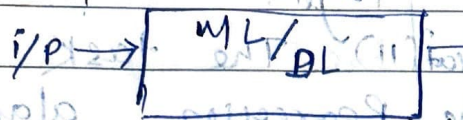
Text Book - Nikhil Buduma, "Fundamental of deep learning : Designing next-generation Machine learning algorithm", O'Reilly, 2017

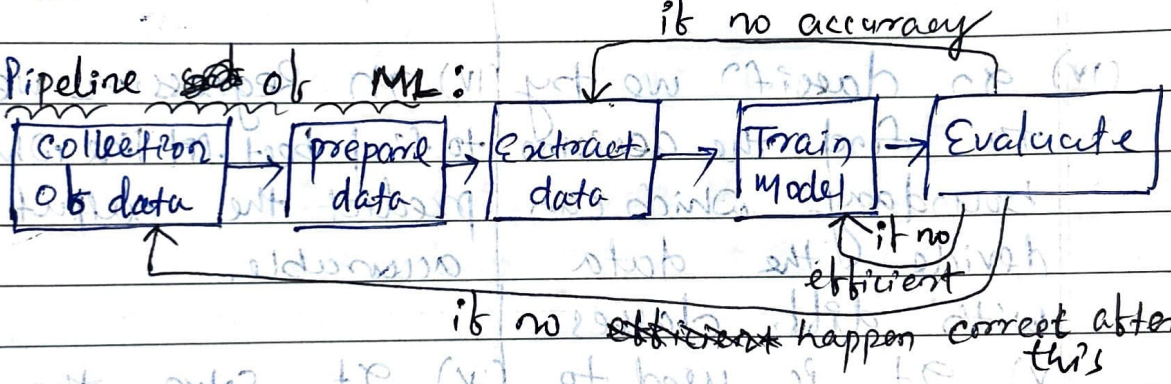


Feature — independent

Target/class

	Sepal length	Sepal width	Petal length	Petal width	Type/class ← dependent
Sample/ Observation	71	3.0	6.9	5.8	Virginica
	50	3.5	6.0	9.8	Setosa

Pipeline of ML :



Datatypes → structured data
→ unstructured data

Types of ML :-

(a) unsupervised Learning :-
- > clustering
- > Dimensionality Reduction

(b) Reforcement learning :-
- > Agent need a environment to make a decision.

(c) Supervised learning :-
- > classification
- > Regression

Q) Diff. bet^n classification and Regression.

classification	Regression
(i) In classification, the output variable must be discreate using activation function like Sigmod, Softmax.	(i) In Regression the output variable must be continuous nature or near value, using a linear activation function in the output layer.
(ii) The task of classif^n algot^m is to map the input value x, with the discreate output variable ..	(ii) The task where the Regression algorithm is to map the input value with the continuous output variable
(iii) Classif^n algot^m are use discrete data.	(iii) Regression algorithm are used continuous data
(iv) In classif^n we try to find the decision boundary which can devide the data with diff. classes	(iv) In Regression we trying to find best fit line which can predict the output more accurable.
(v) It is used to solve the classification problem such as identification of spam emails, Speech Recognisation.	(v) It solve the Regression problem, e.g weather prediction, house price pred^n etc.
(vi) classification algorithm can be devided into binary and multinomy classifier.	(vi) Regression algorithm further devided into linear and non-linar Regression

Metrics for classification :-

True positive : Target value: Yes, Predict value: Yes

True Negetive : NO / NO

False possitive : ~~Yes~~ NO / ~~Yes~~ yes

False Negetive : ~~yes~~ / ~~NO~~

Accurracy : $= \frac{\text{No. of correct prediction}}{\text{Total no. of input sample}}$

$= \frac{TP + TN}{TP + TN + FP + FN}$

Precision $= \frac{TP}{TP + FP}$

Recall / Sensitivity / (TPR) True possitive Rate $= \frac{TP}{TP + FN}$

F1-Score $= 2 \cdot \left[\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}\right]$

(Harmonic Mean bet^n the precision and Recall)

Confusion matrix :

Actual class		+ve predicted class	-ve predicted class
	+ve	TP	FN
	-ve	FP	TN

Specificity / TNR $= \frac{TN}{TN + FP}$

$FPR = \frac{FP}{FP + TN} = 1 - TNR$

ROC curve (Recurcive operating characteristics) :
Plot using TPR vs FPR

	sepal length	sepal width	prediction	Target
1	0.192	2.085	Setosa	setosa
2	1.132	0.726	Virginica	virsicolor
3	-0.9598	2.1738	setosa	setosa
4	2.952	2.138	virginica	virsicolor
5	-0.5055	-2.1499	virsicolor	virginica
6	0.8187	0.6222	Virginica	virginica
7	-0.958	-2.1703	virsicolor	virsicolor
8	0.8777	0.0536	virginica	virginica
9	-4.388	-0.239	versicolor	setosa
10	-1.419	-0.6867	Setosa	Setosa

overall classification :

$$\text{Accuracy} = \frac{6}{10} = 0.6$$

For a particular class → setosa : TP = 3, TN = 6
FP = 0, FN = 1

$$\text{Accuracy} = \frac{TP+TN}{10} = \frac{9}{10} = 0.9$$

$$\text{precision} = \frac{TP}{TP+FP} = \frac{3}{3} = 1$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{3}{3+1} = \frac{3}{4} = 0.7$$

$$\text{F1-score} = 2 \times \left[\frac{1 \times 3/4}{1 + \frac{3}{4}}\right]$$

$$= 2 \times \frac{\frac{3}{4}}{\frac{4+3}{4}} = 2 \times \frac{3}{4} \times \frac{4}{7} = \frac{6}{7}$$

For a class virsicolor =
TP = 1, TN = 5
FP = 2 FN = 2
For a class virginica =
TP = 2 TN = 5
FP = 2 FN = 1

versicolor : Accuracy = $\frac{6}{10}$

Precision = $\frac{1}{3}$

Recall = $\frac{1}{3}$

F1 score = $2 \times \frac{1}{3} \times \frac{1}{3} = \frac{1}{3}$

verginica : Accuracy = $\frac{7}{10}$

Precision = $\frac{2}{4}$

Recall = $\frac{2}{3}$

F1-score = $2 \times \frac{\frac{2}{4} \times \frac{2}{3}}{\frac{6+8}{12}}$

$= 2 \times \frac{1/2}{1} \times \frac{12}{14} = 4/7$

Regression : (for metrics)

(i) Mean absolute error (MAE) = $\frac{1}{n}\sum_{i=1}^{n} |y_{actual} - y_{predicted}|$

(ii) Mean squared error (MSE) = $\frac{1}{n}\sum_{i=1}^{n} (y_{actual} - y_{predicted})^2$

(iii) Root means square error (RMSE) = $\sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_{actual} - y_{predicted})^2}$

→ You need to predict how close you are to the prediction., for this use MAE
→ for DL and ML, for large data set use RMSE.
→ for standard error/less error, use MSE.

Q. Actual Value = (3, 5, 7)
predicted value = (2, 5, 8)

$$MAE = \frac{|3-2| + |5-5| + |7-8|}{3}$$

$$= \frac{1+0+1}{3} = \frac{2}{3}$$

$$MSE = \frac{(3-2)^2 + (5-5)^2 + (7-8)^2}{3}$$

$$= \frac{1+1}{3} = \frac{2}{3}$$

$$RMSE = \sqrt{MSE} \cdot \sqrt{\frac{2}{3}}$$

Actual Value = (5, 6, 9)
predicted value = (3, 5, 7)

$$MAE = \frac{|5-3| + |6-5| + |9-7|}{3}$$

$$= \frac{2+1+2}{3} = \frac{5}{3}$$

$$MSE = \frac{(5-3)^2 + (6-5)^2 + (9-7)^2}{3} = \frac{4+1+4}{3} = \frac{9}{3} = 3$$

$$RMSE = \sqrt{MSE} = \sqrt{3}$$

Logistic loss or cross entropy: find entropy for always 2 value,

$y \in \underbrace{\{0, 1\}}_{\hookrightarrow 2 \text{ value.}}$, $p = Pr(y=1)$

* entropy ↑ for always 2 value,

is more than 2 → $\text{logloss}_{N=2} = -(y \log(p) + (1-y) \log(1-p))$

$y=1$ $\quad$ $y=0$

for $N > 2$,

$$\text{logloss} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log(p) + (1-y_i) \log(1-p))$$

→ It is use of Multiclass, in order to capture the model confidence.

● Advance of ML:
→ Automation of Repeatative task
→ Improved Decision making
→ Pattern Recognisation
→ Continuous Improvement
→ Scalability
→ Cost efficiency
→ Innovation enablement

● Disadvantage of ML:-
→ Data dependency
→ High computational cost
→ Complexity and interpretability.
→ Risk of bias
→ Job Replacement

Q. Find the null set of matrix $A = \begin{bmatrix} 2 & 3 \\ 6 & 9 \end{bmatrix}$

Q. Find the basis of the column space of the matrix.

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

Q. Find the eigen values and corresponding eigen vector of the matrix, $B = \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix}$

Q. Explain the significance of the eigen values and eigen vector in the data science and eigen vector in the data science application?

Q. Find the eigenvalue and corresponding eigen vector of the matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix}$ and

(a)

(b) $C = \begin{bmatrix} 1 & 1 & -2 \\ -1 & 2 & 1 \\ 0 & 1 & -1 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & 2 & 2 \\ 0 & 2 & 1 \\ -1 & 2 & 2 \end{bmatrix}$

Answer:

(Q1) Given matrix $= \begin{bmatrix} 2 & 3 \\ 6 & 9 \end{bmatrix}_{2\times 2}$

take, $Ax = 0$

$$\begin{bmatrix} 2 & 3 \\ 6 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 2x_1 + 3x_2 \\ 6x_1 + 9x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$2x_1 + 3x_2 = 0 \times 3 \Rightarrow 6x_1 + 9x_2 = 0$

$6x_1 + 9x_2 = 0 \Rightarrow 6x_1 + 9x_2 = 0$

$6x_1 + 9x_2 = 0$

$0x_2 = -6x_1 - 9x_2$

$2x_1 + 3x_2 = 0$

$2x_1 = -3x_2 \Rightarrow x_1 = \frac{-3x_2}{2}$

$\Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$

(3) Given matrix $= \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix}$

we know $\Rightarrow |A - \lambda I| = 0$

$$\Rightarrow \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = 0$$

$$\Rightarrow \left| \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right| = 0$$

$$\Rightarrow \left| \begin{bmatrix} 1-\lambda & 2 \\ 5 & 4-\lambda \end{bmatrix} \right| = 0 \quad \text{(i)}$$

$\Rightarrow (1-\lambda)(4-\lambda) - 10 = 0$

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

$\Rightarrow \lambda^2 - 5\lambda - 6 = 0$

$\Rightarrow \lambda^2 - (6-1)\lambda - 6 = 0$

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

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

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

$\Rightarrow \lambda = 6, \lambda = -1$

For $\lambda = 6$, put in eq? (i)

$$\begin{bmatrix} -5 & 2 \\ 5 & -2 \end{bmatrix} = 0$$

$(A - \lambda I)x = 0$

$$\Rightarrow \begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} x = 0$$

$$\Rightarrow \begin{bmatrix} -5 & 2 \\ 5 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$\Rightarrow -5x_1 + 2x_2 = 0 \Rightarrow -5x_1 = -2x_2$

$5x_1 - 2x_2 = 0$

$$\Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

For $\lambda_2 \rightarrow$

$$(A - \lambda I)x = 0$$

$$\Rightarrow \left(\begin{bmatrix} 1 & 2 \\ 5 & 4 \end{bmatrix} - \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \right) x = 0$$

$$\Rightarrow \begin{bmatrix} 2 & 2 \\ 5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\Rightarrow 2x_1 + 2x_2 = 0$$

$$\Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

(4) → Eigenvalues represent the magnitude of variance in the data direction.

→ Eigenvector represent the direction of maximum variance.

→ In PCA eigenvectors = principal components, eigenvalues = importance of those components.

(5) Given matrix, $A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix}$

We know, $|A - \lambda I| = 0$

$$\Rightarrow \left| \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} \right| = 0$$

$$\Rightarrow \begin{vmatrix} 1-\lambda & 2 & 3 \\ 0 & 2-\lambda & 3 \\ 0 & 0 & 3-\lambda \end{vmatrix} = 0$$

$$\Rightarrow (3-\lambda)(2-\lambda)(1-\lambda) = 0$$

$$\Rightarrow (3-\lambda)(2 - 2\lambda - \lambda + \lambda^2) = 0$$

$$\Rightarrow (3-\lambda)(\lambda^2 - 3\lambda + 2) = 0$$

$$\Rightarrow 3 - \lambda = 0 \;,\; \lambda^2 - 3\lambda + 2 = 0$$

$$\Rightarrow \lambda = 3, \quad \lambda^2 - (2+1)\lambda + 2 = 0$$

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

$$\Rightarrow \lambda(\lambda - 2) - 1(\lambda - 2) = 0$$

$$\Rightarrow (\lambda - 2)(\lambda - 1) = 0$$

$$\Rightarrow \lambda = 2, \lambda = 1$$

for $\lambda = 1$,

$$\begin{vmatrix} 0 & 2 & 3 \\ 0 & 1 & 3 \\ 0 & 0 & 3 \end{vmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$$\Rightarrow 0x_1 + 0x_2 + 2x_3 = 0$$

$$\Rightarrow 2x_3 = 0$$

$$x_3 = -2$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix}$$

Q. A fair coin has probability $P(H) = 0.5$ and $P(T) = 0.5$, find its entropy $H(X)$?

Ans: For a fair coin

$$H(X) = -(0.5 \log_2(0.5) + 0.5 \log_2(0.5))$$

(entropy)

=

For a baised coin,

$$P(H) = 0.8, \quad P(T) = 0.2$$

$$H(X) = -(0.8 \log_2 0.8 + 0.2 \log_2 (0.2))$$

For single value

c.

Date ____ Page ____

Q: ~~Refer~~ For two distribution how to find cross entropy ?

for multiple value:

$P = (0.7, 0.3)$, $Q = (0.6, 0.4)$ — (y_1, y_2), (P_1, P_2)

$H(P,Q) = -(0.7\log_2(0.6) + (0.3)\log_2(0.4))$ (cross entropy)

→ why 2 ⇒ ~~value is 2 P = (0.7, 0.3)~~ 2 value

KL-Divergence $(D_{KL}) = H(P,Q) - H(P)$

$H(P) = -(0.7\log_2(0.7) + 0.3\log_2(0.3))$

→ $H(P,Q) - H(P) = $ -do-

Binary cross Entropy :-

Single example, $y = 1$, $\hat{y} = 0.8$

$H(P) = -(y\log_2\hat{y} + (1-y)\log_2(1-\hat{y}))$

Batch of 3 :- $y = (1, 0, 1)$, $\hat{y} = (0.9, 0.25, 0.6)$

$H_1(P) = -\log_2 0.9 + 0$

$H_2(P) = 0 - \log_2(0.75)$

$H_3(P) = +\log_2(0.6)$

$H(P) = \dfrac{H_1(P) + H_2(P) + H_3(P)}{3}$

Date ____ Page ____

Q) Suppose we have a classification problem with 3 classes x, y, z, given that for the probability distribution 'P', $P(x) = 0.7$, $P(y) = 0.2$, $P(z) = 0.1$, calculate the entropy of given probability distribution.

Q) Suppose you have two probability distribution function P and Q, also $P(x) = 0.7$, $P(y) = 0.2$, $P(z) = 0.1$, and $Q(x) = 0.6$, $Q(y) = 0.3$, $Q(z) = 0.1$. Find the cross entropy $CE(P\|Q)$.

Answer :-

(1) $H(P) = -(0.7\log_2(0.7) + 0.2\log_2(0.2) + 0.1\log_2(0.1))$

$= 1.156$

(2) $H(P,Q) = -(0.7\log_2(0.6) + 0.2\log_2(0.3) + 0.1\log_2(0.1))$

$= 1.189$

Defination of Deep learning :-

→ Deep learning is the type of machine learning that uses Artificial neural network that multiple layers to learn and make decision.

→ It replicates just like a human brain as all the neural network connected in the brain which exactly is the concept of deep learning.

→ Deep NN that holds various layers of interconnected nodes elastated with this hirarchical pattern and data feature.

→ Automated learning system, that enhances the data without the intervention of manual feature.

→ Succesfull in various fields such as image recognisation, natural language processing, speech recognisation and recommendation system.

Diff. bet? ML and DL in

Machine learning	Deep learning
→ ML is a study that uses statistical methods enabling the machine to improve with experience.	→ DL is the study that makes use of neural network to imitates the functionalities just like a human brain.
→ ML is the subset of AI	→ DL is the subset of ML.
→ ML allows the system to learn from the data	→ DL is the deep neural networks to analyze the data. and provides the outputs accordingly.
→ The aim is to increase accuracy to caring much about the Ratio	→ DL attempts highest ranks in terms of accuracy when it is trained with large amount of data.
→ If you have the clear idea about the logic involved in behind and u can visualize the complex functionalities, then it defines the ML aspect.	→ In DL you have clear idea about the logic involved in it, but I dont have idea about the feature, so you break the complex functionlities into low dimensional features by adding more layer then it defines the DL aspects.
→ Types of ML → SL, USL, and Reinforcement learning.	→ Types of DL → unsupervised pre-trained network → CNN, RNN (Recurrent) and Recursive NN.
→ ML is less effective than deep learning. as it can not work for longer dimension of higher amount of datas.	→ DL is powerful than ML as it can easily work larger sets of data.
→ ML is a subset of AI, that focuses on developing algorithm that can learn from the data and improve the performance over time without being explicitly programd.	→ DL is the subset of ML that focuses on developing DNN, that can automatically learn extract features from the data.
→ Ex: Virtual personal Assistant, Siri, Alexa, google etc., email spam.	Ex: Centiment based news aggregation, image analysis, caption generation etc.

Importance of Deep learning :-

→ Analysis of large and deep can manage to recognisation and processing where learning algorithms Limitation are surpless.

→ Medical diagnosis and fraud detection are possible because of the high accuracy in identifying the complex and relation-ship in datais.

→ The implementation of deep learning in business, organisation and various sector is visible as it is affordable.

→ deep learning techniques provent other fields like science and technology to understand how the neural how the neural network function and the area of improvement.

Different types of Deep learning :-

→ CNN (Convolutional Neural network)

→ RNN (Recurrent Neural network)

→ DRN (Deep Reinforcement network)

Advantages of deep learning :-

→ High accuracy

→ Automated feature engineering

→ Scalability

→ Flexibility

→ control improvement

Deep learning :-

dataset is easy.

learning algorithm improve image

natural language

the traditionally

Challenges of DL :- (Advantages)

→ Data availability

→ computational Resources

→ Time consuming

→ overfitting and underfitting

→ Bias

→ Hyper Parameter tuning

*

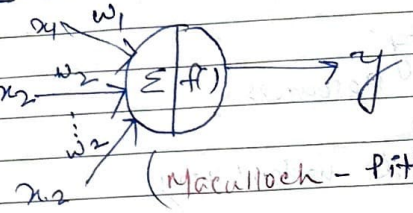
→ Neuron function involves sending electrical impulses and chemical signal to and from the brain.

→ Neuron completes this task using a process called action potential.

→ Action potential is the quick flow of the electrical voltage from the neuron down the axon.

→ This voltage allows a information to be transmitted from neurone to neuron all over the body.

ANN :



(Macculloch - Pitts Neuron)

Macculloch - Pitts Neuron works as follows :-

(i) → The inputs are multiplied by the corresponding weights, the weighted sum of inputs are sumed up.

→ If the sum value is greater than or equal to threshold value then the neuron fiers and the output is '1', otherwise the neuron does not fire and output is zero.

Characteristic of ANN :-

→ A neuron is the mathematical function model on the working of biological neuron.

→ It is an elementary unit in a artificial neural network.

→ one or more inputs are separately weighted.

→ Inputs are sumed of and passed through an non-linear function to produced the outputs.

→ Each neuron holds an internal state, called activation signal.

→ Each connection link is carrying information about the input signal.

→ Every neuron is connected to another neuron to the connection link.

Diff. between biological and Artificial Neural network :

Biological NN	ANN
→ Structure is Dendrites → cell body → Axon → Axon terminal	→ Structure is input → hidden layer → o/p
~~→ Memory can happens in~~	
→ Learning can handles ambigeous noise input.	→ Learning can handle
→ The speed of processing information is slow.	→ Speed of processing is faster compared to biological NN.
→ Storage allocation to new processes can be done easily by adjusting the the interconnection strength.	→ Storage allocation to a new process is not possible as each location is dedicated to a specified process.
→ Massive parallal operation can be conducted simultaneously.	→ only sequential operation can be conducted.
→ Information can be retrived from the sub nodes even its is currupted.	→ once currupted the information can not be retrived.
→ The information is being transmitted and process into the network is not monitored.	→ A control unit monitor of all information and activities on the network.

→ our brain contains about 86 billion and more than 100 synaps.	→ The no. of neuron in artificial neuron is much less than biological neural network.
→ The brain works asynchronosly.	→ ANN works synchronously.
→ Brain have complicated topology.	→ ANN are often in a tree structure.
→ BNN can learn completely new task.	→ ANN are specialized and they can perform one task.

I/P value, $X = [x_1, x_2 \dots x_n]$

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

Advantages of ANN :-

→ Pattern Recognisation
→ Parallel processing
→ Adaptability
→ Generalisation.

Disadvantage of ANN :-

→ large data and computation
→ overfitting
→ Hyperparameter tuning
→ Slow convergence
→ Time consuming
→ Limited data efficiency
→ Categorical data handling

Perceptron :-

→ proposed by Frank Rosenblatt
→ 1957

Q) What is the perceptron ?
Q) What are the components of a perceptron ?
Q) What is meant by the data is linearly separable
Q) can perceptron handle non linearly separable data ?
Q) What are the limitation of perceptron ?

(1) A perceptron is the simplest type of artificial neural network used for binary classification, it takes multiple inputs applies weights sums them, passes the result through an activation function and outputs.

(2) Components of perceptron :-
(i) input layer
(ii) Weight
(iii) Activation function
(iv) output

(3) Data is linearly separable if a ~~Straight~~ Straight line, plane, hyperplane can separate the data points of different classes without error.

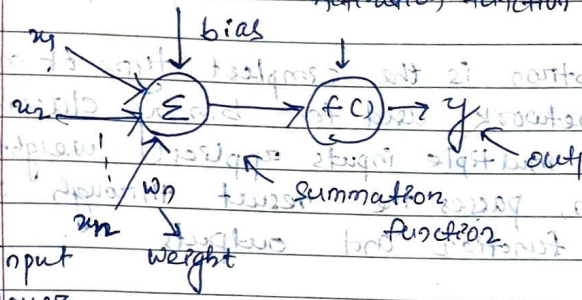
(4) No, a simple perceptron cannot handle non-linearly separable data. for that

multilayer perceptron are used.

(5) Limitations of perceptron :-

→ works only for linearly separable data.
→ Outputs are binary
→ learning may be slow for large dataset
→ cannot solve complex problem like XOR.

Perceptron :-



→ output is produced 0 or 1.
→ perception supervised learning of simple binary classifier

Training algorithm :-

→ If weight value = +ve, & target value = -ve & inverse rel? of targeted value

Bias value :-

Bias = +ve, learnered rate much

$1 \times n \times p \times 1$

Step-1 :-

$$w = x_1w_1 + x_2w_2 + \dots + x_nw_n + b$$
$$= \sum_{i=1}^{n} x_iw_i + b$$
$$= xw^T + b$$

$$x = [x_1 \; x_2 \dots x_n]$$
$$w = [w_1 \; w_2 \dots w_n]$$

$$[x_1 \; x_2 \dots x_n] \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} .$$

Step-2 : $y = f(z)$

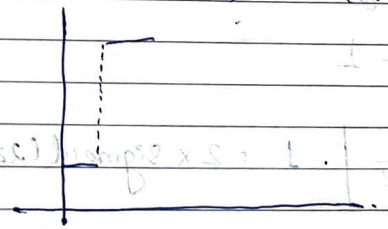
eg step function, $y = 1, \; w > \theta$
$0, \; w < \theta$

Activation function :

→ (1) Linear activation function
(2) non-linear activation function

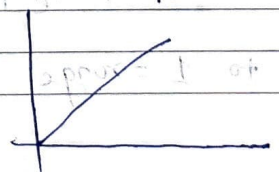
(1) Binary step Activation function :-

$f(z) = 1$, if $z > \theta$
0, if $z < \theta$



(2) → Linear activation function :-

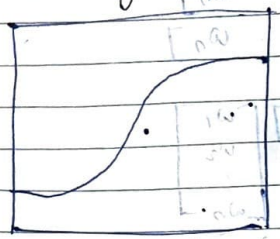
$f(z) = z$; $-\infty$ to $+\infty$



pros :-
→ It gives other than 0 or 1.
→ It can't connect to multi neurons.
→ changes are constt. for error corrects.

③ Non-linear activation function :-

(a) Sigmoid or Logistic activation function :-



* by default it is binary sigmoid function.
* if output is 0 or 1 it is used.

→ Bipolar sigmoidal AF :-

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

→ Binary sigmoid AF :-

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

* -2 to 2 range

④ Tan h AF :-

→ $= \frac{2}{1+e^{-2z}} - 1$

$= 2\left[\frac{1}{1+e^{-2z}}\right] - 1 = 2 \times \text{sigmoid}(2z) - 1$

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

→ -1 to 1 range

⑤ ReLU function :- (Rectify linear unit)

$y = f(z) = \max(0, z)$

→ It ranges from 0 to ∞ $[0, \infty)$
→ non-linear in nature
→ less computational expensive Activation

⑥ Heaviside function :-

heaviside function = $\begin{cases} 0, & \text{if } z<0 \\ 1, & \text{if } z \geq 1 \end{cases}$ Activation

⑦ Softmax function :-

→ It is used multiclass classification function.

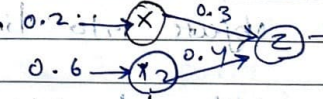
→ $\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$

z_i : output vector
j : class
k : no. of classes of multiclass

output vector

* → If output is binary, then sigmoid AF is used
→ If output is multiclass, then softmax AF is used

→ Multiclass → Branch → CSE, ME
→ multilevel → CSE-A, CSE-B, ...



→ Obtained the output for the network using AF, Binary SF, Bipolar SF, tan-h, ReLU

Activation function :⇒

$z = x_1 w_1 + x_2 w_2 + b$

$= 0.2 \times 0.3 + 0.6 \times 0.4$

$= 0.06 + 0.24 = 0.38$

Binary SF : $y = f(z) = \frac{1}{1+e^{-z}}$

$\Rightarrow \frac{1}{1+e^{-0.38}}$ $z = 0.68$

$= \frac{1}{1+0.68} = 0.595$

Bipolar SF : f_y

Linear neuron :-

→ A linear neuron is a neuron network that uses only linear transformation in its layers.

→ limitation of linear neuron :-

→ It cannot learn complex relationship between the input and output vector.

→ If a neuron network uses a linear activation function all layer of the network collapses into one.

→ It can not be used the back propagation technique, bcoz the derivative of the linear activation function is a constant.

→ It can only classify the objects into linearly separable objects.

→ It can not use the hidden layer.

* What is perceptron function :-

→ It is a function that map it's input 'x' which is multiplied with the learned weight co-efficient and the output value $f(z)$ is generated

$$f(z) = \begin{cases} 1, & \text{if } wx+b \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

bias - that element that adjust the boundary away from the origin without any dependencies on the input value.

x = vector of input value

bias = That element that adjust the boundary away from the origin without any dependdices on the input value.

(1) Characteristic of perceptron model :-

→ It is a ML algorithm for the supervised learning of binary classifier.

→ The weight & co-efficient is automatically learn.

→ Initially weight & and input featured are multiplied.

→ The decision is made whether the nurion should be dismissed or not.

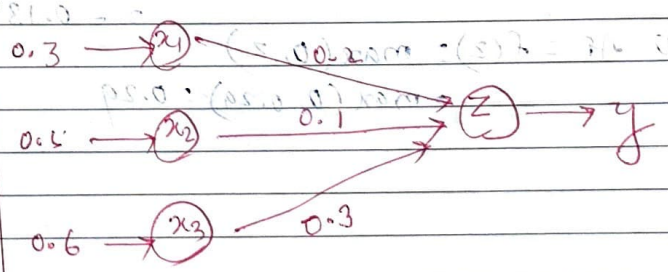
→ That activation function in any perceptron algorithm examples, employees are rule to determine whether the weight function value is higher than xero.

→ If the sum of input all the input value is higher than the threshold value then it should have a output signal

else no output is displayed.

→ The linear decision boundary is plotted and it enables the distinction between the 2 linearly separable classes +1 and -1.

(2)



obtained the output of the network using binary Sigmoid AF, Bipolar Sigmoidal AF, tanh AF, ReLU AF?

Ans:

$Z = \sum x_i w_i + b$

$= 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times 0.3$

$= (0.06 + 0.05 + 0.18) + 0$

$= 0.29$

* Binary Sigmoid AF $(y) = f(z) = \frac{1}{1+e^{-z}}$

$= \frac{1}{1+e^{-0.29}} = \frac{1}{1.748} = 0.572$

* Bipolar sigmoidal AF $= f(z) = \frac{1-e^{-z}}{1+e^{-z}}$

$\Rightarrow \frac{1-1.748}{1+1.748} = \frac{-0.748}{2.748} = -0.272$

* tan h AF $= f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$

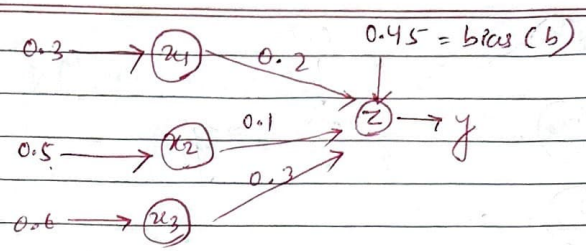
$= \frac{1.336 - 1.748}{1.336 + 1.748} = \frac{-0.412}{3.084}$

$= -0.133$

* ReLU AF $= f(z) = \max(0, z)$

$= \max(0, 0.29) = 0.29$

(3) Q.



Find same as privous question?

$Z = \sum x_i w_i + b$

$= 0.29 + 0.45 = 0.74$

→ Binary Sigmoid AF $(y) = f(z) = \frac{1}{1+e^{-z}}$

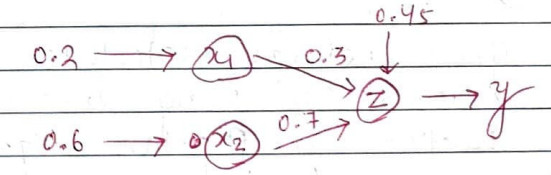
$= \frac{1}{1+e^{-0.74}} = 0.67$

→ Bipolar Sigmoidal AF $= 0.35$

→ tan h $= 0.62$

→ ReLU $= \max(0, z) = \max(0, 0.74) = 0.74$

(4) Q.



→ $Z = \sum x_i w_i + b$

$= (0.2 \times 0.3 + 0.6 \times 0.7) + 0.45$

$= (0.06 + 0.42) + 0.45$

$= 0.48 + 0.45 = 0.93$

→ Binary Sigmoid AF $= 0.71$

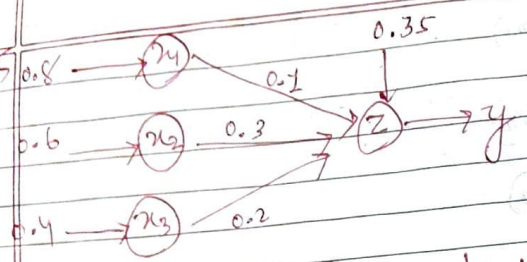
→ Bipolar Sigmoidal AF $= 0.71$

→ tan h =

→ ReLU $= f(z) = \max(0, z) = \max(0, 0.93)$

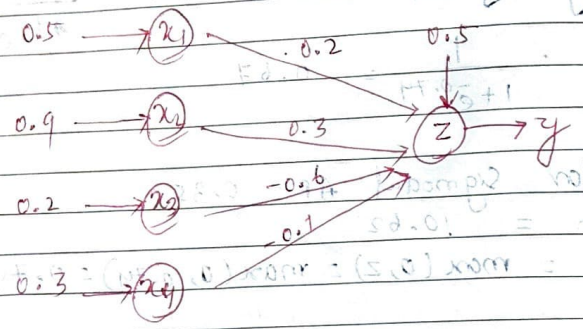
$= 0.93$

Q.5



→ Obtain the output of the nuron y for the network using Binary Sigmoidal Function, Bipolar Sigmoidal SF, and tanh.

Q.6



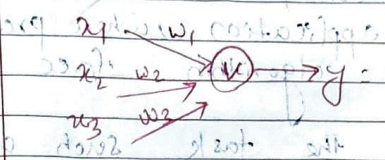
→ For the given network calculate output of the nuron 'y' using binary Sigmoidal function, Bipolar Sigmoidal Function and tanh.

Q.7 The perceptron is having weights $w=(0.4, -0.3, 0.1)$, and a bias is 0, If the input to the perceptron is $x=(0.2, 0.6, 0.5)$ then find the output of the perceptron using binary SF, Bipolar SF and Tanh Function.

NOTE: If output is binary classification, then by default we need to find binary Sigmoidal Function.

7 Ans: $wx + b$

$\Rightarrow [0.4 \ -0.3 \ 0.1]_{1\times3} [0.2 \ 0.6 \ 0.5] + 0$



Suppose the weight considered as figure as shown corresponding to the 3 input have the following values $w_1, w_2, w_3 = (1, 2, -4, 1)$ and the activation function is given by

$f(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{otherwise} \end{cases}$

calculate what will be the output value y of the figure for each of the following input patterns

pattern	P_1	P_2	P_3	P_4
x_1	1	0	1	1
x_2	0	1	0	1
x_3	0	1	1	1

Ans: $P1 = v = \sum x_i w_i + b$, $b = 0$

$x_1 = 1, x_2 = 0, x_3 = 0$

$P1 = 2$, $f(v) = 1$

→ $P2: x_1 = 0, x_2 = 1, x_3 = 1$

$P2 = -4 + 1 = -3$, $f(v) = f(-3) = 0$

$P3: x_1 = 1, x_2 = 0, x_3 = 1$

$P3 = 3$, $f(3) = 1$

$P4: x_1 = 1, x_2 = 1, x_3 = 1$

$P4 = -1$, $f(v) = f(-1) = 0$

Feed Forward Neural Network :-

→ Feed forward neural network are foundational a neural network architecture particularly in application, where predictional machine learning algorithm face limitation.

→ They facilitate the task such as simple classification face recognisation, computer vision and speech recognisation through their unedirectional flow of data.

→ It consist of input and output layers with optional hidden layer in bet".

→ FFN network are efficient for handling noisy data, are relatively straight forward to implement making them versatile tool in various AI appl".

Advantage of FFN :

→ Less complex

→ Easy to design and maintain

→ It is first and speedy., due to one way propagation.

→ It is highly responsive to noisy data.

Disadvantage of FFN :

→ It can not be used in deep learning due to the absence of dense layers and back propagation.

(7) $w = [0.4, -0.3, 0.1]$

$x = [0.2, 0.6, 0.5]$

$b = 0$

$net = 0.4 \times 0.2 + -0.3 \times 0.6 + 0.1 \times 0.5 + 0$

$= 0.08 - 0.18 + 0.05 = -0.05$

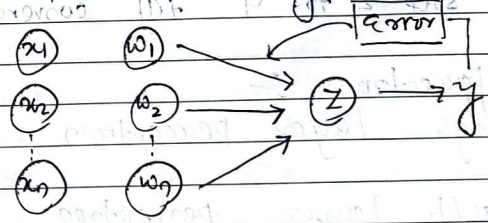
$f(x) = 0, x = -0.05$

Binary SF $= \frac{1}{1+e^{-x}} = \frac{1}{1+0.95} = \frac{1}{1.95} = 0.51$

Bipolar SF $= \frac{1-e^{-x}}{1+e^{-x}} = -0.025$

$\tanh = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = -0.0499$

Perceptron learning Rule :-



Error in perceptron :-

→ In a Perceptron learning rule the predicted output is compared with actual output, if does not match the error is propagated backward to allow the weight adjustment to happen.

formula →

$$w_{new} = w_{old} + \eta (y_{actual} - y_{pred}) x_i$$

η = learning rate (choose, like the minimum iteration is required)

x_i = input value

y_{actual} = output actual

y_{pred} = output predicted

formula $b_{new} = b_{old} + \eta(y_{actual} - y_{predict})$ [$\because x_0 = 1$ (fixed)]

Perceptron algorithm: $b = x_0 w_0 = w_0$

Steps of ~~final~~ Error reduce :-

→ (1) Initilize the weight and bias value Randomly.

(2) calculate the weighted sum z by the given input value.

(3) Applied activation function of the weighted sum and find the predicted output.

(4) IF $y_{predicted}$ is equal to y_{actual} then, no weight updation is required. else update the weight using formula 1.

(5) Repeate step 2 to 4 till conversion

Types of perceptron :-
- Single layer perceptron
- Multi layer perceptron.

Single layer perceptron :-

→ SLP can learn only linearly Separable object.

Step-1: Initialize weight and bias.
For simplicity set weight = 0, bias = 0

(2) → Set the learning rate in the range of 0 to 1 ($\eta = [0 - 1]$)

(3) while stopping condition is False do Step 2 to 6.

(4) For each training pair such that s do step 3 to 5.

(5) Set the activation function of the input unit x_i.

(6) calculate the weighted sum value.

(7) compute the response of the output unit based on the activation function.

(8) Update the weight and bias if a error occur of this pattern. update the weights and bias using w_{new} and b_{new}, else the w_0 and b remain same.

(9) Test the stopping condition.

Limitation of SLP :-

→ A Single layer perceptron, that only classify data that is linearly separable which means that only separate a data in a straight line or a hyper plane.

→ It is not suitable for complex ML problem.

→ Ex: It can not go for XOR problem.

→ It is a batch learning process so it can't learn from a new data points incrementially.

→ It is suceptable to local minima which means it can not find the global optimal solution to a problem.

→ It is prone to overfitting and noise bcoz it tries to fit a straight line to a data.

Implementation of 'OR' using perceptron :-

x_1	x_2	x_1 or x_2 → y_{actual}
0	0	0
0	1	1
1	0	1
1	1	1

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

$$f(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases}$$

case-1 :

$x_1 = 0$, $x_2 = 0$

$z = x_1 w_1 + x_2 w_2 + b = 0 \times 1 + 0 \times 1 - 0.5$

$= -0.5$ (< 0), so $f(z) = 0$

$y_{pred} = f(z) = 0 = y_{actual}$ (No weight update)

case-2 :

$x_1 = 0$, $x_2 = 1$

$z = x_1 w_1 + x_2 w_2 + b = 0 \times 1 + 1 \times 1 - 0.5$

$= 1 - 0.5 = 0.5 > 0$, so $f(z) = 1$

~~$y_{pred} = f(z) = 0.5$ y_{actual}~~

$y_{pred} = f(z) = 1 = y_{actual}$ (No weight update)

case 3 :

$x_1 = 1$, $x_2 = 0$

$z = x_1 w_1 + x_2 w_2 + b = 1 - 0.5 = 0.5 > 0$, $f(z) = 1$

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)

$f(z) = y_{pred}$

case-4 :

$x_1 = 1$, $x_2 = 1$

$z = 1 + 1 - 0.5 = 2 - 0.5 = 1.5 > 0$,

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)

Q. Suppose $w_1 = 0.6$, $w_2 = 0.6$, $b = 0$

$\eta = 0.5$, Θ (activation function) $= 1$

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

Ans case-1 :

$x_1 = 0$, $x_2 = 0$

$z = w_1 x_1 + w_2 x_2 + b = 0 + 0 = 0$

$y_{pred} = f(z) = 0 = y_{act}$ (no weight updated)

case-2 :

$x_1 = 0$, $x_2 = 1$

$z = 0.6 + 0 = 0.6$ (otherwise)

$y_{pred} = f(z) = 0 \neq y_{actual}$

$w_1^{\uparrow} = w_1 + \eta (y_{act} - y_{pred}) x_1$

$= 0.6 + 0.5 (1 - 0) \times 0 = 0.6$

$w_2^{\uparrow} = w_2 + \eta (y_{act} - y_{pred}) x_2$

$= 0.6 + 0.5 (1 - 0) \times 1$

$= 0.6 + 0.5 = 1.1$

$b^{\uparrow} = b + \eta (y_{act} - y_{pred})$

$= 0 + 0.5 (1 - 0) = 0.5$

* case-1 : $x_1 = 0$, $x_2 = 0$

$z = 0 + 0 + 0.5 = 0.5$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

case 2 : $x_1 = 0$, $x_2 = 1$

$z = 0 + 1 \times 1.1 + 0.5 = 1.6 \geq 1 = f(z) = 1$

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)

4x3

case 3: $x_1 = 1, \quad x_2 = 0$

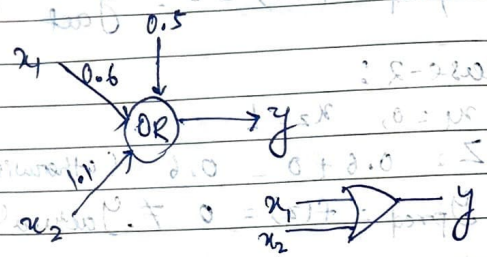
$z = 0.6 \times 1 + 0 + 0.5 = 1.1 \geq 1, \quad f(z) = 1$

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)

case 4: $x_1 = 1, \quad x_2 = 1$

$z = 0.6 + 1.1 + 0.5 = 2.2 \geq 1, \quad f(z) = 1$

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)



How to represent input in a vector format :-

Q.1

$$X = \begin{bmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad W = \begin{bmatrix} w_0 & w_1 & w_2 \\ -0.5 & 1 & 1 \end{bmatrix}$$

but $X.W$ not possible so convert to W^T.

$$W^T = \begin{bmatrix} -0.5 \\ 1 \\ 1 \end{bmatrix}$$

$$XW^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}_{4\times3} \begin{bmatrix} -0.5 \\ 1 \\ 1 \end{bmatrix}_{3\times1}$$

$$= \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ 1.5 \end{bmatrix}_{4\times1}$$

Q2

$$X = \begin{bmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad W^T = \begin{bmatrix} 0 \\ 0.6 \\ 0.6 \end{bmatrix}$$

Z = activation function

$$XW^T = \begin{bmatrix} 0 \\ 0.6 \\ 0.6 \\ 1.2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

y_{pred} y_{actual}

$w_1^T = 0.6 + 0.5(1-0) \times 0 = 0.6$

$w_2^T = 0.6 + 0.5(1-0) \times 1 = 1.1$

$b^T = 0 + 0.5(1-0) = 0.5$

Iter 2

new data :- Step-2 (iteration - 2)

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.6 \\ 1.1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 1.6 \\ 1.1 \\ 2.2 \end{bmatrix}$$

$$y_{pred} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = y_{actual}$$

(Q3) $w_1 = 0.7$, $w_2 = 0.7$, $\eta = 0.4$, Implement 'OR' gate using perceptron ?

If not given, heavy side AF is used

Ans:

$$X = \begin{bmatrix} b & x_1 & x_2 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} 0 \\ 0.7 \\ 0.7 \end{bmatrix}, \quad y_{actual} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$XW = \begin{bmatrix} 0 \\ 0.7 \\ 0.7 \\ 1.4 \end{bmatrix} = \begin{bmatrix} z \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \neq y_{actual}$$

$$w_1^{\uparrow} = 0.7 + 0.4(1-0) \times 0 = 0.7$$

$$w_2^{\uparrow} = 0.7 + 0.4(1-0) \times 1 = 0.7 + 0.4 = 1.1$$

$$b^{\uparrow} = 0 + 0.4(1-0) = 0.4$$

$$X = \begin{bmatrix} 0.4 & 0 & 0 \\ 0.4 & 0 & 1 \\ 0.4 & 1 & 0 \\ 0.4 & 1 & 1 \end{bmatrix}, \quad W = \begin{bmatrix} 1 \\ 0.7 \\ 1.1 \end{bmatrix}$$

$$XW = \begin{bmatrix} 0.4 \\ 1.5 \\ 1.1 \\ 2.2 \end{bmatrix} = \begin{bmatrix} z \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = y_{actual}$$

0.4
1.1
2.2

* NOTE: If Activation function f(z) not given, then heavy side AF is used.

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

Q1 What is deep learning ? How does deep learning differ from ML.
Q2 What ANN ? How does biological neuron similar to ANN ?
Q3 Application of DL.
Q4 What are challenges of DL.

(3) Ans:
* Image recognition : used in face unlock and photo tagging.
* self-driving car : helps detect objects, lanes, and traffic signs.
* Healthcare - used to detect diseases from xrays or scans.
* chatbots - helps computer talk like human.

(4) The challenges of DL are —
→ Needs lots of data
→ Takes long time to train.
→ overfitting
→ High cost.

(5) Implement of AND using Perceptron :-

x_1	x_2	x_1 AND x_2	y_{actual}
0	0	0	
0	1	0	
1	0	0	
1	1	1	

Suppose $w_1 = 1$, $w_2 = 1$, $b = -1.5$

$f(z) = 1$, if $z \geq 0$

0, otherwise

case-1:

$x_1 = 0, x_2 = 0$

$z = x_1 w_1 + x_2 w_2 + b = -1.5 < 0$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight updation is required)

case-2:

$x_1 = 0, x_2 = 1$

$z = 0 \times 1 + 1 \times 1 + (-1.5)$

$= 1 - 1.5 = -0.5 < 0$

$y_{pred} = f(z) = y_{act}$ (not weight updation)

case-3:

$x_1 = 1, x_2 = 0$

$z = -0.5 < 0$

$y_{pred} = f(z) = y_{actual}$ (no weight updation)

case-4:

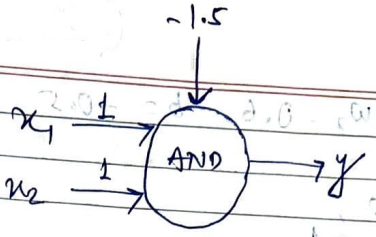
$x_1 = 1, x_2 = 1$

$z = 1 + 1 - 1.5 = 2 - 1.5 = 0.5 > 0 = 1$

$y_{pred} = f(z) = y_{actual}$ (no weight updation)

Vector format :-

$$X_2 = \begin{bmatrix} x_0 & x_1 & x_2 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}_{4 \times 3} \qquad w = \begin{bmatrix} -1.5 \\ 1 \\ 1 \end{bmatrix}_{3 \times 1}$$



Q. $w_1 = 1.2$, $w_2 = 0.6$, $b = 0$, $\eta = 0.5$

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

Ans: **case-1:**

$x_1 = 0, x_2 = 0$

$z = 0$

$y_{pred} = f(z) = 0 = y_{act}$ (no weight update)

case-2:

$x_1 = 0, x_2 = 1$

$z = 0 + 0.6 + 0 = 0.6$

$y_{pred} = f(z) = 0 = y_{act}$ (no weight update)

case-3:

$x_1 = 1, x_2 = 0$

$z = 1.2 > 1$

$y_{pred} = f(z) = 1 \neq y_{actual}$

$w_{new} \Rightarrow$

$w_1^{\uparrow} = w_1^{old} + \eta (y_{actual} - y_{predicted}) \times x_1$

$= 1.2 + 0.5 (0 - 1) \times 1$

$= 1.2 - 0.5 = 0.7$

$w_2^{\uparrow} = 0.6 + 0.5 (0 - 1) \times 0$

$= 0.6$

$b^{\uparrow} = 0 + 0.5 (0 - 1) = -0.5$

$w_1 = 0.7$, $w_2 = 0.6$, $b = -0.5$

case-1 :

$x_1 = 0$, $x_2 = 0$

$z = -0.5 < 0.1$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

case-2 :

$x_1 = 0$, $x_2 = 1$

$z = 0 + 0.6 - 0.5 = 0.1 < 0.1$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

case-3 :

$x_1 = 1$, $x_2 = 0$

$z = 0.7 + 0 - 0.5 = 0.2 < 0.1$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

case-4 :

$x_1 = 1$, $x_2 = 0$

$z = 0.7 + 0.6 - 0.5 = 0.13 - 0.5 = 0.8 < 1$

$y_{pred} = f(z) = 0 \neq y_{actual}$.

$w_1\uparrow = w_1 + \eta (y_{act} - y_{pred}) x_1$

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

$w_2\uparrow = 0.6 + 0.5 (1 - 0) \times 1 = 1.1$

$b\uparrow = -0.5 + 0.5 (1 - 0) = 0$

case-1 :

$x_1 = 0$, $x_2 = 0$

$z = 0$

$y_{pred} = f(z) = 0 = y_{act}$ (no weight update)

case-2 :

$x_1 = 0$, $x_2 = 1$

$z = 1.1$

$y_{pred} = f(z) = 1 \neq y_{actual}$

$w_1\uparrow = 1.2 + 0.5 (0 - 1) \times 0 = 1.2$

$w_2\uparrow = 1.1 + 0.5 (0 - 1) \times 1 = 0.6$

$b\uparrow = 0 + 0.5 (0 - 1) = -0.5$

case-1 :

$x_1 = 0$, $x_2 = 0$

$z = -0.5$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

case-2 :

$x_1 = 0$, $x_2 = 1$

$z = 0.6 - 0.5 = 0.1$

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight updation is required)

case-3 :

$x_1 = 1$, $x_2 = 0$

$z = 1.2 - 0.5 = 0.7 < 1$

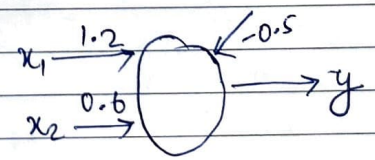
$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

case-4 :

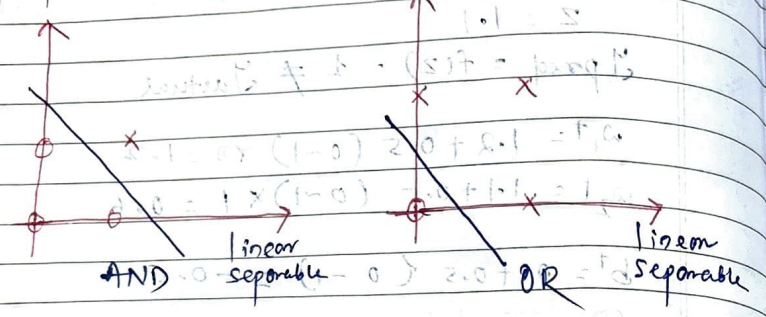
$x_1 = 1$, $x_2 = 1$

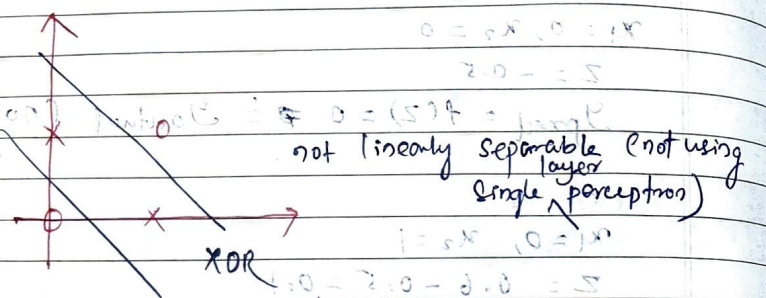
$z = 1.2 + 0.6 - 0.5 = 1.3 > 1$

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)



graphical Representation :-





not linearly separable (not using single layer perceptron)

Linear Separability :-

→ Linear separability of the point is the ability to classify the data points, in the hyper plane by avoiding the overlapping of the classes in the plane.

→ Only single layer perceptron use for linearly separable data.

→ For 'XOR' It is a multiple layer used, so we use multi layer perceptron for non-linearly separable data.

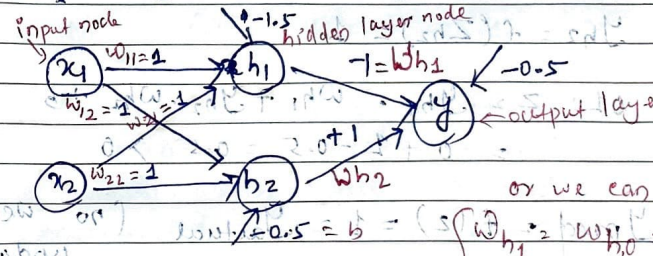
q-learnable parameter, nos of hidden layer inreses = no. of learnable parameter inreses.

XOR : (Architecture of Multilayer perceptron)

$y_{actual} = x_1 x_2' + x_1' x_2$

(H_1, H_2)

x_1	x_2	H_1	H_2	$y_{act} = H_1 + H_2$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



w_{ij} : i = input node, j = hidden node

or we can write

$w_{h_1} = w_{h_1 0} = w_{13}$

$w_{h_2} = w_{h_2 0} = w_{23}$

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

case-1 :

$x_1 = 0, x_2 = 0$

$h1$:

$z = x_1 \cdot w_{11} + x_2 \cdot w_{21} + b$

$= -1.5$

$y_{h1} = f(z_{h_1}) = 0$

$h2$: $z = x_1 \cdot w_{12} + x_2 \cdot w_{22} + b$

$= -0.5$

$y_{h_2} = f(z_{h_2}) = 0$

$O1$: $z = y_{h_1} \cdot w_{h_1} + y_{h_2} \cdot w_{h_2} + b$

$= -0.5 < 0$ (final)

$y_{pred} = f(z) = 0 = y_{actual}$ (no weight update)

Case-2:

$x_1 = 0, x_2 = 1$

$h1 = Z_{h1} = x_1 \cdot w_{11} + x_2 \cdot w_{21} + b$

$= 1 \times 1 - 1.5 = 2 - 1.5 = -0.5 < 0$

$y_{h_1} = f(Z_{h1}) = 0$

$h2: Z_{h_2} = x_1 \cdot w_{12} + x_2 \cdot w_{22} + b$

$= 0 + 1 - 0.5 = 0.5 > 0$

$y_{h2} = f(Z_{h2}) = 1$

$O1: z = y_{h_1} \cdot w_{h_1} + y_{h_2} \cdot w_{h_2} + b$

$= 0 + 1 - 0.5 = 0.5 > 0$

$y_{pred} = f(z) = 1 = y_{actual}$ (no weight update)

Case-3:

$x_1 = 1, x_2 = 0$

$h1: Z_{h_1} = 1 \cdot 1 - 1.5 = -0.5 < 0$

$y_{h_1} = f(Z_{h1}) = 0$

$h2: Z_{h2} = 0 \cdot 1 - 0.5 = 0.5 > 0$

$y_{h_2} = f(Z_{h1}) = 1$

$O1: 0 + 1 - 0.5 = 0.5 > 0$

$y_{pred} = f(z) = 1 = y_{actual}$

Case-4:

$x_1 = 1, x_2 = 1$

$h1: Z_{h1} = 1 + 1 - 1.5 = 0.5 > 0$

$y(h_1) = f(Z_{h1}) = 1$

$h2: Z_{h2} = 1 + 1 - 0.5 = 2 - 0.5 = 1.5 > 0$

$y(h_2) = f(Z_{h2}) = 1$

$O1: 1 + 1 - 0.5 = +0.5 < 0$

$y_{pred} = f(z) \geq 0 = y_{actual}$ (no weight update)

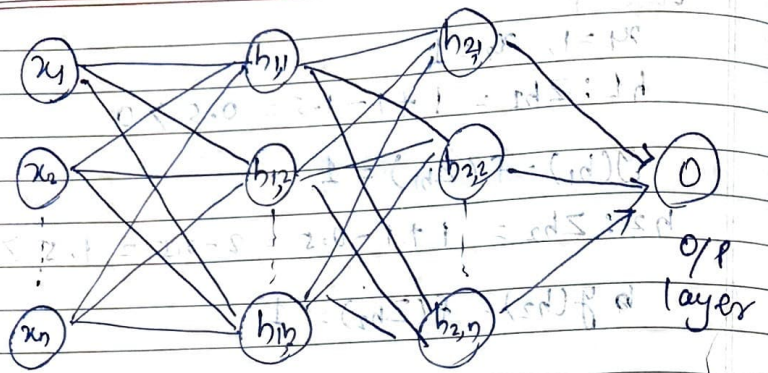
What is multilayer perceptron:-

→ It is an artificial neural network in which all the nodes are interconnected with nodes of different layer.

→ Likewise a single layer perceptron model a multilayer perceptron model also has the same model structer, but greater no. of hidden layers.

→ A multilayer perceptron neural network belongs to the feed forward neural network because it works in the forward direction.

→ It has been widely used in various field including the image recognisation speech recognisation and natural language processing.



⇒ ?

→ The multilayer perceptron also known as back propagation algorithm which executes using two stages as follows –

(i) Forward Stage :–

→ Activation function starts from the and terminates in the output layer.

(ii) Back ward stage :–

→ In backward stage weight and bias value are modified as per the model requirments.

→ In this stage the error between the actual output and the predicted output originates from the output layer and ends on the input layer.

(iii) Multilayer perceptron algorithm :–

① Step-1 : Initialize the weights and bias to small random values near to '0'.

② Set the learning rate (η) in the range of 0 to 1 (both inclusive).

③ Check for the stop condition. If stop condition is false do step 3 to 7.

④ For each training pair do step 4 to 7.

⑤ Set activation function of the output units.

⑥ Calculate the weighted sum.

$$\text{weighted sum}(z) = \sum_{i=1}^{n} x_i w_i$$

⑦ Apply the activation function of the weighted sum.

NOTE :– * For multilayer network based on the no. of layers step 6 and 7 are repeated.

⑧ If the predicted output is not equal to the actual output then update the weight and bias based on the perceptron learning rules.

η = Eta

* $w_{i(new)} = w_{i(old)} + \eta (y_{actual} - y_{pred}) \times x_i$

* $b_{(new)} = b_{(old)} + \eta (y_{actual} - y_{pred})$

⑨ Test for stop condition.

Advantage of Multilayer perceptron :–

→ This model can be used to solve complex non-linear problem.

→ It works well with small and large input data.

→ It helps us to obtain quick prediction after the training.
→ It helps to obtain same accuracy ratio with large as well as small data.

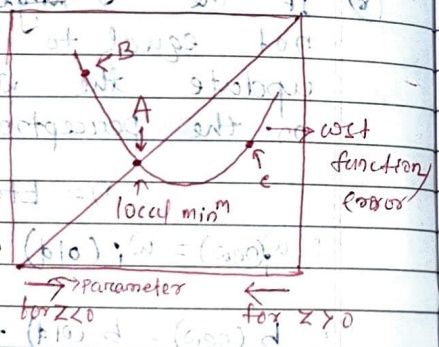
* Disadvantage of ML Pois
1. computation are difficult and time consuming
2. It is difficult to predict how much the dependent variable affect each independent variable.
3. The model functioning depends on the quality of the training.

Gradient descent :-

at A
A. z = 0
f'(z) = 0

at B
B. z < 0, f'(z) < 0

at C,
C. z > 0, f'(z) > 0



→ Gradient decent is a fundamental optimization algorithm in deep learning which is used to minimize the model cost function by iteratively adjusting it's parameter in the direction of the stiffest decent guided by the negetive of the gradients.

* Hidden layer we use ReLU function

→ It is an algo^m used to find the best sol^n to a problem by making small adjustment in the right direction.

Learning rate :-
→ It is defind as a size of the steps towards the desired location or to reach the minimum cost value.
→ It is important hyper parameter in gradient decent that control how big or small the step should be when going towards in gradient for updating the model parameter.

→ To achive the min^m cost we have diff. technique : (to prevent over shooting problem)
(a) Weight Regularization
(b) Gradient clipping
(c) → we need to choose the right learning rate

* Local min^m :-
→ If we move towards the -ve gradient or away from the gradient of a function at a current point then it will give the local min^m of the function.

* local max^m :-
→ When we move towards +ve gradient or towards the gradient of the function at a current point then we will get the local max^m of the function.

$\frac{d}{dx} x^2 = 2x \frac{dx}{dx}$

Mean Squared error :-

$$J(w,b) = \frac{1}{n} \sum_{i=1}^{n} (y_{act} - y_{pred})^2$$

$$y_{pred} = \sum xw + b$$

Gradient w.r.t w :-

$$\frac{\partial J(w,b)}{\partial w} = \frac{\partial}{\partial w} \left[\frac{1}{n} \sum_{i=1}^{n} (y_{actual} - y_{pred})^2 \right]$$

$$= \frac{1}{n} \times 2 \sum_{i=1} (y_{act} - y_{pred}) \frac{d}{dw} (y_{act} - y_{pred})$$

$$= \frac{2}{n} \sum_{i=1} (y_{act} - y_{pred}) \left[\frac{d}{dw} y_{act}^{0} - \frac{d}{dw} (\sum xw + b)^{0} \right]$$

$$= \frac{2}{n} \sum_{i=1} (y_{act} - y_{pred})(0 - x - 0) = \frac{-2x}{n} \sum_{i=1} (y_{act} - y_{pred})$$

$$= \frac{2x}{n} \sum_{i=1} (y_{pred} - y_{act})$$

* Gradient w.r.t b :-

$$\Delta b = \frac{d}{db} J(w,b) = \frac{\partial}{\partial b} \left[\frac{1}{n} \sum_{i=1}^{n} (y_{act} - y_{pred})^2 \right]$$

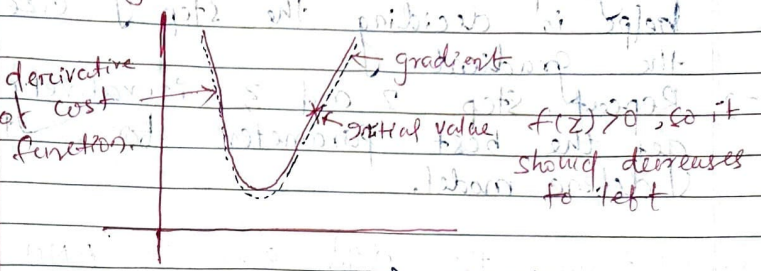
$$= \frac{1}{n} \times 2 \sum (y_{act} - y_{pred}) \frac{\partial}{\partial b} \left[\sum (y_{act} - y_{pred})^{*} \right]$$

$$= \frac{2}{n} \sum (y_{act} - y_{pred}) \left[\frac{\partial}{\partial b} \sum y_{act}^{0} - \frac{\partial}{\partial b} (\sum xw + b) \right]$$

$$= \frac{2}{n} \sum (y_{act} - y_{pred})(0 - 0 - 1) = \frac{-2}{n} \sum (y_{act} - y_{pred})$$

$$= \frac{2}{n} \sum (y_{pred} - y_{act})$$

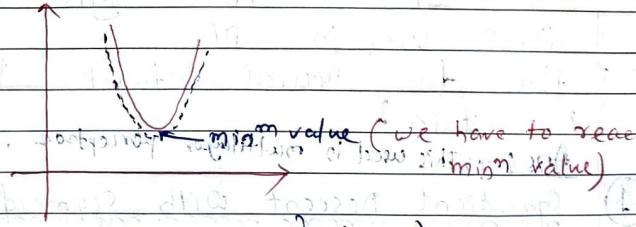
+ve gradient :-



$$w_{new} = w_{old} - \eta \frac{\partial}{\partial w} J(w,b)$$

$$b_{new} = b_{old} - \eta \frac{\partial}{\partial b} J(w,b)$$

-ve gradient :-



$$w_{new} = w_{old} + \eta \frac{\partial}{\partial w} J(w,b)$$

$$b_{new} = b_{old} + \eta \frac{\partial}{\partial b} J(w,b)$$

Step of gradient decent :-

1. Initialize the parameter the model randomly.
2. compute the gradient of the cost function w.r.t each parameter. It involves partial differentiation of cost function with respect to the parameter.
3. update the parameter of the model by taking steps in the opposite direction of the model here we choose the

a hyperparameter i.e learning rate which helps in deciding the step size of the gradient.

→ Repeat step 2 and 3 iteratively to get the best parameter for the define model.

Q.1 What are the diff. layer in ANN? What ~~are the~~ is the notation representing a node of a perticular layer.

Q.2 What is cost function in DL.

Q.3 What are the activation functions in DL and where it is use.

Q.4 What are the diff. types of activation func? used in DL.

Q.5 How do neural network learn from the data?

~~Ans~~ → This used in multilayer perceptron, for back propagation

(2) Gradient Descent with Sigmoid Neuron :-

(1) $\sigma(z)$ (Sigmoid function) :- (binary sigmoid function used)

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

(2) Sigmoid neuron's output ($\hat{y}$) :-

$$\hat{y} = \sigma(z) = \sigma\left(\sum wx + b\right)$$ [For single value $\hat{y} = \sigma(z) = \sigma(wx+b)$]

(3) Cost function (C) :

$$e = \frac{1}{2}\left(y_{\text{actual}} - \hat{y}_{\text{pred}}\right)^2$$

Step-1 :- Derivative of cost function w.r.t w

$$\frac{\partial C}{\partial w_j} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w_j}$$

(i) $\frac{\partial z}{\partial w_j} = \frac{\partial}{\partial w_j}(wx_j + b) = x_j \frac{dw}{dw_j} = x_j$

(ii) $\frac{\partial C}{\partial \hat{y}} = \frac{\partial}{\partial \hat{y}}\left(\frac{1}{2}(y-\hat{y})^2\right)$

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

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

(iii) $\frac{d\hat{y}}{dz} = \frac{\partial}{\partial z}\sigma(z) = \sigma'(z) = \sigma(z)(1-\sigma(z))$

we can write $\frac{d\hat{y}}{dz} = \hat{y}(1-\hat{y})$

$$\frac{\partial C}{\partial w_j} = (\hat{y}-y)(\hat{y}(1-\hat{y}))\, x_j$$

Step-2 : find the derivative of cost func? w.r.t bias.

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial b}$$

(1) derivative of cost func?

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

(3) $\frac{d\hat{y}}{dz} = \hat{y}(1-\hat{y})$

(3) $\frac{\partial z}{\partial b} = \frac{\partial (wx+b)}{\partial b} = 0+1 = 1$

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

update the weight and bias :-

→ $w_j(new) = w_j(old) - \eta \frac{\partial c}{\partial w_j}$

$= w_j(old) - \eta \left[(\hat{y}-y) \cdot \hat{y}(1-\hat{y}) \cdot x_j \right]$

→ $b(new) = b(old) - \eta \frac{\partial c}{\partial b}$

$= b(old) - \eta \left[(\hat{y}-y) \cdot \hat{y}(1-\hat{y}) \right]$

● challenges of Gradient descent algo :-

(1) Local minima and Saddle points :

→ Local min : represent the point, where the loss function reaches the locally min value, compared to near by points.

→ The gradient of the loss function is zero at local minima.

→ saddle point, represents the points where the gradient is zero but not all direction are flat potentially causing optimization difficulties.

→ The gradient of the loss function is also zero at saddle points but not all direction are flat.

→ optimization algo may encounter difficulties as the gradient is '0', making it challenging to escape and continue descending towards the global min.

(2) Vanishing gradient and exploding gradient :-

● vanishing gradient :-

→ vanishing gradient is the phenomenon that occurs during the training the deep neural network where the gradient that are used to update the network become extremely small or vanish as they backpropagate from the output layer to the early layers.

● exploding gradient :-

→ It happens when the gradient is too large causing unstable model ; in this case the model weights will grow too large and they will eventually be represented as NaN.

→ so one solution to this issue is to liverage or dimentionality reduction technique., which can help to minimize the complexity of the model.

● Advantage of gradient descent :-

(1) Ease of Implementation
(2) convergence guarantee
(3) Scalability
(4) versatility
(5) Flexibility

Disadvantages of gradient descent :-

1. Sensitivity to learning rate (more if more complex)
2. Sensitivity to initialization (weight and bias)
3. & coverse speed (depends on learning rate)
4. Termination of algorithm (takes long time)
5. Correctness (best fit)

Assignment :-

Q.1) The input to a single input neuron is 2.0, it's weight is 2.3 and it's bias is -3.

(i) What is the net input to the transfer function.

(ii) What is the neuron output for the following transfer function using Sigmoid activation, tan h activation function and ReLU activation function.

2. Given a 2 input neuron with the following parameter bias = 1.2, $w = [3\ 2]$ and the input $(x) = [-5\ 6]^T$, calculate the neuron output for the following transfer function using bipolar Sigmoidal fun', binary Sigmoidal fun' and tan h activation function.

3. A Single layer neuron network is to have 6 input and 2 outputs, the output are to be limited to and continuous over the range of 0 to 1. What can you tell about the network Architecture ?

(i) Specifically how many neuron are required

(ii) What are the dimension of the weight matrix.

(iii) What kind of transfer function could be used.

(iv) Is Bias required ?

4. The Final layer of the network produces logits for 3 classes, given logits are, $z = [2.0\ 1.0\ 0.1]$, True class is class 1, use softmax to get probability P_i and cross entropy loss function as $L = -\log P_{true}$, compute the probability and loss. (upto 4 decimal places), use numerically stable softmax) (use indexing starting from 0).

class work :-

1. What is overfitting and How to avoid it.
2. Define learning rate in DL.
3. Why is bias value added.
4. How weights are initialized in neural network.
5. What is cross entropy loss function.
6. What is Single layer perceptron ? What is multilayer perceptron and how it is different from Single layer perceptron.
7. How are the number of hidden layers and neuron per hidden are selected.

2241013013

Answer to classwork Question :-

(1) → Overfitting :- When a deep learning model learns the training data too well, including noise and minor details, it performs poorly on new (test) data.

→ ways to avoid overfitting :

(a) use more training data - improves generalization

(b) Early stopping - stop training when validation loss increases.

(c) use simpler models - reduce model complexity.

(2) → Learning rate : It is a hyper parameter that controls how much the model's weights are updated during training after each iteration.

→ when high learing rate, faster learning but may skip optimal point, and more accurate when low learning rate.

(3) → Bias value is added to give the model flexibility to fit data better by allowing the activation function to shift left & right.

→ without bias, all activations would pass the origin (0,0).

(4) → weights initialization means setting the starting value of weights before training begins.

→ common methods -

(a) Zero initialization : All weights = 0 (not used - because it causes symmetry problem)

(b) Random initialization - Small random values (breaks symmetry)

$0^\circ C = 273$ (K) meaningful value

no temp.

→ The response variable is in a dataset is the output variable.

→ Ex : predictor variable / response variable/data

Age	BMI	weight	Insulin level	Diabetic
38	42.5	58	96	No
42	38.6	62	150	Yes
55	30.5	59	94	No
69	48.6	65	194	Yes
72	57.5	70	202	Yes

Q. The following are the attributes of a university student. State if the data are nominal, ordinal, interval or ratio scale.

(a) Gender → Male → nominal

(b) class → Third year → ordinal

(c) Heart rate → 75 beats per minute → ratio

(d) Blood pressure → 80/120 mm of Hg → ratio

(e) Blood type → A positive → nominal

(f) Blood sugar → 4 → ratio

(g) Known allergies → Nil → nominal

(h) Body temperature → 36°C → interval

(5) → cross entropy loss measures the difference between predicted probabilities and actual labels in classification problem.

→ formulae for binary,

$$L = -[y \log p + (1-y) \log (1-p)]$$

y = true label, p = predicted probability

→ It is used in logistic regression, softmax)

(6)

Single layer perceptron	Multilayer Perceptron
(i) It has only one layer of weights.	(i) It has one or more hidden layer bet^n input and output.
(ii) It has only input and output.	(ii) It has hidden layer bet^n input and output.
(iii) It can solve linearly separable problem like AND, OR	(iii) It can solve non-linearly problem like XOR.

Difference :-

→ SLP has one layer and linear decision boundary but MLP has multiple Layer and non-linear decision boundary.

(7) → There is no fixed rule it depends on the problem, but in general

(a) for image or speech ~~is~~ more than 2 layer for complex data.

(b) For simple task, we need 1 or 2 layer.

(c) Neuron per layer usually bet^n input size and output size.

(d) Too many neuron causing overfitting.

(8) What are Feed forward neural network.

(9) When should you choose Deep learning over Machine learning solution.

(10) How do you choose the right deep learning approach for your problem and data.

(11) What are some challenges some common challenges found DL model and how would you overcome them.

(12) input (x) = 2.0
weight (w) = 0.7
bias (b) = -0.1
Find the output of the neuron (value should be upto 4 decimal places)

(8) → Feed forward neural network is a type of neural network where data moves only one direction from input to hidden layer then output with no loops.

→ It is used for tasks like classification and regression.

→ Ex: multilayer perceptron (MLP)

(9) → To choose deep learning, when we have

(a) large amount of data

(b) problem is complex

(c) high computing power

(10) To choose the right deep learning approach —

(a) understand the data type

(b) Consider dataset size

(c) Define the task - Regression or generation

(d) check computational resources

(11) common challenges in deep learning are —

(a) overfitting - use regularization, dropout, more data.

(b) Long training time - use GPU, batch normalization.

(c) Underfitting – increase model complexity.
(d) Data scarcity and poor generalization.

(12) Given, $x = 2.0$, $w = 0.7$, $b = -0.1$
activation function $(z) = wx + b$
$= 0.7 \times 2.0 + (-0.1)$
$= 1.4 - 0.1 = 1.3000$

(13) The perceptron has 2 input and one output where $w_1 = 0.2$, $w_2 = -0.3$, bias $(b) = 0.4$, input $(x) = [1\ 1]$, $y_{actual} = 1$, $\eta = 0.1$, find the updated weights after 1 iteration if the perceptron mis-classify the input.

Ans: Given, $w_1 = 0.2$, $w_2 = -0.3$, $b = 0.4$,
$x = [1\ 1]$, $y_{actual} = 1$, $\eta = 0.1$

$z = w_1x_1 + w_2x_2 + b$
$= 0.2 \times 1 + -0.3 \times 1 + 0.4$
$= 0.2 - 0.3 + 0.4 = 0.3 < 1$,
so $y_{pred} = f(z) = 0 \neq y_{actual}$

so,
$w_1^+ = w_{old} + \eta(y_{act} - y_{pred}) \times x_1$
$= 0.2 + 0.1(1 - 0) \times 1$
$= 0.2 + 0.1 = 0.3$

$w_2^+ = -0.3 + 0.1(1 - 0) \times 1$
$= -0.3 + 0.1 = -0.2$

$b^+ = 0.4 + 0.1(1 - 0)$
$= 0.4 + 0.1 = 0.5$

so updated weight value are:-
$w_1^+ = 0.3$, $w_2^+ = -0.2$

Types of GD:
(i) Batch
(ii)
(iii)

(i) Batch Gradient decent :-

$\theta_{new} = \theta_{old} - \eta \frac{\partial c}{\partial \theta}$ → Gradient of cross fuser for entire dataset
w.r.t θ

(4) → Repeats step 1 to 3, unless the loss function is minimised.

Advantages of batch GD :-

→ It produces less noise in comparision to other gradient decent.

→ It follows the true gradient of the loss function which can lead to a most stable conversion.

→ It is computationaly efficient as all resources are used for all training samples.

→ It consider the entire dataset for each updates providing a global prpective of the loss landscape.

Disadvantage :-

→ For large dataset computing the gradient for the entire dataset can be computationally expensive and slow.

→ This required storing and processing the entire dataset in memory which can be impractical for very large dataset.

Stochastic Gradient decent :- (commonly we used)

$$\theta_{new} = \theta_{old} - \eta \nabla e\left(\theta, \underbrace{x^{(i)}, y^{(i)}}_{\text{it is a } i^{th} \text{ training example}}\right)$$

↓

Gradient of loss function wrt θ, for i^{th} training dataset

→ Stochastic GD is an alternative approach that updates the weight for each training example rather than the entire dataset. This means the gradient are calculated and the weight and bias are updated for each training example.

→ It is easier to store in allocated memory as it required only one training example at a time.

→ ~~Due to frequent updates~~

→ It shows some computational efficiency losses as compared to batch gradient decent as it shows frequent updates that requires more detail and speed.

→ It is treated as a noisy gradient due to frequent updates and it converses faster when the data set is large.

steps :- (update rule)

1. Selects a single training example from the dataset.
2. compute the gradient of the loss function w.r.t each weight and bias using only that training example.
3. update the weight by taking a step in the direction of -ve gradient.
4. Repeat step 1 to 3 for each training example.

Advantages :

→ It is easier to allocate in the desire memory.

→ It is relatively faster to compute than batch gradient decent.

→ It is more efficient than large dataset as it requires less memory

→ It may get new minima.
→ It is sometimes helpful in finding global minima and also escaping the local minima.

(ii) Disadvantages:
→ It can suffer from high variants and noisy updates as each update is based on single example, which may not be representative of the entire data set.
→ It takes more iteration as compared to other gradient decent.
→ The frequent updates can cause the algorithm to over shoot the minimum. especially with the high learning rate.
→ It requires careful tuning of the learning rate to ensure stable and efficient convergence.
→ Eg: CNN (where image processing) and computational power is limited.

(iii) Mini Batch GD :-

$$\theta_{new} = \theta_{old} - \eta \Delta c\left(\theta, \{x^i, y^i\}_{i=1}^{m}\right)$$

gradient of loss function w.r.t θ, for small subset of m, of the dataset.

→ It is a compromise bet? the batch gradient decent and Sochastic GD, instead of using the entire dataset or a single training exam. it computes the gradient and updates the weight using small subset or mini batches of the entire dataset.
→ spliting the training dataset into smaller dataset makes a efficient balance to maintain the computation of batch GD and speed of Sochastic GD.

Steps of Mini GD :-
① → select a mini batch from the training data set.
② → compute the gradient of the loss function w.r.t each weight and bias using only the mini batches.
③ → update the weight by taking a step in the direction of -ve gradient.
④ Repeat step 1 to 3 from different mini batches until the entire data set has been cover.

Advantages :-
→ It strikes a balance bet? the stability of batch gradient and the efficiency of Sochastic gradient decent.
→ It helps to reduce the varience of Sochasticate gradient decent by still being computationally efficient, especially when liveraging vectorised operation on modern hardware.
→ Mini batches allow for better utilization of hardware resources such as GPU, which are optimised to parallel computation.

Disadvantages :
→ It may gets stuck at local minima.

→ It requires careful tunning of the mini batch size and learning rate to ensure optimal performance.

→ While most stable than Sochastic GD it can be still be less stable than batch gradient decent especially if mini batch size is too small.

→ Selecting an inappropriate mini batch size can lead to sub optimal perbormance and convergence issue.

→ Ex: Architecture size is huge and model is more complex and huge amount of dataset.

Why to use Sochastic GD :-

→ Sochastic GD brings several benifits to bussiness and plays a crucial role in machine learning and AI.

~~1. Computational efficiency~~

1. » Computational efficiency
   » faster convergence
   » Memory efficiency
   » Escape local minima

Question 2 :-

1. Minimize the function,
$$f(w) = (w-3)^2$$
By using the GD. ~~minimizing~~ Find the value of w, to minimize $f(w)$.
Given : $w_0 = 0$, $\eta = 0.1$, no. of iteration = 3

2. Minimize the function, $f(x,y) = x^2 + y^2$
Given, Initial point : $(x_0, y_0) = (1,1)$
η is 0.1, No. of iteration = 3

22/01/2023

1. Ans : we know, $\frac{\partial f}{\partial w}(w)$

$$w_{t+1} = w_t - \eta \frac{\partial f}{\partial w}(w)$$

$t=0$, $w_1 = w_0 - \eta \frac{\partial}{\partial w}(w_0-3)^2$ $\quad 2f\,w-3\,\frac{d}{dw}(w-3)$

$= 0 - 0.1\,(2(w_0-3))$

$= -0.1\,(2w_0-6)$

$= -0.2w_0 + 0.6 = 0.6$

$t=1$, $w_2 = w_1 - \eta \frac{d}{dw}(w_1-3)^2$

$= 0.6 - 0.1\,(2w_1-6)$

$= 0.6 - 0.2w_1 + 0.6$

$= -0.2w_1 + 1.2$

$= -0.2(0.6) + 1.2$

$= 0.6 + 0.48$

$= 1.08$

$t=2$, $w_3 = w_2 - \eta \frac{d}{dw}(w_2-3)^2$

$= -0.4w + 1.2 - 0.2w + 0.6$

$= -0.6w + 1.8$

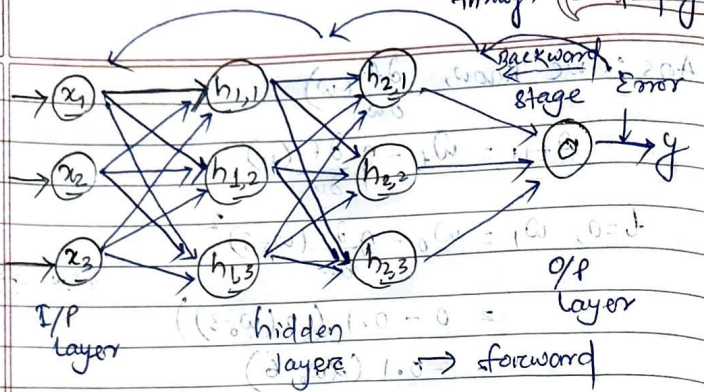
$= 1.08 + 0.1\,(2 \times 1.08 - 6)$

$= 0.696$

2. $t=0$; $x_1 = x_0 - \eta \frac{d}{dx}(x^2+y^2)$

Backpropagation algorithm :-

Adjustment of weight and bias through backpropagation



→ Backpropagation is a method used to train artificial neural network by minimizing the error. It is extensively used to train Feed forward neural network such as CNN, RNN.

→ It enables the used of gradient methods to train multilayer network and update the weights to minimize the error.

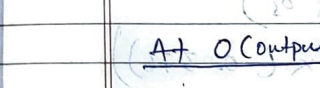
Why Backpropagation is Important?

→ It plays crucial role in how neural improves over time, the reason for this

(a) Efficient weight update

(b) Scalability

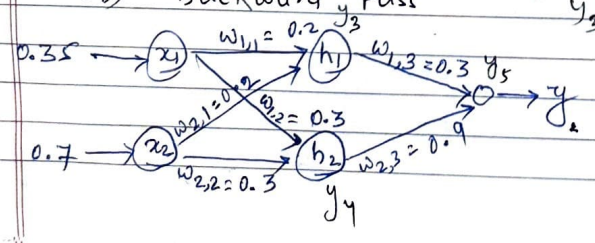
(c) Automated learning



→ The backpropagation algorithm involves

(a) forward pass

(b) Backward pass



y_3, y_4, y_5 = predicted output

Assume sigmoid activation function for forward pass and backward pass.

$y_{act} = 0.5$, $\eta = 1$

Ans: Forward calculation:

hidden layer for h_1 so $w_{1,1}$ & $w_{2,1}$

$z_j = \sum w_{ij} x_i$

at h_1:

$z_1 = w_{11} x_1 + w_{2,1} x_2$

$= 0.2 \times 0.35 + 0.3 \times 0.7$

$= 0.21$

$y_3 = f(z_1) = \frac{1}{1+e^{-z_1}} = \frac{1}{1+e^{-0.21}} = 0.55$

at h_2:

$z_2 = w_{1,2} \times x_1 + w_{2,2} \times x_2$

$= 0.3 \times 0.35 + 0.3 \times 0.7$

$= 0.315$

$y_4 = \sigma(z_2) = \frac{1}{1+e^{-z_2}} = \frac{1}{1+e^{-0.315}} = 0.58$

At O (output layer):

$z_3 = w_{1,3} \times y_3 + w_{2,3} \times y_4$

$= 0.3 \times 0.55 + 0.9 \times 0.58$

$= 0.687$

Sigmoid function of z_3:

$y_5 = \frac{1}{1+e^{-0.67}} = 0.67$;

but actual output is $y_{act} = 0.5$, so

δ = delta (gradient)

Error calculation,

error = ~~y_{actual}~~ $y_{pred.} - y_{act} = 0.67 - 0.5$

$= 0.17$

Backpropagation :

- δ_j = error term j for each unit.

η = learning rate

To compute gradient

$$\Delta w_{ij} = \eta \times \delta_j \times O_j$$ (output)

For 0 :

$\delta_5 = y_5 (1 - y_5) \cdot (y_{act} - y_{pred})$

$= 0.67 \cdot (1 - 0.67) (0.5 - 0.67)$

$= -0.0376$

For h1 :

$\delta_3 = y_3 (1 - y_3) (w_{ij} \times \delta_5)$

$= 0.55 (1 - 0.55) (w_{1,3} \times \delta_5)$

$= 0.55 (1 - 0.55) (0.3 \times (-0.0376))$

$= -0.00279$

For h2 :

$\delta_4 = y_4 (1 - y_4) (w_{2,3} \times \delta_5)$

$= 0.58 (1 - 0.58) (0.9 \times (-0.0376))$

$= -0.00824$

δ_5 = delta

$w_{new} = w_{old} + \Delta w_{ij}$

- Weight update

for output layer (O)

$\Delta w_{2,3} = \eta \times \delta_5 \times y_4$

$= 1 \times (-0.0376) \times 0.58 = -0.0218$

new weight,

$w_{2,3}$ (new) $= 0.9 - 0.0218 = 0.878$

$\Delta w_{1,3} = \eta \times \delta_5 \times y_3$

$= 1 \times (-0.0376) \times 0.55$

$= 0.02$

$w_{1,3}$ (new) $= 0.3 - 0.02 = 0.28$

for hidden layer :

$\Delta w_{1,1} = \eta \times \delta_3 \times x_1$

$= 1 \times (-0.00279) \times 0.35$

$= 0.000976$

$w_{1,1}$ (new) $= 0.2 - 0.000976 = 0.199$

$\Delta w_{1,2} = \eta \times \delta_4 \times x_1$

$= 1 \times (-0.00824) \times 0.35 =$

$= -0.00288$

$w_{1,2}$ (new) $= 0.3 - \Delta w_{1,2} = 0.297$

$\Delta w_{2,1} = \eta \times \delta_3 \times x_2$

Forward calculation:

At h1:

$z_1 = w_{1,1} \times x_1 + w_{2,1} \times x_1$

$= 0.2082$

$y_3 = \sigma(z_1) = \frac{1}{1+e^{-0.2082}} = 0.55$

At h2:

$z_2 = w_{1,2} \times x_2 + w_{2,2} \times x_2$

$= 0.30975$

$y_4 = \sigma(z_2) = \frac{1}{1+e^{-0.30975}} = 0.58$

At O:

$z_3 = w_{1,3} \times y_3 + w_{2,3} \times y_4$

$= 0.571$

$y_5 = \frac{1}{1+e^{-0.571}} = 0.64$

Iteration -2

do - same process ; again calculate back pass then

Algorithm :

Training algo :-

Step-1

→ Initialize the weight to small Random values ; where input training vector is $x = \{x_1, x_2 \dots x_n\}$

Step-4 :

→ While the steps stopping condition is to be false, do step 3 to 10.

Step-3 :

→ For each training pair do step 4 to 9.

Step-4 :

→ Each input unit recieves the signal unit and transmits the signal x_i to all the units.

Step-5 :

→ For each hidden unit calculate the weight sum $z_j = \sum w_{ij} \times x_i + \text{bias}$, then apply the activation function over the weighted sum and sends this signal to the all the units in the layer about.

Step-6

Calculation factors for propagation error :

→ For each output unit y_k, where k is 1 to n recieves the target pattern to an input pattern and error is calculated.

$y_k = f(z_j)$

δ.

1. Initialize the weight to small random value
2. The input is model using true weights for the inputs that arrive through the pre-connected path.
3. calculate the output to each neuron from the input layer to the hidden layer then to the output layer.
4. Calculate the error in the output

   Backpropagation error = $y_{act} - y_{pred}$

5. From the output layer, go back to the hidden layer to adjust the weights in order to reduce the error.
6. Repeat the process until the desired output is achived.

Advantages:

(i) Ease to Implementation.
(ii) It is simple and flexibility.
(iii) Efficiency
(iv) Generalization
(v) Scalability

challenges Disadvantages:-

→ Vanishing Gradient problem
→ Exploding gradient.
→ overfitting