_2 f + b_2\right] = f$$

$$\frac{\partial L}{\partial w_2} = \frac{\hat{y}_i - y_i}{y_i(\cancel{1}-\hat{y}_i)} \times \hat{y}_i(\cancel{1}-\hat{y}_i)\cdot f = \underbrace{(\hat{y}_i - y_i)}_{1\times 1}\cdot f_{4\times 1}$$

$$w_{2new} = w_{old} - \eta(\hat{y}_i - y_i) f$$

$$b_{2new} = b_{old} - \eta(\hat{y}_i - y_i).$$

11/12

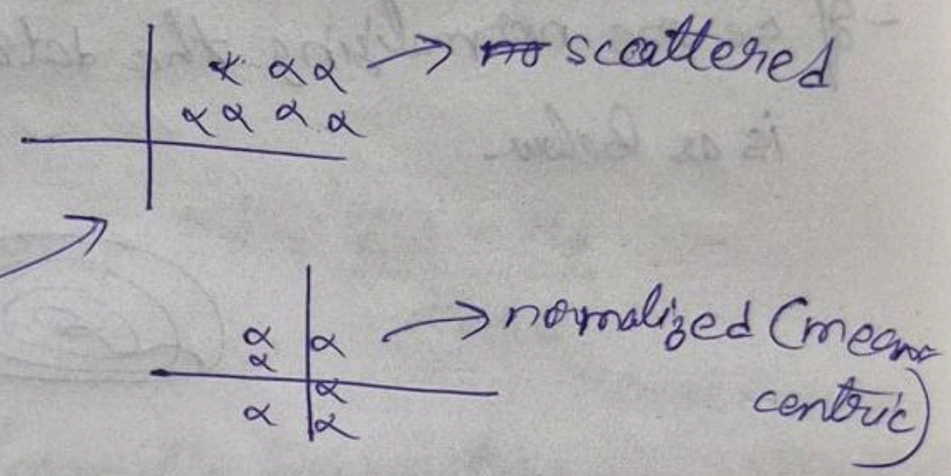
Batch Normalization -

Normalization of data -

mean = 0, SD = 1

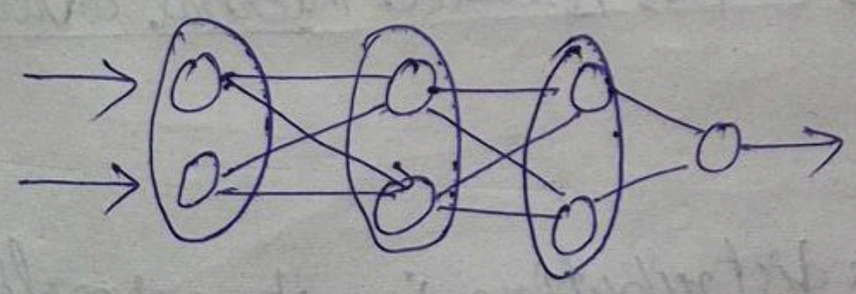
cgpa	iq	placed
80	8	1
90	9	0
70	7	0
60	6	1

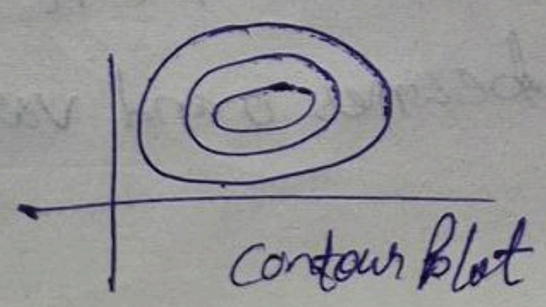
1-100 1-10



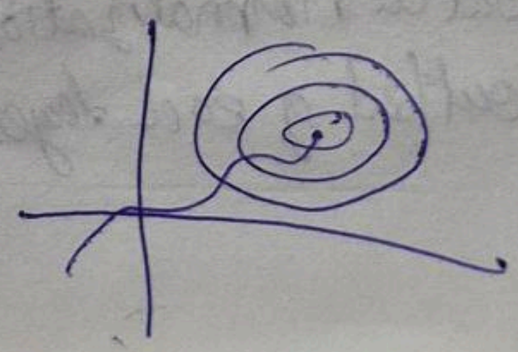
Q> Why we need batch normalization?

- Batch normalization is an algorithmic method which makes the training of deep neural network faster and more stable.
- It consist of normalizing activation vectors from hidden layer using the mean and variance of the current batch. This normalization step is applied right before or right after the non-linear function. By applying batch normalization our training becomes fast i.e ~~the fitt~~ with few epochs it converges towards optimal soln.
- It makes neural network training more stable.

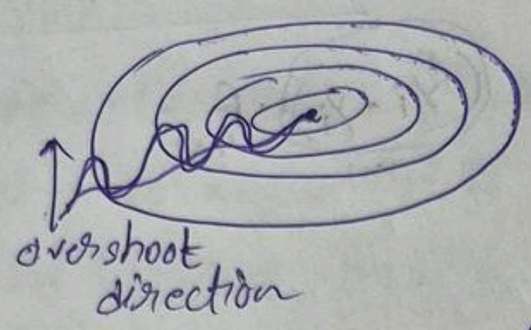




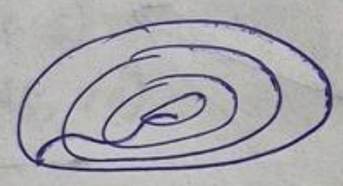
on normalized data



- For unnormalised data, the contour of the cost funcn is below highly elongated in one direction.



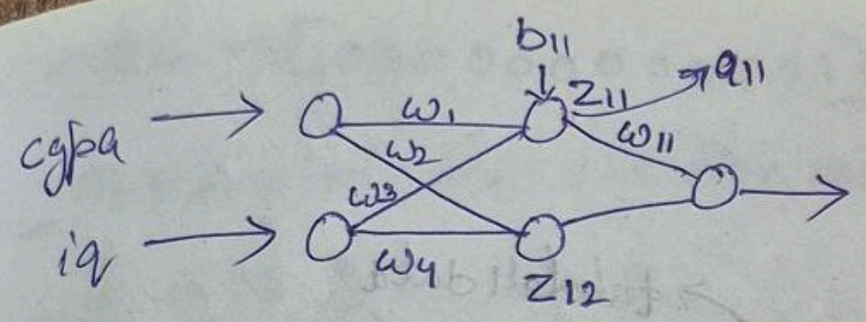
- In this case we can't approach to the minima with high learning rate as we may overshoot in non-elongated direction.
- But, if ~~we normalize the data~~ keep the learning rate small then, we may move with small steps towards the minima. Therefore, training will be slow or unstable.
- If we are normalizing the datapoints the contour of the cost function is as below



- In this case we can approach the minima with high learning rate (long steps) which makes the training faster and more stable.

<u>Internal covariate shift</u> -

- In a neural n/w, a network has many layers, each layer sends o/p the next layer, but during training when model parameters change the o/p distribution of one layer keeps changing. The meaning is <u>mean shift</u> and <u>varience shift</u>. So, the next layer keeps receiving different types of data distribution every time. This is called <u>internal covariate shift</u>.
- Batch normalization fixes the data distribution i.e it normalizes the output of each layer so that the mean becomes 0 and varience becomes 1



$$z_{11} = cgpa\, w_1 + iq\, w_2 + b_1$$

$$g(z_{11}) = a_{11}$$

$$z_{11}^N = \frac{z_{11} - M}{\sigma}$$

$M \to$ Mean, $\sigma \to$ standard deviation

$$z_{11} \to z_{11}^N \to g(z_{11}^N) = a_{11}^N$$ — (More accurate)

$$z_{11} \to g(z_{11}) \to a_{11} \to a_{11}^N$$

$$\begin{bmatrix} z_{11} \\ - - \\ - - \\ - - \end{bmatrix}_{4\times 2} \begin{bmatrix} - - \\ - - \end{bmatrix}_{2\times 2} + [b] = \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{4\times 2}$$

$$M_B = \frac{1}{m} \sum_{i=1}^{n} x_i$$

$$\sigma_B = \sqrt{\frac{1}{m} \sum_{i=1}^{n} (z_{11} - M_B)}$$

$$z_{11}^N = \frac{z_{11} - M_B}{\sigma_B + \epsilon}$$

$\epsilon \to$ const (avoid denominator to be 0)

$$z_{11}^{BN} = z_{11}^N \gamma + \beta$$, γ & $\beta \to$ learnable parameters

$$z_{11} \to z_{11}^N \to g(z_{11}^{BN}) = a_{11}, \quad \gamma = 1$$

$$\beta = 0$$

The parameter γ & β are learnable parameters and will be trained during the training process, if affine is true then this learnable parameter need to be updated, otherwise the mean is substracted from i/p and divided by standard deviation.

~~layer =~~

layer = nn.BatchNorm2d(num_features = 32, eps = 1e-05, momentum = 0.1, affine = True, track_running_stats = True)

ϵ

15/12

Recurrent Neural Network -

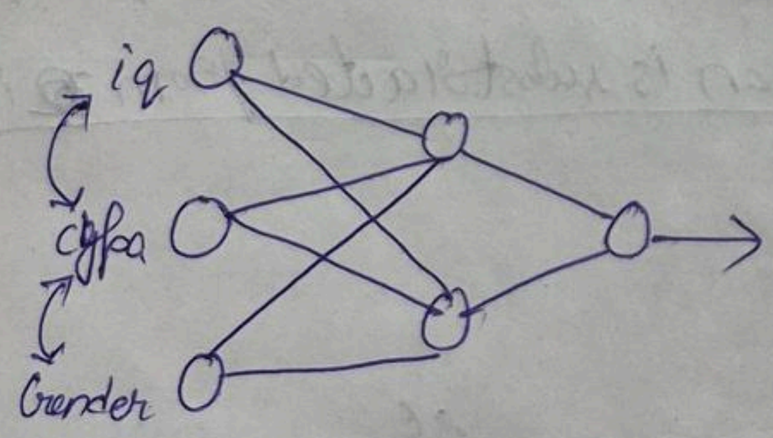
Q> Why RNN ?
Q> Architecture of RNN.
Q> Training of RNN.
Q> Application of RNN.
Q> Types of RNN.
Q> Limitation of RNN.

ANN → tabular data
CNN → Image data
} Used for fixed size data

- CNN works extremely well for fixed size data such as images,
- CNN fails when input length changes, in real world most of the information we deal with comes in the form of sequences.
  Eg: Reading text, listening to audio, watching videos or analyzing stock prices.
- This task requires models that handle variable length.

Non sequential data -

iq	cgpa	gender	placement



- The model o/p don't get effected on sequence of i/p.
- Here, i/p doesn't follow any sequence.

Text

Hi my name is Barnali.
I love travelling.
India won the match.

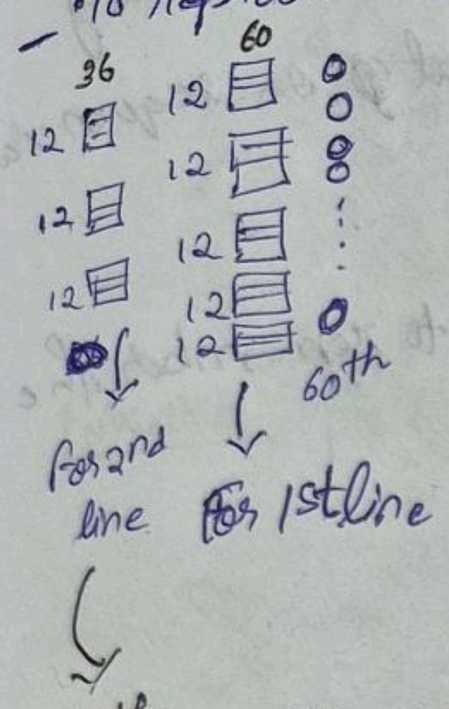
12 words

One hot encoding

Hi → [100000000000]
my → [010000000000]
is → [000100000000]
name → [001000000000]
Barnali → [000010000000]
⋮

match → [000000000001]

- To represent the 1st line we need 5x12 = 60 neurons.



in this case we need 0 padding to make it 60.

* If the word count is more than 5. (i.e 6) Then it will require 6 x 12 = 72 neurons which is not possible, so neural networks breaks.

Application of RNN

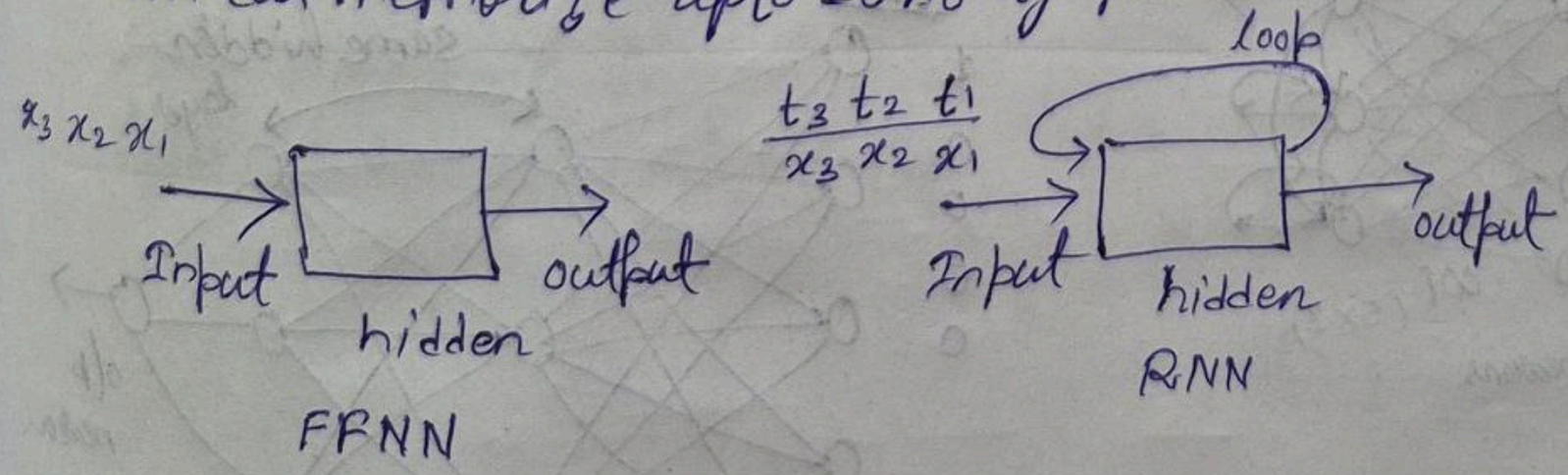
- There are various applications

i) Sentiment analysis.
ii) sentence completion.
iii) Image caption generator.
iv) Machine translation
v) Question and answering system.

Architecture of RNN

Stateless - Model which don't have memory.
Stateful - Model which has memory.

- RNN can memorize upto 10 no. of previous words



- RNNs were introduced to address the limitation of traditional NN, when it comes to processing sequencial data FNN takes inputs and process each i/p independently through a no. of hidden layers without

unable to handle sequential data.

- on the other hand RNNs are a class of network ~~which~~ which has memory it can hold the past i/ps i.e why they ~~want~~ work great on sequential data.

- RNN has a recurrent unit which facilitates it to remember the previous i/p information.

Movie review data for sentiment analysis

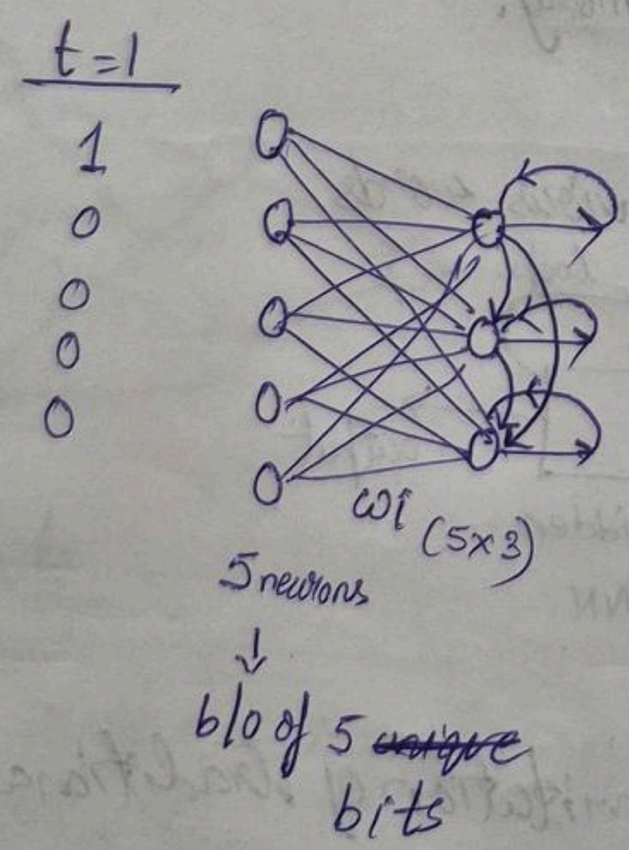
X movie review	sentiment
x_1 movie is good (3) (x_{11} x_{12} x_{13})	1
x_2 Movie is bad (3) (x_{21} x_{22} x_{23})	0
x_3 Movie is not good (4) (x_{31} x_{32} x_{33} x_{34})	0

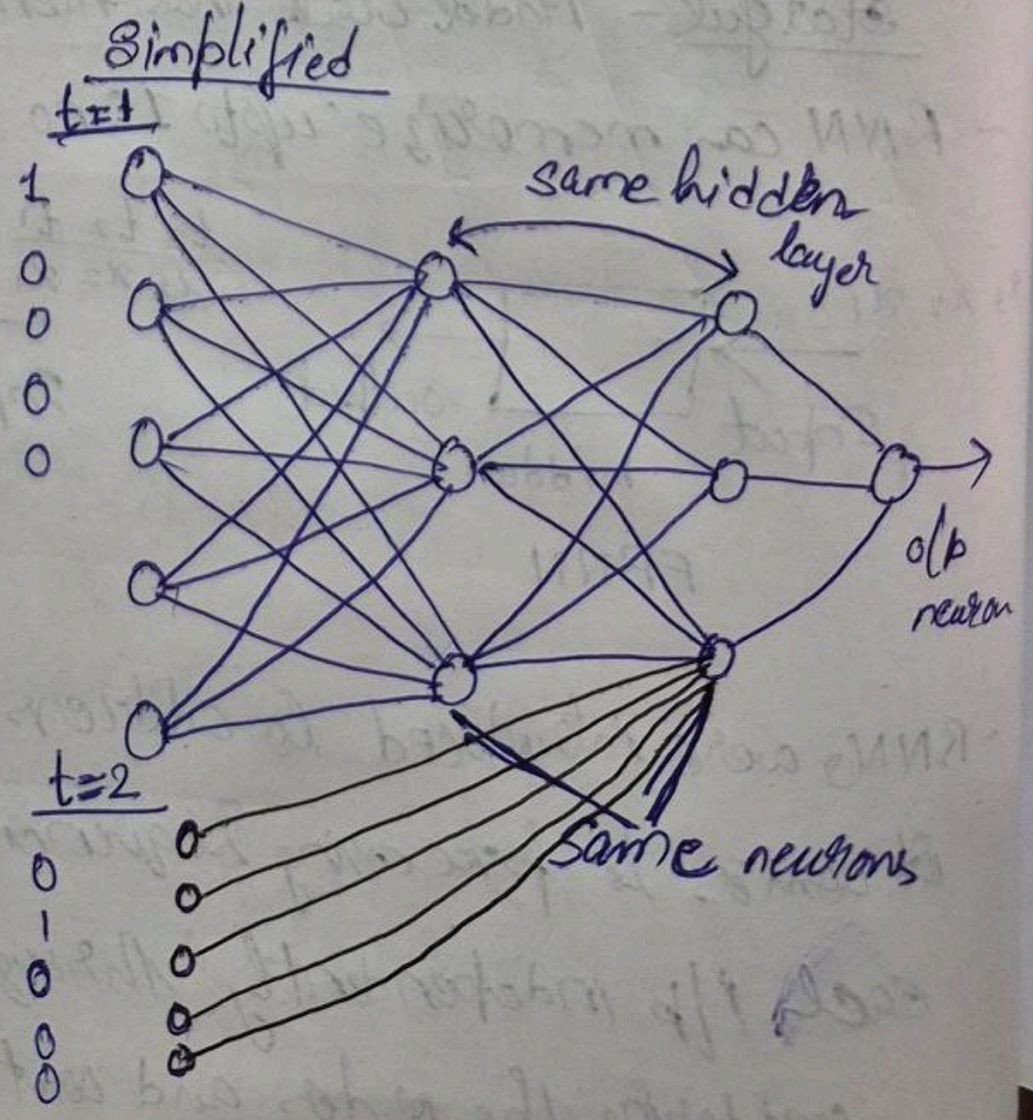
5 word corpus (unique words)

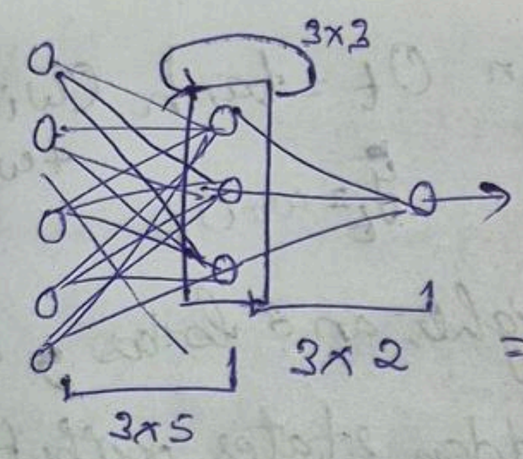
movie → [10000]
is → [01000]
good → [00100]
bad → [00010]
not → [00001]

x_1 → [10000] [01000] [00100] → (3, 5)
x_2 → [10000] [01000] [00010] → (3, 5)
x_3 → [10000] [01000] [00001] [00100] → (4, 5)

time step
word corpus

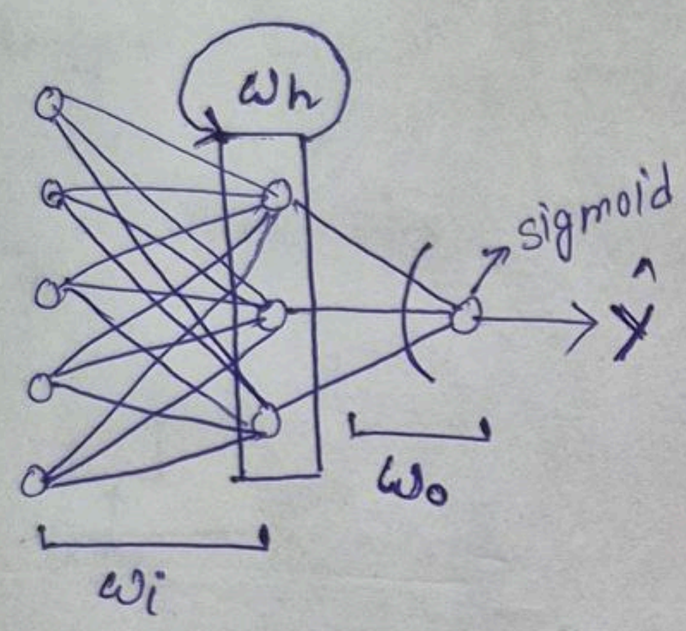


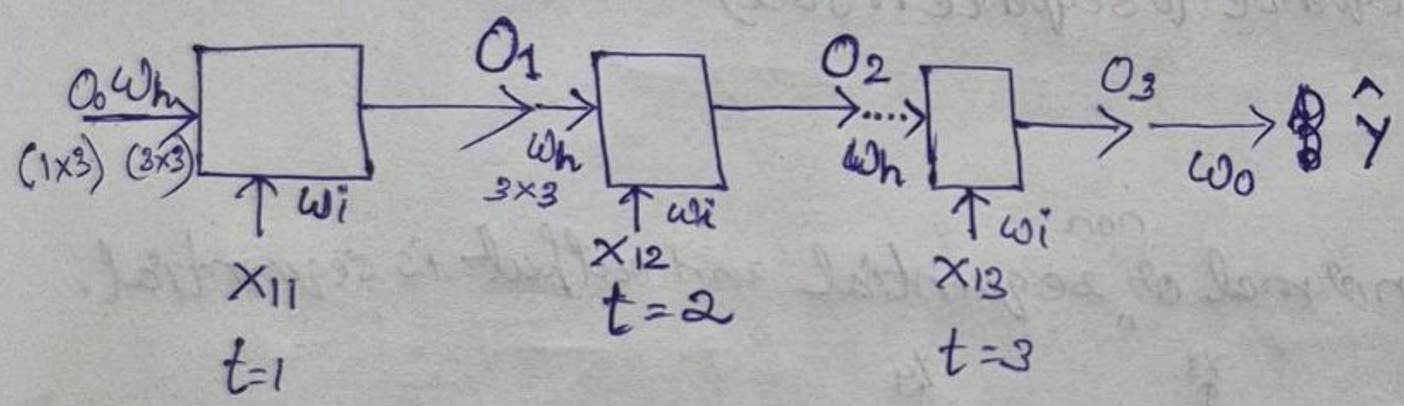




= 27 weights
4 bias

Forward Propagation –



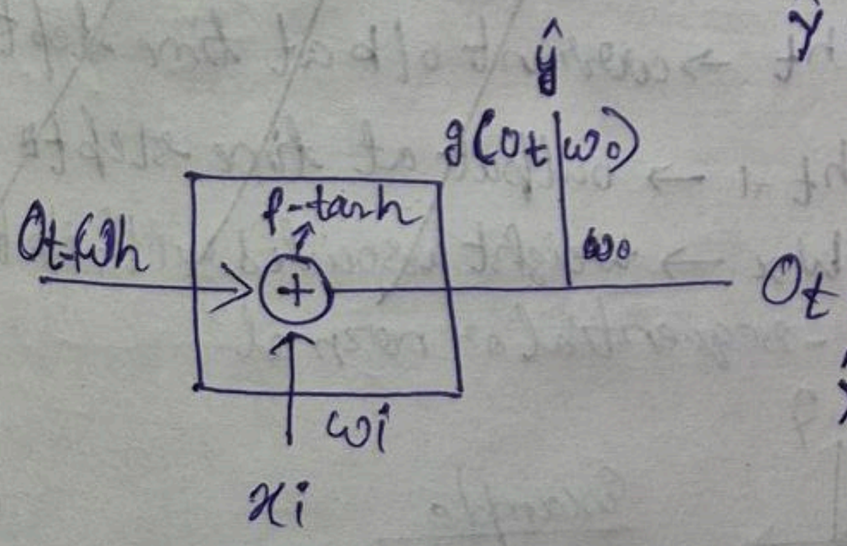


~~$O_1 = (x_{11} \cdot \omega_i + b)$~~ $O_1 = \tanh(x_{11} \cdot \omega_i + b) = \tanh(x_{11} \cdot \omega_i + O_0 \omega_h + b)$

$O_2 = \tanh(\omega_i x_{12} + \omega_h \cdot O_1 + b)$

$O_3 = \tanh(\omega_i x_{13} + \omega_h O_2 + b)$

$\hat{y} = \sigma(O_3 \omega_o)$



$\hat{y} = g(O_3 w_o)$

$t=1$ $O_1 = \tanh(O_0 \omega_h + x_{11} \omega_i)$

$O_2 = \tanh(O_1 \omega_h + x_{12} \omega_i)$

$O_3 = \tanh(O_2 \omega_h + x_{13} \omega_i)$

- With $t=2$, $t=3$ we have 2 inputs but in $t=1$, we have only 1 i/p.
- To have consistency O_0 which is vector of 1x3 with all zero values or random no.'s being associated with the 1st input.

Q1) Consider a simple RNN defined by the eqn $O_t = \tanh(w_i x_i + w_h O_t + b)$

$\hat{y}_t = w_o O_t$

given i/p sequence is $x_1 = 1$, $x_2 = 2$ &

initial hidden state $O_0 = 0$. Weights and biases, $w_i = 0.5$

$w_h = 0.8$ and $b = 0$. Find out the hidden states output O_1, O_2

and $\hat{y}$.

Types of RNN -

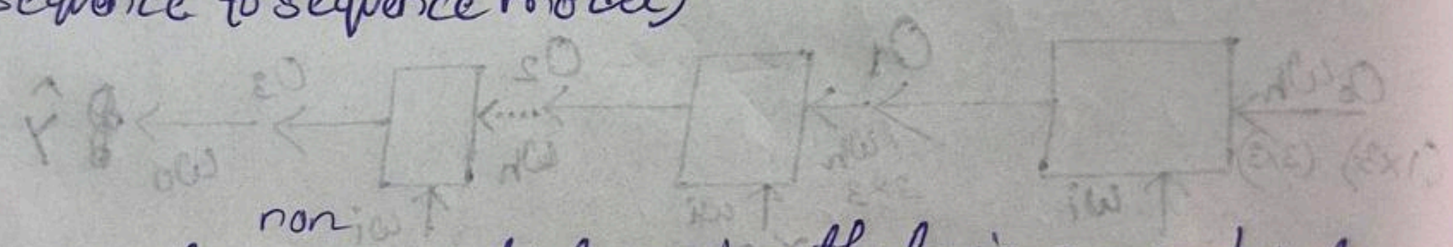
There are 4 types of RNN architecture

i) One to many

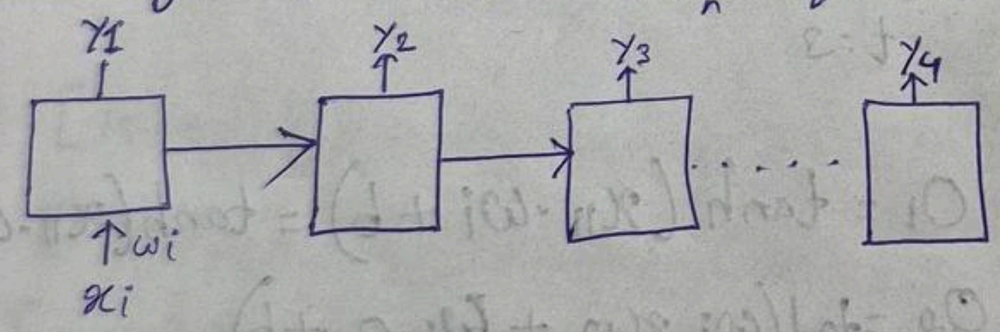
ii) Many to one

iii) Many to many (sequence to sequence model)

iv) One to one



One to many - i/p is normal or non sequential and output is sequential.



Application

i) Used in image captioning

ii) Baby name generation

iii) Music generation

$h_t = \tanh(w_x x + W_h h_{t-1})$

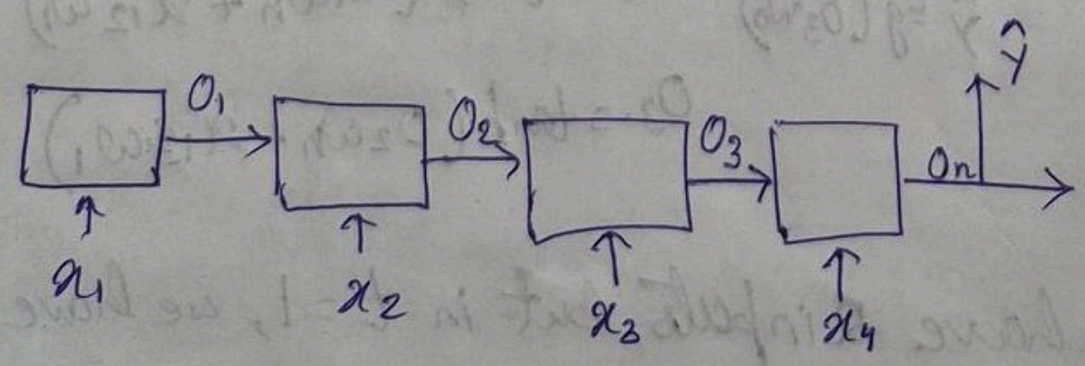
$y_t = w_y h_t$

$h_t \rightarrow$ current o/p at time step t

$h_{t-1} \rightarrow$ output at time step t-1

$W_x \rightarrow$ weight associated with i/p to hid

Many to one - i/p is sequencial, o/p is non-sequential or normal



Example

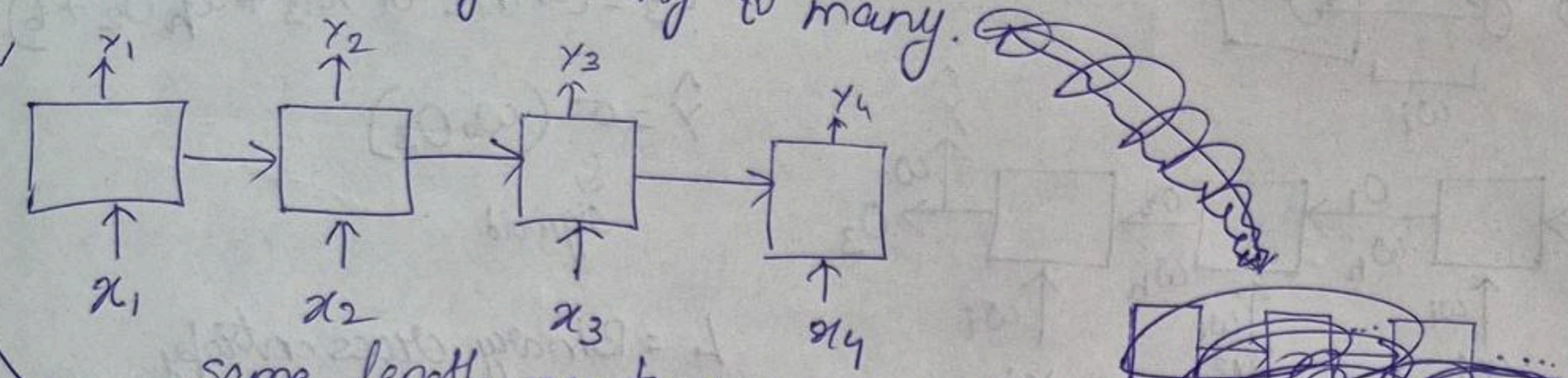
i) Sentiment analysis

ii) Text classification

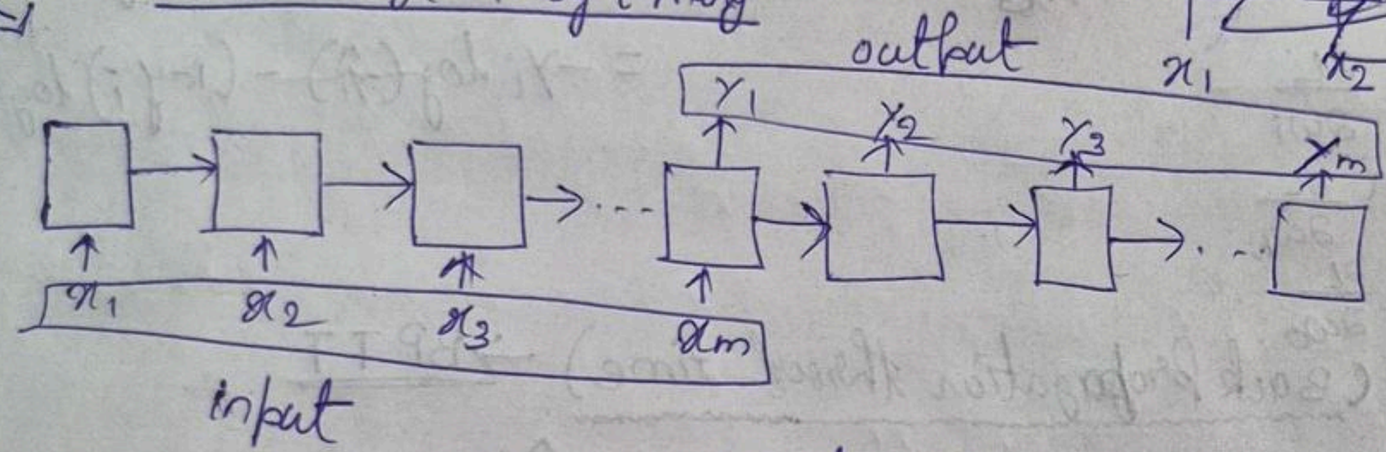
iii) Rating prediction

Many to many - i/p to this is sequential data and o/p is also ~~sequential~~ sequential data.

Sequence to sequence model can be classified into 2 categories
i) Same length many to many
ii) Different length many to many.



same length many to many



variable length many to many

Training of RNN -

i) Forward propagation
ii) Back propagation

Forward propagation - To understand forward & backpropagation we are considering a many to one architecture (sentiment analysis).

text	sentiment
cat rat mat	0
mat mat rat	0
cat cat mat	1

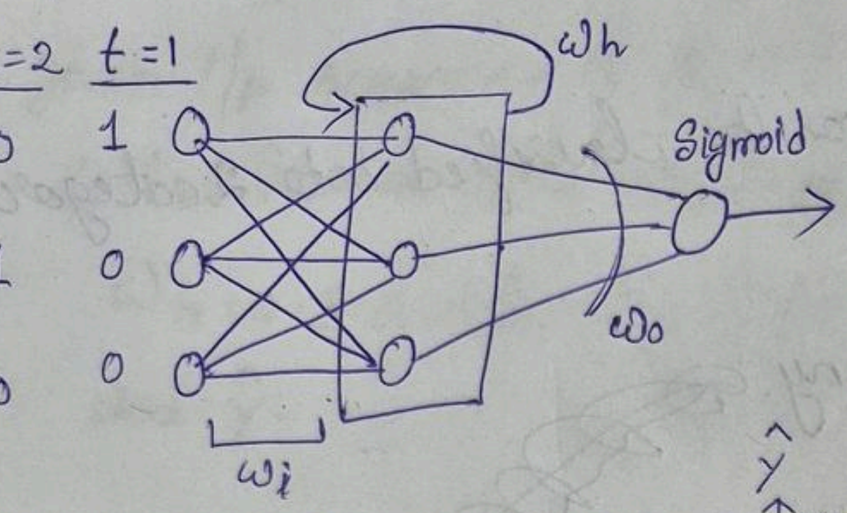
total unique word = 3

cat → [1 0 0]
rat → [0 1 0]
mat → [0 0 1]

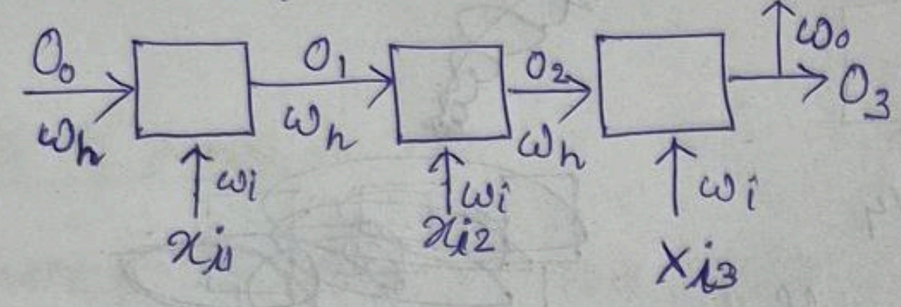
	x_{11}	x_{12}	x_{13}		sentiment
x_1	[1 0 0]	[0 1 0]	[0 0 1]	(3,3)	0
	x_{21}	x_{22}	x_{23}		
x_2	[0 0 1]	[0 0 1]	[0 1 0]	(3,3)	0
	x_{31}	x_{32}	x_{33}		
x_3	[1 0 0]	[1 0 0]	[0 0 1]	(3,3)	1

As feature is 3, we need to take 3 no. of input

$t=3$	$t=2$	$t=1$
0	0	1
0	1	0
1	0	0



$$O_1 = \tanh(w_i \cdot x_{i1} + w_h O_0 + b)$$
$$O_2 = \tanh(w_i x_{i2} + w_h O_1 + b)$$
$$O_3 = \tanh(w_i x_{i3} + w_h O_2 + b)$$
$$\hat{y} = \sigma(w_o O_3)$$
σ → sigmoid



L = Binary cross entropy
$$= -y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)$$

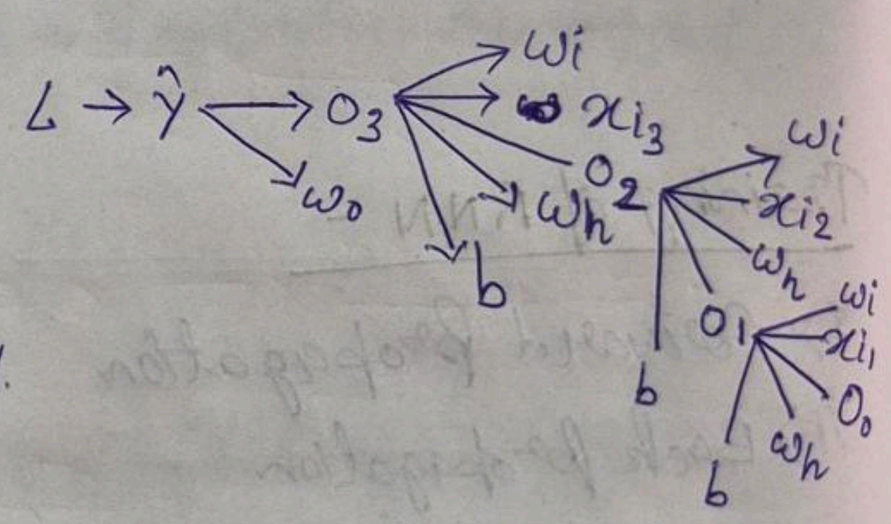
$$w_i = w_i - \eta \frac{\partial L}{\partial w_i}$$
$$w_h = w_h - \eta \frac{\partial L}{\partial w_h}$$
$$w_o = w_o - \eta \frac{\partial L}{\partial w_o}$$

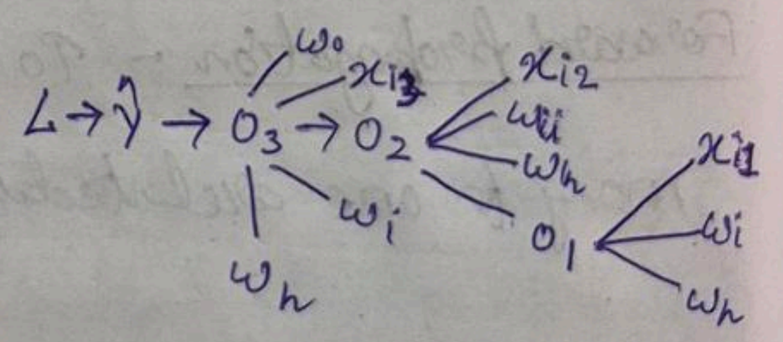
<u>Back Propagation</u> (Back Propagation through time) → <u>BPTT</u>

$$\frac{\partial L}{\partial w_o} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} \times \frac{\partial O_3}{\partial w_o}$$

$L \to \hat{y} \to O_3$, w_o

$$\frac{\partial L}{\partial w_i} = \boxed{\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} \times \frac{\partial O_3}{\partial w_i}} \to \text{short term dependency}$$
$$+ \boxed{\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} \times \frac{\partial O_3}{\partial O_2} \times \frac{\partial O_2}{\partial w_i}} \to \text{middle depency.}$$
$$+ \boxed{\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} \times \frac{\partial O_3}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_i}}$$
↓ long term dependency



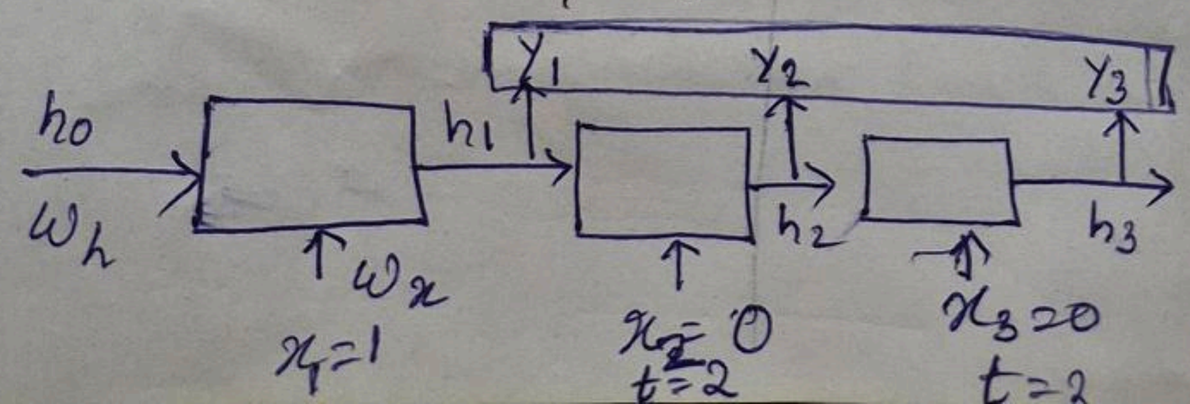


$$\frac{\partial L}{\partial w_h} = \left(\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} * \frac{\partial O_3}{\partial w_h}\right) + \left(\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} \times \frac{\partial O_3}{\partial O_2} \times \frac{\partial O_2}{\partial w_h}\right) + \left(\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial O_3} \times \frac{\partial O_3}{\partial O_2} \times \frac{\partial O_2}{\partial O_1} \times \frac{\partial O_1}{\partial w_h}\right)$$

<u>20/12</u> <u>One to many</u>

Q> $x = 1$, $h_0 = 0$

$w_x = 0.6$, $w_h = 0.5$, $w_y = 2$



<u>Note</u>

In one to many x_2, x_3 the values should be 0.

$\boxed{\text{for } t > 1, x_t \neq 0}$

Sol:

$h_1 = \tanh(0.6 \times 1 + \cancel{0.5 \times 0}^{\,0}) = \tanh(0.6) = 0.537$

$y_1 = h_1 w_y = 0.537 \times 2 = \boxed{1.074}$ Ans

$h_2 = \tanh(0.6 \times 0 + 0.5 \times 0.537) = \tanh(0.2685) = 0.262$

$y_2 = h_2 w_y = 0.262 \times 2 = \boxed{0.524}$ Ans

$h_3 = \tanh(0.6 \times 0 + 0.5 \times 0.262) = \tanh(0.131) = 0.1302$

$y_3 = 0.1302 \times 2 = \boxed{0.260}$ Ans

Formula

$h_t = \tanh(w_x x + w_h h_{t-1})$

$y_t = w_y h_t$

* $h_t \rightarrow$ current o/p at time step t
* $h_{t-1} \rightarrow$ output at time step t-1

$w_x \rightarrow$ weight associated with input to hidden

$w_h \rightarrow$ weight associated with hidden layer (hidden to hidden)

$w_y \rightarrow$ weight associated with hidden to output layer.

Many to one RNN -

Q) input $x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$h_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

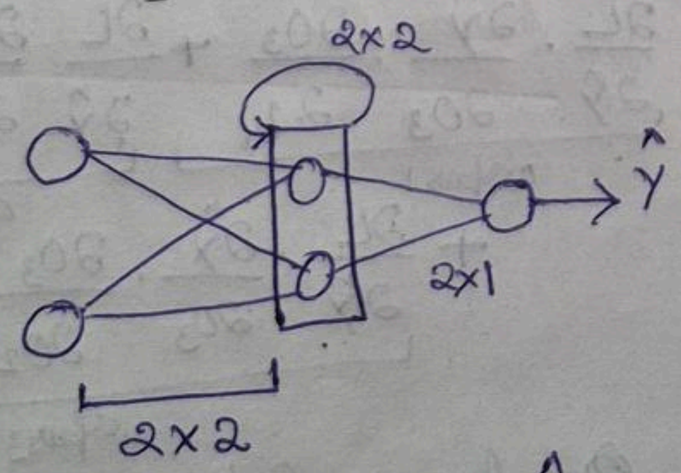
$w_x = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $w_h = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$

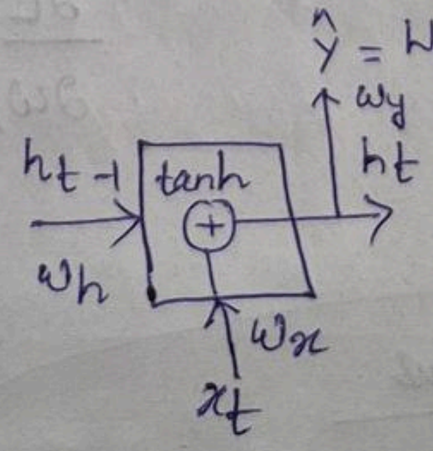
$w_y = [1, 1]$

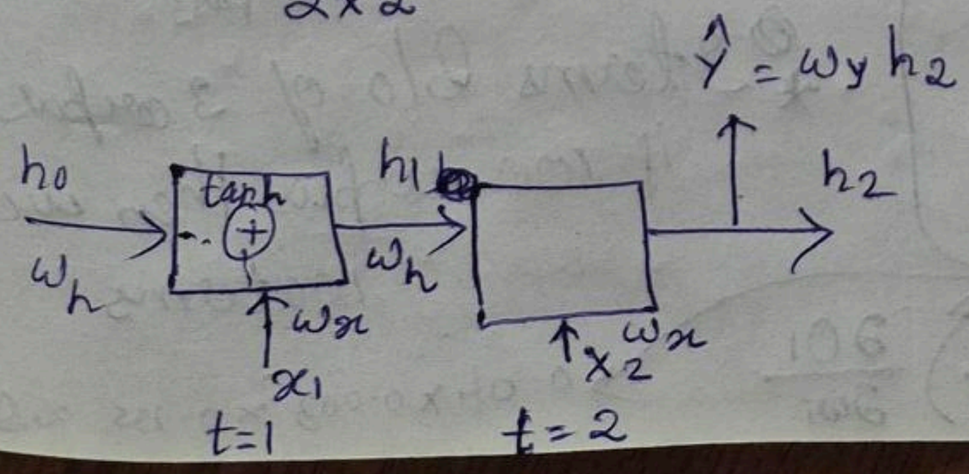
$$\boxed{\begin{aligned} h_t &= \tanh(w_x x_x + w_h \cdot h_{t-1} \ldots) \\ y &= w_y h_t \end{aligned}}$$

Regression function

Sol:







$$h_1 = \tanh\left(\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}0.5 & 0\\ 0 & 0.5\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\right)$$

$$= \tanh\left([\ ]_{2\times1} + [\ ]_{2\times1}\right) = \tanh\left([\ ]_{2\times1}\right) = \begin{bmatrix}0.761\\0\end{bmatrix}$$

~~$h_2 = 0.363$~~ ~~$h_3 = 0.761$~~

$$h_2 = \begin{bmatrix}0.363\\0.761\end{bmatrix}$$

$$\hat{y} = W_y h_2 = [1\ \ 1]\begin{bmatrix}0.363\\0.761\end{bmatrix} = 1.124$$

Problems with RNN -

RNN suffers from 2 major problems -

i) Long term dependency → We will face vanishing gradient problem

ii) stagnated ~~training~~ training

~~Due to~~

$$w_i = w_i - \eta \frac{\partial L}{\partial w_i}, \quad \frac{\partial L}{\partial w_y} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_3} \times \frac{\partial o_3}{\partial w_y}$$

$$w_h = w_h - \eta \frac{\partial L}{\partial w_h}, \quad \frac{\partial L}{\partial w_i} = \underbrace{\frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial o_3} \times \frac{\partial o_3}{\partial w_i}}_{\text{3 gradients (terms)}} + \underbrace{\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial w_i}}_{\text{5 gradient (terms)}} + \underbrace{\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}}$$

$$w_y = w_y - \eta \frac{\partial L}{\partial w_y}$$

$$\frac{\partial L}{\partial w_h} = \underbrace{\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial w_h}}_{\text{corpus 1}} + \underbrace{\frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial o_3} \frac{\partial o_3}{\partial o_2} \frac{\partial o_2}{\partial w_h}}_{\text{corpus 2}} + \underbrace{\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_3} \cdot \frac{\partial o_3}{\partial o_2} \cdot \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_h}}_{\text{corpus 3}}$$

100 word corpus

$$\frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{100}} \cdot \frac{\partial o_{100}}{\partial o_{99}} \cdots \frac{\partial o_2}{\partial o_1} \cdot \frac{\partial o_1}{\partial w_i}$$

$$= \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial o_{100}} \prod_{t=2}^{100} \left(\frac{\partial o_t}{\partial o_{t-1}}\right) \frac{\partial o_1}{\partial w_i}$$

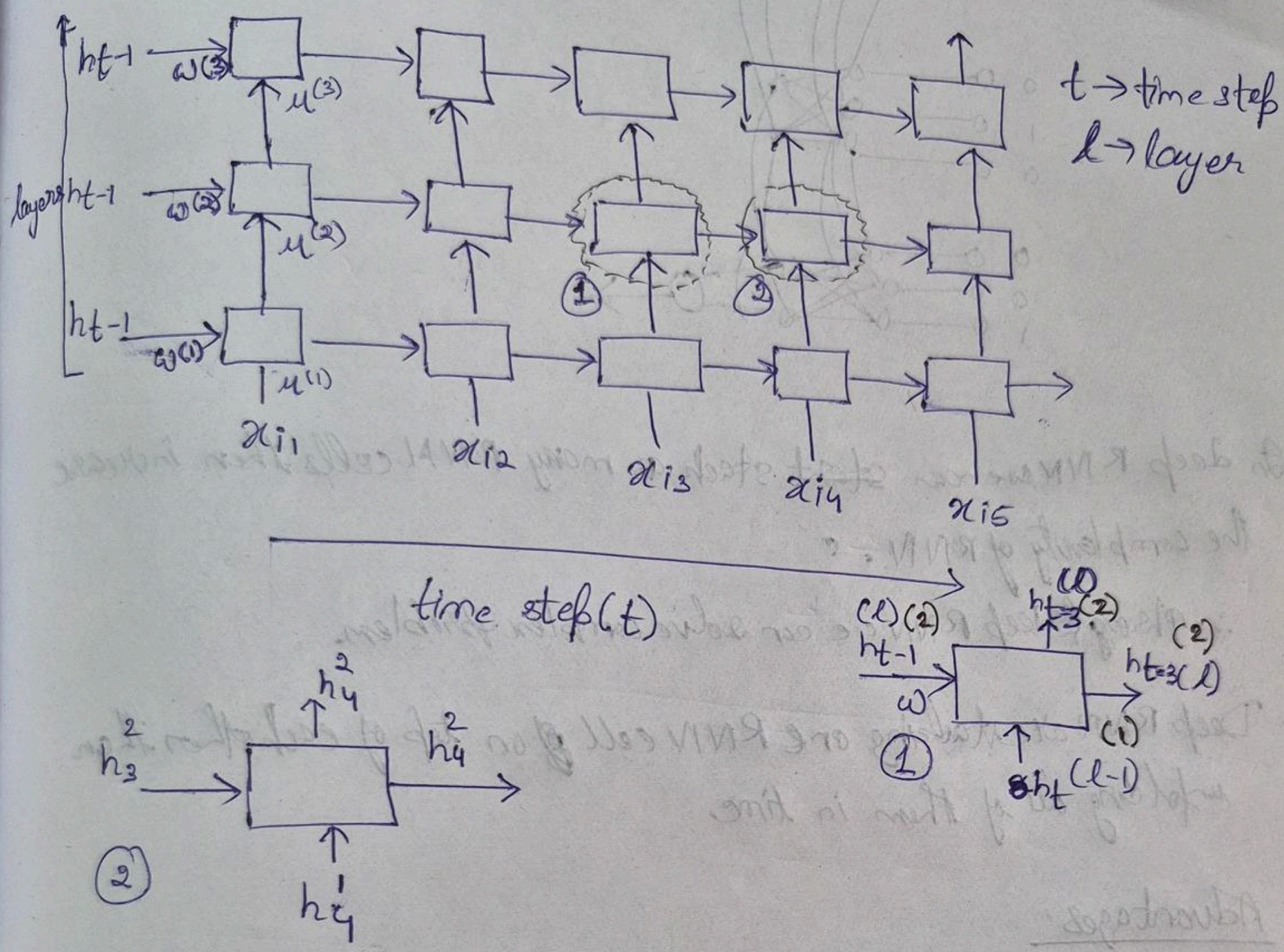
$0.04 \times 0.003 \times 0.135 \approx 0$

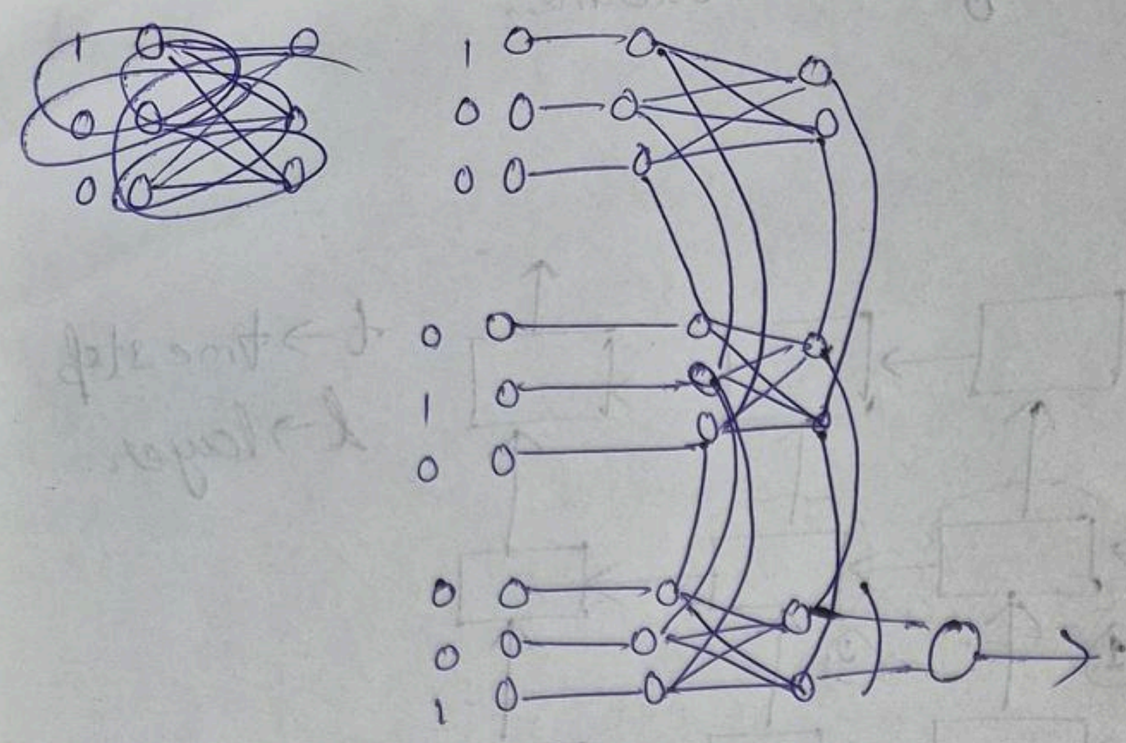
* 3 terms b/c of 3 corpus.
if 100 corpus then we will have 100 terms

- Weight updation happens basically on short terms.

Deep RNNs –

Many to one architecture





- In deep RNN we can ~~start~~ stack as many RNN cells then increase the complexity of RNN.
  ∴ Through deep RNN we can solve complex problem.
- Deep RNN are stacking one RNN cell on top of each other then unfolding all of them in time.

Advantages

- Deep RNNs are used for complex task.
  i) It can be used for large dataset where hierarchy is present.
  - In case of sentiment analysis the 1st layer tries the ~~sent~~ sentiment of the word, the 2nd layers tries of find out the ~~the~~ sentiment in sentence level, the 3rd layer tries to find out the sentiment in paragraph level.

LSTM - (Long Short Term Memory) -

- It is an RNN with a memory system instead of forgetting old inform quickly. It LSTM decides what to keep, what to forget and what to output using special structures called gates.