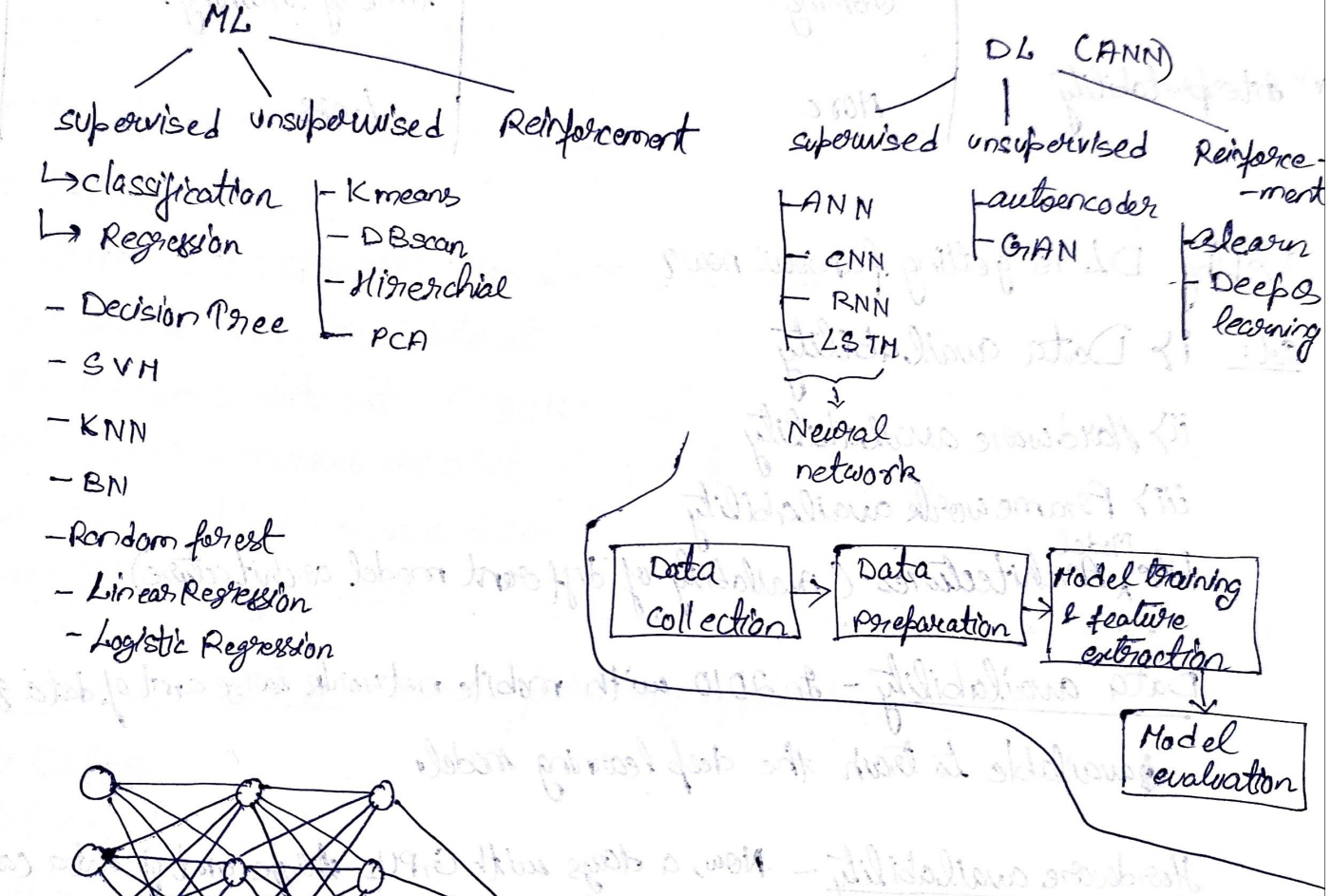
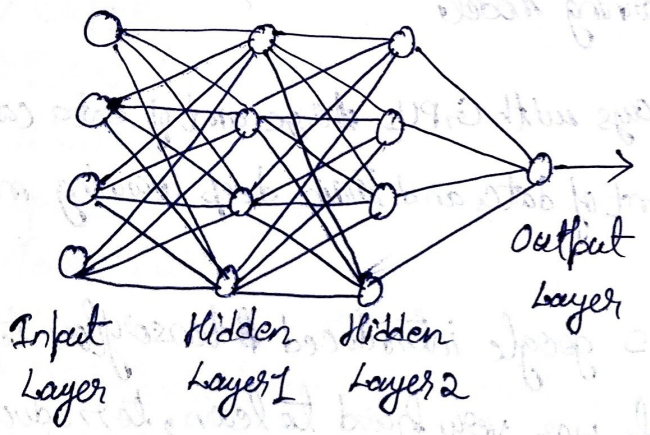
Deep Learning Using Python

6/9

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





- 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
ie Video

	ML	DL
i) Data Dependency	Less data (it does training only).	More data (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 dataset
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 deeplearning applications

i) Caffe
ii) Deep learge 4j
iii) H2O
iv) OpenNN
v) Tensorflow
vi) Keras
vii) Distributed machine learning toolkit
viii) ML-lib
ix) Pytorch

Types of deep learning and application -

Sequential data -

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

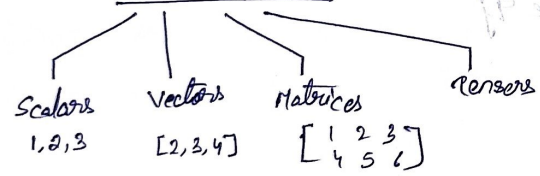
Application -

i) Autonomous car
ii) X ray classification in health care (CNN)
iii) Advanced facial recognisation (CNN)
iv) Visual question and answer (CNN ~~RNN~~)
v) Advanced voice assistence tools and devices (CRNN & LSTM)
vi) Automatic machine translation. (CRNN)
vii) Image recognition and image capturing (CNN) & (CRNN)
viii) Anamoly 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

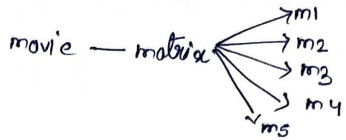
- Linear Algebra is a branch of mathematics that deals with the study of linear systems which are sets of equations involving linear functions of variables.



Q) Why linear algebra for DL?

Sol: i) Good generalization ability.
ii) Good data representation.

Movie Recommendation System



S_1 → The weather is good today
S_2 → My name is Ram
S_3 → How are you
S_4 → ITER is a good college

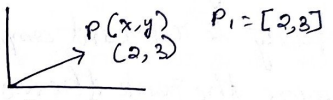
movie data

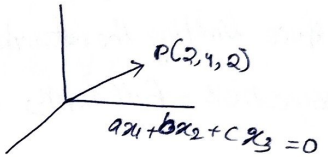
movie name	Summary
m1	S_1
m2	S_2
m3	S_3
m4	S_4

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

S_1 → 1 1 1 1 1 0 0 0 0 0 0 0 0
S_2 → 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.





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

$[a_1\ a_2\ a_3\ \cdots\ a_n] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = 0$

(Row vector) (column vector)

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, i.e 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 $= 2$

Use of column space in neural network – In a fully connected linear neural network equation $\boxed{y = wx + b}$, where $w \rightarrow$ matrix, $x \rightarrow$ is a matrix
($w \rightarrow$ weight matrix, $x \rightarrow$ 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.

Null space – 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 $A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$

Sol: $Ax = 0$

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

Row Echilon form –

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

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

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

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

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

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

For $\lambda = 1$

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

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

$x + y = 0$

For $\lambda = 6$

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

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

$$H(P) = -\sum P_i \log(P_i)$$

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

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

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 tails

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

For case I:

$$H(p,q) = -0.75 \log(0.72) - 0.25 \log(0.28) = 10.814$$

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

For case II:

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

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 ~~keeping~~ 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 -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 ~~proper~~ difference b/w cross entropy and entropy.

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

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

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

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

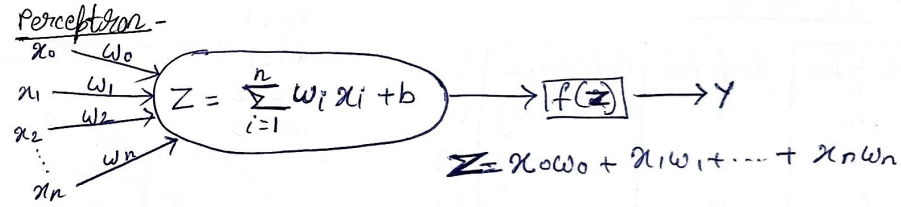
* If a neural network is regression then there will be only one neuron in output layer.

* If a neural network is classification, then there will be min of one neuron and max of 2 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 → 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 dendrites to receive input and the input processed in the nucleus to generate output which is passed through 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 a axon.

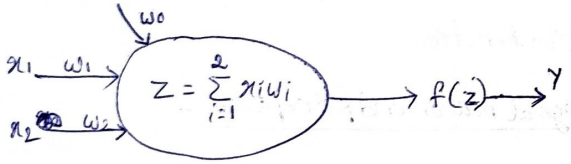
Biological Neuron	Perceptron
i) It has very complex architecture.	i) It is simple architecture
~~ii) The process hap~~	
ii) What processing task is done in nucleus is not known exactly.	ii) Here, we are performing the summation and followed by an activation function to get the output.

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)

$x_0 = 1$
$w_1 = 1$
$w_2 = 2$
$w_0 = 0$



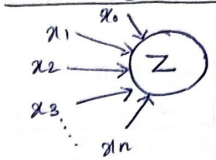
$$Z = x_0 w_0 + x_1 w_1 + x_2 w_2$$
$$= x_0 w_0 + 12 w_1 + 6 w_2 = 0 + 12 + 12 = 24$$

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

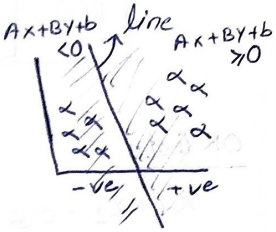
$f(24) = 1$ (as $z > 0$)

- Perceptron is a mathematical model or a mathematical function, which 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 -



$$Z = x_0 w_0 + x_1 w_1 + x_2 w_2 + \cdots + x_n w_n$$
$$Z = x_0 w_0 + \sum_{i=1}^{n} x_i w_i$$
$$\boxed{Z = \sum_{i=1}^{n} x_i w_i + b}$$ line eqn
$$\boxed{Z = XW + b}$$

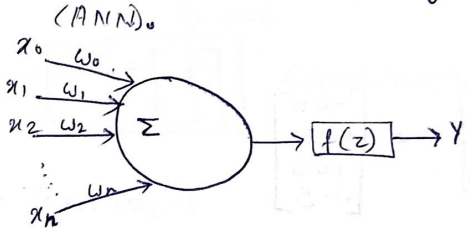


$$\frac{w_1}{A}\frac{x_1}{x} + \frac{w_2}{B}\frac{x_2}{y} + b$$
$$= Ax + By + b$$

- 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 it's 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.



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

$x_1 + x_2 - 1.5$
$x_0 w_0$ (b) $+ x_1 w_1 + x_2 w_2 = 0$

$w_1 = 1, w_2 = 1 \quad b = -1.5 = x_0 w_0$

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

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

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

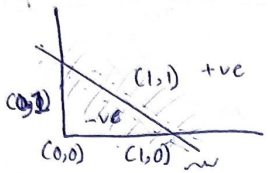
AND Gate

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

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

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

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



There can be ∞ no of lines

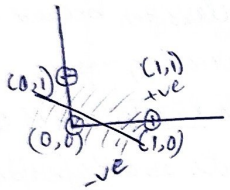
OR Gate

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

$w_1 = 1, w_2 = 1, b = -0.5$

$x_0 = 1$ (always)

$w_0 = -0.5$



TRUTH Table (OR)

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

$w = [-0.5 \quad 1 \quad 1]$

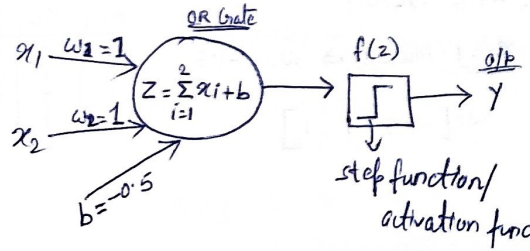
$$x = \begin{matrix} x_0 & x_1 & x_2 \end{matrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

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

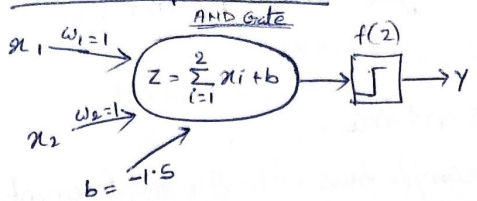
$$f(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

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

Architecture



Architecture of AND Gate



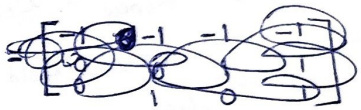
(Q) i) $x_1 + x_2 - 1$ for AND Gate

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

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

$$x = \begin{matrix} x_0 & x_1 & x_2 \end{matrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad w = [-1 \quad 1 \quad 1]$$

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



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

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

$$f(z) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

ii) $2x_1 + 2x_2 - 1$

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

$$x = \begin{matrix} x_0 & x_1 & x_2 \end{matrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad w = [-1 \quad 2 \quad 2]$$

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

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

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

linear activation function in backpropagation:

- If we apply the linear activation function, all layers will collapse to a single layer in multi layer perceptron.
- Sequentially the linear activation funct turns the neural network into just one layer and due to its limited power this doesn't allow the model to create complex mapping btw input and output.

Non-linear activation function -

i) Sigmoid activation function ii) tanh iii) Relu iv) Softmax

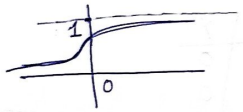
v) Leaky Relu vi) Parameter ReLU vii) Swish viii) GELU ix) SELU

Sigmoid activation function -

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

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

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



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)

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

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

$b = 0.1 + 0.1(1-0) = 0.1 + 0.1 = \underline{0.2}$

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

$f(0.3) = 1$ (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

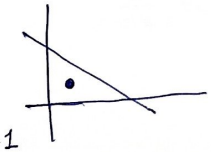
b) $x_1 = 0.5, x_2 = 0.5$

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

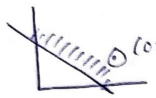
$f(-0.2) = 0$

$x_1 = 0.8, x_2 = 0.2$

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



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



$f(0.3) = 1$ (∵ 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 until 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

$x_0 = 0, x_1 = 0$

$z = 0 \times 0 + 0 \times 0 + 0 = 0$

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

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

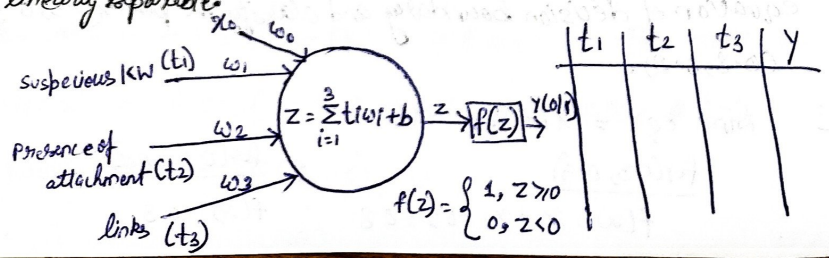
$w_{2new} = 0 + 1(1-0)0 = 0$

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

$x_0 = 0, x_1 = 0$

Q) You are using a perceptron to classify whether an email is spam or not spam. Input features are no. of suspicious 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:



t_1	t_2	t_3	y

Q) Define a perceptron and explain its role as a Binary classifier.
Q) Explain the purpose of weights and bias in perceptron.
Q) Why is a step activation function commonly used in a perceptron?
Q) What is linear separability and why it is important for a perceptron?
Q) Explain why a single layer perceptron can't solve the XOR problem?
Q) How does the choice of activation func affect the learning process of perceptron?

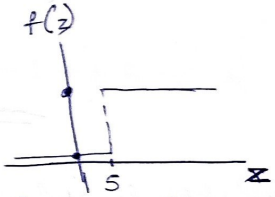
Activation function –
An activation function decides whether a neuron should be activated or not.

3 types –
i) Binary step function
ii) Linear activation function
iii) Non linear activation function.

Binary step function – Depends on a threshold value that decides whether a neuron should be activated or not.

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

z	$f(z)$	$\theta = 5$	y
0	$f(0)$	$\theta = 5$	0
1	$f(1)$	$\theta = 5$	0
4	$f(4)$	$\theta = 5$	0
14	$f(14)$	$\theta = 5$	1



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

Linear activation function –
– Also known as no activation or identity function is where the activation is proportional to input.

$f(x) = x$

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

Limitation – The derivative of the func is constant and has no relation to input "x". Therefore, we can't use the

2019

(b) 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: $z_1 = x_1 w_1 + x_2 w_2 + x_0 w_0$ (b)

$w_0 = 0$, $x_0 = 1$

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

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

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

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

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

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

$x_0 = 0, x_1 = 0$

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

$x_0 = 0, x_1 = 1$

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

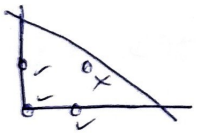
$x_0 = 1, x_1 = 0$

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

$x_0 = 1, x_1 = 1$

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

$= f(0.8) = 0$



updation

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

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

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

$x_0 = 0, x_1 = 0$

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

$x_0 = 0, x_1 = 1$

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

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

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

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

$x_0 = 0, x_1 = 0$

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

$x_0 = 0, x_1 = 1$

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

$x_0 = 1, x_1 = 0$

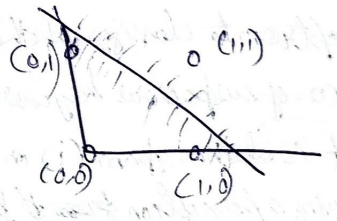
$z = 1 \times 1.2 + 0 \times 0.6 - 0.5 = 1.2 - 0.5 = 0.7 = f(0.7) = 0$

$x_0 = 1, x_1 = 1$

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

$= f(1.3) = 1$

$\therefore$ $1.2x_1 + 0.6x_2 - 0.5 = 0$



(b) 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 $(0.8, 0.2)$

$f(x) = 0.8$

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

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

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)

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

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, update weights and bias

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

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

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

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

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

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

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

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

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

$x_1 = 0, x_2 = 0$

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

$= 0.5$

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

$= 0.5$

$f(z_1) = f(0.5) = 0 \quad (Y = t)$

no updation

$x_1 = 0, x_2 = 1$

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

$f(1.6) = Y = 1 \quad (Y = t)$ No updation

$x_0 = 1, x_2 = 0$

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

$f(1.1) = 1 \quad (Y = t)$ ~~update~~

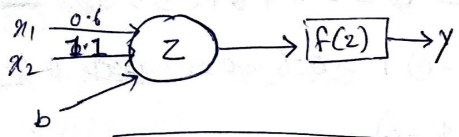
$x_1 = 1, x_2 = 1$

~~$z_2 = 1 \times 0.6 + 1 \times 0.5$~~

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

$= 2.2$

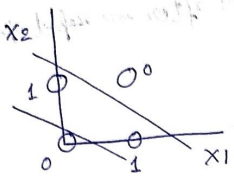
$f(2.2) = 1 \quad (Y = t)$.



$0.6x_1 + 1.1x_2 + 0.5$ Ans

XOR

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



$XOR = (x_1 + x_2) \cdot (\bar{x}_1 + \bar{x}_2)$

OR AND NAND

X_1	X_2	Y	(x_1+x_2)	$(\bar{x}_1+\bar{x}_2)$	$h_1 \cdot h_2$
0	0	0	0	1	0
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	1	0	0

$OR = x_1 + x_2 - 0.5$

$AND = x_1 + x_2 - 1.5$

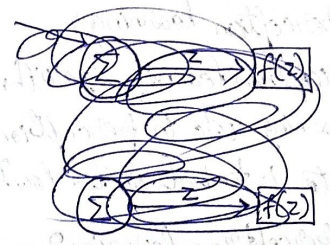
$NAND = -x_1 - x_2 + 1.5$ (NOT AND)

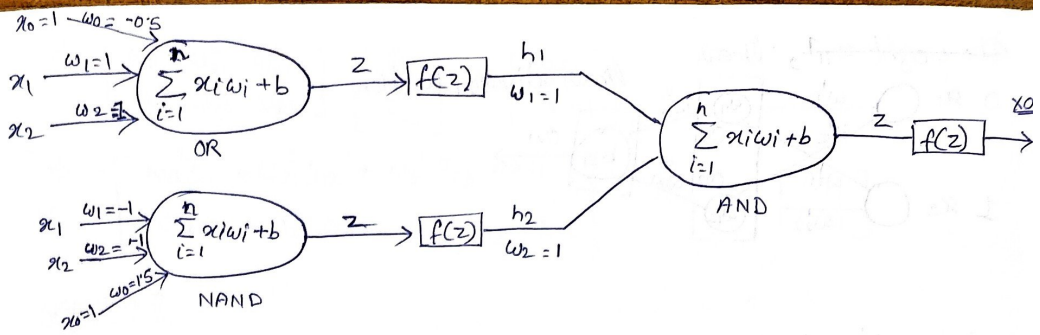
$$f(z) = \begin{cases} 1, z \geq 0 \\ 0, z < 0 \end{cases}$$

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

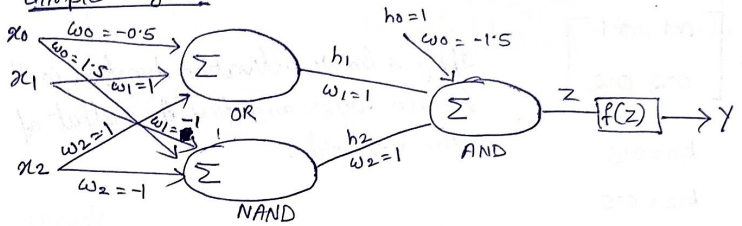
X W Z

Now, $\begin{bmatrix} x_0 & b_1 & h_2 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \end{bmatrix}$ $f(z) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$ = XOR



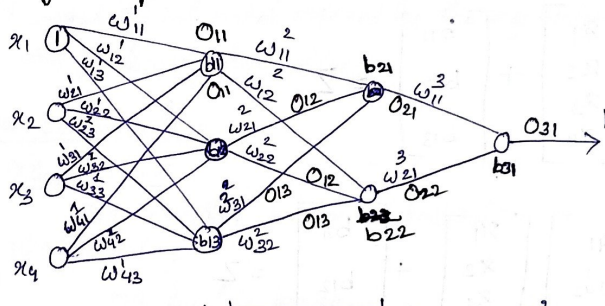


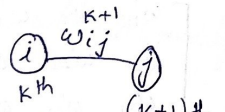
Simple diagram



MLP (Multilayered Perceptron)

Multilayer Perceptron –



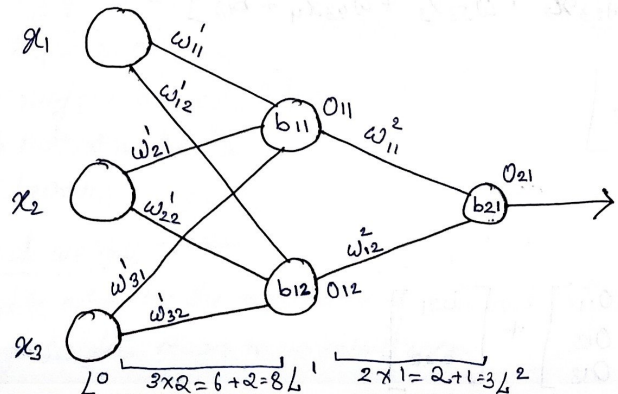


h_{ij} → i layer#, j node#

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 = 11

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

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 ~~epoch~~ 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 is 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 difference 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?

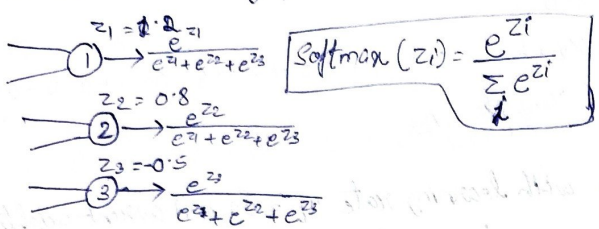
Q9) Under what condition does the perceptron learning algorithm converge?

Q10) What is the 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?

Q) ~~Find~~ 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.



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

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

$z_1 = 0.54,\ z_2 = 0.36,\ z_3 = 0.09$ (Probabilities)

$0.999 \approx 1$

upon adding

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.

$$z_i = \frac{e^{z_i}}{\sum e^{z_i}}$$

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

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

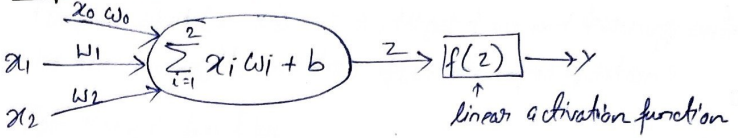
$$z_2 = \frac{e^{1001}}{2.188 \times 10^{435}} = 2.955 \times 10^{245}$$

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

Highest value is z_1. So sample belongs to class 1

6/10/25

Perceptron as a Regression -



$x_1 w_1 + x_2 w_2 + b = z$

$f(z) = z$

$f(x) = x$ → linear activation function

Perceptron training (Linear)

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

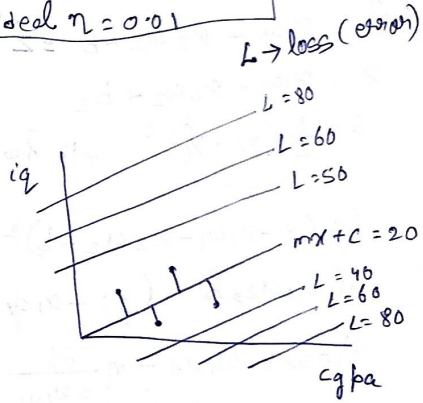
$w_{2new} = w_{2old} + \eta(t-y)x_2$

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

$\eta = 0$ 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

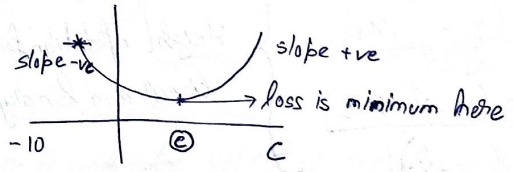
$\downarrow$

$\hat{y}_i = mx_i + c$

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

$L(c) = (y_i - 78.5x_i - c)^2$, $m = 78.5$ (calculated)

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



slope in -ve (value is increasing)

slope in +ve (value is decreasing)

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

$f(x) = x^2$

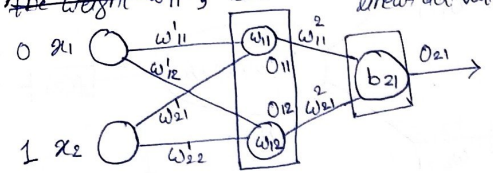
$\frac{d}{dx} f(x)\Big|_{x=2} = \frac{d}{dx} x^2$

$= 2x$

$= 4$

~~The weight w_{11}~~, Linear

Linear activation function



The weight

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

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

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

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

Layer #1 W

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

W^T

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

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

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

Layer #2 W

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

W^T

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

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

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 model~~ Artificial Neural 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.

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

v) Repeat the above steps iteratively over many epochs, until 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, output of 1 layer becomes input to the next layer.

Continue layer by layer until 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 func of the parameters of Machine learning parameter. If we adjust the parameters the loss function value changes.

Categories of loss function -

Regression	classification	Autoencoder	GAN
MSE	Primary cross entropy	KL Divergence	Discrimination 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 if the dataset is free of outliers.

② Mean Absolute Error (MAE) -

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

For layer 2

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

$$\Rightarrow \begin{bmatrix} 0.1 & 0.1 & 0.2 \\ 0.1 & 0.1 & 0.2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix} =$$

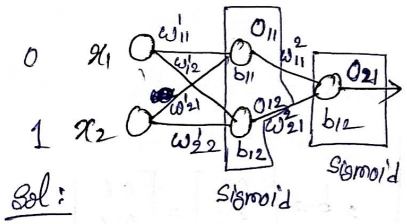
$$\Rightarrow \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \\ 0.2 & 0.2 \end{bmatrix}^T \begin{bmatrix} 0.22 \\ 0.23 \\ 0.24 \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 0.593 \\ 0.693 \end{bmatrix}$$

For layer 3

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

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

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

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

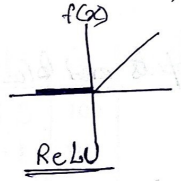
$$= 0.3963 + 0.6 = 0.99636$$

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

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 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.

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 until 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 ~~perceptron~~ perceptron stops updating its weights and bias bc 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 ~~oscillation~~ oscillation and failure to converge

Too low

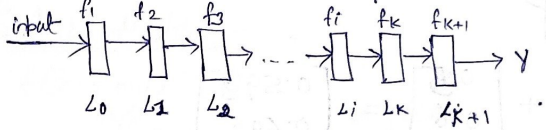
- Learning becomes slow.
- May take long time to converge.

11) Stopping condition for perceptron learning are -

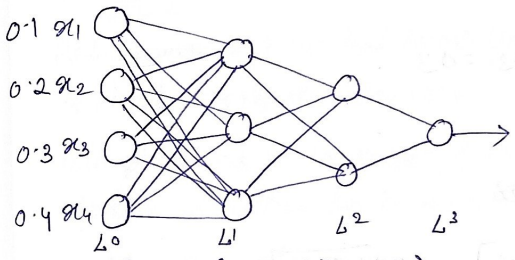
i) All training examples are correctly classified.
ii) Max no. of iterations (epochs) reached.

13/10

Neural network is a cascade of non-linear function



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



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

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

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

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

$a[1]$, $a[2]$, $a[3]$

For 1st layer

12 weights, 3 bias

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

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

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

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 is robust to outliers.

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

Disadvantages

1. It is not differentiable.

③ Huber loss → acts as MSE

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

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

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

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

25/10

Q1) What's 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 their pros and cons?

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

Q5) What's 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

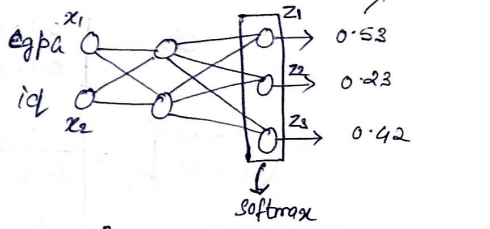
$$L = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

K = no. of classes

y_j → true class

$\hat{y}_j$ → predicted class

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



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

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

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

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

Sol: a) Layer #1

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

W X z

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

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

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

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

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

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

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

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

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

1/10

Training the MLP -
- Forward propagation
- Back propagation

Neural network performance enhancement -

i) Tuning the hyper parameters
a) Hidden layer b) Batch
c) No. of neurons d) Learning rate
e) Activation function
f) Optimizers
g) Epochs

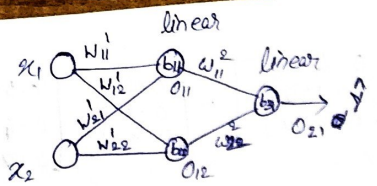
ii) Solving neural network problems
a) Vanishing gradient
b) ~~Verifying problem~~
b) overfitting problem.

Variants of Gradient descent -
i) Batch GD
ii) Stochastic GD
iii) Mini Batch GD

Batch GD

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

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



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

9 learnable parameters are there in the above diagram

Minimize the loss

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

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

$z = \hat{y} = o_{11}w_{11}^2 + o_{12}w_{21}^2 + b_{21}$

$\hat{y} = \underbrace{[x_1w_{11}^1 + x_2w_{21}^1 + b_{11}]}_{o_{11}} w_{11}^2 + [x_1w_{12}^1 + x_2w_{22}^1 + b_{12}]w_{21}^2 + b_{21}$

1st

$w_{11}^2 \text{new} = w_{11}^2 \text{old} - \eta \frac{\partial L}{\partial w_{11}^2} = 2(\hat{y}-y)o_{11}$

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

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

$w_{11}^1 \text{new} = w_{11}^1\text{old} - \eta \frac{\partial L}{\partial w_{11}^1}$

$w_{12}^1 \text{new} = w_{12}^1\text{old} - \eta \frac{\partial L}{\partial w_{12}^1}$

$w_{21}^1 \text{new} = w_{21}^1\text{old} - \eta \frac{\partial L}{\partial w_{12}^1}$

$w_{22}^1 \text{new} = w_{22}^1\text{old} - \eta \frac{\partial L}{\partial w_{22}^1}$

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

$b_{12}\text{new} = b_{12}\text{old} - \eta \frac{\partial L}{\partial b_{12}}$

$\frac{\partial L}{\partial w_{11}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{11}^2}$ ($2(\hat{y}-y)$, 1, o_{11})

$\frac{\partial L}{\partial w_{21}^2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_{21}^2}$ ($2(\hat{y}-y)$, 1, o_{12})

$\frac{\partial L}{\partial b_{21}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial b_{21}}$ ($2(\hat{y}-y)$, 1, 1)

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

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

$\frac{\partial z}{\partial w_{11}^2} = o_{11}$

2nd

$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f} \cdot \frac{\partial z_f}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{11}^1} = 2(\hat{y}-y)\, w_{11}^2\, x_1$ ($2(\hat{y}-y)$, 1, w_{11}^2, 1 (as linear activation function), x_1)

$L \to \hat{y} \to z_f \to o_{11} \to z_{11} \to w_{11}^1$

$\frac{\partial L}{\partial w_{21}^1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f} \cdot \frac{\partial z_f}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial w_{21}^1} = 2(\hat{y}-y)\, w_{21}^2\, x_2$ (x_2)

$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_f} \cdot \frac{\partial z_f}{\partial o_{11}} \cdot \frac{\partial o_{11}}{\partial z_{11}} \cdot \frac{\partial z_{11}}{\partial b_{11}} = 2(\hat{y}-y)\, w_{11}^2$ (1)

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 from the following information

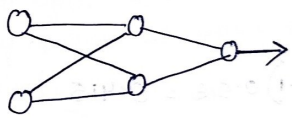
$x = \begin{bmatrix} 0.6 \\ 0.9 \end{bmatrix}$, $y_{\text{true}} = 1$, $\eta = 0.1$

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

$w^{[2]} = [0.3 \quad -0.5]$, $b^{[2]} = 0.05$

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

Sol:



* If a MLP solves a regression problem the output layer should contain only 1 neuron and on that neuron linear activation function 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 MSE, MAE, huber loss.
* " " Binary function " " " binary cross entropy.

$\Rightarrow L = -1\log(0.53) - 0\log(0.23) - 0\log(0.42)$

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

6) Design a MLP with the following information.

input(2) → hidden(2) → output(3)
(sigmoid) (softmax)

a) Find the predicted value using forward propagation

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

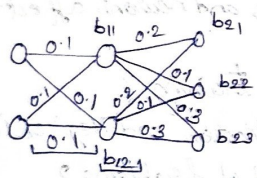
$w^{[2]} = \begin{bmatrix} 0.2 & 0.2 \\ 0.1 & 0.1 \\ 0.3 & 0.3 \end{bmatrix}$ $b^{[2]} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$

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

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

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

Sol:



a) Layer #1

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

$\Rightarrow \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}\begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.1\times0.2 + 0.1\times0.3 \\ 0.1\times0.2 + 0.1\times0.3 \end{bmatrix} = \begin{bmatrix} 0.05 \\ 0.05 \end{bmatrix} \begin{matrix} z_0 \\ z_1 \end{matrix}$

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

Layer #2

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

w O b

Softmax

$z_1 = \frac{e^{z_1}}{e^{z_1}+e^{z_2}+e^{z_3}}$, $z_2 = \frac{e^{z_2}}{e^{z_1}+e^{z_2}+e^{z_3}}$ & $z_3 = \frac{e^{z_3}}{e^{z_1}+e^{z_2}+e^{z_3}}$

$= 0.298$ $= 0.297$ $= 0.403$

class of the sample is 3.

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

$L = -Y_1\log(\hat{Y}_1) - Y_2\log(\hat{Y}_2) - Y_3\log(\hat{Y}_3)$

$= -0.298\log($

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

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

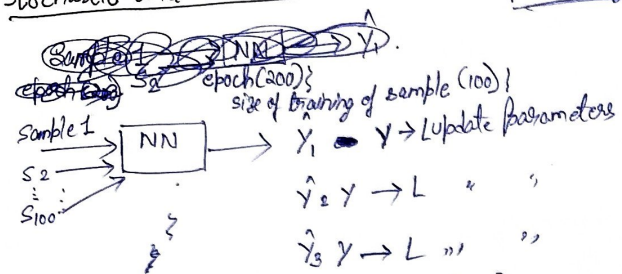
- Given an artificial neural network and an error function the methods calculate the gradient of the error function with respect to neural network, weights, and biases.
- Backpropagation is an algorithm to train a neural network and training a neural network means for a given data it finds the optimum values of weights and biases.

```
for i in range(epochs):
    for j in range(x.shape[0])
        select row(random)
        predict using forward propagation
        calculate loss (using a suitable loss function)
        update weights and bias using GD (Gradient descent)
            Wn = Wo - η ∂L/∂Wo
Calculate the average loss.
```

So, no. of updates = no. of epochs (Batch GD)

Stochastic Gradient Descent –

100 training Data



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

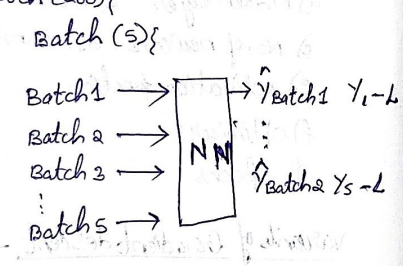
Mini Batch Gradient Descent –

100 training samples

into no. of mini batches

batch size = 20

no. of batches = 5



∴ 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 –

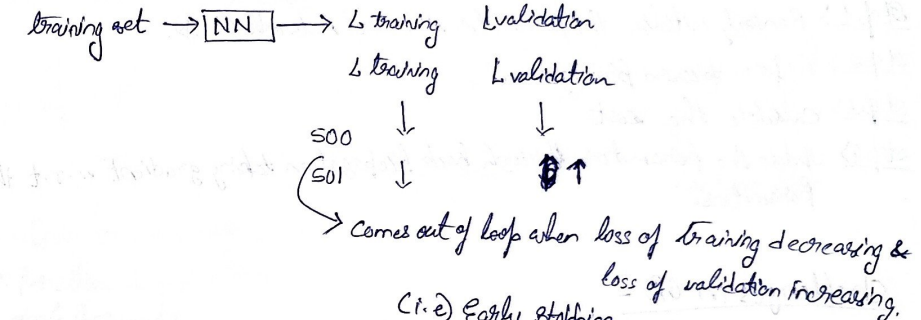
i) Early stopping
ii) Dropout
iii) Regularization
iv) Data augmentation

Early stopping –

Before testing the model splitting ratio is –

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

1000{



(i.e) Early stopping.

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

No. of neurons

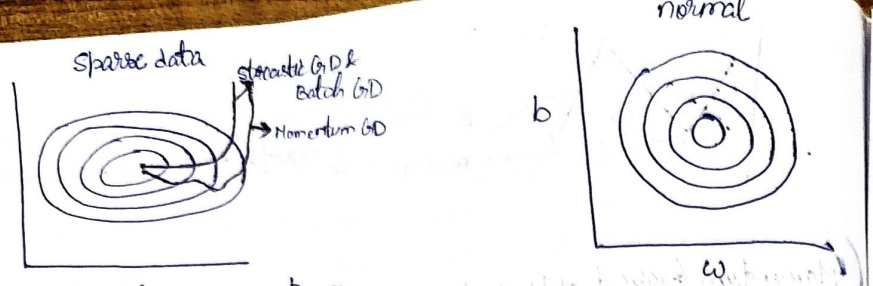
n_h (no. of hidden layer) $= \frac{n_{input} + n_{out}}{2}$ (linear)

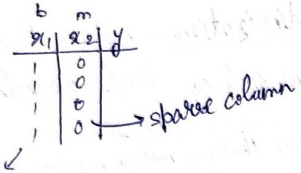
* hidden layer activation function → sigmoid / tanh / ReLU

Vanishing gradient problem –

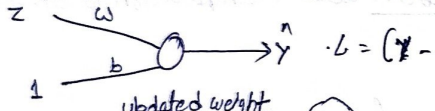
If we have a lot of hidden layer (eg: 30, 40) it is known as deep neural network.

- 0.1×0.1×0.1×0.1 = 0.00001 ≈ 0 (vanishing)
- ReLU activation fun is used to avoid vanishing gradient problem.
- If η is high it will keep oscillating in a the range





- In elongated 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 in other axis. In this case our normal optimization algo doesn't solve well, to solve this problem the adagraph algo is used.



updated weight

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

$$b_t = b_{t-1} - \eta \frac{\partial L}{\partial b_{t-1}}$$

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

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

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

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

- ~~$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla w_t$~~ ~~$v_t = v_{t-1} + (\nabla w_t)^2$~~

- $w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla w_t$ (learning rate factor)

$v_t = v_{t-1} + (\nabla w_t)^2$

↓ current velocity, ↘ old velocity, ↘ (Gradient)^2

$b_{t+1} = b_t - \frac{\eta}{\sqrt{v_t + \epsilon}}$ → it will become small; → if it is very large

$v_t = v_{t-1} + (\nabla b_t)^2$

→ as it is not sparse 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 reach 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.

- By calculating ~~expo~~ 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 sudden trends in time series data.

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

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).

Eg:

	temb
θ_1	25
θ_2	13
θ_3	17
θ_4	23
θ_5	5

$V_0 = 0$

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

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

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

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

$V_0 = 0$

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

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

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

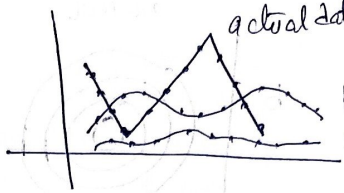
$= \beta^2(1-\beta)\theta_1 + \beta(1-\beta)\theta_2 + (1-\beta)\theta_3$

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

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

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

=



} Graph becoming more smooth.

Momentum based optimization – 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

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

(βV_{t-1} : momentum)

Adagrad optimization

Adaptive gradient optimization –

- 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 where adagrad works well in comparision to other optimization techniques

case I : The i/p feature scales are different.

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

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

cgpa	iq	IIT	Package
–	–	1	
		0	
		?	

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

w_{11}^{1} new $= w_{11}^{1}$ old $- \eta \frac{\partial L}{\partial w_{11}^{1}}$ → 0 minimize the loss

$\frac{\partial L}{\partial w_{11}^{1}} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_8} \times \frac{\partial z_8}{\partial o_{11}} \times \frac{\partial o_{11}}{\partial z_{11}} \times \frac{\partial z_{11}}{\partial w_{11}}$ → 0 (derivation of sigmoid activation function or tanh activation function).

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

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).

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

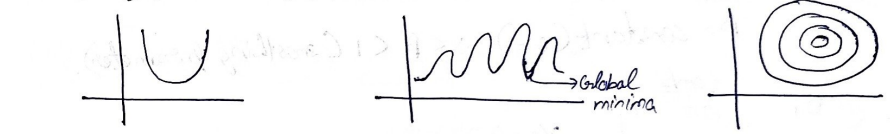
ii) So, to overcome this particular problem, we can use learning rate scheduling.
- Can't be applied to all neural network

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 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) Saddle point - 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 nor a local maxima.

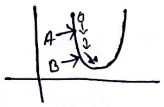
- To remove this problem we can use some other optimizers -
  i) Momentum based optimizer
  ii) Adagrad
  iii) RMS prop
  iv) Adam



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

non convex

Contour plot



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 means have to increase the learning rate but this is not possible in our all three Gradient Descents.
(Batch, shocastic, mini Batch). based GD.
Therefore, η should be adaptive.

Therefore, learning

Exponentially weighted moving average -

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 build cool optimizers.