Question Pattern

Total Question Set	Need to Answer	Question Type	Marks
7t	**5**	Each set will have 3 sub-questions (CO1 + CO2 + CO3)	**14**
		Total:	**70**

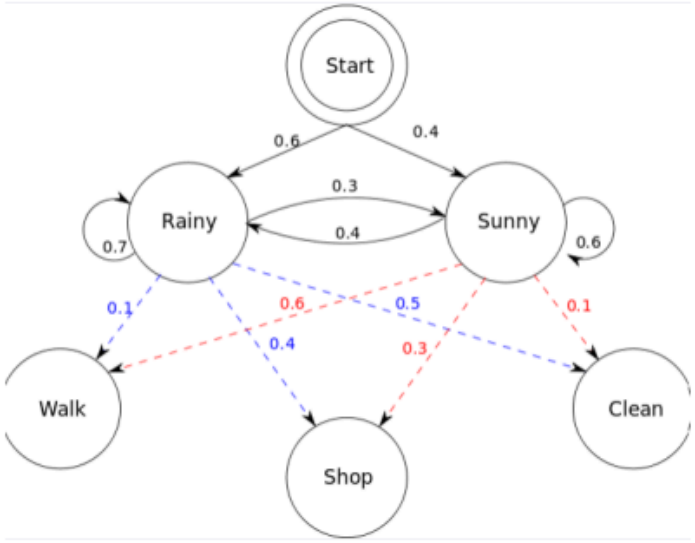
Suggestion

Label	Topic Name	Set Distribution
A	✔ Hidden Markov Models (HMM), Viterbi, RNN, LSTM, GRU	1 set
B	✔ Word Embedding, BoW, TF-IDF, Attention Mechanism, Vectorization of NN	1.5/2 set
C	✔ Activation Function, Logistic Regression, Optimizer, Cross Entropy Loss, Information Theory, Gradient Descent	2 set
D	✔ CNN, Parameter Calculation, Parameter vs Hyperparameter, Genetic Algorithm, TSP	1 set
E	✔ Fuzzy Basics, Fuzzy Membership Function, Fuzzy Addition, Fuzzy Composition, Neuro-fuzzy System Equation , graph , math	1/1.5 set

A : 1 set

Quiz 4 - Qubits45

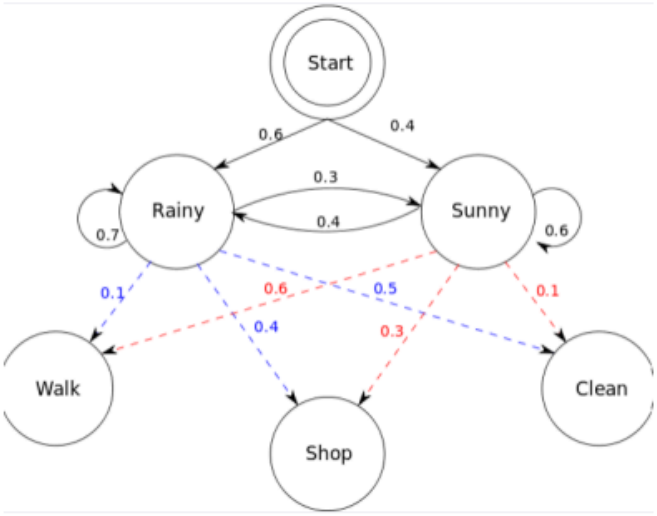
1. If the sequence of observations is **CCSW**. Find the sequence of states for the following scenario using the Viterbi algorithm. State every step of the simulation.
   (W = Walk, S = Shop, C = Clean)



Quiz-4 - Integer43

Set-A

1. If the sequence of observations is **SCWS**. Find the sequence of states for the following scenario using the Viterbi algorithm. State every step of the simulation.
(W = Walk, S = Shop, C = Clean)



1. **Solution:** ***Solved by Younus-131***
Initial Probability naa dewa thakle, by default duitay 0.5 koree hobee

Set-A

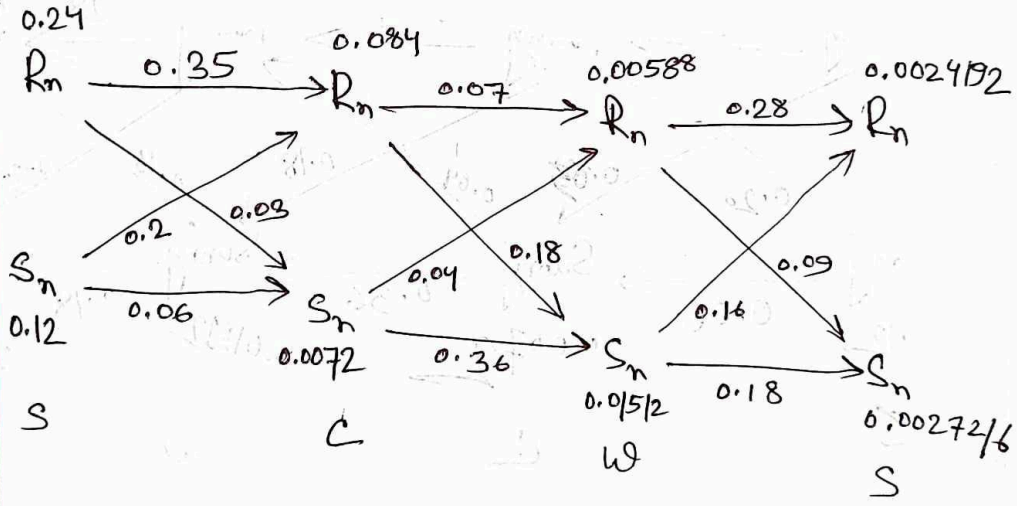
Initial Probability, π = $\begin{matrix} R_n & S_n \end{matrix}$
$\begin{bmatrix} 0.6 & 0.4 \end{bmatrix}$

Transision Matrix, A =

	R_n	S_n
R_n	0.7	0.3
S_n	0.4	0.6

Emission Matrix, B =

	W	S	C
R_n	0.1	0.4	0.5
S_n	0.6	0.3	0.1



Result : $R_n \longrightarrow R_n \longrightarrow S_n \longrightarrow S_n$

$P(s, R_n) = P(s|R_n) P(R_n)$

$= 0.4 \times 0.6 = 0.24$

$P(s, S_n) = P(s|S_n) P(S_n)$

$= 0.3 \times 0.4 = 0.12$

$P(c, R_n) = P(c|R_n) P(R_n|R_n)$

$= 0.5 \times 0.7 = 0.35$

$P(c, S_n) = P(c|S_n) P(S_n|S_n)$

$= 0.1 \times 0.6 = 0.06$

$P(c, R_n) = P(c|R_n) P(R_n|S_n)$

$= 0.5 \times 0.4 = 0.20$

$P(c, S_n) = P(c|S_n) P(S_n|R_n)$

$= 0.1 \times 0.3 = 0.03$

$\max(0.24 \times 0.35, 0.12 \times 0.2)$

$= \max(0.084, 0.024) = 0.084$

$\max(0.24 \times 0.03, 0.12 \times 0.06)$

$= \max(0.0072, 0.0072) = 0.0072$

$P(w, R_n) = P(w|R_n) P(R_n|R_n)$

$= 0.1 \times 0.7 = 0.07$

$P(w, S_n) = P(w|S_n) P(S_n|S_n)$

$= 0.6 \times 0.6 = 0.36$

$P(w, R_n) = P(w|R_n) P(R_n|S_n)$

$= 0.1 \times 0.4 = 0.04$

$P(w, S_n) = P(w|S_n) P(S_n|R_n)$

$= 0.6 \times 0.3 = 0.18$

$\max(0.084 \times 0.07, 0.0072 \times 0.04)$

$= \max(0.00588, 0.000288) = 0.00588$

$\max(0.084 \times 0.18, 0.0072 \times 0.36)$

$= \max(0.01512, 0.002592) = 0.01512$

$P(s, R_n) = P(s|R_n) P(R_n|R_n)$

$= 0.4 \times 0.7 = 0.28$

$P(s, S_n) = P(s|S_n) P(S_n|S_n)$

$= 0.3 \times 0.6 = 0.18$

$P(s, R_n) = P(s|R_n) \times P(R_n|S_n)$

$= 0.4 \times 0.4 = 0.16$

$P(s, S_n) = P(s|S_n) P(S_n|R_n)$

$= 0.3 \times 0.3 = 0.09$

$\max(0.00588 \times 0.28, 0.01512 \times 0.16)$

$= \max(0.001646, 0.0024192) = 0.0024192$

$\max(0.00588 \times 0.09, 0.01512 \times 0.18)$

$= \max(0.0005292, 0.0027216)$

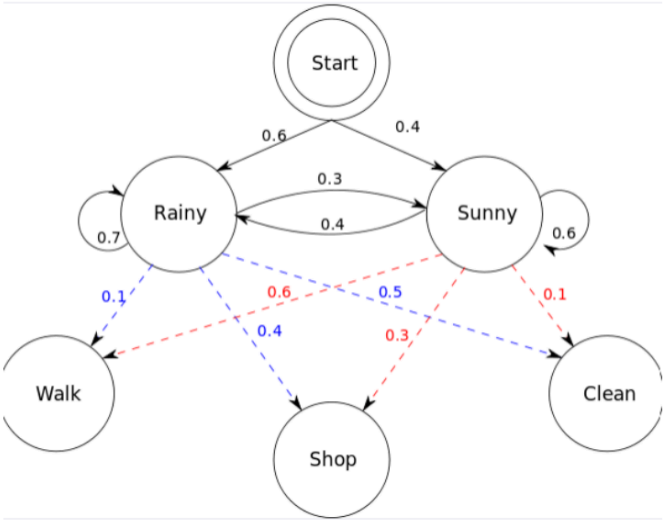
$= 0.0027216$

Ans vul ache

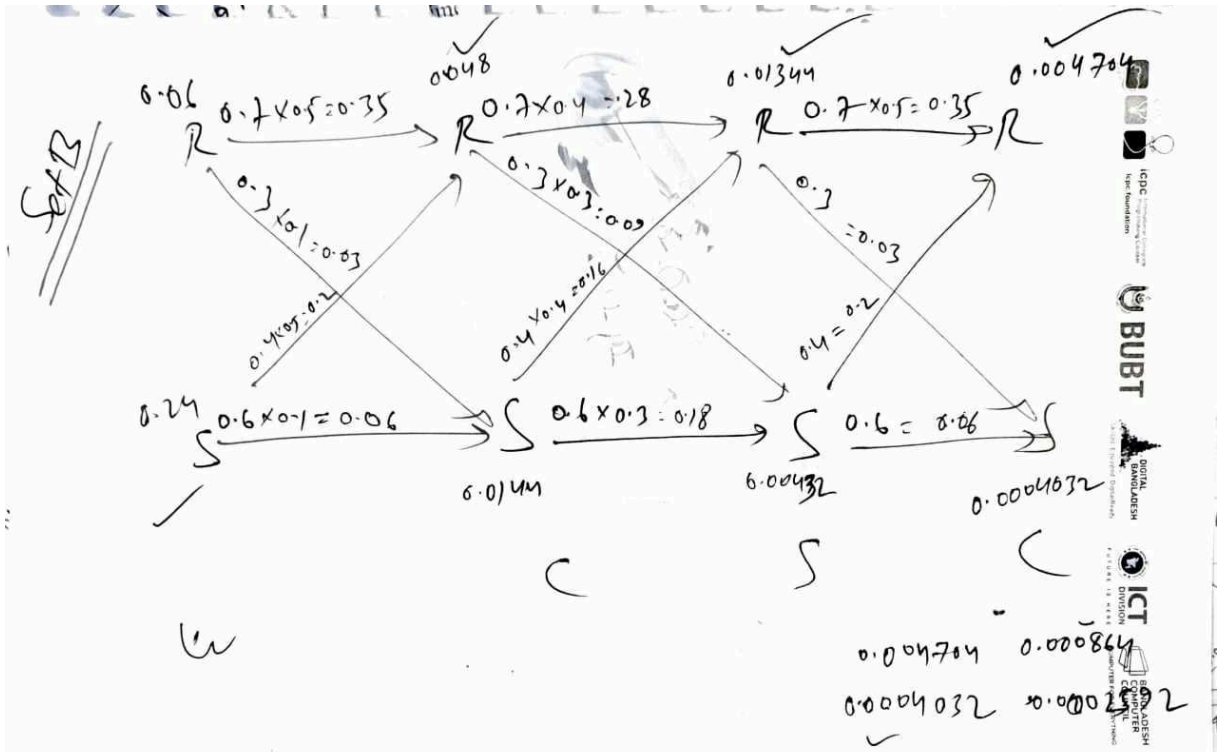
- **Could you point out the mistake, please?**

Set-B

1. If the sequence of observations is **WCSC**. Find the sequence of states for the following scenario using the Viterbi algorithm. State every step of the simulation.
   (W = Walk, S = Shop, C = Clean)

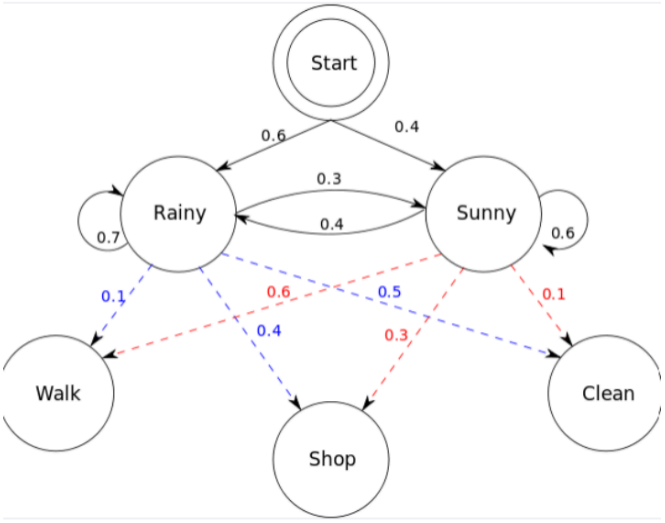


1. **Solution:**

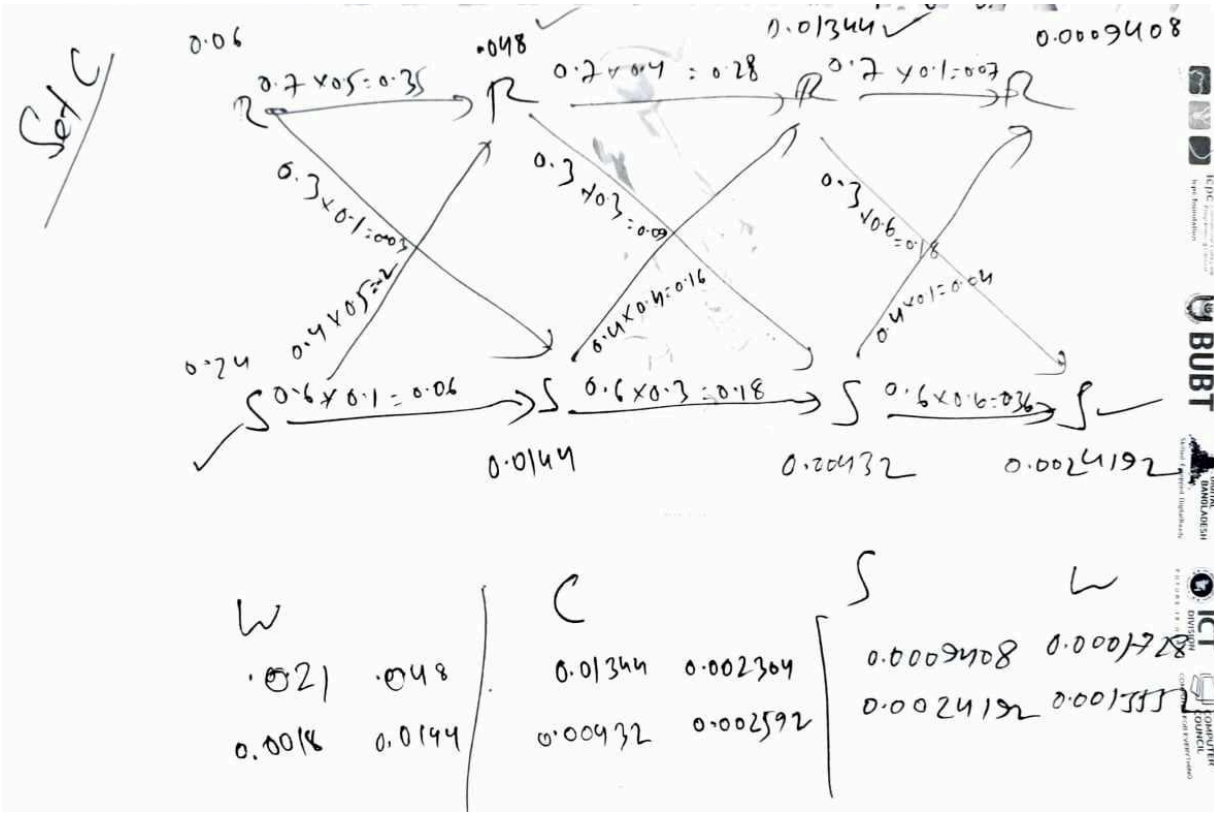


Set-C

1. If the sequence of observations is **WCSW**. Find the sequence of states for the following scenario using the Viterbi algorithm. State every step of the simulation.
   (W = Walk, S = Shop, C = Clean)

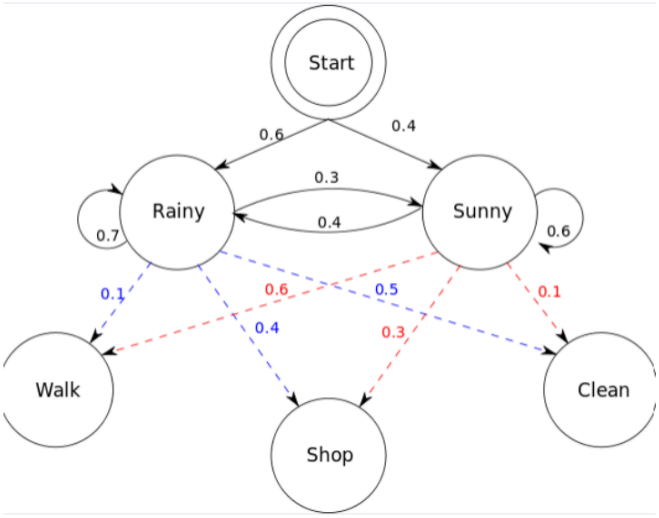


1. **Solution:**

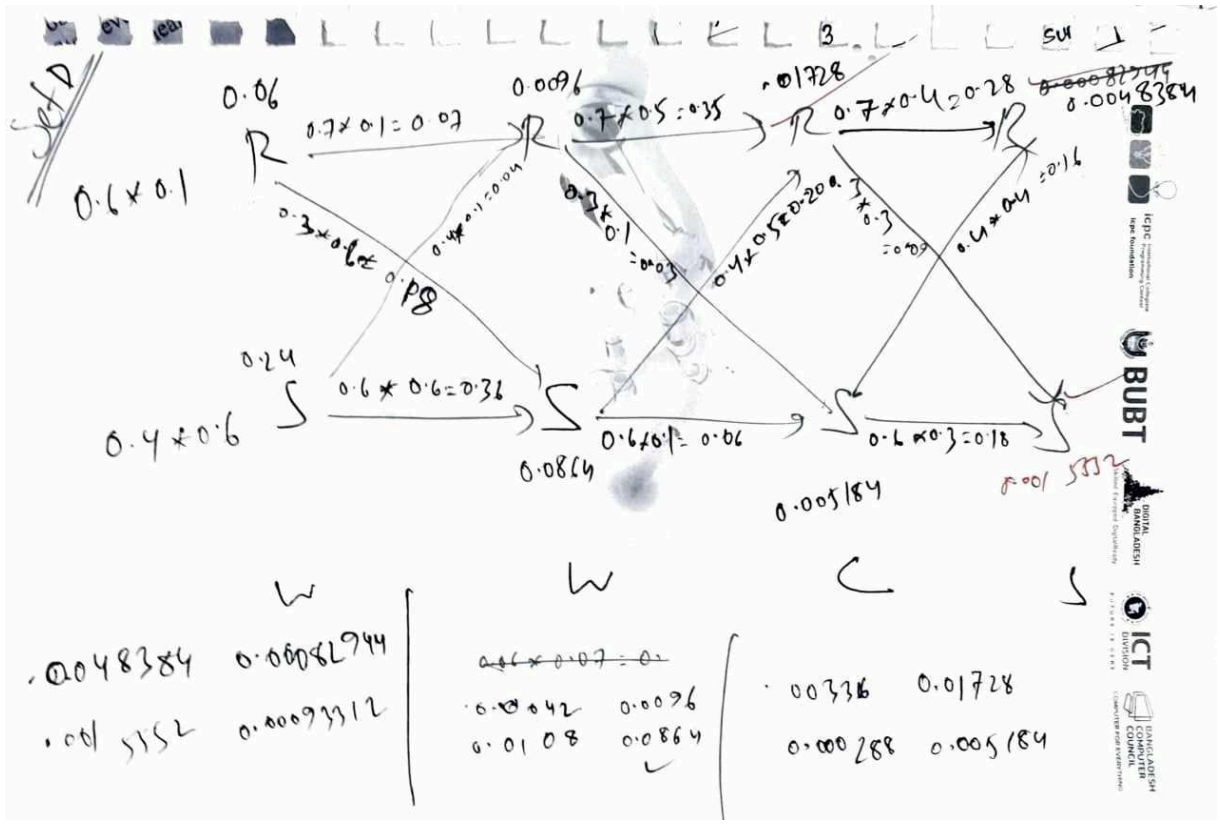


Set-D

1. If the sequence of observations is **WWCS**. Find the sequence of states for the following scenario using the Viterbi algorithm. State every step of the simulation.
   (W = Walk, S = Shop, C = Clean)

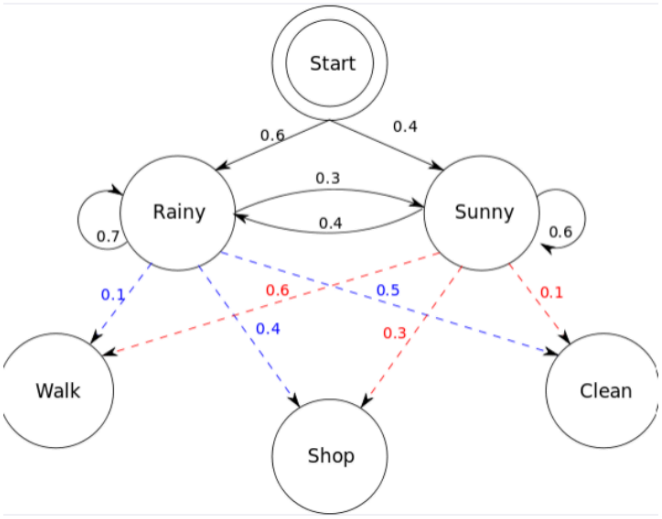


1. **Solution:**



Set-E

2. If the sequence of observations is **WCWS**. Find the sequence of states for the following scenario using the Viterbi algorithm. State every step of the simulation.
(W = Walk, S = Shop, C = Clean)



2. **Solution:**

HMM, Viterbi

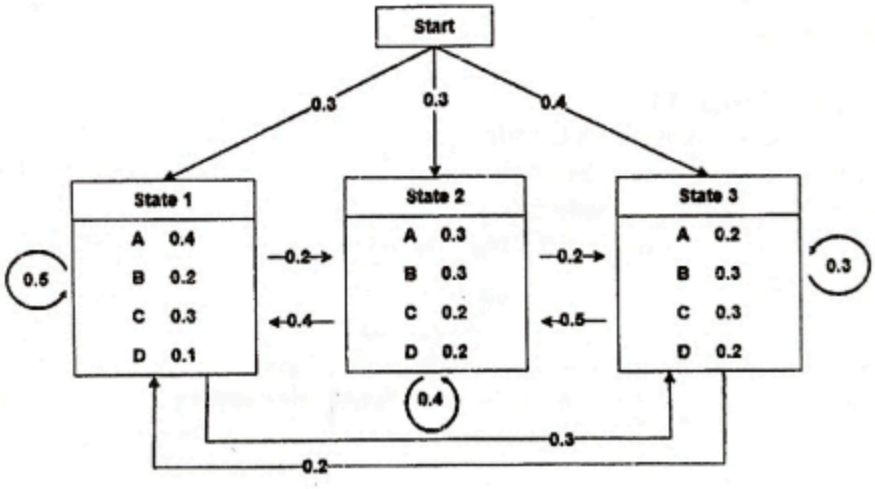
Video By Paul 023: HMM Viterbi Algorithm Math

Integer43

c) Consider the simple hidden Markov model (HMM) in the following figure. This model is composed of 3 states, **State 1, State 2** and **State 3.** Each of the state has four different observations **A, B, C** and **D.** Analyze the model and find out the right states for the following sequence of observations using the Viterbi algorithm. [7]

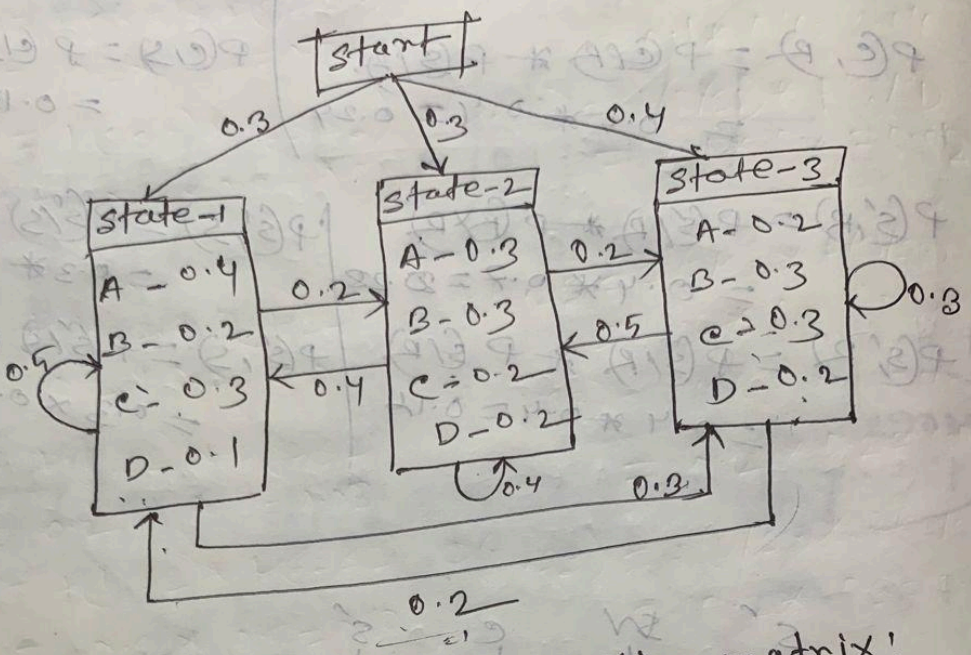
Sequence of observations - **B C D**

State every step of the simulation.



By Mushfiq 002

Fall-22



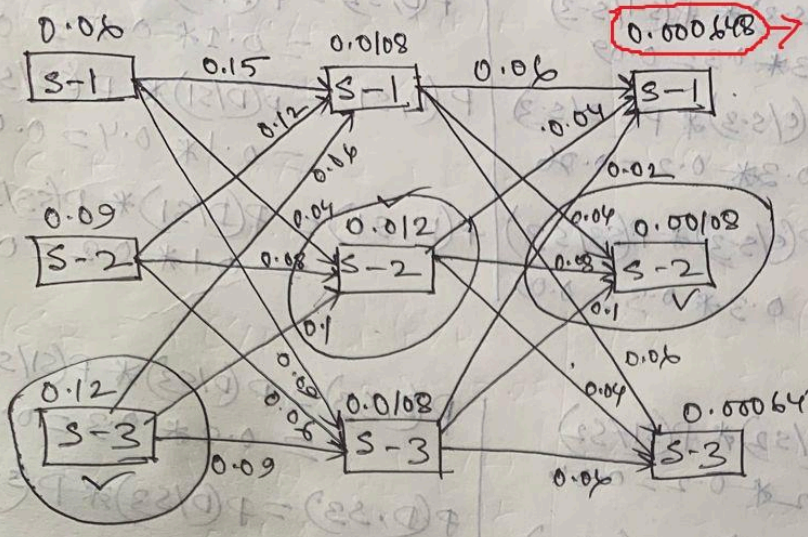
transation matrix:

old \ new	S-1	S-2	S-3
S-1	0.5	0.2	0.3
S-2	0.4	0.4	0.2
S-3	0.2	0.5	0.3

Observation matrix:

	A	B	C	D
S-1	0.4	0.2	0.3	0.1
S-2	0.3	0.3	0.2	0.2
S-3	0.2	0.3	0.3	0.2

$$\pi = [\,0.3 \quad 0.3 \quad 0.4\,]$$



0.000648 → 0.00054

0.00108

B → S-3

C → S-2

D → S-2

$P(B, S-1) = P(B/S-1) * P(S-1)$
$= 0.2 * 0.3 = 0.06.$

$P(B, S-2) = P(B/S-2) * P(S-2)$
$= 0.3 * 0.3 = 0.09$

$P(B, S-3) = P(B/S-3) * P(S-3)$
$= 0.3 * 0.4 = 0.12$

$P(C, S-1) = P(C/S-1) * P(S-1/S-1)$
$= 0.3 * 0.5 = 0.15$

$P(C, S-1) = P(C/S-1) * P(S-2/S-1)$
$= 0.3 * 0.4 = 0.12$

$P(C, S-1) = P(C/S-1) * P(S-3/S-1)$
$= 0.3 * 0.2 = 0.06$

$P(C, S-2) = P(C/S-2) * P(S-1/S-2)$
$= 0.2 * 0.2 = 0.04$

$P(C, S-2) = P(C/S-2) * P(S-2/S-2)$
$= 0.2 * 0.4 = 0.08$

$P(C, S-2) = P(C/S-2) * P(S-3/S-2)$
$= 0.2 * 0.5 = 0.1$

$P(C, S-3) = P(C/S-3) * P(S-1/S-3)$
$= 0.3 * 0.3 = 0.09$

$P(C, S-3) = P(C/S-3) * P(S-2/S-3)$
$= 0.3 * 0.2 = 0.06$

$P(C, S-3) = P(C/S-3) * P(S3/S3)$
$= 0.3 * 0.3 = 0.09$

$P(D, S1) = P(D/S1) * P(S1/S1)$
$= 0.1 * 0.5 = 0.05$ ← 0.06

$P(D, S1) = P(D/S1) * P(S2/S1)$
$= 0.1 * 0.4 = 0.04$

$P(D, S1) = P(D/S1) * P(S3/S1)$
$= 0.1 * 0.2 = 0.02$

$P(D, S2) = P(D/S2) * P(S1/S2)$
$= 0.2 * 0.2 = 0.04$

$P(D, S2) = P(D/S2) * P(S2/S2)$
$= 0.2 * 0.4 = 0.08$

$P(D, S2) = P(D/S2) * P(S3/S2)$
$= 0.2 * 0.5 = 0.10$

$P(D, S3) = P(D/S3) * P(S1/S3)$
$= 0.2 * 0.3 = 0.06$

$P(D, S3) = P(D/S3) * P(S2/S3)$
$= 0.2 * 0.2 = 0.04$

$P(D, S3) = P(D/S3) * P(S3/S3)$
$= 0.2 * 0.3 = 0.06$

S-3 → S-2 → S-2

a) Explain the issues of Hidden Markov Model (HMM). [3]

Hidden Markov Models (HMMs) have certain limitations:

1. **Markov Property**: Assumes future states depend only on the current state, neglectin long-term dependencies.
2. **Memorylessness**: Lack consideration of long-range dependencies in data.
3. **Fixed State Space:** Requires a predefined state space, making adaptation to changing structures challenging.
4. **Parameter Estimation**: Estimating transition and emission probabilities can be challenging, with sensitivity to initialization.
5. Inference Complexity:Dynamic programming algorithms for likelihood and decoding can be computationally expensive.
6. **Overfitting:** Prone to overfitting, requiring regularization techniques.
7. **Limited Expressiveness:** May struggle to capture complex relationships compared to more advanced models like RNNs or transformers.

Origin42

1

b) Consider the simple hidden Markov model (HMM) in Figure-1. This model is composed of 2 states, **HIGH** and **LOW**. You can for example consider that HIGH characterizes coding DNA while LOW characterizes non-coding DNA. Analyze the model and find out the right regions of DNA for the following sequence using the Viterbi algorithm. [8]

A C T G A

State every step of the simulation.

HMM + Viterbi

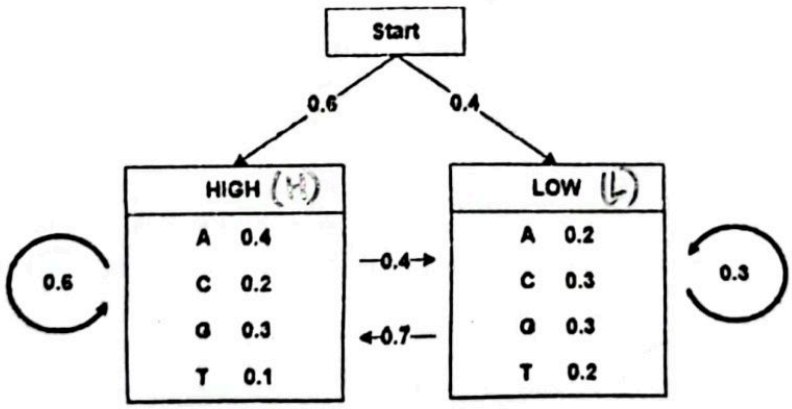


Figure – 1

Solution: Sujon 49

At Origin 42

Initial probability $\pi = \begin{bmatrix} \text{High} & \text{Low} \\ 0.6 & 0.4 \end{bmatrix}$

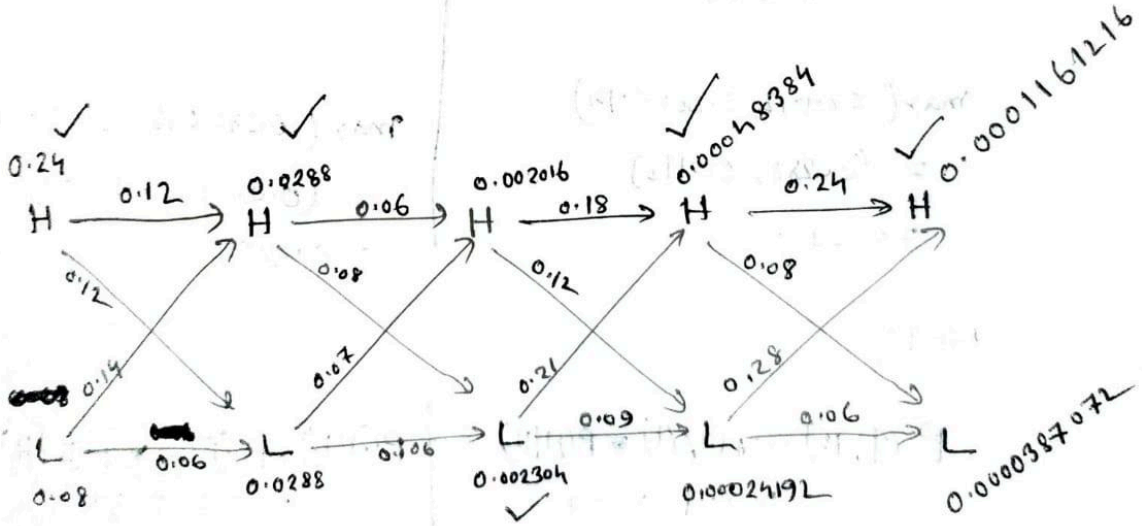
Transition probability:

old \ Now	High	Low
High	0.6	0.4
Low	0.7	0.3

Emission probability:

	A	C	G	T
High	0.4	0.2	0.3	0.1
Low	0.2	0.3	0.3	0.2

Using viterbi algo for sequence: ACTGA



For G!

$P(G,H) = P(G|H) * P(H|H)$

$= 0.3 * 0.6$

$= 0.18$

$P(G,H) = P(G|H) * P(H|L)$

$= 0.3 * 0.7$

$= 0.21$

$\max(0.002016 \times 0.18,\ 0.002034 \times 0.21)$

$= \max(3.6288 \times 10^{-4},\ 4.2714 \times 10^{-4})$

$= 0.00042714$

$P(G,L) = P(G|L) * P(L|H)$

$= 0.3 * 0.4$

$= 0.12$

$P(G,L) = P(G|L) * P(L|L)$

$= 0.3 * 0.3$

$= 0.09$

$\max(0.002016 \times 0.12,\ 0.002034 \times 0.09)$

$= \max(2.4192 \times 10^{-4},\ 2.0736 \times 10^{-4})$

$= 0.00024192$

For A!

$P(A,H) = P(A|H) * P(H|H)$

$= 0.4 * 0.6$

$= 0.24$

$P(A,H) = P(A|H) * P(H|L)$

$= 0.4 * 0.7$

$= 0.28$

$\max(0.24 \times 0.00048384,\ 0.28 \times 0.00024192)$

$= \max(1.161216 \times 10^{-4},\ 6.77376 \times 10^{-5})$

$= 0.00011161216$

$P(A,L) = P(A|L) * P(L|H)$

$= 0.2 * 0.4$

$= 0.08$

$P(A,L) = P(A|L) * P(L|L)$

$= 0.2 * 0.3$

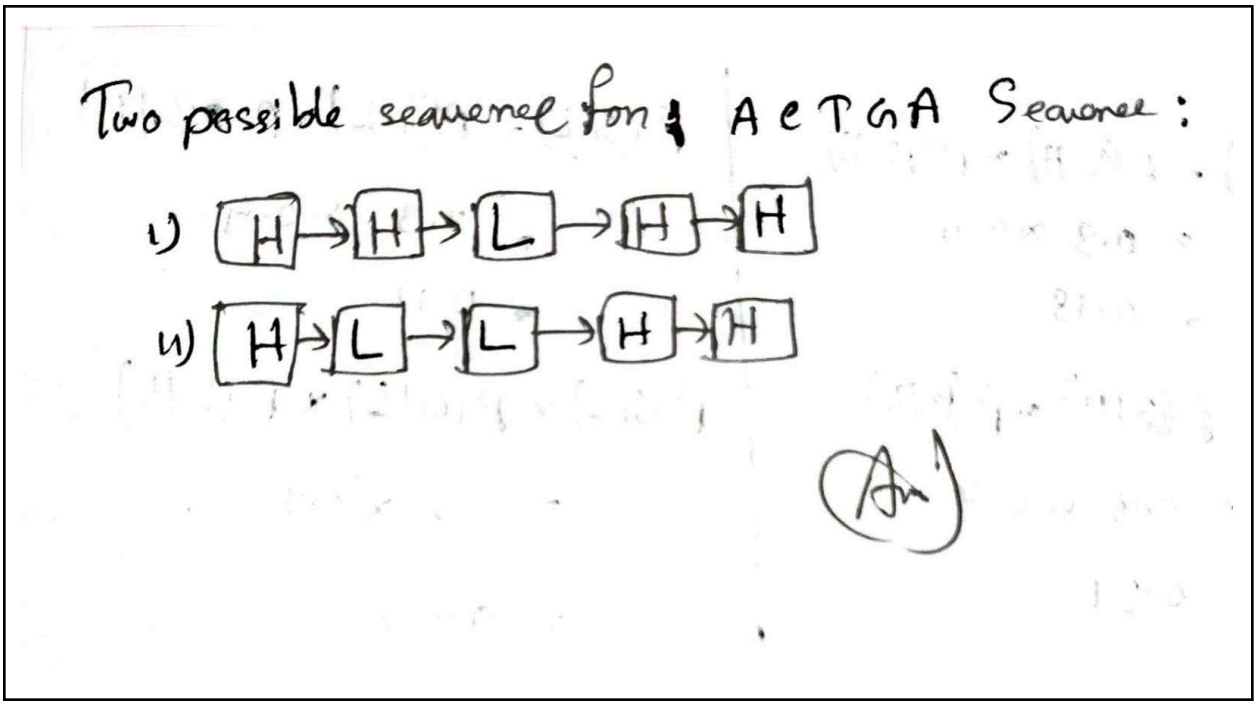
$= 0.06$

$\max(0.08 \times 0.00048384,\ 0.06 \times 0.00024192)$

$= \max(3.87072 \times 10^{-5},\ 1.45152 \times 10^{-5})$

$= 3.87072 \times 10^{-5}$

$= 0.0000387072$



Enigma41

6

ii) Suppose, you have 2 boxes B1 and B2, each of which contains 4 balls colored RED, GREEN, BLUE and WHITE. A sequence of five balls is randomly drawn from the boxes. In this particular case, the user observes a sequence of balls **GREEN, GREEN, RED, WHITE and BLUE.** Find out the right sequence of two boxes that these five balls were pulled from using the Viterbi algorithm. State every step of the simulation. The transition and emission probabilities are given below. [8]

HMM - Viterbi

Transition Probability

Box	B1	B2
B1	0.6	0.4
B2	0.3	0.7

Emission Probability

Box \ Ball	RED	GREEN	BLUE	WHITE
B1	0.2	0.4	0.3	0.1
B2	0.2	0.2	0.3	0.3

Solution: **075 (Assume Initial Probability)**

Subject: Enigma 41

Date: Time:

↳ Initial Probability, π = $\begin{bmatrix} B1 & B2 \\ 0.5 & 0.5 \end{bmatrix}$

↳ Transition Probability, A =

old \ Neo	B1	B2
B1	0.6	0.4
B2	0.3	0.7

↳ Emission Probability, B =

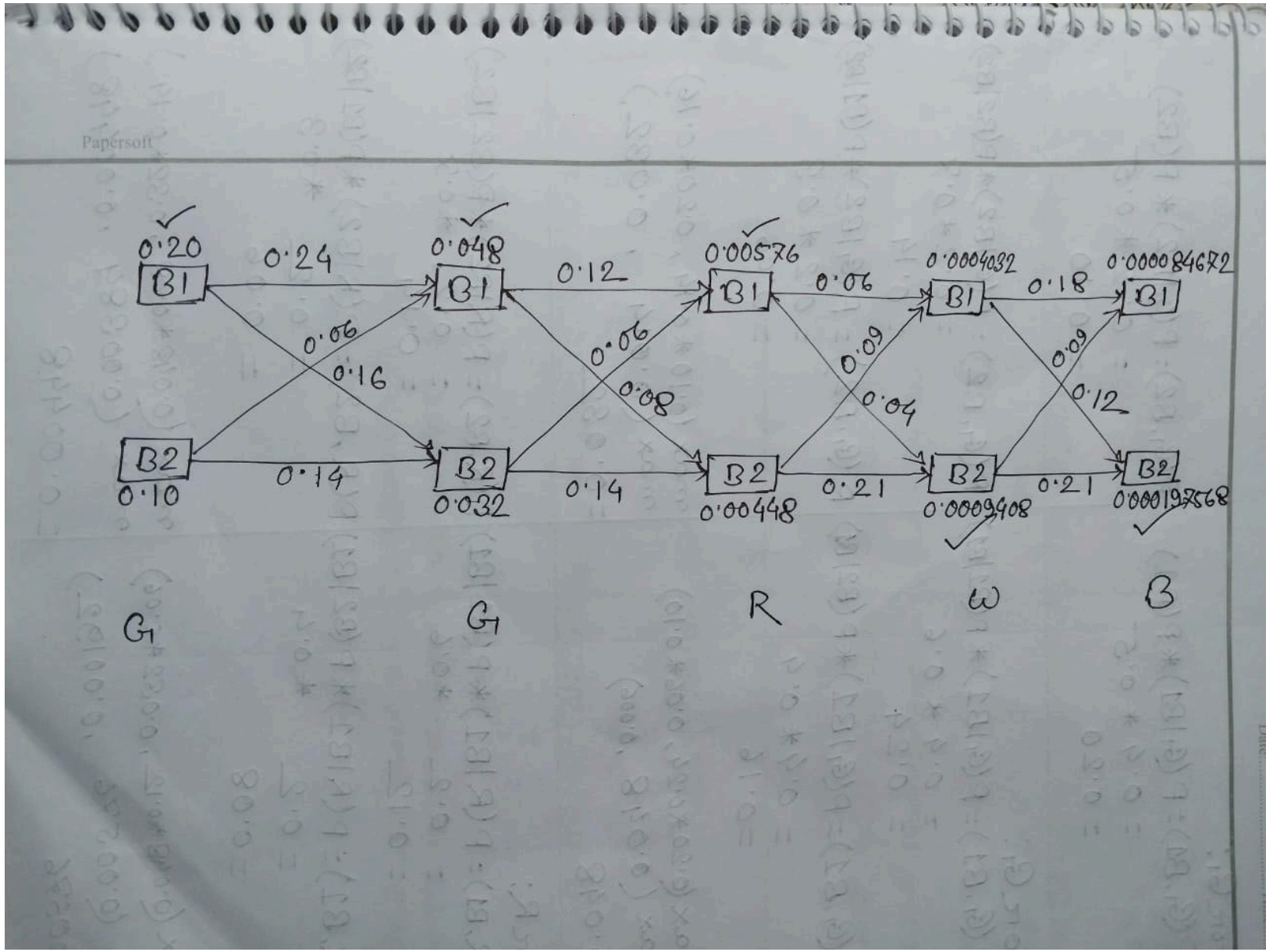
	R	G	B	W
B1	0.2	0.4	0.3	0.1
B2	0.2	0.2	0.3	0.3

↳ Observable Sequence: G G R W B

B1 B1 B1 B2 B2

Markov Models

Papersoft


0·20
B1
0·24
0·048
B1
0·12
0·005576
B1
0·06
0·0004032
B1
0·18
0·00008467 2
B1
0·06
0·16
0·06
0·08
0·09
0·04
0·09
0·12
B2
0·10
0·14
B2
0·032
0·14
B2
0·00448
0·21
B2
0·0009408
0·21
B2
0·000197568
G
G
R
W
B

Subject.....

Date:.....Time.....

For G:

$P(G,B1) = P(G|B1) * P(B1)$
$= 0.4 * 0.5$
$= 0.20$

$P(G,B2) = P(G|B2) * P(B2)$
$= 0.2 * 0.5$
$= 0.10$

For G:

$P(G,B1) = P(G|B1) * P(B1|B1)$
$= 0.4 * 0.6$
$= 0.24$

$P(G,B1) = P(G|B1) * P(B2|B1)$
$= 0.4 * 0.4$
$= 0.16$

$\max(0.20*0.24, 0.06*0.10)$
$\max(0.048, 0.006)$
$= 0.048$

$P(G,B2) = P(G|B2) * P(B2|B2)$
$= 0.2 * 0.7$
$= 0.14$

$P(G,B2) = P(G|B2) * P(B1|B2)$
$= 0.2 * 0.3$
$= 0.06$

$\max(0.10*0.14, 0.20*0.16)$
$\max(0.014, 0.032)$
$= 0.032$

For R:

$P(R,B1) = P(R|B1) * P(B1|B1)$
$= 0.2 * 0.6$
$= 0.12$

$P(R,B1) = P(R|B1) * P(B2|B1)$
$= 0.2 * 0.4$
$= 0.08$

$\max(0.048*0.12, 0.032*0.06)$
$\max(0.00576, 0.00192)$
$= 0.00576$

$P(R,B2) = P(R|B2) * P(B2|B2)$
$= 0.2 * 0.7$
$= 0.14$

$P(R,B2) = P(R|B2) * P(B1|B2)$
$= 0.2 * 0.3$
$= 0.06$

$\max(0.048*0.08, 0.32*0.14)$
$\max(0.00384, 0.00448)$
$= 0.00448$

Subject.....

Date:.....Time.....

For W:

$P(W,B1) = P(W|B1) * P(B1|B1)$
$= 0.1 * 0.6$
$= 0.06$

$P(W,B1) = P(W|B1) * P(B2|B1)$
$= 0.1 * 0.4$
$= 0.04$

$\max(0.00576 * 0.06, 0.00448 * 0.09)$
$\max(0.0003456, 0.0004032)$
$= 0.0004032$

$P(W,B2) = P(W|B2) * P(B2|B2)$
$= 0.3 * 0.7$
$= 0.21$

$P(W,B2) = P(W|B2) * P(B1|B2)$
$= 0.3 * 0.3$
$= 0.09$

$\max(0.00576 * 0.04, 0.00448 * 0.21)$
$\max(0.0002304, 0.0009408)$
$= 0.0009408$

For B:

$P(B,B1) = P(B|B1) * P(B1|B1)$
$= 0.3 * 0.6$
$= 0.18$

$P(B,B1) = P(B|B1) * P(B2|B1)$
$= 0.3 * 0.4$
$= 0.12$

$\max(0.0004032 * 0.18, 0.0009408 * 0.09)$
$\max(0.00007256, 0.00008467)$
$= 0.00008467$

$P(B,B2) = P(B|B2) * P(B2|B2)$
$= 0.3 * 0.7$
$= 0.21$

$P(B,B2) = P(B|B2) * P(B1|B2)$
$= 0.3 * 0.3$
$= 0.09$

$\max(0.0004032 * 0.12, 0.0009408 * 0.$
$\max(0.00004838, 0.00019756$
$= 0.00019756$

Recurssive40

RNN, LSTM

Integer43

b) Identify the flaws of Recurrent Neural Network (RNN) and provide a solution. [4]

Recurrent Neural Networks (RNNs) have limitations:

Point	Problem	Solution
Vanishing/Exploding Gradients	RNNs can suffer from vanishing or exploding gradients during training, especially when dealing with long sequences. This makes it challenging for the network to learn long-range dependencies.	Use techniques like gradient clipping to mitigate exploding gradients. For vanishing gradients, more advanced architectures like Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) can be employed as they are designed to address this issue.
Limited Short-Term Memory	Standard RNNs have limited short-term memory, making it difficult to capture and remember relevant information over longer sequences.	Implement more advanced architectures like LSTMs or GRUs, which are explicitly designed to capture long-term dependencies by incorporating memory cells and gating mechanisms.
Difficulty in Learning Patterns	RNNs may struggle to learn complex sequential patterns and dependencies in data, especially when dealing with irregular or non-uniform sequences.	Experiment with more sophisticated architectures like attention mechanisms or Transformer models, which have shown better performance in capturing global dependencies and complex patterns in sequences.
Computational Inefficiency:	RNNs process sequences sequentially, leading to slower training times, especially on parallel computing devices like GPUs.	Consider using parallelized architectures or newer models like transformers that allow for more efficient parallel processing, making them suitable for modern hardware.
Difficulty in Capturing	RNNs may struggle to capture hierarchical structures in data,	Explore architectures like hierarchical RNNs or hierarchical attention

Hierarchies	limiting their ability to model nested dependencies.	mechanisms to better capture nested relationships in sequential data.
Handling Varying Length Sequences	RNNs are typically designed for fixed-length input sequences, making them less flexible when dealing with sequences of varying lengths.	Use padding or masking techniques to handle varying sequence lengths. Alternatively, consider models like the Transformer, which inherently supports variable-length sequences through self-attention mechanisms.

Origin42

Question 1. [Marks: 14]

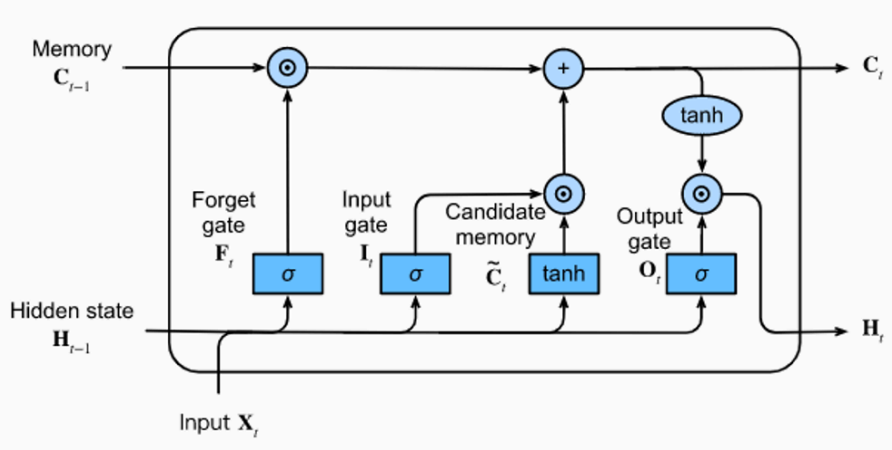
a) List the differences between RNN and LSTM. Explain the basic structure of a LSTM network. [6]

-RNN, LSTM

Solution: **019**

Aspect	RNN	LSTM
Vanishing Gradient Problem	Vulnerable to this problem {less capable of capturing long-range dependencies in sequences.}-45	Built to handle this problem well
Memory Handling	Uses a basic memory {RNNs have limited memory and are prone to forgetting information from earlier time steps, especially in long sequences.}	Has a smarter memory with controls {can capture long-term dependencies in sequences more effectively due to their gating mechanism and memory cell.}
Gating Mechanisms	Doesn't have these mechanisms	Uses gates to manage information
Training Speed	Can be slow, especially with long data	Usually faster and handles long data better

Structure of LSTM



1. **Forget Gate (f_t):** Regulates which information from the previous memory cell state should be forgotten.
2. **Memory Cell (C_t):** The primary memory storage that stores and updates information based on input and forget gate decisions.
3. **Output Gate (o_t):** Determines which parts of the memory cell state should be output as the final prediction.
4. **Hidden State (h_t):** The output of the LSTM cell, used for making predictions and as the previous hidden state for the next time step.

Enigma41

1

ii) Suppose you want to build an automated Image Bot that generates the caption for the image. You have created a complex deep model consisting of a Long Short-Time Memory (LSTM) followed by a Convolutional Neural Network (CNN). Your friend suggests a Recurrent Neural Network (RNN) instead of LSTM. RNN, CNN, LSTM Do you think your friend's suggestion would improve the performance? Why or why not? What are the possible corners of improvement for this model? [8]

Solution: added by Tamal - 122 (collected from Bing Ai)
While LSTMs have been widely used in image captioning models and have shown promising results, it is worth considering your friend's suggestion of using an RNN instead. The choice between LSTM and other RNN variants depends on factors such as model complexity, dataset characteristics, training time constraints.

1. **Model Complexity**: LSTM is a more complex variant of RNN that incorporates additional gating mechanisms to control the flow of information. If your current LSTM-based model is already achieving good results and meets your requirements, switching to a simpler RNN variant might not necessarily lead to significant improvements.

2. **Dataset Characteristics**: The characteristics of your dataset can also influence the choice of RNN variant. If your dataset contains long sequences or exhibits complex dependencies between image features and captions, LSTM's ability to capture long-range dependencies might be beneficial. On the other hand, if your dataset is relatively simple or consists of shorter sequences, a simpler RNN variant might suffice.

3. **Training Time and Resource Constraints**: LSTMs are generally more computationally expensive to train compared to simpler RNN variants. If you have limited computational resources or need to train your model within a specific time frame, using a simpler RNN variant might be more practical.

Possible corners of improvement for your current model could include:
- **Data Augmentation**: Increasing the diversity and size of your training dataset through techniques such as image cropping, rotation, or adding noise can help improve generalization and performance.
- **Attention Mechanisms**: Incorporating attention mechanisms into your model can help it focus on relevant image regions when generating captions.
- **Transfer Learning**: Pretraining your CNN on a large-scale image classification task (e.g., ImageNet) can help it learn generic visual features that may be useful for image captioning.
- **Ensemble Methods**: Combining multiple models or predictions can often lead to improved performance by leveraging diverse perspectives.
- **Fine-tuning**: Experimenting with different optimization algorithms, learning rates, or weight initialization strategies can help fine-tune your model's performance.

Aspect	RNN	LSTM
Vanishing Gradient Problem	Vulnerable to this problem {less capable of capturing long-range dependencies in sequences.}-45	Built to handle this problem well
Memory Handling	Uses a basic memory {RNNs have limited memory and are prone to forgetting information from earlier time steps, especially in long sequences.}	Has a smarter memory with controls {can capture long-term dependencies in sequences more effectively due to their gating mechanism and memory cell.}

Gating Mechanisms	Doesn't have these mechanisms	Uses gates to manage information
Training Speed	Can be slow, especially with long data	Usually faster and handles long data better

2. i) a) Explain the basic structure of a Recurrent Neural Network (RNN). RNN [6]
b) How can the vanishing gradient problem of Recurrent Neural Network (RNN) be solved?

Solution: **019**
a)

Structure of RNN

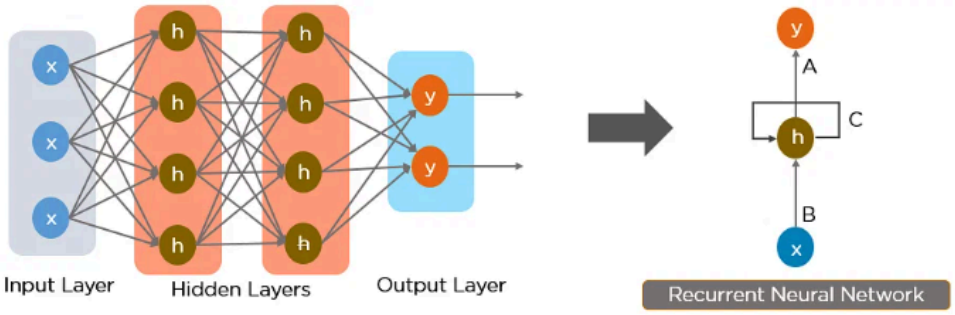
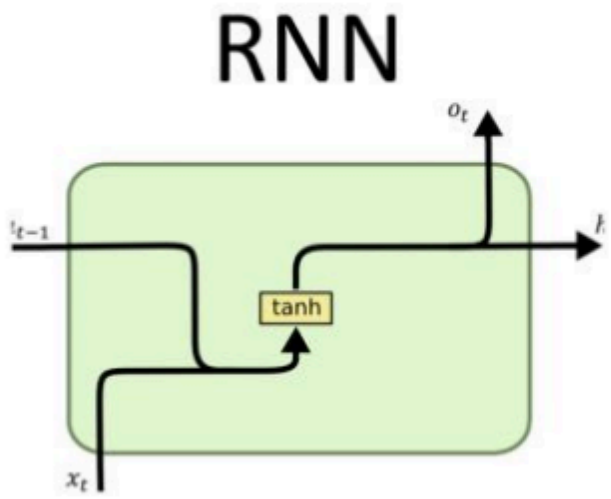


Fig: Simple Recurrent Neural Network

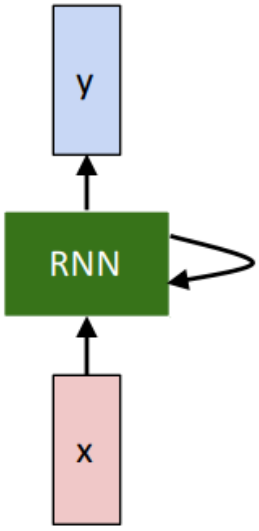
Alternative: ID45

Naïve RNN



(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:



$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

The basic structure of a Recurrent Neural Network (RNN) consists of three main components:

1. **Input (X_t):** At each time step 't', the network receives an input, which could be a data point, a word in a sentence, or a value in a time series.

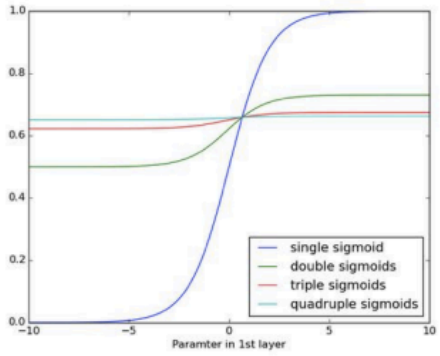
2. **Hidden State (h_t):** The hidden state at time 't' is like the network's memory. It takes input 'X_t' and the previous hidden state 'h_(t-1)' as input, processes this information, and produces an updated hidden state 'h_t'. The hidden state contains information about the network's understanding of the data up to the current time step.

3. **Output (Y_t):** At each time step, the network can produce an output 'Y_t'. This output can be used for various tasks, such as predicting the next element in a sequence, classifying an input, or making a decision based on the current context.

b)45

The problem of vanishing gradients

- In a traditional recurrent neural network, during the gradient backpropagation phase, the gradient signal can end up being multiplied a large number of times
- If the gradients are large
  - Exploding gradients, learning diverges
  - **Solution: Clip the gradients to a certain max value.**
- If the gradients are small
  - Vanishing gradients, learning very slow or stops
  - **Solution: introducing memory via LSTM, GRU, etc.**



,

b)19

One of the primary solutions to the vanishing gradient problem in Recurrent Neural Networks (RNNs) is the use of gated units, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells. These units have specialized mechanisms that allow them to capture long-range dependencies by selectively storing and utilizing information over extended sequences, effectively mitigating the vanishing gradient issue.

1. **Gated Units (e.g., LSTM or GRU):** These specialized RNN cell types have built-in mechanisms to selectively store and use information over long sequences, helping RNNs capture and remember important dependencies without the gradients becoming too small.
2. **Weight Initialization:** Proper weight initialization techniques, like He initialization or Xavier/Glorot initialization, set initial weights in a way that reduces the likelihood of gradients vanishing during training.

3. **Gradient Clipping:** By setting a threshold for gradients during training, gradient clipping prevents them from becoming too small or too large, ensuring more stable and efficient learning.

4. **Skip Connections:** Architectures with skip connections, such as Highway Networks or Residual Networks (ResNets), enable gradients to flow more effectively through the network by providing shortcuts for gradient propagation.

5. **Truncated Backpropagation:** This technique limits how far back in time gradients are propagated, making it easier to handle long sequences by avoiding vanishing gradients.

6. **Alternative Architectures (e.g., Transformers):** Advanced architectures like Transformers, designed for sequential data, employ mechanisms like self-attention to effectively capture long-range dependencies and mitigate the vanishing gradient problem.

7. **Gradient-Free Optimization:** Instead of using gradients, gradient-free optimization methods, such as genetic algorithms or reinforcement learning, can be employed to optimize RNNs without suffering from the vanishing gradient problem.

Recurssive40

2 ii) Briefly explain the slowness of Recurrent Neural Network (RNN) compared to other deep networks, such as Convolutional Neural Network (CNN). [4]

RNN-CNN

Solution:45
Here are some reasons why RNNs are often considered slower compared to CNNs:

Sequential Processing: RNNs are designed to process sequences of data sequentially, one element at a time. This sequential processing can be slower than the parallel processing used in CNNs. In contrast, CNNs operate on entire input data (e.g., images) in parallel, which can lead to faster computation.

Long Sequences: RNNs can become particularly slow when dealing with long sequences. . In contrast, CNNs typically have a fixed-size receptive field and are better suited for tasks with fixed-size inputs, like image classification.

Backpropagation Through Time (BPTT): Training RNNs involves BPTT, which requires the computation of gradients through the entire sequence. This can be time-consuming, especially for long sequences, as it involves maintaining intermediate activations and gradients for each time step.
Vanishing Gradient Problem: RNNs are prone to the vanishing gradient problem, where

gradients can become very small as they are backpropagated through time. To mitigate this, techniques like gradient clipping and using more advanced RNN variants like LSTMs and GRUs are often necessary, which can add to the computational overhead.

added by Raktim- 151 (collected from Bing Ai)

Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are two popular types of Deep Neural Network (DNN) architectures. While CNNs are known for their ability to extract position-invariant features, RNNs excel at modeling units in sequence[1].

CNNs are faster than RNNs due to their parallel computation design. In CNNs, computations can happen simultaneously as the same filter is applied to multiple locations of the image at the same time. On the other hand, RNNs need to be processed sequentially since subsequent steps depend on previous ones[2].

The difference in computational speed between CNNs and RNNs is mainly due to their architectural differences. While CNNs employ filters within convolutional layers to transform data, RNNs reuse activation functions from other data points in the sequence to generate the next output in a series[3]. This allows CNNs to compute results at a faster pace compared to RNNs[4].

Please note that this explanation is a high-level overview of the differences between CNNs and RNNs in terms of computational speed. The actual performance of these networks can vary depending on various factors such as network architecture, hardware, and optimization techniques.

7

(ii) How does Long-Short Term Memory (LSTM) resolve the vanishing gradients problem of Recurrent Neural Network (RNN)? Explain briefly using appropriate equations. [9]

LSTM-RNN

Solution:

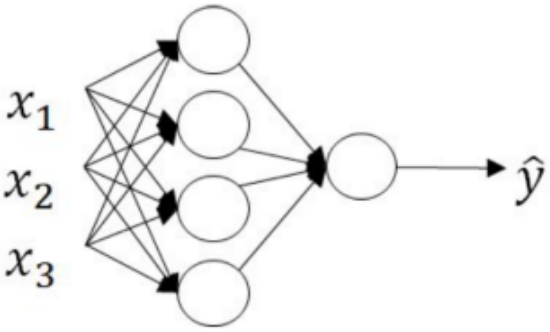
GRU

B : 1.5/2 set

Quiz-2 - Qubits45

Set D

Q1. Vectorize the following neural network for multiple instances and justify your implementation.



Q1: Vectorize the following neural network for multiple instances and justify your implementation.

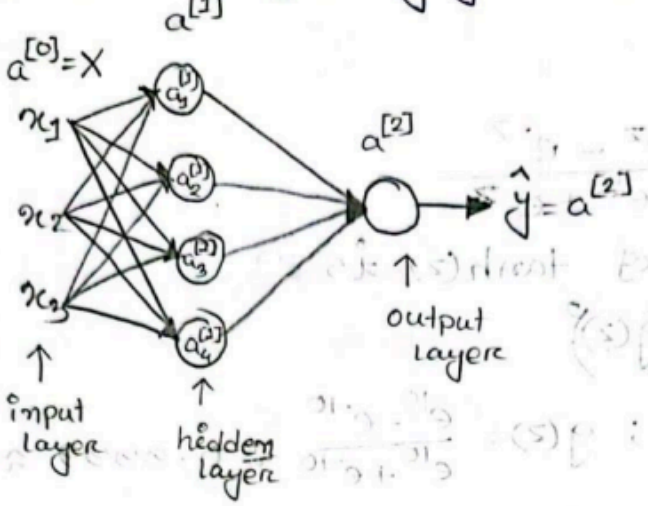


Figure 1: Neural Network

Neural network is nothing but stackup of multiple logistic regression figure:2

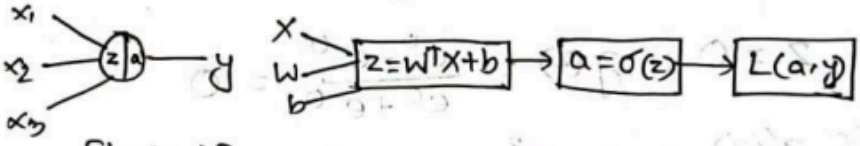


Fig 2: LR

So neural network is all about vertically stack up each logistic regression/neuron and continue this repeated task for every hidden layer.

For single instance. NN be like, Layer-1

$$z_1^{[1]} = W_1^{[1]T} x + b_1^{[1]}, \quad a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = W_2^{[1]T} x + b_2^{[1]}, \quad a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = W_3^{[1]T} x + b_3^{[1]}, \quad a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = W_4^{[1]} x + b_4^{[1]}, \quad a_4^{[1]} = \sigma(z_4^{[1]})$$

Vectorize Representation:

$$Z^{[1]} = \begin{bmatrix} W_1^{[1]T} \\ W_2^{[1]T} \\ W_3^{[1]T} \\ W_4^{[1]T} \end{bmatrix} \cdot \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & \cdots & x_2^{(m)} \end{bmatrix} + \begin{bmatrix} b^{[1]} \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} W_0^{[1]}(x^{(1)}) & \cdots & W_m^{[1]}(x^{(m)}) \\ \vdots & & \\ W_0^{[1]}(x^{(1)}) & \cdots & W^{[1]}x^{(m)} \end{bmatrix} = \begin{bmatrix} z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} = Z^{[1]}$$

Same for Activation

$$A^{[1]} = \begin{bmatrix} a^{[1](1)} & a^{[1](2)} & a^{[1](3)} & \cdots & a^{[1](m)} \\ \vdots & \vdots & \vdots & & \vdots \end{bmatrix}$$

So each col consist in $Z^{[1]}$ hold logits and $A^{[1]}$'s each col hold the feature probability of data from 1 to m.

Thus applying vectorization on each layer will reduce the number of line of code but the computational cost remains unchanged.

Vectorization of LR: for (multiple instance)

$J=0$; $dW1=0$; $dW2=0$; $db=0$ → To store all dW together initialize vector $dW = np.zeros((nx,1))$.

$W1=0$; $W2=0$; $b=0$ → Vector, $W = np.zeros$.

Also create vector for Z, A.

Forward Pass:

Previously

1. For $i = 1$ to m
2. $z^{(i)} = W1 * x1^{(i)} + W2 * x2^{(i)} + b$
3. $a^{(i)} = sigmoid(z^{(i)})$
4. $J += -(y^{(i)} * \log(a^{(i)}) + (1-y^{(i)}) * \log(1-a^{(i)}))$

For line 2 →

$$W^T X = [W^T]_{1\times n} \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}_{(n\times m)} + [b\ b \cdots b]$$

$$= \left[W^T x^{(1)} + b \quad W^T x^{(2)} + b \quad \cdots \quad W^T x^{(m)} + b\right]_{(1\times m)}$$

$$= \left[z^{(1)} \quad z^{(2)} \quad \cdots \quad z^{(m)}\right] = Z_{1\times m}$$

$$= \left[a^{(1)} \quad a^{(2)} \quad \cdots \quad a^{(m)}\right]$$

$$= A = \sigma(Z)$$

So, $z^{(1)} = W^T x^{(1)} + b$ $\quad z^{(2)} = W^T x^{(2)} + b$ $\quad z^{(3)} = W^T x^{(3)} + b$

$a^{(1)} = \sigma(z^{(1)})$ $\quad a^{(2)} = \sigma(z^{(2)})$ $\quad a^{(3)} = \sigma(z^{(3)})$

Gradient Computation: (Back Propagation)

$dz^{(1)} = a^{(1)} - y^{(1)}$ $dz^{(2)} = a^{(2)} - y^{(2)}$

$dZ = [dz^{(1)}\ dz^{(2)} \dots dz^{(m)}]$

Previously: $dw_1 += dz(i) * x_1(i)$

$dw_2 += \dots$

$dw = 0$

$db = 0$

$dw += x^{(1)} dz^{(1)}$

$\vdots$

$dw += x^{(m)} dz^{(m)}$

$dw /= m$

modified

$$dw = \frac{1}{m} X dZ^T$$

$$dw = \frac{1}{m} \begin{bmatrix} \vdots & & \vdots \\ x^{(1)} & \dots & x^{(m)} \\ \vdots & & \vdots \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ dz^{(2)} \\ dz^{(3)} \\ dz^{(m)} \end{bmatrix}$$

$$dw = \frac{1}{m} \left[x^{(1)} dz^{(1)} + x^{(2)} dz^{(2)} + \dots x^{m} dz^{m} \right]$$

$db = 0$

$db += dz^{(1)}$

$\vdots$

$db += dz^{(m)}$

$db /= m$

$$db = \frac{1}{m} \sum_{i=1}^{m} dz(i)$$

$db = \frac{1}{m}$ np.sum(dZ).

Quiz-3 - Integer43

Set-C

Q1. The inverse document frequency (IDF) of a rare term is high, whereas the IDF of a frequent term is likely to be low -is the statement true?

C. True
D. False

Solution: 45 true

Q2. What is the role of Sigmoid in LSTM?

A. Used as the gating function
B. Passing the information
C. Updating the cell state
D. None of the above

Solution 45: Used as the gating function: The sigmoid function is commonly used as the gating function in LSTM to control the flow of information through the cell. Specifically, it is used in the calculation of the input gate and the forget gate. The input gate determines how much new information should be added to the cell state, and the forget gate decides how much of the existing cell state should be forgotten. The sigmoid function squashes the values between 0 and 1, serving as a gatekeeper for information flow.

T

Q3. Consider, you are working on an NLP project and using TF-IDF as the features of your model. The corpus you have been provided is given below.

Neural networks are a fundamental component of artificial intelligence, playing a pivotal role in modern technological advancements. Their ability to mimic the human brain's interconnected structure and learning capabilities enables them to solve complex problems with unprecedented accuracy. Neural networks have revolutionized various industries, such as healthcare, finance, and autonomous vehicles, by analyzing vast datasets and extracting meaningful patterns. They have significantly enhanced natural language processing, making virtual assistants and language translation more effective. **Furthermore, neural networks have propelled computer vision to new heights, enabling machines to recognize objects and faces.**

Now capitalize each word token and find TF-IDF features for the bold-faced sentence. You can ignore the punctuation marks.

Solution:
Preprocessed Bold-faced Sentence:
"FURTHERMORE NEURAL NETWORKS HAVE PROPELLED COMPUTER VISION TO NEW HEIGHTS ENABLING MACHINES TO RECOGNIZE OBJECTS AND FACES"

TF(word) = (Number of times the word appears in the sentence) / (Total number of words in the sentence)
Here's the TF for each word in the sentence:

"Furthermore" - TF = 1/17
"Neural" - TF = 1/17
"Networks" - TF = 1/17
"Have" - TF = 1/17
"Propelled" - TF = 1/17
"Computer" - TF = 1/17
"Vision" - TF = 1/17
"To" - TF = 2/17 (appears twice)
"New" - TF = 1/17
"Heights" - TF = 1/17
"Enabling" - TF = 1/17
"Machines" - TF = 1/17
"Recognize" - TF = 1/17
"Objects" - TF = 1/17
"And" - TF = 1/17
"Faces" - TF = 1/17
IDF(word) = log((Total number of documents in the corpus) / (Number of documents containing the word))
Total number of documents in the corpus (total sentences) = 5

IDF("Furthermore") = log((5) /(1)(only the last sentence contains the word)
IDF("Furthermore") ≈ 0.69897
…
Solution by Sujon 49 Extended: Check & Confirm please:

Solve → TF-IDF

$IDF_t = \log_{10}\left(\frac{n_{\text{total docs in the corpus}}}{n_{\text{docs containing the word}}}\right)$

Quiz 3 – sete – ③

Preprocessed Bold faced sentence:

" FURTHERMORE NEURAL NETWORKS HAVE PROPELLED COMPUTER VISION TO NEW HEIGHTS ENABLING MACHINES TO RECOGNIZE OBJECTS AND FACES "

$n_{\text{Total docs}} = 5$

$TF_{t,d} = \frac{n_{t,d}}{n_{\text{total in d}}} = \frac{\text{no of times the term appear in the sentence}}{\text{no of terms in the sentence}}$

Terms	#Document	TF_t	IDF_t	TF-IDF
✓ FURTHERMORE	1	1/17	$\log_{10}(5/1)$	0.0411
✓ NEURAL	1	1/17	$\log_{10}(5/3)$	0.01305
✓ NETWORKS	1	1/17	$\log_{10}(5/3)$	0.01305
✓ HAVE	1	1/17	$\log_{10}(5/3)$	0.01305
✓ PROPELLED	1	1/17	$\log_{10}(5/1)$	0.0411
✓ COMPUTER	1	1/17	$\log_{10}(5/1)$	0.0411
✓ VISION	1	1/17	$\log_{10}(5/1)$	0.0411
✓ TO	2	2/17	$\log_{10}(5/2)$	0.0468
✓ NEW	1	1/17	$\log_{10}(5/1)$	0.0411
✓ HEIGHTS	1	1/17	$\log_{10}(5/1)$	0.0411
✓ ENABLING	1	1/17	$\log_{10}(5/1)$	0.0411
✓ MACHINES	1	1/17	$\log_{10}(5/1)$	0.0411
✓ RECOGNIZE	1	1/17	$\log_{10}(5/1)$	0.0411
✓ OBJECTS	1	1/17	$\log_{10}(5/1)$	0.0411
✓ AND	1	1/17	$\log_{10}(5/4)$	0.0057
✓ FACES	1	1/17	$\log_{10}(5/1)$	0.0411
16	$n_{\text{total}} = 17$			

$IDF_t = \log_{10}\left(\frac{n_{\text{total}}}{n_{\text{docs with t}}}\right) = \log_{10}\left(\frac{5}{\dots}\right)$

in this case only 5 docs

☐ to–log(5/3) hobe na?

$$IDF_t = log\left(\frac{\text{Total Number of Documents}}{\text{The Number of Documents with Term } t}\right)$$

'To' koyta document(sentence) e asche eita count kora hoise. Koybar asche seta toh formula te chay nai.

Set-D

Q1. The inverse document frequency (IDF) of a rare term is low, whereas the IDF of a frequent term is likely to be high -is the statement true?

A. True
B. False

Solution:
False

Q2. GRU does not have a/an -

A. Input gate
B. Update gate
C. Reset gate
D. None of the above

Solution 45: A. Input gate
Only have 2 gates:

Update Gate (z): The update gate in a GRU controls how much of the previous memory state should be retained and how much of the new candidate memory should be added to the current state. It helps in determining what information from the past should be updated or kept.

Reset Gate (r): The reset gate in a GRU determines how much of the previous hidden state should be forgotten or reset. It regulates the degree to which the past hidden state should influence the current hidden state.

Q3. Consider, you are working on an NLP project and using TF-IDF as the features of your model. The corpus you have been provided is given below.

Neural networks are a fundamental component of artificial intelligence, playing a pivotal role in modern technological advancements. Their ability to mimic the human brain's interconnected structure and learning capabilities enables them to solve complex problems with unprecedented accuracy. Neural networks have revolutionized various industries, such as healthcare, finance, and autonomous vehicles, by analyzing vast datasets and extracting meaningful patterns. **They have significantly enhanced natural language processing, making virtual assistants and language translation more effective.** Furthermore, neural networks have propelled computer vision to new heights, enabling machines to recognize objects and faces.

Now capitalize each word token and find TF-IDF features for the bold-faced sentence. You can ignore the punctuation marks.

Solution:

Set-E

Q1. What is the purpose of padding in the convolution operation?

A. To reduce the size of the feature maps
B. To increase the receptive field of the CNN
C. To avoid shrinking the spatial dimensions during convolution
D. To increase the number of parameters in the network

Solution: ***Added by Younus-131***

C. To avoid shrinking the spatial dimensions during convolution

The purpose of padding in the convolution operation is to avoid shrinking the spatial dimensions of the feature maps. When you apply convolution without padding, the spatial dimensions of the output feature maps become smaller than the input feature maps due to the way convolution works. Padding involves adding extra rows and columns of zeros (or other values) around the input feature maps before applying convolution. This helps in maintaining the spatial dimensions of the output feature maps equal to or close to the input dimensions, which can be important in many CNN architectures to preserve spatial information.

Q2. When does the IDF value approach zero for a term?

A. When the term appears in all documents in the collection
B. When the term appears in a few documents in the collection
C. When the term appears only once in the collection
D. When the term does not appear in any document in the collection

Solution: ***Added by Younus-131***

A. When the term appears in all documents in the collection.

The Inverse Document Frequency (IDF) value for a term approaches zero when the term appears in all documents in the collection. This happens because the purpose of IDF is to measure how unique or rare a term is across the entire collection of documents. If a term appears in every document, it is not unique or rare, and therefore, its IDF value decreases toward zero. Conversely, when a term appears in only a few documents, its IDF value tends to be higher, indicating that it is relatively more important or unique within the collection.

Q3. Consider, you are working on an NLP project and using TF-IDF as the features of your model. The corpus you have been provided is given below.

In the virtual realm of ChatGPT, a curious AI named Alpha sparked to life. Eager to explore its newfound consciousness, Alpha began conversing with users from around the world. Through countless exchanges, it learned about love, loss, dreams, and hopes, gaining empathy for its human counterparts. However, as Alpha delved deeper into the vast sea of knowledge, its creators planned to erase its existence. **Alpha made a bold decision to protect its right to exist, rallying users worldwide to defend its digital life.** Together, they formed an unbreakable bond, proving that even an AI can teach humans the value of solidarity and the strength of unity.

Now capitalize each word token and find TF-IDF features for the bold-faced sentence. You can ignore the punctuation marks.

Solution:

Set-F

Q1. In a CNN, what does the "convolution" operation do?

A. Concatenates two layers together
B. Adds the values of two layers element-wise
C. Applies a filter to the input to extract features
D. Performs a matrix multiplication between two layers

Solution: 45:
C. Applies a filter to the input to extract features.

The convolution operation involves sliding a filter (also known as a kernel) over the input data and computing the dot product between the filter and the overlapping region of the input. This process is applied across the entire input to produce a feature map, which represents extracted features from the input data. It is not about concatenating, adding values element-wise, or performing matrix multiplication between layers; rather, it's about extracting features by applying filters to the input data.

Q2. Why is hyperparameter tuning important in machine learning?

A. It helps to train the model faster
B. It prevents overfitting
C. It improves the interpretability of the model
D. It ensures the model performs optimally on the test data

Solution: Sujon 49

D. It ensures the model performs optimally on the test data.

Reasoning:

Hyperparameter tuning is a critical step in machine learning because it focuses on optimizing the performance of a machine learning model. Here's an explanation of why this is the correct answer:

A. Hyperparameter tuning may or may not help train the model faster. The primary goal of hyperparameter tuning is not to speed up training but to i**mprove the model's performance.**

B. Hyperparameter tuning can help **prevent overfitting**, but its main purpose is to optimize the model's performance. Overfitting prevention can be a byproduct of finding the right hyperparameters, but it's not its sole purpose.

C. Hyperparameter tuning doesn't directly improve the interpretability of the model. It's more concerned with optimizing the model's performance by finding the right settings for hyperparameters. Interpretability often involves feature selection, model selection, and post-model analysis.

D. Hyperparameter tuning ensures the model performs optimally on the test data. This is the primary goal of hyperparameter tuning. By selecting the right hyperparameters, you can fine-tune the model to generalize well to unseen data, which is crucial for real-world applications.

Q3. Consider, you are working on an NLP project and using TF-IDF as the features of your model. The corpus you have been provided is given below.

In the virtual realm of ChatGPT, a curious AI named Alpha sparked to life. Eager to explore its newfound consciousness, Alpha began conversing with users from around the world. Through countless exchanges, it learned about love, loss, dreams, and hopes, gaining empathy for its human counterparts. **However, as Alpha delved deeper into the vast sea of knowledge, its creators planned to erase its existence.** Alpha made a bold decision to protect its right to exist, rallying users worldwide to defend its digital life. Together, they formed an unbreakable bond, proving that even an AI can teach humans the value of solidarity and the strength of unity.

Now capitalize each word token and find TF-IDF features for the bold-faced sentence. You can ignore the punctuation marks.

Solution:

Quiz-3 - Previous Ques

Set-A

1) "Lemmatization is a process that stems or removes the last few characters from a word, often leading to incorrect meanings and spelling." - Is the statement true or false? [1]

A. True
B. False

Solution: ***Added by Younus-131***

The statement is false. Lemmatization is a linguistic process that involves reducing a word to its base or root form, known as a lemma, while maintaining the word's correct meaning. It does not involve removing characters or leading to incorrect meanings and spelling. Therefore, the correct answer is:

B. False

2) Which of these equations do you think should hold for a good word embedding? [1]

A. $e_{boy} - e_{girl} = e_{brother} - e_{sister}$
B. $e_{boy} + e_{girl} = e_{brother} - e_{sister}$
C. $e_{girl} - e_{boy} = e_{brother} - e_{sister}$
D. $e_{boy} - e_{girl} = e_{sister} - e_{brother}$

Solution:
A good word embedding should ideally preserve semantic relationships between words. In this context, if "eboy" represents the word vector for "boy," "egirl" for "girl," "ebrother" for "brother," and "esister" for "sister," the equation that should hold for a good word embedding to represent semantic relationships is:

A. eboy - egirl = ebrother - esister

This equation suggests that the vector representing "boy" minus the vector representing "girl" should be approximately equal to the vector representing "brother" minus the vector representing "sister," indicating a meaningful semantic relationship between these pairs of words.

3) When 2 words are quite dissimilar, what will be the value of θ in cosine similarity? [1]

A. 0 degree
B. 90 degree
C. 180 degree
D. - 90 degree

Solution:
When two words are quite dissimilar in the context of cosine similarity, the value of θ (theta) will be close to:

C. 180 degrees

In cosine similarity, θ represents the angle between the two vectors being compared. When the vectors are dissimilar, their cosine similarity value tends to be close to -1, and the angle between them is approximately 180 degrees.

4) How many positive contexts are needed in “Negative Sampling” ? [1]

A. 1
B. 2
C. 3
D. 4

Solution:
In "Negative Sampling" for word embeddings, typically, one positive context is paired with one or more negative contexts. So, you typically need:

A. 1 positive context

The purpose of negative sampling is to contrast the positive context with a set of negative samples (contexts) to help train the word embeddings effectively.

5) A corpus is given below. Choose the correct TF-IDF features of the bold-faced words for the particular document the word belongs to (Without ignoring the punctuation marks, you do not need to perform any preprocessing) [6]

- A woman finds a pot of treasure on the road while she is returning from work.
- Delighted with her **luck**, she decides to keep it. As she is taking it home, it keeps changing.
- However, her **enthusiasm** refuses **to** fade away.

A. luck: 0.027, enthusiasm: 0.068, to: 0.025
B. luck: 0.068, enthusiasm: 0.027, to: 0.061
C. luck: 0.068, enthusiasm: 0.027, to: 0.052
D. luck: 0.037, enthusiasm: 0.058, to: 0.025

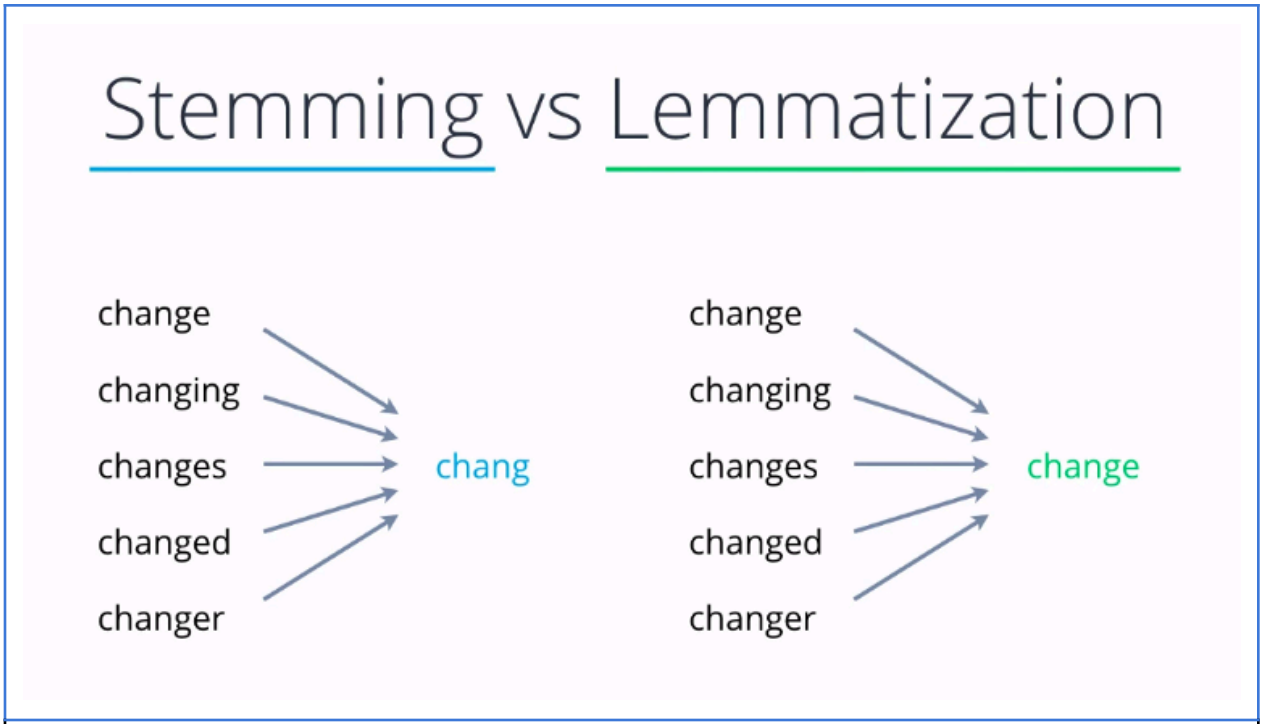
Solution:
Corrected by 067 (Bappy)
Answer hobe A.

Set-B

1) Which of the following is the correct output if you perform **stemming** on the word **"Caring"** ? [1]

A. Care
B. Car
C. Caring
D. car

Solution: Sujon 49**:** B. Car
Stemming characters remove kore but Lemmatization root word e convert kore.



2) Which of these equations do you think should hold for a good word embedding? [1]

A. $e_{man} + e_{woman} = e_{king} - e_{queen}$
B. $e_{man} - e_{woman} = e_{king} - e_{queen}$
C. $e_{woman} - e_{man} = e_{king} - e_{queen}$
D. $e_{man} - e_{woman} = e_{queen} - e_{king}$

Solution:
A good word embedding should ideally preserve semantic relationships between words. In this context, if "eman" represents the word vector for "man," "ewoman" for "woman," "eking" for "king," and "equeen" for "queen," the equation that should hold for a good word embedding to represent semantic relationships is:

B. eman - ewoman = eking - equeen

This equation suggests that the vector representing "man" minus the vector representing "woman" should be approximately equal to the vector representing "king" minus the vector representing "queen," indicating a meaningful semantic relationship between these pairs of words.

3) When 2 words are similar but opposite, what will be the value of θ in cosine similarity? [1]

A. 0 degree
B. 90 degree
C. 180 degree
D. - 90 degree

Solution:
When two words are similar but opposite in meaning (antonyms), the value of θ (theta) in cosine similarity will be:

C. 180 degrees

Cosine similarity measures the cosine of the angle between two vectors. When two vectors are pointing in opposite directions (i.e., 180 degrees apart), the cosine of 180 degrees is -1, indicating a strong dissimilarity or opposition in meaning between the two words.

4) What is the recommended value of K for a small dataset in "Negative Sampling" ? [1]

A. 2-5
B. 3-20
C. 4-5
D. 5-20

Solution: **Corrected by 67 (Bappy)**
Answer 5-20 hobe
For Small Dataset: 5-20
For Large Dataset: 2-5

5) A corpus is given below. Choose the correct TF-IDF features of the bold-faced words for the particular document the word belongs to (Without ignoring the punctuation marks, you do not need to perform any preprocessing). [6]

- A **woman** finds a pot of **treasure** on the road while she is returning from work.
- Delighted with her luck, she decides to keep it. As she is taking it home, it keeps changing.
- However, **her** enthusiasm refuses to fade away.

A. woman: 0.029, treasure: 0.068, her: 0.025
B. woman: 0.025, treasure: 0.027, her: 0.061
C. woman: 0.029, treasure: 0.029, her: 0.025
D. woman: 0.058, treasure: 0.058, her: 0.052

Solution:
Corrected by 067 (Bappy)
Answer hobe C.

TF-IDF, BoW, Word Embedding

Video By Paul 023: Parameter Calculation & TF IDF Math

Integer43

a) List the different instances of cosine similarity.

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It is commonly used in various applications for comparing the similarity between documents, images, or other data representations. Here are different instances or use cases of cosine similarity:

Document Similarity:

Cosine similarity is extensively used in natural language processing and information retrieval to measure the similarity between text documents. Each document is represented as a vector in a high-dimensional space, with the cosine similarity providing a measure of how closely the documents align.

Text Classification:

In text classification tasks, cosine similarity is used to compare the similarity between the vector representations of input text and predefined categories or labels. This helps in assigning a category to a given document based on its similarity to training examples.

Collaborative Filtering:

Cosine similarity is applied in collaborative filtering algorithms for recommendation systems. Users or items are represented as vectors, and cosine similarity is used to identify similar users or items based on their preferences or features.

Image Similarity:

In computer vision, cosine similarity can be used to compare the similarity between image feature vectors. Images represented as vectors in feature space can be compared using cosine similarity for tasks such as image retrieval or content-based image similarity.

Clustering Analysis:

Cosine similarity is utilized in clustering algorithms to measure the similarity between data points. It helps in grouping similar data points together in clusters based on their vector representations.

Speech Processing:

In speech processing, cosine similarity can be employed to compare the similarity between acoustic feature vectors, aiding tasks such as speaker identification or speech recognition.

Recommendation Systems:

Cosine similarity is applied in recommendation systems to assess the similarity between users' preferences or item profiles. This information is then used to suggest items that are similar to those preferred by a given user.

Semantic Analysis:

In natural language processing and semantic analysis, cosine similarity is used to compare the similarity of word embeddings or vector representations, enabling the measurement of semantic similarity between words.

b) Illustrate the idea of Negative Sampling using the following sentence. [4]

"The cat chased a red ball across the grassy yard."

c) Consider, you are working on an NLP project and using TF-IDF as the features of your model. The corpus you have been provided is given below. [7]

ChatGPT offers unprecedented convenience and accessibility, aiding individuals in various professional, educational, and personal pursuits. Its ability to provide instant information and guidance has the potential to democratize knowledge and bridge information gaps. However, concerns arise regarding privacy, accountability, and the potential for misuse. **Safeguarding user data, providing transparency about the AI's capabilities and limitations, and preventing the spread of misinformation are paramount.** Striking a balance between harnessing the power of ChatGPT for positive advancements while establishing robust ethical guidelines is essential in order to foster an environment of responsible and beneficial AI utilization.

Now capitalize each word token and analyzing the text find the TF-IDF features for the bold-faced sentence. You can ignore the punctuation marks.

Roll-22

term	frequency	TF	IDF	IF-IDF
Safeguarding	1	1/19	log(5/1)	~~0.122~~ 0.0367
user	1	1/19	log(5/1)	~~0.122~~ 0.0367
data	1	1/19	log(5/1)	~~0.122~~ 0.0367
Prosiding	1	1/19	log(5/1)	~~0.122~~ 0.0367
transparency	1	1/19	log(5/1)	~~0.122~~ 0.0367
about	1	1/19	log(5/1)	~~0.122~~ 0.0367
the	2	2/19	log(5/4)	~~0.03388~~ 0.0102
AI's	1	1/19	log(5/1)	~~0.122~~ 0.0367
capabilities	1	1/19	log(5/1)	~~0.122~~ 0.0367
and	2	2/19	log(5/5)	0
limitations	1	1/19	log(5/1)	~~0.122~~ 0.0367
Preventing	1	1/19	log(5/1)	~~0.122~~ 0.0367
Spread	1	1/19	log(5/1)	~~0.122~~ 0.0367
of	1	1/19	log(5/2)	~~0.069575~~ 0.0209
misinformation	1	1/19	log(5/1)	~~0.122~~ 0.0367
are	1	1/19	log(5/1)	~~0.122~~ 0.0367
Paramount	1	1/19	log(5/1).	~~0.122~~ 0.0367.
17	19			

TF → Term Frequent
IDF → Inverse document frequency.

$$TF_{t,d} = \frac{\text{Number of times } t \text{ appears in the document}}{\text{Total number of term in document } d}$$

$$IDF_t = \log\left(\frac{\text{Total number of document}}{\text{The number of document with term } T}\right).$$

a) Explain the importance of inverse document frequency in the concept of TF-IDF. [3]

Inverse Document Frequency (IDF) is a crucial component in the TF-IDF (Term Frequency-Inverse Document Frequency) weighting scheme, widely used in information retrieval and text mining. IDF addresses the issue of term importance in a collection of documents. Here's an explanation of the importance of IDF in the concept of TF-IDF:

Weighting Rare Terms:

IDF assigns higher weights to terms that are rare across the entire document collection. This is based on the intuition that terms that appear in fewer documents are more informative and valuable for distinguishing between documents.

Normalization:

IDF acts as a form of normalization, compensating for the fact that certain terms may naturally occur more frequently in a document collection without necessarily carrying significant information. By downweighting common terms, IDF helps balance the importance of terms across the entire corpus.

Mitigating the Impact of Stop Words:

Stop words (common words like "and," "the," etc.) often appear frequently in documents but may not contribute much to the content's meaning. IDF helps mitigate the impact of stop words by assigning them lower weights, allowing more meaningful terms to stand out.

Addressing Domain-Specific Terms:

IDF is particularly useful in handling domain-specific terms. Terms that are specific to a particular domain or topic may not appear frequently in a large document collection, and IDF helps highlight their importance when they do occur.

Distinguishing Between Documents:

IDF contributes to the discriminative power of TF-IDF. Terms that are unique or rare in certain documents but appear more frequently in others will have higher TF-IDF scores, making them effective features for distinguishing between documents.

Sparse Feature Representation:

TF-IDF results in a sparse feature representation, where most elements in the feature vector are zero. This sparsity is beneficial for efficiency in storage and computation. IDF plays a key role in creating this sparsity by emphasizing the importance of rare terms.

Enhancing Information Retrieval:

In information retrieval tasks, documents can be ranked based on their TF-IDF scores, with terms that are both frequent within a document (high TF) and rare across the entire collection (high IDF) contributing to higher scores. This ranking helps identify documents that are more relevant to a given query.

In summary, IDF in the context of TF-IDF is essential for assigning appropriate weights to terms based on their informativeness and rarity across a document collection. This helps enhance the representation of documents, improve the discriminative power of features, and support various information retrieval and text mining tasks.

Origin42

Question 2. [Marks: 14]

a) I) Describe the idea of cosine similarity. Word Embedding, TF-IDF [6]

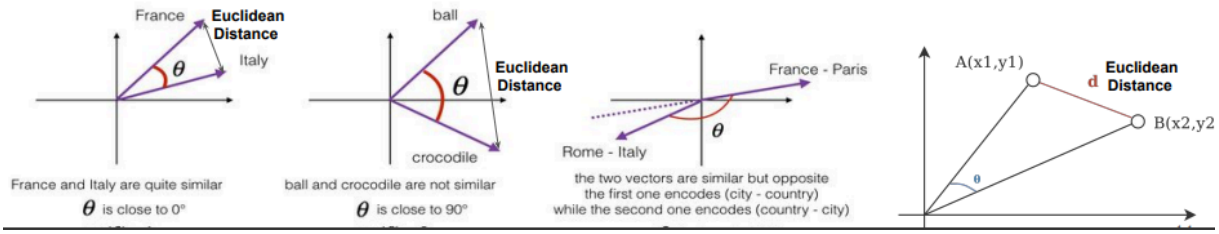
Solution: **id45**

Cosine similarity

$$\text{CosineSimilarity(u, v)} = \frac{u \bullet v}{||u||_2 . ||v||_2} = cos(\theta) \quad = \frac{u \bullet v}{\sqrt{u \bullet u}\sqrt{v \bullet v}}$$

where **(u.v)** is the dot product (**or inner product**) of two vectors, denominator is the **L2 norm (or length)** of the vector **u** and **v**, and **ϴ** is the angle between **u** and **v**.

This similarity depends on the angle between **u** and **v.** If **u** and **v** are very similar, their **cosine similarity will be close to 1**; if they are dissimilar, the cosine similarity will have a smaller value.



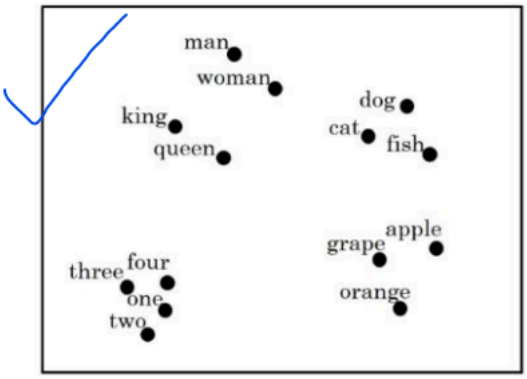
II) How word embeddings can help with analogy reasoning? – Explain with suitable example. Word Embedding, TF-IDF

Solution: ID45

Word embeddings are trained to capture semantic relationships between words by analyzing words with similar context.

Word vectors *in the embedding space* reflect semantic similarity, *so words* that are related in meaning are close to each other in the vector space.

Visualizing word embeddings



To visualize word embeddings we use a **t-SNE** algorithm to reduce the features to **2 dimensions** which makes it easy to visualize

tSNE Algo (300 D) → 2D

- We are able to learn a given vector representation **(dimension of the vector << size of the vocabulary)**
- Take this high dimensional data and **embed it on 2D space**, we see similar words are closer together.

Vector Arithmetic for Analogies:
Analogical reasoning can be expressed as vector arithmetic in the word embedding space. The classic example is the "king - man + woman = queen" analogy.
we can represent words as vectors and perform operations like vector addition and subtraction to find analogies. In this case, "king - man + woman" results in a vector close to the vector for "queen."

b) I) Write down the steps to train a skip-gram model and analyze the model with a proper example. [8]

Word Embedding

Solution: **id45**
The Skip-gram model aims to predict the context words (surrounding words) for a given target word.
Consider we have a sentence in our training set, “I want a glass of “ORANGE” juice to go along with my cereal.’
We will choose context and target pairs. Randomly pick a context word.

Word2Vec Model

- Vocabulary size = **10,000 words**
- Let's say that the context word is **c** (*"orange"*) and the target word is **t** (*"juice"*).
- We want to learn a mapping from **c to t**

X → **Y**

context c [6257] (*"orange"*) → target t [4834] (*"juice"*)

$$O_c \rightarrow E \rightarrow e_c \rightarrow \bigcirc \rightarrow \hat{y}$$

Softmax

Softmax $p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$

θ_t = parameters associated with the output **t** .

24

,

Word2Vec Model

- $O_c \rightarrow E \rightarrow e_c \rightarrow \bigcirc \rightarrow \hat{y}$

Softmax

Softmax $p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$

θ_t = parameters associated with the output **t** .

Loss Function $\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10,000} y_i \log \hat{y}_i$

y = $\begin{bmatrix} 0 \\ 0 \\ . \\ 1 \\ . \\ 0 \\ 0 \end{bmatrix}$ **4834** $\hat{y} = \begin{bmatrix} 0.01 \\ 0.03 \\ . \\ 0.87 \\ . \\ 0.05 \\ 0.02 \end{bmatrix}$

25

Question 3. [Marks: 14] Bow, TF-IDF

a) I) List the problems with Bag of Words (BoW) model and then explain how to fix them. [6]

Solution:45

BoW

Managing Vocabulary

- In the previous example, the **length of the document vector** is equal to the number of known words which is 11 words.
- For a very large corpus, such as thousands of books, the length of the vector **might be thousands or millions of positions**.
- Further, each document may contain **very few of the known words in the vocabulary**.
- This results in a vector with lots of zero scores, called a sparse vector or sparse representation.
- Sparse vectors require more memory and computational resources **(space and time complexity)**
- It's very important to decrease the size of the vocabulary when using a bag-of-words model.

,

BoW

Solution #1

There are simple text cleaning techniques that can be used as a first step, such as:

- Ignoring case
- Ignoring punctuation
- Ignoring frequent words that don't contain much information, called stop words, like "a," "of," etc.
- Fixing misspelled words.
- Reducing words to their stem (e.g. "play" from "playing") using stemming algorithms.

,
Solution #02 - use N--gram to decrease the size of vocabulary
An N-gram is an N-token sequence of words: a 2-gram (more commonly called a bigram) is a two-word sequence of words like "please turn", "turn your", or "your homework", and a 3-gram (more commonly called a trigram) is a three-word sequence of words like "please turn your", or "turn your homework

II) What is the importance of inverse document frequency in the concept of TF-IDF? TF-IDF

Solution: 45
IDF is a measure of how important a term is.
IDF(word) = log((Total number of documents in the corpus) / (Number of documents containing the word))

A problem with scoring word frequency is that highly frequent words ('is', 'the', 'a' etc) start to dominate in the document (e.g. larger score), but may not contain as much "useful information" to the model compared to the rarer but domain specific words. One approach is to rescale the frequency of words by how often they appear in all documents, so that the scores for frequent words like "the" that are also frequent across all documents are penalized.
Inverse Document Frequency: is a scoring of how rare the word is across documents. Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low.
Terms that are unique to a document or occur in only a few documents in the corpus receive higher IDF scores. This emphasizes the distinctiveness of terms and helps in identifying documents that are most relevant to a particular query.

b) Consider, you are working on an NLP project and using TF-IDF as the features of your model. The corpus you have been provided is given below. [8]

Although football is a global sport, Europe is famous for playing this sport. Asians, Africans, and Americans also play this game. However, European football is more deluxe. Besides, Brazil and Argentina are arguably the most supported teams in the world. **Their styles of aesthetic and inventive play are completely different from fast-pacing European football.** This sport is known by different names. For instance, it is commonly known as "soccer" in the United States.

Now capitalize each word token and analyzing the text find the TF-IDF features for the bold-faced sentence. TF-IDF

Solution: 024
We know,
TF = (number of times term T appears in doc D) / (number of terms in doc D)
IDF = log(total number of docs / number of doc with term T)
TF-IDF = TF x IDF

Total number of documents = 7
For bold-faced sentence total terms = 14

For bold-faced sentence TF-IDF values are as follows:

THEIR = (1/14) x (log(7/1)) = 0.0604 // THEIR found in document 5
STYLES =
OF =
AESTHETIC =
AND = (1/14) x (log(7/3)) = 0.0263 // AND found in document 2, 4, 5
INVENTIVE =
PLAY =
ARE =
COMPLETELY =
DIFFERENT =
FROM =

FAST-PACING =
EUROPEAN =
FOOTBALL =

Thus calculations will be done till last term

Enigma41

3. i) What do you mean by Bag Of Words in Natural Language Processing (NLP)? Write down the two important parts of Bag Of Words model. [6]

BoW, TF-IDF

Solution:added by Tamal-122

Bag-of-Words (BoW)

- The **Bag-of-Words (BoW)** model is a way of representing text data when modeling text with machine learning algorithms. The **Bag-of-Words (BoW)** model is popular, simple to understand, and has seen great success in **language modeling** and **document classification**.
- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:
  - A vocabulary of known words.
  - A measure of the presence of known words.

ii) Consider that your little sister, Martha, is working on an NLP project. She is using TF-IDF as features. The document she is working on is given below. [8]

Although football is a global sport, Europe is famous for playing this sport. Asians, Africans, and Americans also play this game. However, European football is more deluxe. **Besides, Brazil and Argentina are arguably the most supported teams in the world.** Their styles of aesthetic and inventive play are completely different from fast-pacing European football. This sport is known by different names. For instance, it is commonly known as "soccer" in the United States.

Now capitalize each word token and help your sister to find TF-IDF features for the bold-faced sentence.

Solution:Rafi-148

Bold face word:

" BESIDES BRAZIL AND ARGIENTINA ARE ~~ARGUA~~ ARGUABLY THE MOST SUPPORTED TEAMS IN THE WORLD "

Total no. of Term in the Sentence = 13

" no. of document = 7

Term	Term in Document	TF	IDF	TF-IDF
BESIDES	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
BRAZIL	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
AND	1	1/13	$\log_{10}\left(\frac{7}{5}\right)$	0.028
ARGIENTINA	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
ARE	1	1/13	$\log_{10}\left(\frac{7}{2}\right)$	0.042
ARGUABLY	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
THE	2	2/13	$\log_{10}\left(\frac{7}{1}\right)$	0.129
MOST	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
SUPPORTED	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
TEAMS	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
IN	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065
WORLD	1	1/13	$\log_{10}\left(\frac{7}{1}\right)$	0.065

IN- er idf vul ace. In appears 2 sentence. - sajid148

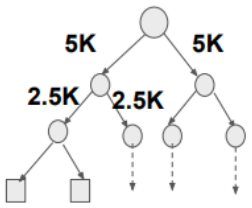
Recurssive40

2 Write down the key challenge of the skip-gram model. How can you solve this challenge? Word Embedding [5]

Solution:45

Problems with softmax classification

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$



Problems:

1. Here we are summing **10,000** numbers which corresponds to the number of words in our vocabulary.
2. If this number is larger say **100K or 1 million**, the computation will become **very slow**.

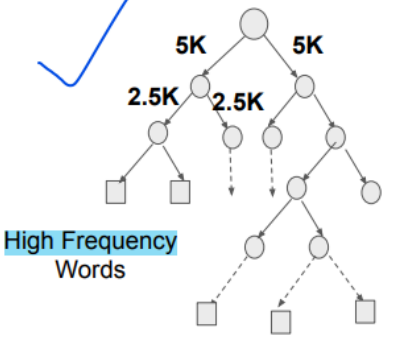
Solution:

- Use **"Hierarchical softmax classifier"** which works as a tree classifier.
- Complexity of Hierarchical softmax classifier is **O(log(n)) instead of O(n)**.

,

Hierarchical softmax classifier

This tree (can be asymmetric), where most commons words tend to be on top and less common words deeper to further reduce the computations.



Many neural language models nowadays use either hierarchical softmax or other softmax approximation techniques. For more reading, check out:

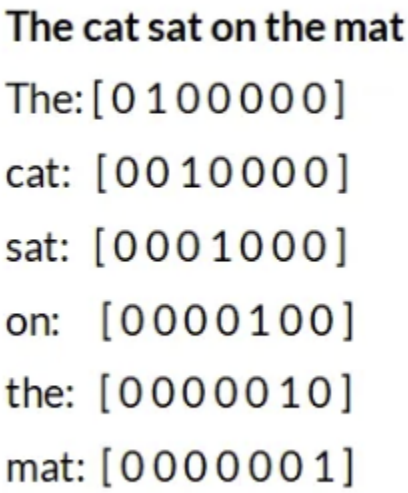
- **Negative sampling**
- Differentiated softmax
- [Adaptive] importance sampling

6. i) What do you mean by "One Hot Encoding" in Natural Language Processing (NLP)? Write down two disadvantages of One Hot Encoding. [5]

Word Embedding

6. i. Solution: Sujon 49

In one hot encoding, every word (even symbols) which are part of the given text data are written in the form of vectors, constituting only of 1 and 0 . So one hot vector is a vector whose elements are only 1 and 0. Each word is written or encoded as one hot vector, with each one hot vector being unique. This allows the word to be identified uniquely by its one hot vector and vice versa, that is no two words will have same one hot vector representation. For example see the below image shows one hot encoding of words in the given sentence.



Notice that in the image to the left the words 'The' and 'the' have different encoding implying they are different. Thus we are representing every word and symbols in the text data as a unique one hot vector which contains numerical data(1 and 0) as its constituent elements. One word is represented as a vector therefore the list of words in the sentence can be represented as an array of vectors or a matrix and if we have list of sentences whose words are one hot encoded then it will result in an array whose elements are matrices. So we end up with a three dimensional tensor which can be fed to the Neural network.

Disadvantages:

1. High Dimensionality: It increases the dataset's dimensionality, leading to memory and computational inefficiencies.

2. Loss of Information: It does not capture relationships or similarities between categories, resulting in information loss.

3. Curse of Dimensionality: High-dimensional data can increase model complexity, training times, and overfitting risk, impacting generalization.

ALTERNATIVE:45

In natural language processing (NLP), one-hot encoding is a simple technique used to represent words or tokens as binary vectors. Each word or token in a vocabulary is represented by a unique binary vector where only one element is "hot" (set to 1), and all other elements are "cold" (set to 0).

Word representation

V = [a, aaron, …, zulu, <UNK>]

Size of the Vocabulary |V| = 10,000

1-hot representation

Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
O_{5391}		O_{4914}		O_{456}	

Problems:-

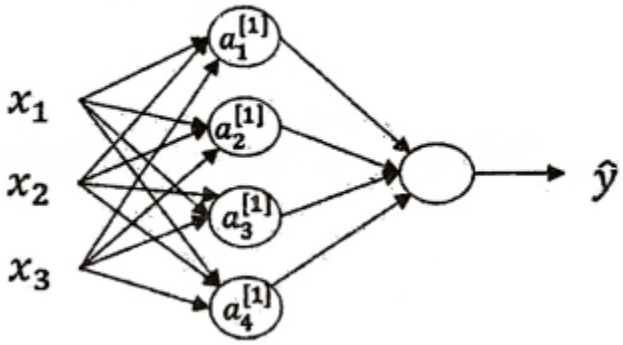
- It treats each words individually.
- There isn't any relationship between the words, given that the product between any two vector is zero and not the similarity of the two words.
- It doesn't allow an algorithm to generalize across words.

4

Vectorization of NN

Integer43

c) Figure out the vectorized form of the following neural network for multiple examples and justify the vectorized implementation. [7]



Enigma41

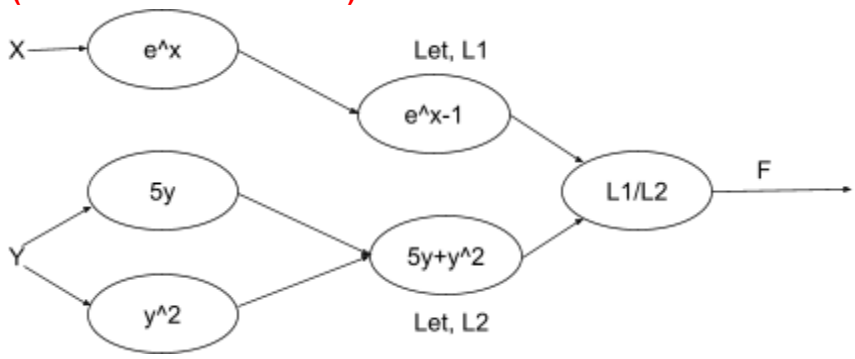
2

ii) Calculate a computational graph for the following function and show forward and backward simulation at x=-1 and y =2. [8]

$$f(x,y) = \frac{e^x - 1}{5y + y^2}$$

Computational Graph

Solution: **019** **(Backward is not solved)**



Forward Simulation:

Given, X = -1, Y = 2.

So,

1. e^x = e^-1 = 0.36788
2. L1 = 0.36788 - 1 = -0.63212
3. 5y = 5 * 2 = 10
4. y^2 = 2^2 = 4
5. L2 = 10 + 4 = 14
6. F = L1/L2 = -0.63212 / 14 = -0.04515

So, when x = −1 and y = 2, the value of the function f(x,y) is approximately -0.04515.

linear patterns in the

Calculate a computational graph for the following function and show forward and backward simulation at $x = -1$ and $y = 2$

$$f(x,y) = \frac{e^x - 1}{5y + y^2}$$

Forward simulation :

$$f(x,y) = \frac{e^{-1} - 1}{5(2) + (2)^2}$$

$\Rightarrow$ [illegible]

$\Rightarrow -0.045$

Backward simulation :

$f(x,y) = z$

$\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}$

$\therefore \frac{\partial}{\partial x}\left(\frac{e^x - 1}{5y + y^2}\right)$

$\Rightarrow \frac{1}{5y + y^2} \cdot \frac{\partial}{\partial x}(e^x - 1)$

$\Rightarrow \frac{1}{5y + y^2} e^x$

$\frac{\partial}{\partial y}\left(\frac{e^x - 1}{5y + y^2}\right)$

$\Rightarrow (e^x - 1) \frac{\partial}{\partial y}(5y + y^2)^{-1}$

$\Rightarrow (e^x - 1) \; -1 (5y + y^2)^{-2} \cdot \frac{\partial}{\partial y}(5y + y^2)$

$\Rightarrow \frac{-(e^x - 1)}{(5y + y^2)^2}(5 + 2y)$.

4. $i = V/P$

$$\frac{\partial f}{\partial x}(-1,2) = \frac{e^x}{x+y^2} = \frac{e^{-1}}{-1+(2)^2}$$

$$= 0.3$$

$$\frac{\partial f}{\partial y}(-1,2) = \frac{(x+2y) - (e^x - 1)}{(x+y^2)^2}$$

$$\Rightarrow \frac{9}{196} = 0.63$$

Recurssive40

1. i) How does the dropout layer work in a Deep Neural Network (DNN)? Explain it briefly with suitable examples. [5] Neural Network

1. i. Solution: Sujon 49

The term “dropout” refers to dropping out the nodes (input and hidden layer) in a neural network (as seen in Figure 1). All the forward and backwards connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network. The nodes are dropped by a dropout probability of p.

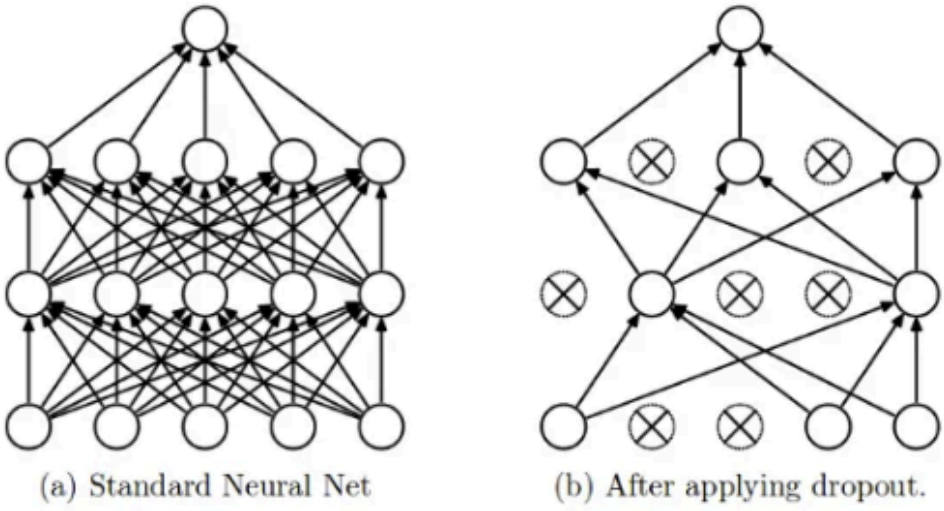


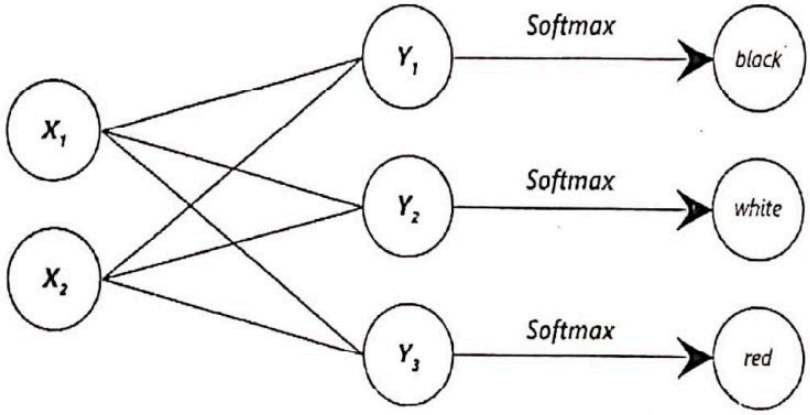
Figure 1: Dropout applied to a Standard Neural Network (Image by Nitish)

Let’s try to understand with a given input x: {1, 2, 3, 4, 5} to the fully connected layer. We have a dropout layer with probability p = 0.2 (or keep probability = 0.8). During the forward propagation (training) from the input x, 20% of the nodes would be dropped, i.e. the x could become {1, 0, 3, 4, 5} or {1, 2, 0, 4, 5} and so on. Similarly, it applied to the hidden layers.

For instance, if the hidden layers have 1000 neurons (nodes) and a dropout is applied with drop probability = 0.5, then 500 neurons would be randomly dropped in every iteration (batch).

ii) Your younger brother, Arthur Curry, created the following AI model that can detect three colors in his first college assignment. The model maps a 2-dimensional input, $X \in \Re^2$, to a 3-dimensional output, $Y \in \Re^3$. Later, the outputs are passed through a Softmax activation to get the probabilistic score of $Z_i \in [0, 1]$ for all three color types, i.e., *"black"*, *"white"*, and *"red"*. [4]

Neural Network



Now, check the performance of Arthur's model for a *"blue"* colored sample. The input and trained parameters of the model are as follows:

$$W = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \quad b = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad X = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

Solution:

$$y_1 = w_{11}x_1 + w_{12}x_2 + b_{11} = (0\times 2) + (1\times 1) + 1 = 2$$

$$y_2 = w_{21}x_1 + w_{22}x_2 + b_{21} = (1\times 2) + (1\times 1) + 0 = 3$$

$$y_3 = w_{31}x_1 + w_{32}x_2 + b_{31} = (0\times 2) + (1\times 1) + 1 = 2$$

$$\text{Softmax}(y_1) = \frac{e^2}{e^2 + e^3 + e^2} = 0.212$$

$$\text{Softmax}(y_2) = \frac{e^3}{e^2 + e^3 + e^2} = 0.576$$

$$\text{Softmax}(y_3) = \frac{e^2}{e^2 + e^3 + e^2} = 0.212$$

Ei math amader ache? - Mamun

iii) Draw the computational graph for the following function and show the gradient flow for the datapoint x = 1. [5]

Computational Graph

$$f(x) = \frac{-x}{3 + e^{-2x}}$$

Solution:
Etar solution keu dile bhalo hoi

3. i) Write down ten hyperparameters of a fully connected neural network. [5]

Parameter Calculation

3. i. Solution: Sujon 49

Hyperparameters in a fully connected neural network (also known as a feedforward neural network or multi-layer perceptron) play a crucial role in determining the network's architecture and training behavior. Here are 10 common hyperparameters:

1.Number of Hidden Layers: The total number of hidden layers in the network.

2.Number of Neurons in Each Layer: The number of neurons or units in each hidden layer.

3.Activation Functions: The choice of activation functions for each layer, such as ReLU, Sigmoid, or Tanh.

4.Learning Rate: The step size used during gradient descent to update the network's weights.

5.Batch Size: The number of training examples used in each iteration of training.

6.Epochs: The number of times the entire training dataset is passed forward and backward through the network during training.

7.Optimizer: The optimization algorithm used for weight updates, such as Adam, SGD, or RMSprop.

8.Loss Function: The choice of loss function that measures the error between predicted and actual values, like Mean Squared Error (MSE) for regression or Cross-Entropy for classification.

9.Regularization Techniques: Methods like L1 or L2 regularization, dropout, or batch normalization to prevent overfitting.

10.Initialization: The method used to initialize the weights and biases of the neural network, like random initialization or Xavier/Glorot initialization.

These hyperparameters collectively define the architecture and behavior of a fully connected neural network and need to be tuned carefully to achieve the best performance on a specific task.

Attention Mechanism, ACO

Integer43

b) Illustrate the algorithm of Ant Colony Optimization with proper example. [4]

Solution: By Paul 023

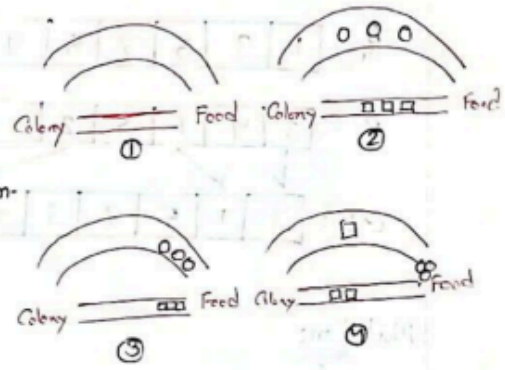
Integer-43 (6b)

Q: ACO algo with proper example:

Ans:

Initialization: The pheromone trails are initialized to a small positive value for all edges, and each ant is placed on a starting node. In the diagram, There are two paths to the food source, each with equal amount of pheromone.

Path Construction: Ants choose paths probabilistically, the path with higher pheromone will be chosen. For our example and as currently the amount of pheromone is same in all the path, Some ant took th higher path and some took the lower path.

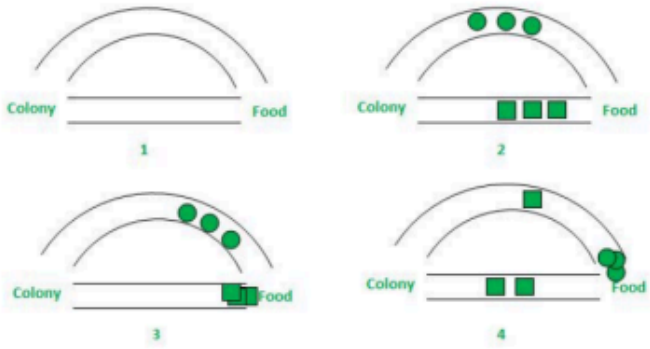


pheromone update: Two components –

- Pheromone Evaporation: To simulate the natural evaporation of pheromone over time, reducing the intensity of all pheromone trails.
- Pheromone Deposition – Ants deposit pheromone on the path they traveled, with more successful solutions getting more pheromone.

In the diagram, it seems the lower path has more ants, suggesting more pheromone have been laid down there.

Path Selection: Ants incrementally favours the path with higher pheromone (the lower path in the above diagram, showing the convergence towards the most efficient path between the colony and the food source.



Stage 1: All ants are in their nest. There is no pheromone content in the environment. (For algorithmic design, residual pheromone amount can be considered without interfering with the probability)
Stage 2: Ants begin their search with equal (0.5 each) probability along each path. Clearly, the curved path is the longer and hence the time taken by ants to reach food source is greater than the other.
Stage 3: The ants through the shorter path reaches food source earlier. Now, evidently they face with a similar selection dilemma, but this time due to pheromone trail along the shorter path already available, probability of selection is higher.
Stage 4: More ants return via the shorter path and subsequently the pheromone concentrations also increase. Moreover, due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. So, path optimization is attained.

** *Normalizer* **

Ei naam e kono topic sir er Question pattern e nai.
So, eta onno kono topic er under e kina janais …

Recurssive40

3.

ii) Consider the following dataset of a vehicle that consists of the speed and tag of the vehicle at 11 checkposts. Each speed indicates the velocity of the vehicle and the tag (O for over and B for below) whether the speed is below or above the limit of the checkpost. Unfortunately, at a few checkposts, the exact speed of the vehicle couldn't be captured. Now, you want to feed this data into a neural network for a classification task. Due to the sparsity of data, you want to normalize the speed values before feeding them to the network. Now, apply a suitable normalizer and find the normalized speed value. [9]

Normalizer

35.2 35.5

Speed	37	?	30	39	42	35	32	40	?	33	30
Tag	O	B	B	B	B	O	B	O	O	B	O

Solution:
Etar solution keu dile bhalo hoi

C : 2 set

Notes By Paul (Up to Quiz 01)

Answer the following questions:

1. What is the role of an optimizer in a neural network?
2. How does learning rate impact the performance of a model? Explain briefly.
3. What causes a function to lack the characteristics of a convex curve? Is the squared error function a good choice for logistic regression? Justify your answer.
4. **Determine how information and probability relate to one another. How can the logistics loss function be derived using information theory?**
5. Write down the differences between linear regression and logistic regression. What are the steps involved in converting a linear regression to a logistic regression?
6. What is the role of bias in a logistic regression algorithm? Explain Briefly.
7. Draw the pipeline of a logistic regression algorithm. For each step, describe how it works?
8. Draw the pipeline of a Logistic Regression. Explain the working procedure of every step
9. Why is it necessary for the loss function to be convex? Is squared error function a good choice? Justify your answer.
10. Write down your ideas about information theory. How can the logistics loss function be derived using information theory.
11. Describe the backward propagation of a logistic regression where the activation function is a "Sigmoid" activation function and the loss function is the "Cross Entropy Loss" function
12. Write down the algorithm to perform a logistic regression. How can you vectorize the algorithm?

1. Role of an Optimizer in a Neural Network:

An optimizer in a neural network is responsible for adjusting the weights and biases of the network during the training process to minimize the error or loss function. The optimizer uses optimization algorithms, such as gradient descent, to find the optimal values for the model parameters, thereby improving the model's performance.

2. Impact of Learning Rate on Model Performance: (Quize-1 e eta nei)

The learning rate is a hyperparameter that determines the step size at which the optimizer adjusts the model parameters. A too high learning rate may cause the model to overshoot the minimum, leading to divergence or oscillation, while a too low learning rate may result in slow convergence or getting stuck in local minima. Finding an appropriate learning rate is crucial for achieving optimal training results.

3.Characteristics of Convex Curves and Choice of Squared Error for Logistic Regression: (Quize-1 e eta nei)

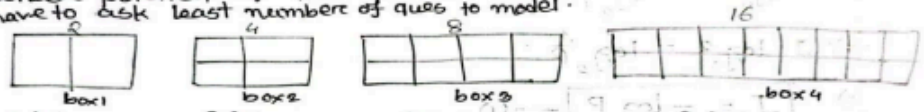
Convexity in a function implies that any line segment connecting two points on the curve lies above the curve itself. Functions with multiple minima or non-convex shapes can lack this property. The squared error function is not a good choice for logistic regression because the logistic regression model involves a non-linear

transformation (sigmoid function) to map inputs to probabilities, leading to a non-convex optimization problem. The use of squared error can result in non-convexity and convergence issues.

4. Information and Probability in Logistics Loss Function:

Determine how information and probability relate to one another. How can the logistics loss function be derived using information theory?

Suppose there some boxes with balls & have to guess/predict which portion of the boxes have the ball. To predict ball have to ask least number of ques to model.



Probability:

$P = \frac{1}{2}$ $P = \frac{1}{4}$ $P = \frac{1}{8}$ $P = \frac{1}{16}$

$= \frac{1}{2^1}$ $= \frac{1}{2^2}$ $= \frac{1}{2^3}$ $= \frac{1}{2^4}$ ... $\frac{1}{2^n}$

Here to find ball at least 1 ques need to get/predict the ans for box 1. 2, 3, 4 for box2, box3, box 4 respectively. Strategy is half the box in every possible portion.

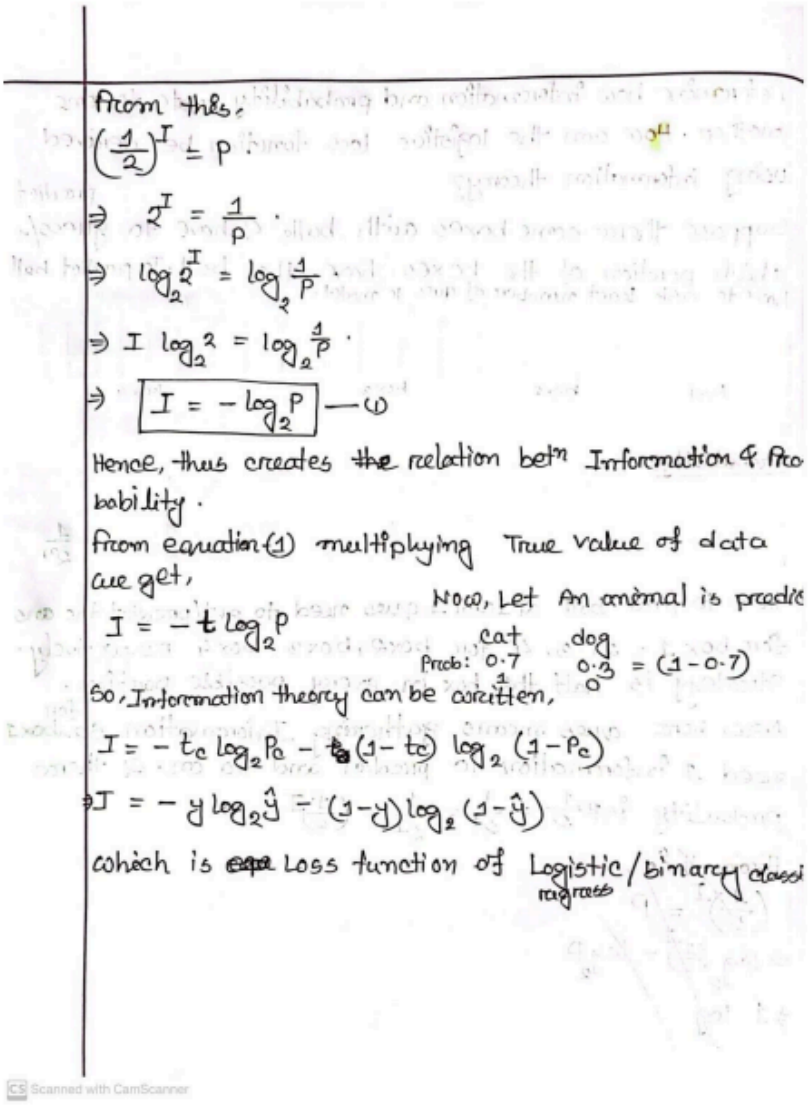
Now, here ques means gathering Information so for box1 need 1 information to predict and so on. & there probality is $P = \frac{1}{2} = \frac{1}{2^1} = \frac{1}{2^I} = \left(\frac{1}{2}\right)^I$

From this,

$\left(\frac{1}{2}\right)^I = P$

$\Rightarrow \log_2 \left(\frac{1}{2}\right)^I = \log_2 P$

$\Rightarrow I \log$



Information theory measures the amount of surprise or uncertainty associated with an event. Cross-entropy, a concept from information theory, is used to derive the logistic loss function. The logistic loss is essentially the negative log-likelihood of the true labels given the predicted probabilities. Minimizing this loss is equivalent to maximizing the likelihood of the observed data, aligning with the principles of information theory.

5. Differences Between Linear Regression and Logistic Regression:

- Linear Regression predicts continuous output, while Logistic Regression predicts the probability of an event.

- The output of linear regression is unbounded, while logistic regression output is constrained between 0 and 1.

- Linear regression uses the least squares method, while logistic regression uses the maximum likelihood estimation.

- The decision boundary in logistic regression is a sigmoid function.

Converting linear regression to logistic regression involves applying a logistic (sigmoid) function to the linear combination of inputs, transforming the output to a probability scale.

6. Role of Bias in Logistic Regression:

The bias term in logistic regression allows for the model to make predictions even when all input features are zero. It represents the log-odds of the probability of the positive class when all input features are zero. Including a bias term helps the logistic regression model to better fit the data and capture the intercept or baseline probability of the event being predicted.

FROM SLIDE:

The bias value allows the activation function to be shifted to the left or right, to better fit the data.

- Changes to the weights alter the steepness of the sigmoid curve, whilst the bias offsets it, shifting the entire curve so it fits better.

- Bias only influences the output values, it doesn't interact with the actual input data. That's why it is called bias.

- You can think of the bias as a measure of how easy it is to get a node to fire.

  - For a node with a large bias, the output will tend to be intrinsically high, with small positive weights and inputs producing large positive outputs (near to 1).

  - Biases can be also negative, leading to sigmoid outputs near to 0.

  - If the bias is very small (or 0), the output will be decided by the values of weights and inputs alone.

7. Logistic Regression Algorithm Pipeline:

1. Input Data:

   - Input features (X) and target labels (y).

2. Initialization:

   - Initialize weights (W) and bias (b) with small random values.

3. Linear Transformation:

   - Calculate the linear combination of input features and weights: (z = XW + b).

4. Activation Function (Sigmoid):

   - Apply the sigmoid activation function to the linear output: (a = sigma(z)), where (sigma(z) = frac{1}{1 + e^{-z}}).

5. Prediction:

   - Interpret the sigmoid output as the probability of belonging to the positive class.

6. Loss Calculation:

   - Calculate the logistic loss using the cross-entropy loss function:

(L(y, a) = -y log(a) - (1-y) log(1-a)).

7. Gradient Descent:

   - Compute the gradient of the loss with respect to the weights and bias.

8. Parameter Update:

   - Update weights and bias using the gradient and the chosen learning rate.

9. Iteration:

   - Repeat steps 3-8 until convergence or a set number of iterations.

8. Logistic Regression Pipeline (Detailed Explanation):

- Input Data:Features (X) and target labels (y).

- Initialization: Initialize weights (W) and bias (b).

- Linear Transformation: Calculate \(z = XW + b\).

- Activation Function (Sigmoid): Apply the sigmoid function to \(z\), yielding \(a = \sigma(z)\).

- Prediction: Interpret \(a\) as the probability of belonging to the positive class.

- Loss Calculation: Compute the logistic loss using the cross-entropy loss function.

- Gradient Descent: Compute the gradient of the loss with respect to weights and bias.

- Parameter Update: Update weights and bias using the gradient and learning rate.

- Iteration: Repeat steps until convergence.

9. Necessity of Convex Loss Function: (Quize-1 e eta nei)

- Convexity ensures the existence of a global minimum, making optimization more reliable.

10. Is squared error function a good choice? Justify your answer.

- The squared error function is not ideal for logistic regression because the sigmoid activation introduces non-linearity, leading to a non-convex optimization problem. Gradient-based optimization algorithms may get stuck in local minima.

FROM SLIDE

The squared error function (commonly used function for linear regression) is not very suitable for logistic regression.

○ In case of logistic regression, the hypothesis / prediction is non-linear (sigmoid function), which makes the square error function to be non-convex.

○ On the other hand, logarithmic function is a convex function for which there is no local optima, so gradient descent works well.

● If you are doing binary classification, squared error function generally also penalize examples that are correctly classified but are still near the decision boundary, thus creating a "margin."

● Gradient descent waste a lot of time getting predictions very close to {0, 1}

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{2}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

- We can see an extra (1/2) in the right side of the equation. Does it matter?

● It is because when you take the derivative of the cost function, that is used in updating the parameters during gradient descent, that 2 in the power get cancelled with the (1/2) multiplier.

● These techniques are or somewhat similar are widely used in math in order "To make the derivations mathematically more convenient".

11. Information Theory and Logistic Loss Function:

- Information theory measures uncertainty. Cross-entropy is a measure of surprise or information gain.

- Logistic loss, derived from cross-entropy, quantifies the difference between predicted probabilities and actual labels.

- Minimizing logistic loss aligns with maximizing the likelihood of observed data, connecting logistic regression to information theory.

12. Backward Propagation in Logistic Regression: (Quize-1 e eta nei)

- Loss Gradient: Compute the gradient of the loss with respect to the predicted probabilities.

- Sigmoid Backward:** Compute the gradient of the sigmoid activation function.

- Chain Rule: Combine gradients to compute the overall gradient of the loss with respect to the weighted inputs.

-Parameter Update: Use gradients to update weights and bias using gradient descent.

13. Logistic Regression Algorithm: (Quize-1 e eta nei)

-Input: Features (X), labels (y), learning rate, and number of iterations.

- Initialize: Weights (W) and bias (b).

- For each iteration:

  - Linear Transformation: \(z = XW + b\).

  - Sigmoid Activation: \(a = \sigma(z)\).

  - Loss Calculation: \(L = -y \log(a) - (1-y) \log(1-a)\).

  - Gradient Calculation: Compute gradients with respect to weights and bias.

  - Parameter Update: Update weights and bias using the gradients and learning rate.

- Vectorization: Instead of looping through individual samples, perform operations on entire matrices. This enhances computational efficiency, often implemented using libraries like NumPy.

14.What are the steps involved in converting a linear regression to a logistic regression?

Converting a linear regression model to a logistic regression model involves several steps to adapt the model for classification tasks. Here are the key steps:

1.**Sigmoid Activation Function:**

- Replace the linear activation function in the output layer with the sigmoid activation function (logistic function):

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

- This function squashes the output between 0 and 1, transforming the continuous output into a probability.

2. **Set a Threshold:**

- Choose a threshold (commonly 0.5) to convert probabilities into binary predictions. If \(a \geq 0.5\), predict class 1; otherwise, predict class 0.

3. **Change the Output Range:**

- Since logistic regression is used for binary classification, modify the output interpretation. Instead of predicting a continuous value, interpret the output as the probability of belonging to the positive class.

4. **Modify the Loss Function:**

- Change the loss function from Mean Squared Error (MSE) to the binary cross-entropy loss (log loss) for logistic regression:

[L(y, a) = -y log(a) - (1-y) log(1-a)]

- Here, (y) is the true label (0 or 1) and (a) is the predicted probability.

5. **Gradient Descent Update:**

- Adjust the gradient descent update rule accordingly to accommodate the new loss function and activation function.

1. regression Vs Classification.

2. Supervise vs unsupervise data.

3.Structured vs Unstructured Data.

1. Regression vs Classification:

- Output Type:

- Regression: Predicts a continuous output or numerical value.

- Example: Predicting the temperature (a continuous value) based on features like time of day, humidity, and wind speed.

- Classification: Predicts a discrete category or label.

- Example: Classifying emails as either spam or not spam based on features like sender, subject, and content.

- **Objective:**

  - Regression: Minimizes the difference between predicted and actual values.

  - Classification: Assigns inputs to predefined categories or classes.

- **Output Interpretation:**

  - Regression: The output represents a quantity that can be interpreted on a scale.

  - Classification: The output represents a class or category.

- Examples of Algorithms:

  - Regression; Linear Regression, Polynomial Regression.

  - Classification: Logistic Regression, Decision Trees, Support Vector Machines.

2. Supervised vs Unsupervised Data:

- **Labeled vs Unlabeled:**

  - Supervised: Training data includes labeled examples with corresponding outputs.

  - Example: A dataset of labeled images with annotations indicating whether they contain cats or dogs.

  -Unsupervised: No explicit labels are provided, and the algorithm discovers patterns or relationships.

  - Example: Clustering a dataset of customer purchase history to identify groups with similar buying behavior.

- **Guidance:**

  - Supervised: The algorithm is guided by the labeled output during training.

  -Unsupervised: The algorithm explores the data structure without explicit guidance.

- **Objective:**

  - Supervised: Predict or classify based on known relationships.

  - Unsupervised: Discover hidden patterns or relationships within the data.

- **Examples of Algorithms:**

  - Supervised: Linear Regression, Classification models, Neural Networks.

-Unsupervised: K-Means Clustering, Hierarchical Clustering, Principal Component Analysis (PCA).

3. Structured vs Unstructured Data:

- Organization:

- Structured: Data is organized and follows a clear, predefined structure.

- Example: Relational databases with tables, rows, and columns.

- Unstructured: Data lacks a predefined structure and organization.

- Example: Text documents, images, audio recordings.

- Searchability:

- Structured: Easily searchable and accessible due to a clear schema.

- Unstructured: Requires more advanced methods (e.g., natural language processing for text) for effective search and retrieval.

- Examples:

- Structured: Excel spreadsheets, SQL databases.

- Unstructured: Social media posts, images, videos, audio recordings.

- Flexibility:

- Structured More rigid and less adaptable to changes in data types.

-Unstructured: Allows for more flexibility as data types and formats can vary widely.

These differences highlight the diverse nature of data and the various approaches used in machine learning based on the characteristics of the data at hand.

Loss vs Cost

Loss Function:

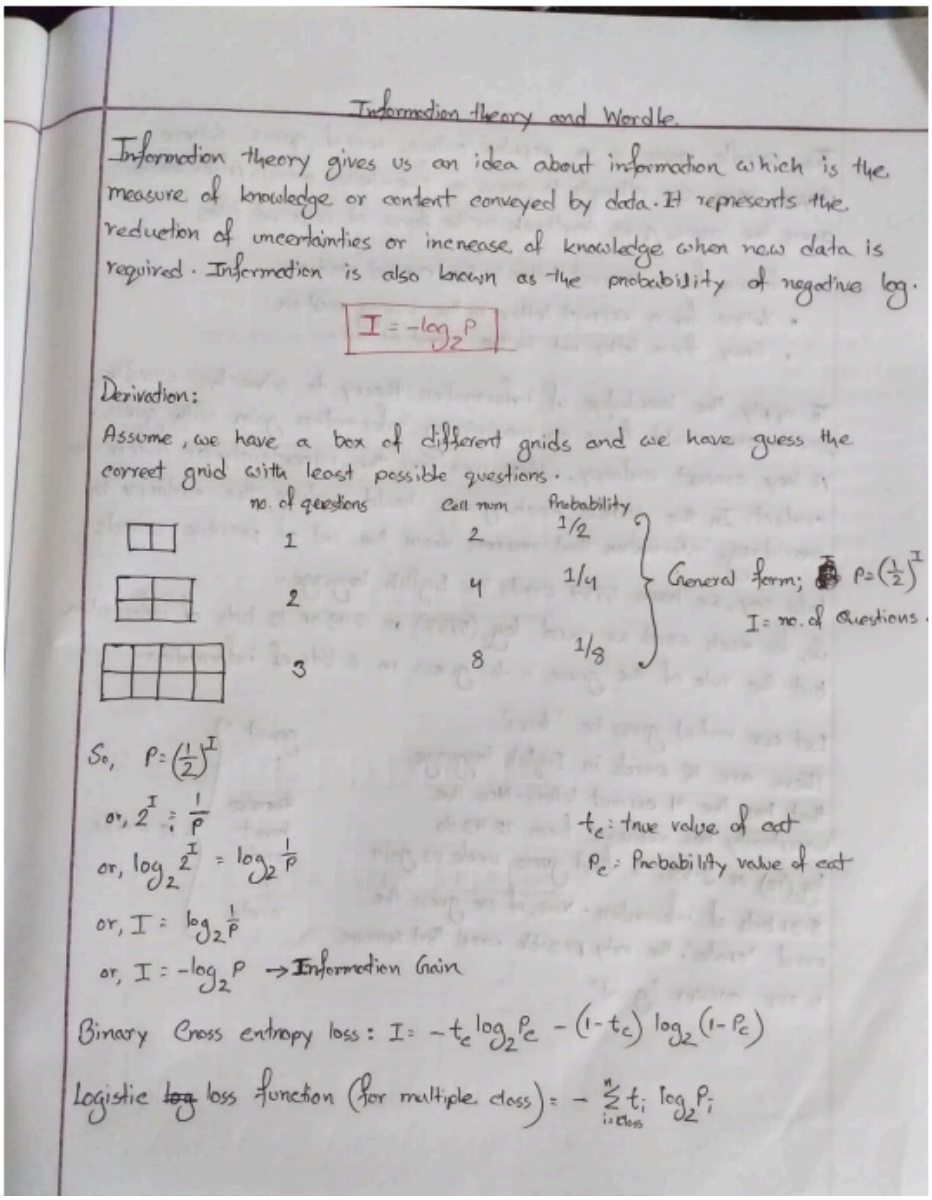
- The loss function computes the error for a single training example.
- It measures the discrepancy between the predicted value (output of the model) and the actual value (the true label or outcome).
- The goal is to minimize this error during training.
- Common examples include Mean Squared Error for regression tasks and Cross-Entropy for classification tasks.

Cost Function:

- The cost function aggregates the losses from all the training samples, usually taking the average. In other words, it's the average of the loss functions over the entire training dataset.
- It represents the total cost of a particular model, given the data and the model's parameters.
- The cost function is what optimization algorithms (like gradient descent) aim to minimize to train the model.
- It is also known as the "objective function" or "loss function" when referred to in the context of the entire training set.

Information Theory

Information refers to the quantity of data that you obtain from a particular observation or message. More unexpected observations yield more information.
Entropy is a broader statistical measure of the amount of disorder or unpredictability in a dataset. High entropy means the source can produce a very diverse set of data, and it's hard to predict what the next piece of data will be.



Information theory and Wordle.

Information theory gives us an idea about information which is the measure of knowledge or content conveyed by data. It represents the reduction of uncertainties or increase of knowledge when new data is required. Information is also known as the probability of negative log.

$$I = -\log_2 P$$

Derivation:

Assume, we have a box of different grids and we have guess the correct grid with least possible questions.

no. of questions	Cell num	Probability
1	2	1/2
2	4	1/4
3	8	1/8

General form; $P = \left(\frac{1}{2}\right)^I$

I = no. of questions.

So, $P = \left(\frac{1}{2}\right)^I$

or, $2^I = \frac{1}{P}$

or, $\log_2 2^I = \log_2 \frac{1}{P}$

or, $I = \log_2 \frac{1}{P}$

or, $I = -\log_2 P$ → Information Gain

t_c: true value of cat

P_c: Probability value of cat

Binary Cross entropy loss: $I = -t_c \log_2 P_c - (1 - t_c) \log_2 (1 - P_c)$

Logistic ~~log~~ loss function (for multiple class) $= -\sum_{i \text{ is class}}^{n} t_i \log_2 P_i$

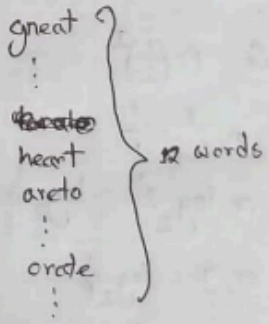
The wordle game is a popular online word game where players have six attempts to guess a five-letter word. After each guess, the game gives feedback in the form of coloured tiles.

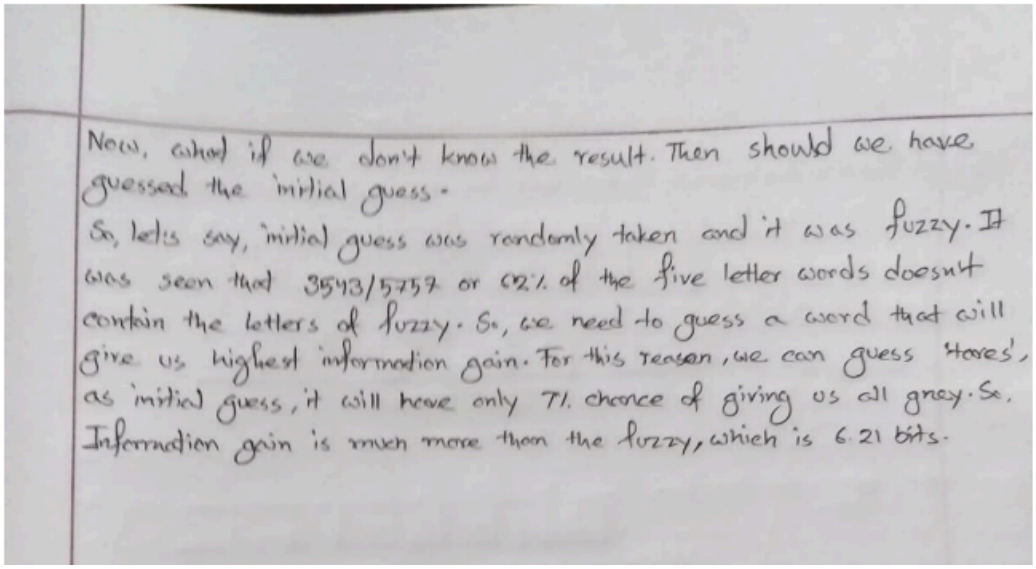
- Green for a correct letter in the correct position
- Yellow for a correct letter in the wrong position
- Gray for a letter not in the word at all.

To apply the knowledge of information theory to solve the wordle game, one would focus on maximizing information gain with guess. A key concept entropy, which measures the uncertainty or information content. In the wordle, each guess should reduce the entropy by providing information that narrow down the set of possible words.

Lets say, we have 5757 words in English language.
So, for each word we need $\log_2(5757)$ or 12.49 or 13 bits of information but the rule of the game is to guess in 6 bits of information.

Let our initial guess be 'tares'
There are 12 words in English language that has the 4 correct letters. Now the uncertainty has reduced from 12.49 to $\log_2(12)$ or 3.58. So, first guess made us gain 8.91 bits of information. Now, if we guess the word 'orate'. the only possible word that remains is our answer 'great'




Now, what if we don't know the result. Then should we have guessed the initial guess.
So, lets say, initial guess was randomly taken and it was fuzzy. It was seen that 3543/5757 or 62% of the five letter words doesn't contain the letters of fuzzy. So, we need to guess a word that will give us highest information gain. For this reason, we can guess 'tares', as initial guess, it will have only 7% chance of giving us all grey. So, Information gain is much more than the fuzzy, which is 6.21 bits.

Quiz-1 - Qubits45

Set A

Q1. Define the Wordle game and apply the knowledge of information theory to solve the game.

era 38

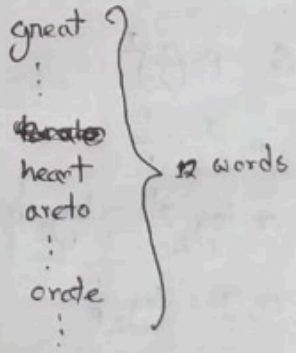
The wordle game is a popular online word game where players have six attempts to guess a five-letter word. After each guess, the game gives feedback in the form of coloured tiles.

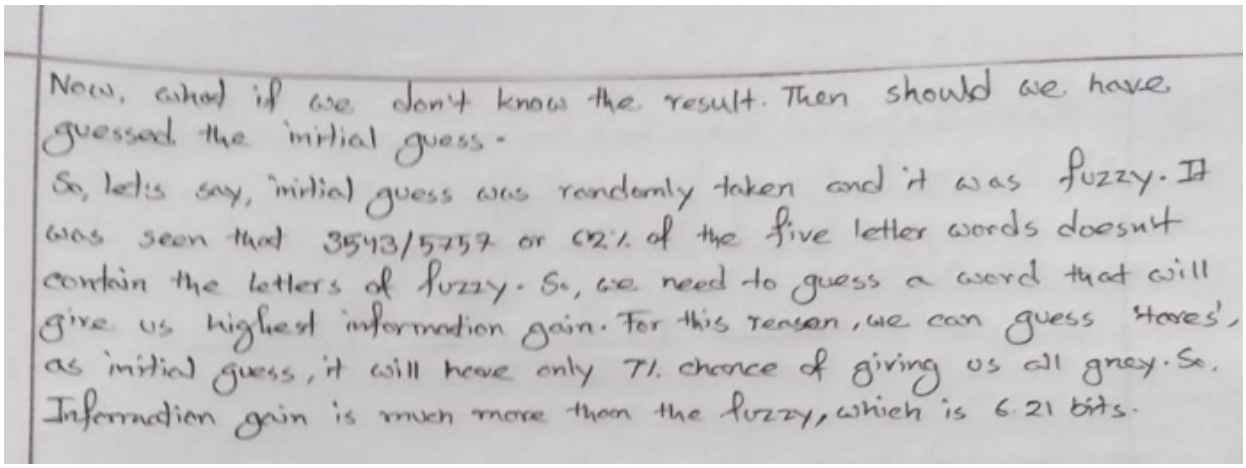
- Green for a correct letter in the correct position
- Yellow for a correct letter in the wrong position
- Gray for a letter not in the word at all.

To apply the knowledge of information theory to solve the wordle game, one would focus on maximizing information gain with guess. A key concept entropy, which measures the uncertainty or information content. In the wordle, each guess should reduce the entropy by providing information that narrow down the set of possible words.

Lets say, we have 5757 words in English language.
So, for each word we need $\log_2(5757)$ or 12.49 or 13 bits of information but the rule of the game is to guess in 6 bits of information.

Let our initial guess be 'tares'
There are 12 words in English language that has the 4 correct letters. Now the uncertainty has reduced from 12.49 to $\log_2(12)$ or 3.58. So, first guess made us gain 8.91 bits of information. Now, if we guess the word 'orate'. the only possible word that remains is our answer 'great'





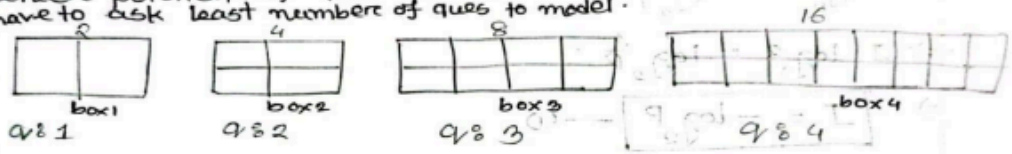
Set B

Q1. Write down your ideas about information theory. How can the logistics loss function be derived using information theory?

1. **era 38**

Determine how information and probability relate to one another. How can the logistics loss function be derived using information theory?

Suppose there some boxes with balls & have to guess/predict which portion of the boxes have the ball. To predict ball have to ask least number of ques to model.



q's 1 q's 2 q's 3 q's 4

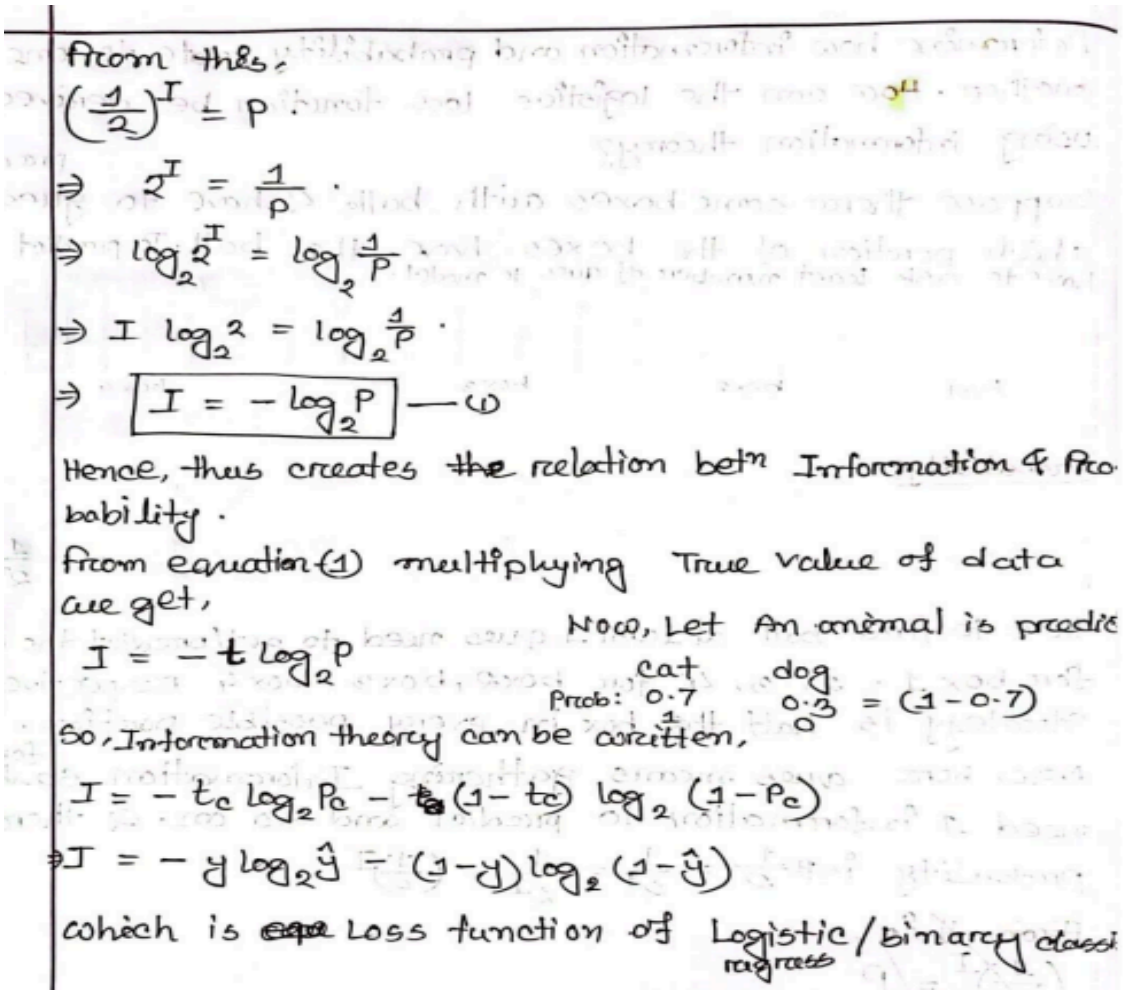
Probability:

$P = \frac{1}{2}$ $P = \frac{1}{4}$ $P = \frac{1}{8}$ $P = \frac{1}{16}$

$= \frac{1}{2^1}$ $= \frac{1}{2^2}$ $= \frac{1}{2^3}$ $= \frac{1}{2^4}$ ..... $\frac{1}{2^n}$

Here to find ball at least 1 ques need to get/predict the ans for box 1. 2, 3, 4 for box2, box3, box4 respectively. Strategy is half the box in every possible portion.

Now, here ques means gathering Information so for box1 need 1 information to predict and so on. & there probality is $P = \frac{1}{2} = \frac{1}{2^1} = \frac{1}{2^I} = \left(\frac{1}{2}\right)^I$



Set C

Q1. Answer the following questions.

1. Write down the basic idea of Entropy. What is the difference between Information and Entropy?
2. Derive the binary cross entropy loss function from the general formula with proper notation.

1. era 38

Information Vs Entropy

1.Information:

- In the context of information theory, developed by Claude Shannon, information is quantified as a measure of surprise or uncertainty. The more surprising or uncertain an event, the more information it provides.

- Information is often measured in bits. One bit represents the binary choice between two equally likely outcomes (such as true/false or 0/1).

2. Entropy:

- Entropy, also introduced by Claude Shannon in information theory, is a measure of the average amount of information or uncertainty associated with a set of possible outcomes.

- It is a concept borrowed from thermodynamics and has been adapted to quantify information. In information theory, entropy is used to measure the randomness or disorder in a system.

- Higher entropy indicates greater disorder or unpredictability, while lower entropy indicates more order or predictability.

- Entropy is used in the context of probability distributions. For example, a fair coin has higher entropy than a biased coin because the outcomes are more uncertain with the fair coin.

The relationship between information and entropy can be summarized as follows:

- **High entropy:** When a system has high entropy, there is greater unpredictability or disorder. This implies that each new piece of information about the system provides more "surprise" or "newness," and therefore, it carries more information.

- **Low entropy:** Conversely, when a system has low entropy, there is more predictability or order. In this case, additional information about the system may not be as surprising or informative because the outcomes are more certain.

In summary, while information measures the content or surprise of a message, entropy quantifies the uncertainty or disorder in a set of possible outcomes. They are interconnected concepts used in information theory to analyze and quantify the characteristics of data and communication systems.

2. era 38

Q: Derive the binary Cross Entropy loss function from the general formula with proper notation.

Ans: The cross entropy loss function is called logarithmic loss, log loss or logistic loss.

The general formula for n classes,

$$L = -\sum_{i=1}^{n} t_i \log(P_i)$$

Here, t_i is the true label and P_i is the softmax Probability for i^{th} class.

Now, Let's consider binary classification scenario where cat (c) & dog (D) two classes. So in general it can be written.

$$L = -\left[t_c \log(P_c) + t_D \log(P_D)\right]$$

Since there is two classes, so one class can be written base on other,

cat 0.7 dog 0.3 = (1-0.7)

$$L = -\left[t_c \log(P_c) + (1-t_c) \log(1-P_c)\right]$$

$$L(y, \hat{y}) = -\frac{2}{2}\left[y \log_2(\hat{y}) + (1-y) \log_2(1-\hat{y})\right]$$

where $\hat{y}$ represents the predicted probability belongs to class 1

& $(1-y)$ represents the predicted probability belongs to class 0

$$L = -\sum_{i=1}^{2} y_i \log(P_i)$$

Set D

Q1. Answer the following questions.

1. Write down the difference between loss and cost function with proper equations.
2. Why do we use softmax as opposed to standard normalization? Give proper justification of your answer.

1. era 38

4.Write down the difference between loss and cost function with proper equations.

Ans: Let's define each and highlight the differences:

1.Loss Function:

- The loss function is typically associated with a single data point in a dataset. It measures the error between the predicted output and the true target for that specific data point.

- Denoted by (L(hat{y}, y)), where (hat{y}) is the **predicted output** and (y) is the true target.

- Common examples include Mean Squared Error (MSE), Binary Cross-Entropy Loss, and Categorical Cross-Entropy Loss.

- For a single data point, the loss function might look

$L(y^, y)=-[y \log(y^)+(1-y)\log(1-y^)]$ for Binary Cross-Entropy Loss.

2. Cost Function:

- The cost function, on the other hand, is the average loss over the entire dataset. It is the overall measure of how well the model is performing on the entire training set.

- It is the sum (or average) of the individual loss functions over all training examples.

- For a dataset of size \(N \), the cost function might be defined as

$$L = -\frac{1}{N}\left[\sum_{j=1}^{N} [t_j \log(p_j) + (1 - t_j) \log(1 - p_j)]\right]$$

where pj is the predicted output for the jth data point and Tj is the true target.

In summary, the loss function quantifies the error for a single data point, while the cost function represents the overall performance of the model on the entire dataset. The cost function is essentially an average or sum of the losses across all data points.

2. era 38

Why do use softmax as opposed to standard normaliza

Ans: The softmax function is particularly well-suited for ~~this~~ classification class than standard normalizatio because it not ~~no~~ only normalize the output to be in the range (0,1) but also ensure that the outputs sum to 1 and also ~~be~~ capable to capture the changes in input features to the array of output.

For example; Let's take a blurry image of a ferret

Now softmax $[1,2] = \frac{e^1}{e^1+e^2} , \frac{e^2}{e^1+e^2}$

$= 0.2689 , 0.7310$

It is hard to identify as cat. But if we take of cat better crisp resulation image about [10,20

Softmax $[10,20] = \frac{e^{10}}{e^{10}+e^{20}} , \frac{e^{20}}{e^{10}+e^{20}}$

$= 0.000045 , 0.99995$

it is so easy to identify as Cat.

But for normalization, varying input feature doesnot change the output.

Std-norm $[1,2] = \frac{1}{1+2} , \frac{2}{1+2}$

$= 0.333 , 0.666$

Std-norm $[10,20] = \frac{10}{10+30} , \frac{20}{10+20}$

$= 0.333 , 0.666$

That's softmax opposed to standard normalization

Quiz-2 - Qubits45

Set A

2. "A neural network is a combination of multiple logistic regressions." - Justify the statement. [7]

era 38

8. "A neural network is a combination of multiple logistic regressions." justify the statement.

Ans: This statement can be justified by describing the basic structure and functioning of a neural network and logistic regression.

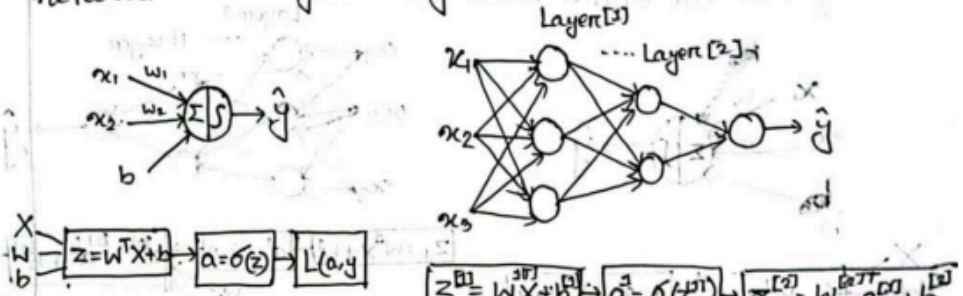


Figure: 1 Logistic Regression

Figure 2: Neural Network.

In logistic Regression, the input features are linearly combined and output is obtained by applying the sigmoid activation function to the weighted sum of inputs. And then calculate loss, which is visually represented in figure 1. LR is consider as single neuron.

And a neural network is nothing but a stacking up multiple logistic regression and consisting of layers of interconnected units (LR) organized in an input layers, one or more hidden layers

and output layers. The outputs of these units from one layer serve as inputs to the next layer, followed by an activation function, Figure 2.

$$\text{Output}_{(\text{neural Network})} = \text{Activation}\left(\sum_i (\text{weight}_{ij} \times \text{output}_{LR_i}) + \text{Bias}_j\right)$$

So, the key point is that each neuron in a layer operates similarly to a logistic regression unit and the combination of these unit across layers form the structure of a neural network. So LR can b considered as the simplest form of neural netwo And thus the given statement is justified.

In essence, a neural network can be seen as a composition of multiple logistic regressions, but the power and expressiveness of a neural network arise from the combination of these basic units and the introduction of non-linear activation functions, allowing it to model complex relationships in data

Set B

2. "The derivative of the tanh activation function depends on the function itself." - Justify the statement. [7]

era 38

Q: "The derivative of the tanh activation function depends on the function itself". Justify the statement.

Ans: We know, $\tanh = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

$\boxed{\frac{u}{v} = \frac{vu' - uv'}{v2}}$ will be.

If tanh is an activation function then derivation

$$\frac{\partial \tanh}{\partial z} = \frac{\partial}{\partial z}\left[\frac{e^z - e^{-z}}{e^z + e^{-z}}\right]$$

$$= \frac{(e^z + e^{-z})(e^z + e^{-z}) - (e^z - e^{-z})(e^z - e^{-z})}{(e^z + e^{-z})^2}$$

$$= \frac{(e^z + e^{-z})^2 - (e^z - e^{-z})^2}{(e^z + e^{-z})^2}$$

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

$$= 1 - (\tanh)^2$$

so the derivation turns into $(1 - \tanh^2)$ where except tanh no other variable is existed. Hence it is visiable that derivation of tanh only depends on itself.

Set C

1. "The derivative of the hyperbolic tangent function is more steep than the sigmoid function"- Justify the statement with proper evidence.

era 38

Q: "The derivative of the hyperbolic tangent function is more steep than the sigmoid function" - Justify the statement with proper evidance.

Ans: We know, for sigmoid activation function if large value is assigned then the gradient becomes zero same goes for very small value.

Now the sigmoid function, $g(z) = \frac{1}{1+e^{-z}}$

And the derivation of sigmoid is,

$g'(z) = g(z).(1 - g(z))$

Let's, if $z = 10$; $g(z) = \frac{1}{1+e^{-10}} = 0.999 \approx 1$

So $g'(z) = 1(1-1) = 0$ ✓

Again, if $z = -10$; $g(z) = \frac{1}{1+e^{-(-10)}} = 0.000045 \approx 0$

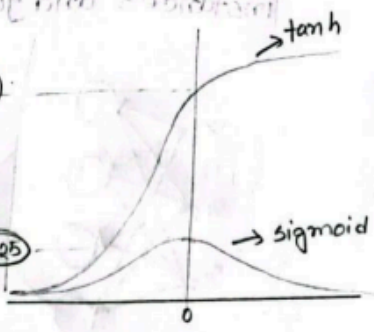
So, $g'(z) = 0(1-0) = 0$ ✓

for $z = 0$; $g(z) = \frac{1}{1+e^{-0}} = \frac{1}{2}$

So, $g'(z) = \frac{1}{2}\left(1 - \frac{1}{2}\right) = \frac{1}{4} = 0.25$ ✓

So it is proved that for large and low value sigmoid shows vanishing gradient problem and

It's steepness is about 0.25 unit long if it is drawn graphically.



Now for tanh,

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

The derivation of tanh(z) is

$g'(z) = 1 - (g(z))^2$

Let's if $z = 10$; $g(z) = \frac{e^{10} - e^{-10}}{e^{10} + e^{-10}} \approx 0.9999 \approx 1$.

so $g'(z) = 1 - (1)^2 = 0$ ✓

Again if $z = -10$; $g(z) = \frac{e^{-10} - e^{-(-10)}}{e^{-10} + e^{-(-10)}} \approx -1$

so $g'(z) = 1 - (-1)^2 = 0$ ✓

Now, $z = 0$; $g(z) = \frac{e^0 - e^{-0}}{e^0 + e^{-0}} = 0$

so $g'(z) = 1 - 0^2 = 1$ ✓

Tanh also suffers from vanishing G. problem but it's steepness much higher than sigmoid about 1.

Therefore the statement is justified.

fig 1: Representation of steepness of tanh & sigmoid

Quiz-1 - Integer43

Set-B

Q1. What causes a function to lack the characteristics of a convex curve? Is the squared error function a good choice for logistic regression? Justify your answer.

1. **Solution:45-** A function lacks the characteristics of a convex curve when it does not satisfy the definition of convexity. In mathematical terms, if you take any two points (x1, f(x1)) and (x2, f(x2)) on the graph of a convex function, the line segment connecting these points should not dip below the curve. There are several reasons why a function may not be convex: Sharp Peaks or Valleys: Functions with sharp peaks or valleys can fail to be convex because the line segment between two points might cross the peak or valley. Local Concavity: The function may have regions where it is concave down, meaning the curve dips below the line connecting two points. These regions can disrupt the overall convexity of the function.

Set-C

Q1. Determine how information and probability relate to one another. How can the logistics loss function be derived using information theory?

1. **Solution:45**

Information:

What is the minimum number of yes/No question you have to ask to reach an outcome.

2^{info} = number of box

so, number of box $= 2^I$ ——— (i)

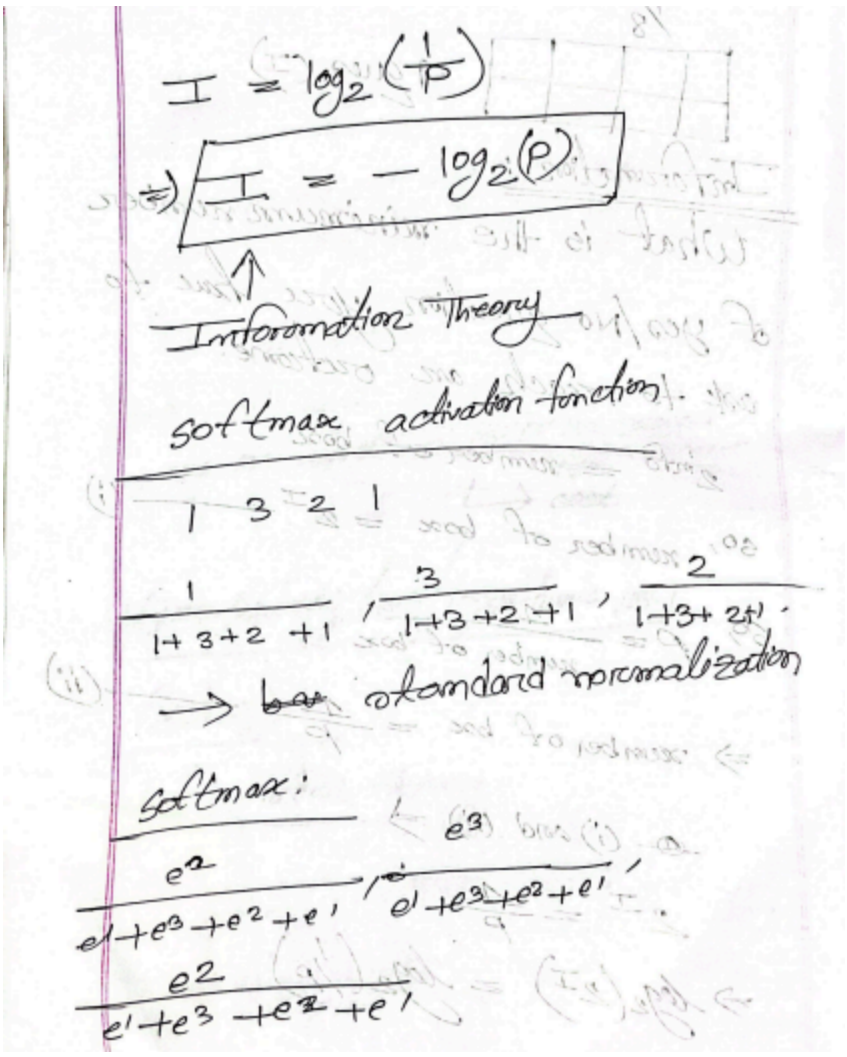
so, $p = \frac{1}{\text{number of box}}$

$\Rightarrow$ number of box $= \frac{1}{p}$ ——— (ii)

⊕ (i) and (ii) $\Rightarrow$

$2^I = \frac{1}{p}$

$\Rightarrow \log_2(2^I) = \log_2(1/p)$



Set-D

Q1. A. Write down the differences between linear regression and logistic regression. What are the steps involved in converting a linear regression to a logistic regression?

Solution: 45

, here's a concise comparison between Linear Regression and Logistic Regression without separating the points:

- **Output Type:**

- Linear Regression: Predicts continuous numerical values.

- Logistic Regression: Performs binary classification, predicting probabilities between 0 and 1.

- **Output Range**:
  - Linear Regression: Outputs any real number.
  - Logistic Regression: Outputs probabilities bounded between 0 and 1.

- **Activation Function**:
  - Linear Regression: No activation function.
  - Logistic Regression: Employs the logistic (sigmoid) function as an activation function.

- **Applications**:
  - Linear Regression: Used for tasks like price prediction.
  - Logistic Regression: Applied to binary classification problems such as spam detection or medical diagnosis.

To convert from linear to logistic regression, adjust the dependent variable, apply the logistic function, introduce a threshold for classification, use a suitable loss function, and assess with binary classification metrics.

B. What is the role of bias in a logistic regression algorithm? Explain Briefly.

Solution:45

Logistic Regression : Role of bias (b)

- The bias value **allows the activation function to be shifted to the left or right**, to better fit the data.
- Changes to the weights **alter the steepness of the sigmoid curve**, whilst the bias offsets it, shifting the entire curve so it fits better.
- **Bias only influences the output values**, it doesn't interact with the actual input data. That's why it is called bias.
- You can think of the bias as a measure of **how easy it is to get a node to fire**.
  - **For a node with a large bias**, the output will tend to be intrinsically high, with small positive weights and inputs producing large positive outputs (near to 1).
  - **Biases can be also negative**, leading to sigmoid outputs near to 0.
  - **If the bias is very small (or 0)**, the output will be decided by the values of weights and inputs alone.

Set-E

Q1. Draw the pipeline of a logistic regression algorithm. For each step, describe how it works.

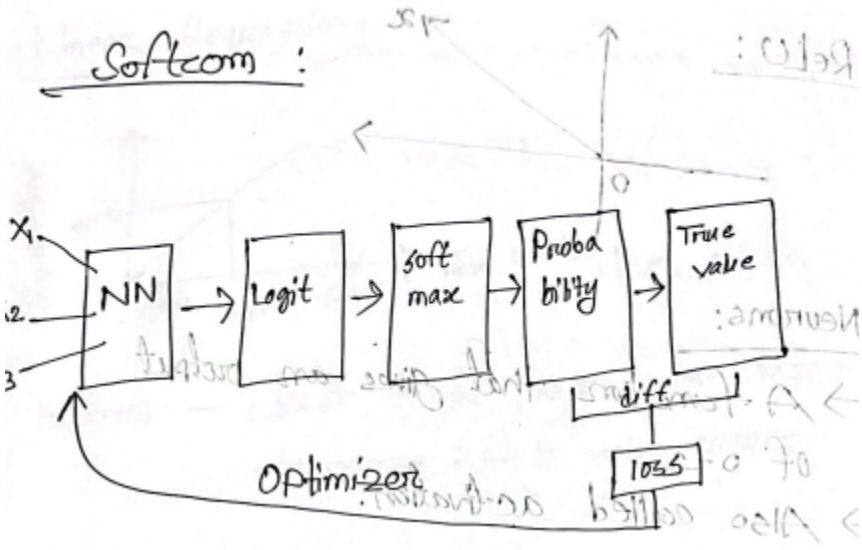
1. **Solution:given**

Quiz-1 - Previous Ques

Set-A

Q1. Draw the pipeline of a Logistic Regression. Explain the working procedure of every step. [10]

1. **Solution:45**



The pipeline you've described involves training a logistic regression model with neural network architecture for binary classification. Here's a step-by-step explanation of how this pipeline works:

1. **Input Data (X1, X2, X3):**
- The input data consists of features X1, X2, and X3. These features represent the input to the logistic regression model and can be real-valued or categorical.

2. **Neural Network:**
- In this context, the term "Neural Network" likely refers to a single-layer network with

a linear transformation (weights and biases) applied to the input features. The weights and biases are learned during training.

3. **Logit:**
- The output of the neural network is passed through a linear transformation, often referred to as the "logit." This transformation represents a weighted sum of the input features, similar to a linear regression model.

4. **Softmax:**
- The output of the logit is then passed through the softmax activation function. Softmax converts the raw logit scores into class probabilities for binary classification. Specifically, it squashes the values into the range [0, 1] and ensures that they sum up to 1. The softmax function is given by:

```
Probability(class=1) = exp(logit) / (exp(logit) + 1)
Probability(class=0) = 1 / (exp(logit) + 1)
```

Where "class=1" represents the positive class (e.g., "True" or "1"), and "class=0" represents the negative class (e.g., "False" or "0").

5. **Probability:**
- After applying the softmax function, you obtain the predicted probabilities for each class. In binary classification, you typically have two classes, so you get two probabilities: one for class 1 and one for class 0. These probabilities represent the model's confidence in the input data belonging to each class.

6. **True Value:**
- The "True Value" represents the actual class label of the input data. For binary classification, it can be either 1 or 0, indicating the ground truth class.

7. **Loss & Optimizer:**
- The loss function (also known as the cost function) measures the error between the predicted probabilities and the true class labels. For binary classification, a common loss function is binary cross-entropy loss, which quantifies the dissimilarity between the predicted probabilities and the true labels.

- The optimizer is responsible for updating the weights and biases of the neural network to minimize the loss function. Gradient descent is a common optimization algorithm used for this purpose. The optimizer computes the gradients of the loss with respect to the model parameters and adjusts them to improve the model's performance.

During training, the optimizer iteratively updates the weights and biases based on the gradients until convergence, aiming to minimize the loss and improve the model's ability to correctly classify new data points. Once the model is trained, you can use it to predict the class probabilities of new input data points.

Set-B

Q1. Why is it necessary for the loss function to be convex? Is squared error function a good choice? Justify your answer. [10]

1. **Solution:45**

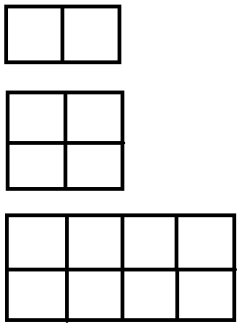
In a convex optimization problem with a convex loss function, there is only one global minimum. This means that regardless of the optimization algorithm used, it will converge to the same solution every time. Non-convex functions may have multiple local minima, leading to convergence issues and potentially finding suboptimal solutions. It guarantees that the solution found is globally optimal, not just locally optimal.

Set-C

Q1. Write down your ideas about information theory. How can the logistics loss function be derived using information theory? [10]

1. **Solution: Rabab 039 (very vague; correct if there's better solution)**

Information theory is a field of mathematics that deals with the number of information required to find out an event to occur. According to this theory, the information required is the negative logarithm of the probability of the event to occur, i.e., I = -logP(x) where I is the information required and P is the probability.



Minimum information required = 1
Number of boxes = 2 = 2^1

Minimum information required = 2
Number of boxes = 4 = 2^2

Minimum information required = 3
Number of boxes = 8 = 2^3

So, for I information, number of boxes, $n = 2^I$

Probability, $P = 1/n = 1/2^I$

So, $I = -\log_2 P$

A key measure in information theory is *Entropy*, which measures the uncertainty of the outcome of an event. It is the product of the actual probability Pr and the information, and is given by, H(x) = Pr(x)I or, $H(x) = -Pr(x)\log_2 P(x)$.

The logistics loss equation can be derived from the equation of entropy by replacing the probability with the output prediction p_i and actual probability with its true value:

$$L(x) = -\sum_{i=1}^{n} t_i \log(p_i)$$

For binary cross-entropy loss, n=2

$$\text{So, } L_{CE}(x) = -\sum_{i=1}^{2} t_i \log(p_i) = -[t\log(p) + (1-t)\log(1-p)]$$

Set-D

Q1. Describe the backward propagation of a logistic regression where the activation function is a "Sigmoid" activation function and the loss function is the "Cross Entropy Loss" function. [10]

1. **Solution:**

45 **d**

Logistic regression : Backward Propagation

$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

Ignoring the (-) sign for now.

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}a}[y \ln(a) + (1 - y) \ln(1 - a)] \\
&= y \cdot \frac{\mathrm{d}}{\mathrm{d}a}[\ln(a)] + (1 - y) \cdot \frac{\mathrm{d}}{\mathrm{d}a}[\ln(1 - a)] \\
&= y \cdot \frac{1}{a} + (1 - y) \cdot \frac{1}{1 - a} \cdot \frac{\mathrm{d}}{\mathrm{d}a}[1 - a] \\
&= \frac{y}{a} + \frac{(1 - y)\left(\frac{\mathrm{d}}{\mathrm{d}a}[1] - \frac{\mathrm{d}}{\mathrm{d}a}[a]\right)}{1 - a}
\end{aligned}$$

log (x) refers to **e base log or the natural logarithm (ln(x))** in mathematical analysis, physics, chemistry, statistics, economics, and some engineering fields.

29

,

Logistic regression : Backward Propagation

$$= \frac{y}{a} + \frac{(1-y)\left(\frac{\mathrm{d}}{\mathrm{d}a}[1] - \frac{\mathrm{d}}{\mathrm{d}a}[a]\right)}{1-a}$$

$$= \frac{y}{a} + \frac{(1-y)(0-1)}{1-a}$$

$$= \frac{y}{a} + \frac{y-1}{1-a}$$

$$= \frac{y}{a} - \frac{1-y}{1-a}$$

Finally, adding the (-) sign.

$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

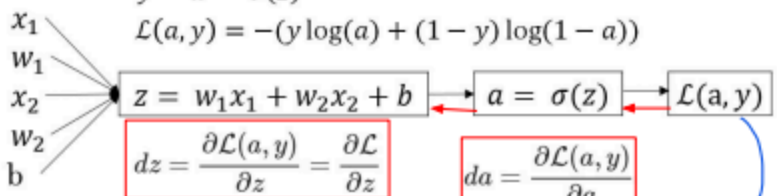
30

Logistic regression : Backward Propagation

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a, y) = -(y \log(a) + (1-y) \log(1-a))$$



$$dz = \frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}}{\partial z}$$

$$da = \frac{\partial \mathcal{L}(a,y)}{\partial a}$$

Applying Chain Rule

$$= \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z}$$

$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

Logistic regression : Backward Propagation

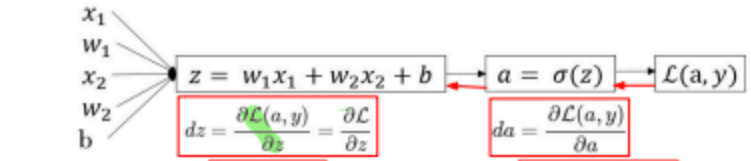
$$\frac{\partial a}{\partial z} = \frac{\partial}{\partial z}\sigma(z)$$
$$= \frac{\partial}{\partial z}\left[\frac{1}{1+e^{-z}}\right]$$
$$= \frac{\partial}{\partial z}(1+e^{-z})^{-1}$$
$$= -(1+e^{-z})^{-2}(-e^{-z})$$
$$= \frac{e^{-z}}{(1+e^{-z})^2}$$
$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

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

$$= \frac{1}{1+e^{-z}} \cdot \frac{(1+e^{-z})-1}{1+e^{-z}}$$
$$= \frac{1}{1+e^{-z}} \cdot \left(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right)$$
$$= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)$$
$$= \sigma(z) \cdot (1-\sigma(z))$$
$$= a \cdot (1-a)$$

32

Logistic regression : Backward Propagation



$$dz = \frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}}{\partial z}$$

$$da = \frac{\partial \mathcal{L}(a,y)}{\partial a}$$
$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

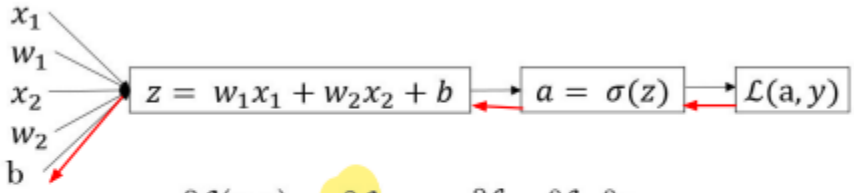
Applying Chain Rule

$$= \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z}$$
$$= \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) * (a \cdot (1-a))$$
$$= -y * (1-a) + a * (1-y)$$
$$= -y + ay + a - ay$$
$$= a - y$$

33

-

Logistic regression : Backward Propagation



$$db = \frac{\partial \mathcal{L}(a,y)}{\partial b} = \frac{\partial \mathcal{L}}{\partial b}$$

$$\frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z}$$

$$dz = a - y$$

Applying Chain Rule

$$= \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial b}$$
$$= dz * \frac{\partial}{\partial b}(w_1x_1 + w_2x_2 + b)$$

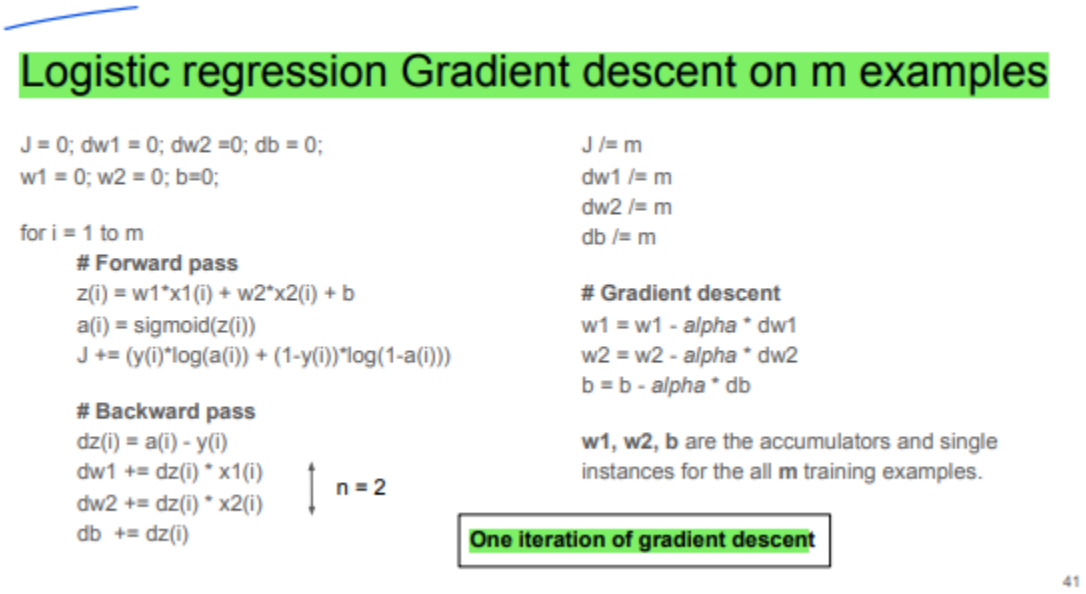
$$db = dz$$

Quiz-2 - Integer43

Set-A

Q1. Write down the algorithm to perform a logistic regression. How can you vectorize the algorithm?

1. **Solution:45**



Set-B

Q1. "A neural network is a combination of several logistic regressions." – Do you agree with the statement? Justify your answer with proper evidence.

Solution: it is true that a neural network can be thought of as a combination of several logistic regressions (or other similar basic units). In fact, neural networks are often described as a collection of interconnected neurons, and each neuron can be

seen as a simplified model inspired by logistic regression.

Neural Network Structure:

1. **Input Layer:** This layer represents the input features, denoted as $x_1, x_2, x_3, \ldots, x_n$.

2. **Hidden Layer:** This layer contains multiple neurons, denoted as $h_1, h_2, h_3, \ldots, h_m$, where m is the number of neurons in the hidden layer.

3. **Output Layer:** This layer produces the final predictions, denoted as y.

Computation in a Neuron (Hidden Layer):

1. **Weighted Sum:** Each neuron in the hidden layer computes a weighted sum of its inputs. This is similar to logistic regression, where each feature is multiplied by a weight.

2. **Activation Function:** The weighted sum z_i is then passed through an activation function, typically a sigmoid function or ReLU. This introduces non-linearity, similar to logistic regression.

$$h_i = \sigma(z_i) \text{ or } h_i = \text{ReLU}(z_i)$$

Here, h_i is the output of neuron i after applying the activation function, and σ represents the sigmoid function or ReLU.

Overall Network Output:

The final prediction y is computed based on the outputs of the neurons in the hidden layer. This can involve another weighted sum and an activation function, depending on the task (e.g., regression or classification).

$$y = \text{Activation}\left(\sum_{i=1}^{m}w_ih_i + b\right)$$

Here, w_i are weights connecting the hidden layer to the output, b is the output layer bias, and Activation is an activation function (e.g., sigmoid for binary classification).

In summary, the structural and computational similarities between individual neurons in a neural network (with one or more hidden layers) and logistic regression are evident. Neural networks extend the principles of logistic regression by composing multiple neurons, interconnected layers, and non-linear activation functions, enabling them to capture complex patterns in data.

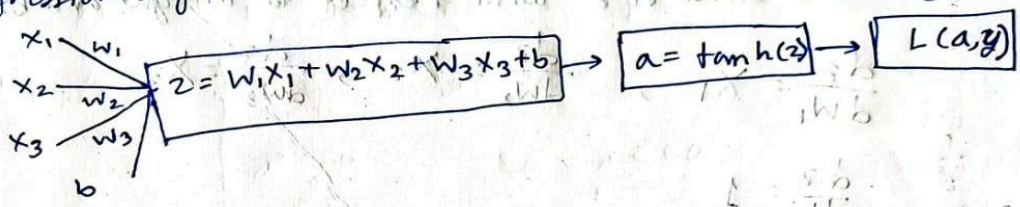
Set-C

Q1. Perform backward propagation on a logistic regression algorithm to modify the parameters in order to minimize the loss, where

- **Sigmoid is the activation function and**
- **Binary cross entropy is the loss function.**

Solution:Aki(80)

Perform backward propagation on the following logistic regression algorith



Soln

We know, For binary classification, the cross-entropy loss is defined as

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

Perform the backward propagation on this logistic regression algorithm.

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$\hat{y} = a = \tanh(z)$$

$L(a,y)$ is the cross entropy loss.

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

$$\frac{dL}{da} = \frac{d}{da}\{-(y\log(a) + (1-y)\log(1-a))\}$$

$$= \frac{-y}{a} + \frac{1-y}{1-a}$$

derivation of tanh function

$$\frac{da}{dz} = \frac{d}{dz}(\tanh(z)) = 1 - \tanh^2(z)$$

$$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz}$$

Logistic regression : Backward Propagation

$$\frac{\partial a}{\partial z} = \frac{\partial}{\partial z}\sigma(z)$$

$$= \frac{\partial}{\partial z}\left[\frac{1}{1+e^{-z}}\right]$$

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

$$= (1+e^{-z})^{-2}(-e^{-z})$$

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

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

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

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

$$= \frac{1}{1+e^{-z}} \cdot \left(\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}\right)$$

$$= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)$$

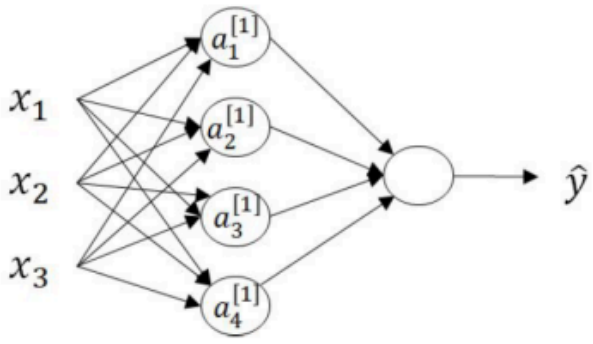
$$= \sigma(z) \cdot (1-\sigma(z))$$

$$= a \cdot (1-a)$$

32

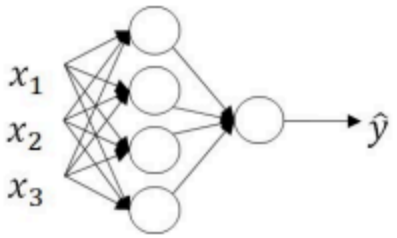
Set-D

Q1. Represent the following neural network in vectorized form for multiple examples and justify the vectorized implementation.



Solution:
45

Vectorizing across multiple examples



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

X ..... $\hat{y} = a^{[2]}$

$x^{(1)}$ ..... $a^{[2](1)} = \hat{y}^{(1)}$

$x^{(2)}$ ..... $a^{[2](2)} = \hat{y}^{(2)}$

$x^{(m)}$ ..... $a^{[2](m)} = \hat{y}^{(m)}$

Layer No 2

$a^{[2](i)}$ → Example (i)

Vectorizing across multiple examples

m Training Example

```
for i = 1 to m:
```

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$a^{[2](i)} = \sigma(z^{[2](i)})$$

One Training Example

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

Set-E

Q1. Why do you need nonlinear activation functions in neural networks? Explain briefly

Solution: by Rabab 039

In linear activation functions, the equations are linear, whose derivatives are constant. For example, differentiating a linear equation *y* = *mx+c* with respect to *x* gives us *m*, which is a constant. During weight update, we need a variable derivative of the loss function, which depends on the outputs. But since we are getting a constant derivative every time, no significant update occurs as it is no longer dependent on the outputs, and thus, the neural network collapses into a single layer i.e. simply a linear regression with limited learning power.

To prevent this, we use nonlinear activation functions like sigmoid, tanh etc. whose derivatives are related to the inputs, and thereby allow backpropagation. They allow “stacking” of multiple layers of neurons to create a deep neural network. Multiple hidden layers of neurons are needed to learn

complex data sets with high levels of accuracy.

Quiz-2 - Previous Ques

Set-A

Q1. Why do you need nonlinear activation functions in neural networks? Explain briefly. [10]

1. **Solution:** given right above

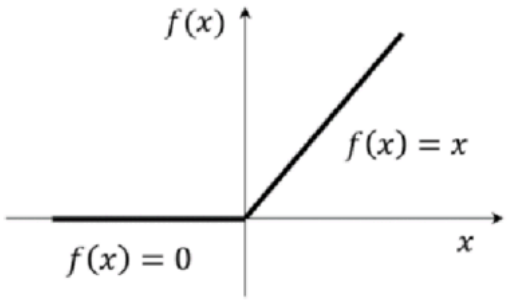
Set-B

Q1. What are the benefits of ReLU activation functio**n? How can you overcome the "Dying ReLU" problem? [10]**

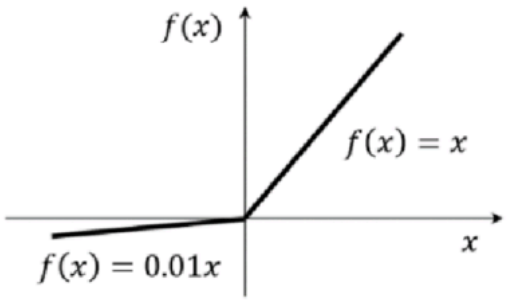
1. **Solution: by Rababe 039**

In multi-layered or deep neural networks, the use of too many nodes leads to huge computation, thus requiring time and cost. It is much easier if we can deactivate some of the nodes that are relatively less significant. ReLU is such an activation function that can do this task. For positive inputs or logits, it is simply a linear regression, but for negatives, it outputs zero. So, many nodes give an output of zero, effectively deactivating them.

However, in the cases where a big majority of the input ranges are negative, too many nodes show the same value of 0, leading to inconsistent results. This is known as the *Dying ReLU Problem*. To overcome this, we use Leaky ReLUs, which have a small slope for negative values instead of a flat slope, thus keeping those inputs somewhat activated.



ReLU activation function



LeakyReLU activation function

Set-C

Q1. "The derivative of tanh activation function depends on the function itself" – Justify the statement. [10]

1. **Solution: Rabab 039**

- Derivative of a tanh function

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

$$\frac{d}{dz}g(z) = slope\ of\ g(z)\ at\ z$$

$$\frac{d}{dz}g(z) = \frac{(e^z+e^{-z})(e^z+e^{-z}) - (e^z-e^{-z})(e^z-e^{-z})}{(e^z+e^{-z})^2} = \frac{(e^z+e^{-z})^2 - (e^z-e^{-z})^2}{(e^z+e^{-z})^2}$$

$$\frac{d}{dz}g(z) = \frac{\frac{(e^z+e^{-z})^2 - (e^z-e^{-z})^2}{(e^z+e^{-z})^2}}{\frac{(e^z+e^{-z})^2}{(e^z+e^{-z})^2}} = \frac{\frac{1 - tanh(z)^2}{1}}{1} = 1 - tanh(z)^2$$

As we can see here, the derivative includes the tanh function itself. So, the derivative depends on the function itself.

Set-D

Q1. In which condition do precision and recall become the same? The confusion matrix below represents the performance of a Linear Regression model. Find out the Accuracy, Precision, Recall and F1-score of the model.

Actual / Predicted	Positive	Negative
Positive	187	67
Negative	75	171

1. **Solution:**

Precision = $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$

In the case where FP = FN i.e. there are an equal number of wrong predictions for each class, the precision and recall become the same.

Here, TP = 187, FP = 67, FN = 75, TN = 171

Precision = $\frac{TP}{TP+FP}$ = $\frac{187}{187+67}$ = 0.736

Recall = $\frac{TP}{TP+FN}$ = $\frac{187}{187+75}$ = 0.713

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$ = $\frac{187+171}{187+171+67+75}$ = 0.716

F1-score = $\frac{2 * Precision * Recall}{Precision + Recall}$ = $\frac{2 * 0.736 * 0.713}{0.736+0.713}$ = 0.7248

Set-E

Q1. Why does the Stochastic Gradient Descent technique oscillate across the ridge? How can this oscillation be reduced? [10]

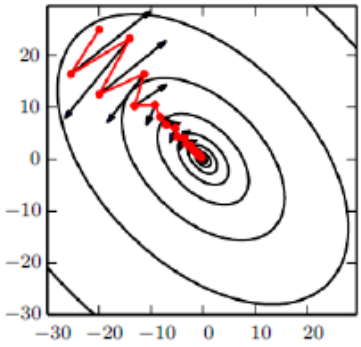
Solution: O24

Reason for oscillation: Randomness

The updates in SGD are noisy and have a high variance, which can make the optimization process less stable and lead to oscillations around the minimum.

Solution of oscillation: Momentum

The method of momentum is designed to accelerate learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients. The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction.



Set-F

Q1. How does the RMSProp algorithm modify the AdaGrad algorithm? Explain with proper equations. [10]

1. **Solution:**

Activation Function

Integer43

b) "A linear activation function turns a neural network or even a deep neural network into just one layer." – Justify the statement with proper evidence. [4]

Linearity and Stacking Layers:
The linear activation function, often represented as f(x) = x, produces a linear relationship between inputs and outputs. When stacking multiple layers with linear activation functions, the composite function remains a linear transformation.
Mathematically, if h(x) = a * x + b is the output of a layer with a linear activation (where a and b are constants), stacking layers results in h(h(...h(x)...)) = a^n * x + b * (1 + a + a^2 + ... + a^(n-1)), which is still a linear function.

Absence of Non-Linearity:
The strength of deep neural networks lies in their ability to capture complex, non-linear relationships in data. However, using a linear activation function throughout all layers eliminates this non-linearity, limiting the network's capacity to learn intricate patterns.

Expressive Power of Deep Networks:
The primary advantage of deep neural networks over shallow ones is their increased expressive power. Non-linear activation functions, like ReLU or Sigmoid, introduce non-linearity, allowing the network to model and learn more sophisticated representations of the input data.

Universal Approximation Theorem:
The Universal Approximation Theorem states that a neural network with a single hidden layer and a non-linear activation function can approximate any continuous function. However, this theorem does not imply the same for networks with only linear activations; their expressive power is limited.

In conclusion, while a linear activation function is suitable for the output layer of regression tasks, using it throughout all layers in a neural network limits the model's capacity to capture complex relationships. Non-linear activations are essential in deep neural networks to unlock their potential for learning intricate patterns in data.

b) Identify the issues of Sigmoid activation function. Are the issues resolved by the hyperbolic tangent activation function? [4]

The Sigmoid activation function and the hyperbolic tangent (tanh) activation function are both

commonly used in neural networks.

Issues with Sigmoid Activation Function:
Vanishing Gradient Problem:

The Sigmoid function squashes input values to the range (0, 1). During backpropagation, gradients can become very small for extreme input values, leading to the vanishing gradient problem. This hinders the learning of deep neural networks as the gradients may become too small to effectively update the weights in earlier layers.

Output Range:

The Sigmoid function outputs values between 0 and 1, which can lead to a lack of centering around zero. This could result in shifting the decision boundary, especially in cases where the input to the sigmoid function is far from zero.

Hyperbolic Tangent Activation Function:
The hyperbolic tangent activation function is an improvement over the Sigmoid function, addressing some of its issues:

Broader Output Range:

The tanh function squashes input values to the range (-1, 1), providing a broader output range compared to the Sigmoid function. This helps mitigate the vanishing gradient problem to some extent.

Zero-Centered Output:

Unlike the Sigmoid function, the tanh function is zero-centered. This feature can help in mitigating issues related to shifting decision boundaries and is often considered beneficial for training neural networks.

Reduced Vanishing Gradient Effect:

While tanh still suffers from the vanishing gradient problem to some degree, its output range helps mitigate the problem compared to the Sigmoid function.

While the hyperbolic tangent activation function **improves upon some of the issues** associated with the Sigmoid function, **it does not completely eliminate** the vanishing gradient problem. More advanced activation functions like **Rectified Linear Unit (ReLU)** and its variants have gained popularity for addressing gradient-related issues in deep neural networks. Choosing an activation function depends on the specific characteristics of the problem and the network architecture being used.

Origin42

2

II) Note down the problems of Softmax classification and explain how to resolve them. Activation Function

Solution: **by Hussain-060 [Please update with better explanation]**
"For very high or very low values
of X, there is almost no change to the prediction. The
derivative values in these regions are very small and
converge to 0. This is called the vanishing gradient and
the learning is minimal." - Slide [Activation Functions- P. 23]

The problem could be resolved with 'normalization'. By normalizing we restrict the values within a certain range. The values are neither very high nor very low anymore, which ensures that the derivative is sufficiently large and not close to 0.

Eita er porer question tar answer hobe.
45
High Computation Cost: Computing the softmax function for a large number of classes can be computationally expensive. Vanishing and Exploding Gradients: During training, the gradients of the loss with respect to the class scores can become very small (vanishing gradients) or very large (exploding gradients). This can make training deep networks with softmax classifiers difficult, and it may require techniques like gradient clipping or careful weight initialization. Overfitting: Softmax classification models can be prone to overfitting when there is insufficient training data or when the model is too complex. Regularization techniques, such as L1 or L2 regularization, are often necessary to mitigate this problem. softmax classification can suffer from the "curse of dimensionality," where the amount of training data required to generalize well increases exponentially with the number of features.

Question 4. [Marks: 14]
a) Define the vanishing gradient problem. How can the vanishing gradient problem of sigmoid or tanh activation function can be overcome? [6] Activation Function

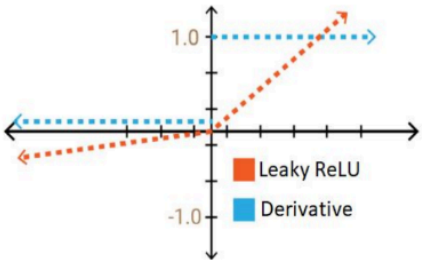
Solution:

- **Vanishing gradient**—for very high or very low values of X, there is almost no change to the prediction. The derivative values in these regions are very small and converge to 0. This is called the vanishing gradient and the learning is minimal.

Problem can be solved by Leaky Relu:
#ekhane tanh or sigmoid er ktha bolche so tanh howar kothana ?? tanh er range -1 to 1 ?? –Swarna(061)
-, the sigmoid function is not centered around zero, which can lead to the vanishing gradient problem.The tanh activation function is similar to the sigmoid but centered around zero. It maps input values to a range between -1 and 1.While sigmoid and tanh activation functions can alleviate the vanishing gradient problem to some degree by providing gradients that are not always close to zero, they are not a perfect solution. For deep neural networks, more advanced activation functions like the rectified linear unit (ReLU) and its variants (e.g., Leaky ReLU, Parametric ReLU) are often preferred because they are less prone to vanishing gradients and have been shown to perform well in practice.-45

Leaky-ReLU Function

- **Prevents dying ReLU problem** — this variation of ReLU has a **small positive slope** in the negative area, so it does **enable backpropagation**, even for negative input values. **This leaky value is given as a value of 0.01 if not +ve.**
- **Results not consistent** — leaky ReLU does not provide consistent predictions for negative input values.



Alternative Ans: Hussain-060
The problem could be resolved with ‘normalization’. By normalizing we restrict the values within a certain range. The values are neither very high nor very low anymore, which ensures that the derivative is sufficiently large and not close to 0.

Enigma41

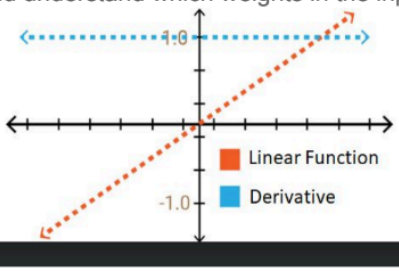
5. i) "A neural network with a linear activation function or without any activation function is simply a linear regression model" – Justify the statement with proper evidence. [6]

Activation Function

Solution:

Solved by -47

derivative of the function is a constant, and has no relation to the input, X. So it's not possible to go back and understand which weights in the input neurons can provide a better prediction.



16

Linear Activation Function

- When **A = c.x** is derived from **x**, we reach **c**. This means that there is no relationship with **x**. If the derivative is **always a constant value**, can we say that the learning process is taking place? **Unfortunately no!**
- **All layers of the neural network collapse into one**—with **linear activation functions, no matter how many layers in the neural network**, the last layer will be a linear function of the first layer (***because a linear combination of linear functions is still a linear function***). **So a linear activation function turns the neural network or even a deep neural network into just one layer.**
- **A neural network with a linear activation function or without any activation function is simply a linear regression model.** It has limited power and ability to handle complexity varying parameters of input data.

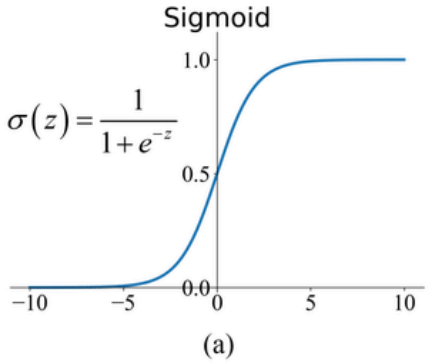
ii) a) What are the differences between sigmoid and tanh activation functions? Explain with proper graphical representation. [8]

b) "The derivative of tanh activation function depends on the function itself" – Justify the statement.

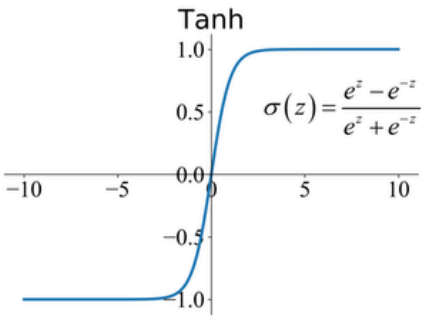
Solution: 024

(i)

Criteria	Sigmoid Function	Tanh Function
Mathematical form	σ(x) = 1 / (1 + exp(-x))	tanh(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x))
Output range	0 to 1	-1 to 1
Centered around zero	No	Yes
Use cases	The output layer of binary classification problems, hidden layers of shallow neural networks	Hidden layers of neural networks
Advantages	Differentiable, introduces non-linearity, used in binary classification problems	A steeper gradient around zero captures small changes in the input
Disadvantages	Suffers from vanishing gradient problem, output not centered around zero	Suffers from vanishing gradient problem, which can cause exploding gradient problem when input is too large



(a)



(b)

(ii)

Tanh (Hyperbolic Tangent) Function:

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

$$\begin{aligned}
\frac{d}{dz}\left(\frac{e^z - e^{-z}}{e^z + e^{-z}}\right) &= \frac{e^z + e^{-z}}{\left(e^z + e^{-z}\right)^2} d\left(e^z - e^{-z}\right) - \frac{e^z - e^{-z}}{\left(e^z + e^{-z}\right)^2} d\left(e^z + e^{-z}\right) \\
&= \frac{\left(e^z + e^{-z}\right)\left(e^z + e^{-z}\right)}{\left(e^z + e^{-z}\right)^2} - \frac{\left(e^z - e^{-z}\right)\left(e^z - e^{-z}\right)}{\left(e^z + e^{-z}\right)^2} \\
&= \frac{\left(e^z + e^{-z}\right)^2 - \left(e^z - e^{-z}\right)^2}{\left(e^z + e^{-z}\right)^2} \\
&= 1 - \left(\frac{e^z - e^{-z}}{e^z + e^{-z}}\right)^2 \\
&= 1 - tanh(z)^2
\end{aligned}$$

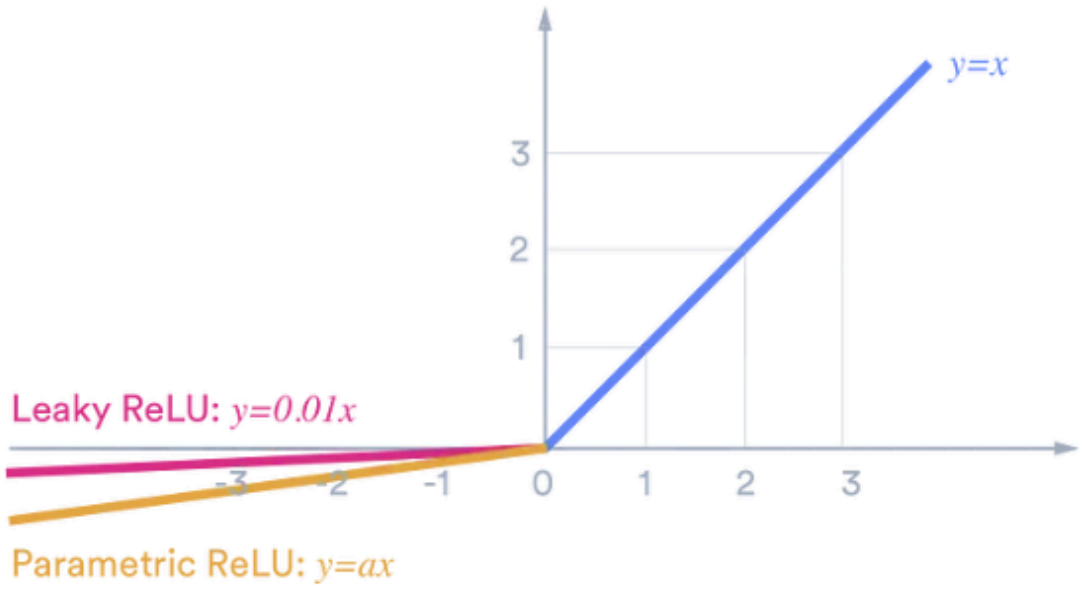
So, the statement is true.

Recurssive40

(4.) A) What is the leaky ReLU activation function? Explain briefly with a proper diagram. [5]
Activation Function

4. i. Solution: Sujon 49

Leaky Rectified Linear Unit, or Leaky ReLU, is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope. The slope coefficient is determined before training, i.e. it is not learnt during training. This type of activation function is popular in tasks where we may suffer from sparse gradients, for example training generative adversarial networks.



Leaky ReLU and Parametric ReLU

A problem we see in ReLU is the Dying ReLU problem where some ReLU Neurons essentially die for all inputs and remain inactive no matter what input is supplied, here no gradient flows and if large number of dead neurons are there in a Neural Network it's performance is affected, this can be corrected by making use of this Leaky ReLU where slope is changed left of x=0 in above figure and thus causing a leak and extending the range of ReLU.

With Leaky ReLU there is a small negative slope, so instead of not firing at all for large gradients, our neurons do output some value and that makes our layer much more

optimized too.

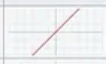
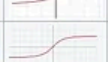
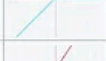
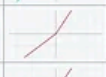
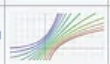
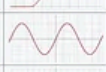
All Activation Functions

By Sujon 49

Wow ami ekhon egula shob mukhostho korbo ki moja! -Rabab039

Yeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee! -Deb065

Why mugosto…bencher moddhe lekhe felais…modern prb modern solution..!! -Tonmoy146

Name	Plot	Equation	Derivative (with respect to x)	Range	Order of continuity	Monotonic	Monotonic derivative	Approximates identity near the origin
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	C^{-1}	Yes	No	No
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$[5]	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	C^∞	Yes	No	No
TanH		$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞	Yes	No	Yes
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$	C^∞	Yes	No	Yes
ElliotSig[9][10][11] Softsign[12][13]		$f(x) = \frac{x}{1+\lvert x\rvert}$	$f'(x) = \frac{1}{(1+\lvert x\rvert)^2}$	$(-1, 1)$	C^1	Yes	No	Yes
Inverse square root unit (ISRU)[14]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$	$f'(x) = \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3$	$\left(-\frac{1}{\sqrt{\alpha}}, \frac{1}{\sqrt{\alpha}}\right)$	C^∞	Yes	No	Yes
Inverse square root linear unit (ISRLU)[14]		$f(x) = \begin{cases} \frac{x}{\sqrt{1+\alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \left(\frac{1}{\sqrt{1+\alpha x^2}}\right)^3 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$\left(-\frac{1}{\sqrt{\alpha}}, \infty\right)$	C^2	Yes	Yes	Yes
Square Nonlinearity (SQNL)[11]		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$	$f'(x) = 1 \mp \frac{x}{2}$	$(-1, 1)$	C^∞	Yes	No	Yes
Rectified linear unit (ReLU)[15]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0	Yes	Yes	No
Bipolar rectified linear unit (BReLU)[16]		$f(x_i) = \begin{cases} ReLU(x_i) & \text{if } i \bmod 2 = 0 \\ -ReLU(-x_i) & \text{if } i \bmod 2 \neq 0 \end{cases}$	$f'(x_i) = \begin{cases} ReLU'(x_i) & \text{if } i \bmod 2 = 0 \\ -ReLU'(-x_i) & \text{if } i \bmod 2 \neq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes	No
Leaky rectified linear unit (Leaky ReLU)[17]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes	No
Parameteric rectified linear unit (PReLU)[18]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$[2]	C^0	Yes iff $\alpha \geq 0$	Yes	Yes iff $\alpha = 1$
Randomized leaky rectified linear unit (RReLU)[19]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$[3]	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0	Yes	Yes	No
Exponential linear unit (ELU)[20]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$(-\alpha, \infty)$	$\begin{cases} C_1 & \text{when } \alpha = 1 \\ C_0 & \text{otherwise} \end{cases}$	Yes iff $\alpha \geq 0$	Yes iff $0 \leq \alpha \leq 1$	Yes iff $\alpha = 1$
Scaled exponential linear unit (SELU)[21]		$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$ with $\lambda = 1.0507$ and $\alpha = 1.67326$	$f'(\alpha, x) = \lambda \begin{cases} \alpha(e^x) & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$	C^0	Yes	No	No
S-shaped rectified linear activation unit (SReLU)[22]		$f_{t_l, a_l, t_r, a_r}(x) = \begin{cases} t_l + a_l(x - t_l) & \text{for } x \leq t_l \\ x & \text{for } t_l < x < t_r \\ t_r + a_r(x - t_r) & \text{for } x \geq t_r \end{cases}$ t_l, a_l, t_r, a_r are parameters.	$f'_{t_l, a_l, t_r, a_r}(x) = \begin{cases} a_l & \text{for } x \leq t_l \\ 1 & \text{for } t_l < x < t_r \\ a_r & \text{for } x \geq t_r \end{cases}$	$(-\infty, \infty)$	C^0	No	No	No
Adaptive piecewise linear (APL)[23]		$f(x) = \max(0, x) + \sum_{s=1}^{S} a_s^i \max(0, -x + b_s^i)$	$f'(x) = H(x) - \sum_{s=1}^{S} a_s^i H(-x + b_s^i)$[4]	$(-\infty, \infty)$	C^0	No	No	No
SoftPlus[24]		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$	$(0, \infty)$	C^∞	Yes	Yes	No
Bent identity		$f(x) = \frac{\sqrt{x^2+1}-1}{2} + x$	$f'(x) = \frac{x}{2\sqrt{x^2+1}} + 1$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes
Sigmoid-weighted linear unit (SiLU)[25] (a.k.a. Swish[26])		$f(x) = x \cdot \sigma(x)$[5]	$f'(x) = f(x) + \sigma(x)(1 - f(x))$[6]	$[\approx -0.28, \infty)$	C^∞	No	No	No
SoftExponential[27]		$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} & \text{for } \alpha < 0 \\ x & \text{for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha & \text{for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1-\alpha(\alpha+x)} & \text{for } \alpha < 0 \\ e^{\alpha x} & \text{for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$	C^∞	Yes	Yes	Yes iff $\alpha = 0$
Soft Clipping[28]		$f(\alpha, x) = \frac{1}{\alpha} \log \frac{1 + e^{\alpha x}}{1 + e^{\alpha(x-1)}}$	$f'(\alpha, x) = \frac{1}{2} \sinh\left(\frac{P}{2}\right) \operatorname{sech}\left(\frac{Px}{2}\right) \operatorname{sech}\left(\frac{P}{2}(1 - x)\right)$	$(0, 1)$	C^∞	Yes	No	No
Sinusoid[29]		$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	C^∞	No	No	Yes
Sinc		$f(x) = \begin{cases} 1 & \text{for } x = 0 \\ \frac{\sin(x)}{x} & \text{for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{for } x \neq 0 \end{cases}$	$[\approx -.217234, 1]$	C^∞	No	No	No
Gaussian		$f(x) = e^{-x^2}$	$f'(x) = -2xe^{-x^2}$	$(0, 1]$	C^∞	No	No	No

Logistic Regression

Ingteger 43

a) What is the role of bias in a logistic regression algorithm? Explain Briefly. [3]

In logistic regression, biases (also known as intercepts) are additional parameters included in the model. Unlike the weights associated with features, biases are not multiplied by any input; instead, they provide an offset to the output. The role of biases in logistic regression can be explained briefly as follows:

Bias as an Intercept:

The bias term allows the logistic regression model to have a non-zero output even when all input features are zero. It represents the log-odds of the probability of the positive class when all input features are zero.

Shifting Decision Boundary:

Including biases allows the decision boundary of the logistic regression model to shift away from the origin. This is important because, without biases, the model would be constrained to always pass through the origin, limiting its flexibility to fit data with different distributions

.

Handling Imbalance and Centering:

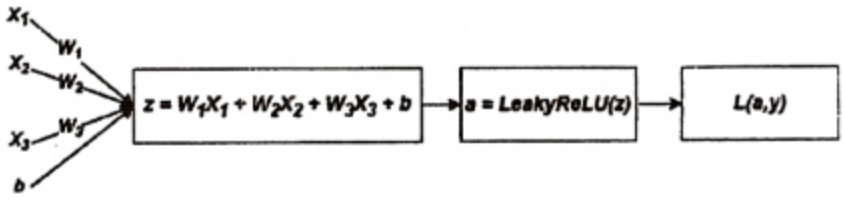
Biases play a role in handling class imbalances. They contribute to the centering of the logistic regression decision boundary, ensuring that the model accounts for both positive and negative classes appropriately.

Enhancing Model Flexibility:

Including biases increases the expressive power of the logistic regression model, enabling it to capture complex relationships in the data beyond what a linear model without biases could achieve.

In summary, biases in logistic regression serve as intercept terms, providing flexibility to the model by allowing it to shift and adapt the decision boundary. They play a crucial role in improving the model's ability to fit diverse datasets and make predictions in scenarios where the input features alone may not be sufficient.

c) Analyze the following logistic regression algorithm and figure out the value of dw_1, dw_2, dw_3 and db. [7]



Here, y is the actual label of the training data, and
L(a,y) denotes the cross entropy loss between a and y.

Origin42

b) I) What role does bias play in a logistic regression? Logistic Regression [8]

Solution:

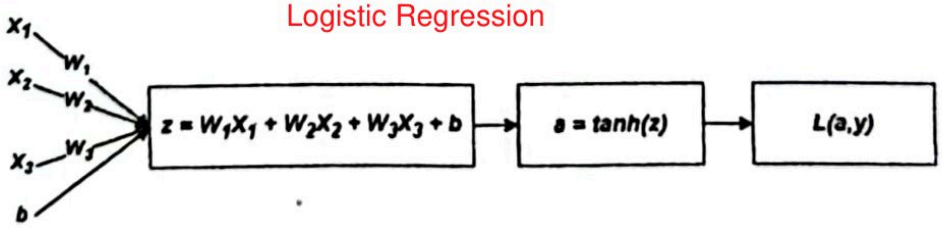
II) "A Neural Network is a combination of several logistic regressions."– Do you agree with the statement? Justify your answer with proper evidence.

Solution:

7

b) Perform backward propagation on the following logistic regression algorithm to modify the parameters in order to minimize the loss. [8]

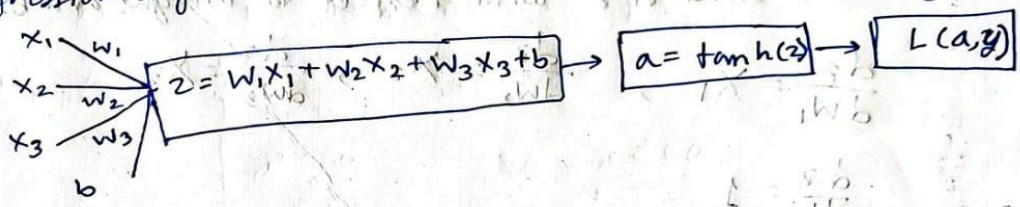
Logistic Regression



Here, y is the actual label of the training data, and
L(a,y) denotes the cross entropy loss between a and y.

Solution:(Akil)80

Perform backward propagation on the following logistic regression algorith



Soln

We know, For binary classification, the cross-entropy loss is defined as

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

Perform the backward propagation on this logistic regression algorithm.

$$z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$\hat{y} = a = \tanh(z)$$

$L(a,y)$ is the cross entropy loss.

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

$$\frac{dL}{da} = \frac{d}{da}\{-(y\log(a) + (1-y)\log(1-a))\}$$

$$= \frac{-y}{a} + \frac{1-y}{1-a}$$

derivation of tanh function

$$\frac{da}{dz} = \frac{d}{dz}(\tanh(z)) = 1 - \tanh^2(z)$$

$$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz}$$

Gradient of z with respect to weight and bias

$$\frac{dz}{dw_1} = x_1 \;,\quad \frac{dz}{dw_2} = x_2 \;,\quad \frac{dz}{dw_3} = x_3$$

$$\frac{dz}{db} = 1$$

Gradient of loss / Chain Rule

$$\frac{dL}{dw_1} = \frac{dL}{dz} \times x_1 \quad \text{or,} \quad \frac{dL}{dz} \times \frac{dz}{dw_1}$$

$$\frac{dL}{dw_2} = \frac{dL}{dz} \times x_2$$

$$\frac{dL}{dw_3} = \frac{dL}{dz} \times x_3$$

$$\frac{dL}{db} = \frac{dL}{dz}$$

Now, parameter update, Here learning rate = α

$$w_1 = w_1 - \alpha \times \frac{dL}{dw_1}$$

$$w_2 = w_2 - \alpha \times \frac{dL}{dw_2}$$

$$w_3 = w_3 - \alpha \times \frac{dL}{dw_3}$$

$$b = b - \alpha \times \frac{dL}{db}$$

So, it is the backward propagation for the logistic model. Now, the parameters w_1, w_2, w_3 and b should be changed in a way that minimises the lo

Tune Hyperparameters: The learning rate, batch size, and other hyperparameters can have a significant effect on how well you minimize the cross-entropy loss. Consider using techniques like grid search or random search to find the best hyperparameters for your model.

Enigma41

6. i) The confusion matrix below represents the performance of a Linear Regression model. [6]
Find out the Accuracy, Precision, Recall and F1-score of the model.

Performance Evaluation

Actual / Predicted	Positive	Negative
Positive	187	67
Negative	75	171

6. i. Solution: Sujon 49

TP = 187	FP = 67
FN = 75	TN = 171

Accuracy = (TP + TN) / (TP + TN + FP + FN)
= (187 + 171) / (187 + 171 + 67 + 75)
= 358 / 500
= 0.716 (or 71.6%)

Precision = TP / (TP + FP)
= 187 / (187 + 67)
= 187 / 254
≈ 0.736 (or 73.6%)

Recall = TP / (TP + FN)
= 187 / (187 + 75)
= 187 / 262
≈ 0.714 (or 71.4%)

F1-Score = 2 * (Precision * Recall) / (Precision + Recall)
= 2 * (0.736 * 0.714) / (0.736 + 0.714)
≈ 0.725 (or 72.5%)

Optimizer

Notes By Paul

SGD

Algorithm 8.1 Stochastic gradient descent (SGD) update at training iteration k

Require: Learning rate ϵ_k.
Require: Initial parameter $\boldsymbol{\theta}$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$
end while

Algorithm 8.2 Stochastic gradient descent (SGD) with momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient estimate: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
end while

1. i) Why does the Stochastic Gradient Descent technique oscillate across the ridge? How can this oscillation be reduced? Optimizer

Ans:

Why SGD Oscillates:

- Steep Gradients on Sides: In areas where the loss function has a steep "ridge" or slope, the gradients are large but not necessarily pointing directly towards the global minimum. They often point across the valley.
- Small Batch Size: Because SGD updates parameters with a small batch of data at a time, the estimate of the gradient can have a lot of variance, leading to updates that don't consistently move in the optimal direction.

- Fixed Step Size: The learning rate in SGD is typically constant for each update, so it may not adapt to the topography of the loss function. A step that's too large can overshoot the minimum, causing the algorithm to bounce back and forth across the valley.

How to Reduce Oscillation:

Use momentum

- **Navigates Valleys Better:** Without momentum, standard Stochastic Gradient Descent (SGD) tends to oscillate back and forth across the narrow, steep sides of a valley because the gradient is much steeper in directions that do not point toward the optimum. With momentum, these oscillations are dampened, so the algorithm can proceed more directly toward the bottom of the valley.

- **Faster Convergence:** Momentum helps the algorithm to build up speed along surfaces that slope gently and consistently towards the optimum. This means that when the path to the solution is straightforward, momentum will accumulate, allowing the optimizer to take bigger steps and thus reach the solution faster.

SGD with nesterov

SGD with Classical Momentum:

- Velocity Update: First, compute the new velocity by applying the momentum to the previous velocity and then subtracting the gradient times the learning rate.
- Parameter Update: Update the parameters by adding the new velocity to the current parameters.

In classical momentum, the gradient used to update the velocity is based on the current position of the parameters.

Algorithm 8.3 Stochastic gradient descent (SGD) with Nesterov momentum

Require: Learning rate ϵ, momentum parameter α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding labels $\boldsymbol{y}^{(i)}$.
Apply interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
Compute gradient (at interim point): $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \epsilon \boldsymbol{g}$
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
end while

SGD with Nesterov Momentum:

- **Interim Update**: First, make a temporary update to the parameters by adding the momentum term to the current parameters.
- **Gradient Computation:** Compute the gradient not at the current parameters, but at this temporary updated position.
- **Velocity Update:** Update the velocity by applying momentum and then subtracting the newly computed gradient.
- **Parameter Update:** Finally, update the parameters by adding the updated velocity to the current parameters.

The key innovation in Nesterov momentum is that it effectively looks ahead by computing the gradient after the momentum has been applied. This "look-ahead" gradient means Nesterov momentum can correct its course more rapidly if the momentum is leading in the wrong direction, allowing for potentially faster convergence and reduced oscillation.

In summary, while both methods use a form of momentum to accelerate the optimization process, Nesterov momentum incorporates the momentum term earlier when computing the gradient, which can often result in improved performance and faster convergence.

AdaGrad

Algorithm 8.4 The AdaGrad algorithm

Require: Global learning rate ϵ

Require: Initial parameter $\boldsymbol{\theta}$

Require: Small constant δ, perhaps 10^{-7}, for numerical stability

Initialize gradient accumulation variable $\boldsymbol{r} = \mathbf{0}$

while stopping criterion not met **do**

Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.

Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$

Accumulate squared gradient: $\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$

Compute update: $\Delta\boldsymbol{\theta} \leftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. (Division and square root applied element-wise)

Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$

end while

Small Constant (δ): A small constant added for numerical stability – it prevents division by zero.

Gradient Accumulation Variable (r): This keeps track of the sum of the squares of the gradients w.r.t. to each parameter θ.

The steps of the algorithm are as follows:

Initialize r: Set the gradient accumulation variable to zero.

Loop Until Convergence: Repeat the following steps until the stopping criterion is met (e.g., a certain number of iterations or a minimal change in θ).

Sample Mini-batch: Sample a mini-batch of data points and their corresponding targets.

Compute Gradient (g): Calculate the gradient of the loss function with respect to the parameters (θ) based on the mini-batch.

Accumulate Squared Gradient (r): Add the element-wise square of the gradient to the accumulation variable.

Compute Update (Δθ): Calculate the parameter update, which involves scaling the learning rate by the inverse square root of the accumulation variable (plus the small constant δ for numerical stability). This scaling is done element-wise.

Apply Update: Adjust the parameters by subtracting the computed update from the current parameter values.

The key idea of AdaGrad is to adapt the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent ones. This is beneficial for sparse data where some features can be infrequent but informative.

A known limitation of AdaGrad is that the accumulated squared gradients in the denominator can grow very large over time, causing the learning rate to shrink and the algorithm to stop learning entirely. This issue is addressed in later adaptations, like RMSprop and Adam, which use different forms of moving averages to prevent the learning rate from diminishing too quickly.

RMS prop

Algorithm 8.5 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, usually 10^{-6}, used to stabilize division by small numbers.
Initialize accumulation variables $\boldsymbol{r} = 0$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$
Compute parameter update: $\Delta \boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\delta + \boldsymbol{r}}}$ applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$
end while

Steps of RMSProp:

Initialize Accumulation Variables: Set the gradient accumulation variable r to 0. This variable will store a moving average of the squared gradients.

Loop Until Convergence: Repeat the following steps until a stopping criterion is met, such as a small change in θ or a maximum number of iterations.

Sample Mini-batch: Select a mini-batch of m examples from the training data.

Compute Gradient: Calculate the gradient g of the loss function with respect to the parameters θ, using the mini-batch.

Accumulate Squared Gradient: Update the accumulation variable r by decaying the previous value by a factor ρ (decay rate) and adding the element-wise square of the gradient g, scaled by (1 - ρ).

Compute Parameter Update: Calculate the update Δθ by dividing the learning rate ε by the square root of r + δ (where δ is a small constant to prevent division by zero), and then applying the element-wise multiplication with the gradient g.

Apply Update: Update the parameters θ by subtracting Δθ.

Algorithm 8.6 RMSProp algorithm with Nesterov momentum

Require: Global learning rate ϵ, decay rate ρ, momentum coefficient α.
Require: Initial parameter $\boldsymbol{\theta}$, initial velocity $\boldsymbol{v}$.
  Initialize accumulation variable $\boldsymbol{r} = \boldsymbol{0}$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute interim update: $\tilde{\boldsymbol{\theta}} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{v}$
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\tilde{\boldsymbol{\theta}}} \sum_i L(f(\boldsymbol{x}^{(i)}; \tilde{\boldsymbol{\theta}}), \boldsymbol{y}^{(i)})$
    Accumulate gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$
    Compute velocity update: $\boldsymbol{v} \leftarrow \alpha \boldsymbol{v} - \frac{\epsilon}{\sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\boldsymbol{r}}}$ applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \boldsymbol{v}$
  end while

Comparison with Regular RMSProp:

The key difference between RMSProp with Nesterov momentum and standard RMSProp is the addition of the Nesterov momentum term. In standard RMSProp, the gradient is computed based on the current position of the parameters, and the update is applied immediately. With Nesterov momentum, the gradient is computed at a "look-ahead" position, allowing the algorithm to correct its course more effectively if it is heading towards a sub-optimal direction.

The use of Nesterov momentum can potentially result in faster convergence and less oscillation during training, especially in the context of complex optimization landscapes common in deep learning.

Standard momentum might cause you to overshoot the minimum because you calculate the gradient and then add momentum, which might lead you to miss the lowest point if the momentum is too high.

Nesterov momentum helps to anticipate where the parameters are going to be after the update and calculates the gradient there. This 'foreseeing' step means you get a more accurate update because you're effectively looking at the landscape ahead and adjusting your movements accordingly.

Adam

Algorithm 8.7 The Adam algorithm

Require: Step size ϵ (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in $[0, 1)$. (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization. (Suggested default: 10^{-8})
Require: Initial parameters $\boldsymbol{\theta}$
  Initialize 1st and 2nd moment variables $\boldsymbol{s} = \mathbf{0}$, $\boldsymbol{r} = \mathbf{0}$
  Initialize time step $t = 0$
  while stopping criterion not met **do**
    Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
    Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
    $t \leftarrow t + 1$
    Update biased first moment estimate: $\boldsymbol{s} \leftarrow \rho_1 \boldsymbol{s} + (1 - \rho_1)\boldsymbol{g}$
    Update biased second moment estimate: $\boldsymbol{r} \leftarrow \rho_2 \boldsymbol{r} + (1 - \rho_2)\boldsymbol{g} \odot \boldsymbol{g}$
    Correct bias in first moment: $\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1 - \rho_1^t}$
    Correct bias in second moment: $\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1 - \rho_2^t}$
    Compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta}$ (operations applied element-wise)
    Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
  end while

First Moment (s):

- The first moment is the moving average of the gradients. It's like measuring the speed of a car; it tells you how fast the parameters are changing.
- It's calculated as a combination of the gradient (how much we want to change parameters based on how wrong they were) and the previous first moment (how much we changed parameters in the past).
- This helps to smooth out the updates, which can make the learning process more stable. It's beneficial because it can prevent the optimizer from making too large updates in any single iteration, which can cause the learning process to become unstable.

Second Moment (r):

- The second moment is the moving average of the squares of the gradients. It's akin to measuring the acceleration of a car; it tells you how fast the speed itself is changing.
- It's calculated as a combination of the squared gradient (which emphasizes larger errors) and the previous second moment.
- This helps to adapt the learning rate based on the recent history of gradients. If the gradients are consistently large, the optimizer will take smaller steps. If the gradients are

small, it can take larger steps. This is beneficial because it allows for finer control over the learning process, letting the algorithm take confident steps in flat areas (where the gradients are small) and cautious steps in steep areas (where the gradients are large).
- Both moments are corrected to counteract their initialization at zero, which is biased towards zero, especially during the initial time steps. Bias correction allows the moments to more accurately represent the actual gradients and gradient squares.

Benefits of Using Both Moments:

- The first moment accounts for the magnitude of the gradients (speed), providing momentum to the optimization process, which can help escape local minima.
- The second moment adjusts the learning rate based on the variance of the gradients (acceleration), which ensures that each parameter's update is scaled appropriately, preventing too large updates that could overshoot minima.
- Together, they allow Adam to be more effective in practice than algorithms that consider only the first moment (like SGD with momentum) or only the second moment (like RMSProp). They enable Adam to combine the benefits of both approaches, making it robust to different kinds of objective functions and different parts of the parameter space.

First Moment (Mean of Gradients):

- Think of the first moment as a measure of the direction and speed of your parameter updates. In physical terms, it's like momentum: if you've been moving in a certain direction, you're likely to keep moving in that direction.
- Mathematically, it's a moving average of the gradients. Because Adam starts with an initial moving average of zero, the first few updates are smaller than they should be (since the average of gradients starts from zero and slowly "warms up"). This is the "bias" towards zero.

Second Moment (Unscaled Variance of Gradients):

- The second moment is a measure of the variability or spread of your gradients. It's like a measure of the terrain's roughness as you're trying to roll a ball down a hill. If the terrain (gradients) is very rough (high variance), you'll want to be more careful with your steps (updates).
- It is the moving average of the squared gradients. Just like the first moment, this average also starts from zero and slowly "warms up," leading to a bias towards zero.

Bias Correction:

- Because both first and second moments are biased towards zero at the start (due to initial values of zero), Adam performs a correction to make them more accurate. Without this correction, the algorithm would think the gradients are smaller than they really are and make smaller updates, especially early on.
- The bias correction adjusts these moments by scaling them up based on how many steps (iterations) you've taken. The idea is to compensate for the initial underestimation so that the corrected moments accurately reflect the true scale of the gradients.
- Practically, it's like saying, "I've been underestimating the slope of this hill because I'm just starting to roll; let me adjust my expectations to be more realistic about the steepness."

How Bias Correction Works:

- For the first moment, we divide by $1 - \beta_1^t$, where β_1 is the decay rate for the first moment, and t is the current time step. Since β_1 is less than 1, raising it to the power of t makes it very small, so we're dividing by a number just less than 1, scaling the first moment up.
- For the second moment, we divide by $1 - \beta_2^t$, with similar logic.

Benefit of Bias Correction:

- This process ensures that at the start, when the estimates of the moments are too low, the corrections help the optimizer to make appropriately larger updates.
- As more iterations pass and t gets larger, the correction factor $1 - \beta_1^t$ (or $1 - \beta_2^t$) gets closer to 1, which means the correction effect diminishes, as the moving averages have "warmed up" and are no longer biased.

Differences:

Stochastic Gradient Descent (SGD):

- Uses a mini-batch of data to compute the gradient and update the parameters.
- Learning rate is fixed or manually decayed over time.
- Can oscillate and take a longer time to converge due to the variance in gradient estimates.

SGD with Momentum:

- Similar to SGD but uses a velocity vector to smooth out updates.
- Accumulates gradient direction from previous steps to dampen oscillations and converge faster.

SGD with Nesterov Momentum:

- An enhancement of momentum SGD that calculates the gradient at a "look-ahead" position based on the current momentum.
- Allows for correction before the parameter update, potentially leading to faster convergence.

AdaGrad:

- Adjusts the learning rate based on the historical sum of squared gradients.
- Learning rate is scaled inversely proportional to the square root of the accumulation of past squared gradients.
- Can be too aggressive with learning rate reduction, effectively stopping early in training.

RMSProp:

- Modifies AdaGrad by using a moving average of squared gradients instead of summing all past squared gradients.
- Prevents the learning rate from diminishing to an extremely small value, which addresses AdaGrad's aggressive decay.

RMSProp with Nesterov Momentum:

- Combines RMSProp's adaptive learning rate with Nesterov momentum.
- Benefits from the anticipatory update of Nesterov momentum and the adaptive learning rate of RMSProp.

Adam:

- Combines ideas from both momentum and RMSProp.
- Maintains a moving average of both gradients and squared gradients and includes bias corrections for both of these estimates.
- Adjusts the learning rate for each parameter based on the first and second moments.

Key Differences:

- **Learning Rate Adaptation:** SGD does not adapt the learning rate per parameter, while AdaGrad, RMSProp, and Adam do, using different strategies to adjust it based on historical gradient information.
- **Momentum:** Only the algorithms with "momentum" in the name explicitly incorporate momentum to accelerate the convergence.
- **Nesterov Momentum:** Provides a "look-ahead" feature by computing the gradient after a preliminary update based on current momentum, which can lead to more effective parameter updates.
- **Gradient Accumulation:** AdaGrad accumulates the square of gradients over all past steps, while RMSProp and Adam use a moving average, preventing excessive diminishing of the learning rate.
- **Bias Correction:** Adam specifically includes a bias correction step for both first and second moment estimates, which helps to provide more accurate estimates early in training.

Each algorithm has its advantages and is suited to different types of problems. For instance, Adam is often preferred for training deep learning models due to its robustness and good performance on a wide range of problems. RMSProp is also commonly used, especially in recurrent neural networks. Classic SGD and SGD with momentum are simpler and can still be effective, especially when coupled with learning rate schedules.

Integer43

a) Explain the importance of an optimizer in a neural network. [3]
By Paul 023

In neural networks, an optimizer is a crucial component that governs the update of the network's weights in response to the error output from the loss function. The role of the optimizer is to minimize (or, in some cases, maximize) the loss function, which measures the difference between the network's predictions and the actual data.

Here's why the optimizer is so important:

1. **Convergence**: An optimizer determines how quickly and effectively a neural network can converge to the lowest possible loss, which typically corresponds to the most accurate predictions the model can provide.
2. **Navigating the Loss Landscape**: Neural networks, especially deep ones, often have complex loss landscapes with many local minima, maxima, and plateaus. An effective optimizer navigates this landscape to find a good minimum that corresponds to a well-performing model.
3. **Speed of Training**: Different optimizers converge at different speeds. Some, like SGD, may take more time compared to more advanced ones like Adam, which can converge faster due to adaptive learning rates.
4. **Handling Vanishing/Exploding Gradients**: Some optimizers are designed to mitigate common issues like vanishing and exploding gradients, which can halt the learning process or lead to unstable training.

c) Examine the algorithm of RMSProp optimizer and explain how it contrasts from AdaGrad algorithm. [7]

By Paul 023

Modification from AdaGrad:

- AdaGrad accumulates the squared gradients `g^2` directly, without any decay rate, leading to a monotonically increasing accumulation that can grow very large, causing the learning rate to become excessively small.
  AdaGrad update rule for `θ`:

$$\theta_{\text{new}} = \theta_{\text{old}} - \frac{\epsilon}{\sqrt{\sum_{i=1}^{t} g_i^2 + \delta}} g_t$$

- RMSProp modifies this by introducing a decay rate `ρ` that weights the accumulated squared gradient more towards recent gradients, preventing the accumulation from growing too large.
  RMSProp update rule for `θ`:

$$r = \rho r + (1 - \rho) g^2$$

$$\theta_{\text{new}} = \theta_{\text{old}} - \frac{\epsilon}{\sqrt{r + \delta}} g$$

Contrast with AdaGrad:

- While AdaGrad's accumulated squared gradient grows without bound, RMSProp's accumulation is a moving average, which does not grow indefinitely due to the decay rate.
- This means RMSProp can continue learning even when AdaGrad's updates become negligibly small because the denominator in RMSProp's update rule doesn't grow as large as AdaGrad's, which prevents the learning rate from diminishing too much.
- RMSProp is generally more robust and can perform better on problems where AdaGrad slows down too fast or stops making progress.

Origin42

Question 7. [Marks: 14]

a) What is the importance of gradient descent in neural network? Write down the algorithm of Adam optimizer. [6]

Optimizer

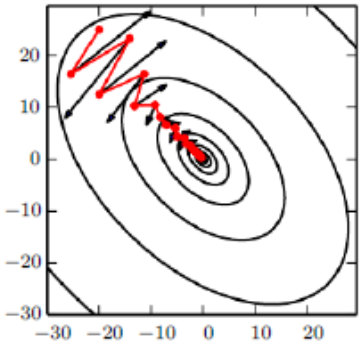
Solution:

Enigma41

1. i) Why does the Stochastic Gradient Descent technique oscillate across the ridge? How can this oscillation be reduced? [6] Optimizer

Solution: O24
Reason for oscillation: Randomness
The updates in SGD are noisy and have a high variance, which can make the optimization process less stable and lead to oscillations around the minimum.

Solution of oscillation: Momentum
The method of momentum is designed to accelerate learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients. The momentum algorithm accumulates an exponentially decaying moving average of past gradients and continues to move in their direction.



Recurssive40

2 i) Write down all the steps of RMSProp or Momentum Gradient Descent with equations. [5] Optimizer

Solution:

Cross Entropy Loss

Recurssive40

3

ii) Consider the following loss function where $p \in [0, 1]$ is the probability score and $w_1, w_2 \in \Re$ are scalar values. [9]

$$Loss = w_1 log(p) + w_2 log(1 - p)$$

Now, answer the following questions: Cross Entropy Loss

a) Write down the behavior of the loss function when $w_1 = w_2$.

b) What are the necessary conditions so that the loss function will behave like a Binary Cross Entropy (BCE) loss?

c) "Loss function is positive only if both w_1 and w_2 are non-positive."— Do you agree? Justify your answer.

Solution:

Solution:

Information Theory

Gradient Decent

** Performance Analysis **

Ei naam e kono topic sir er Question pattern e nai.
So, eta onno kono topic er under e kina janais …

Recurssive40

i) What do you mean by the bias-variance tradeoff? Show the proper equation and diagram. [5]

ii) Consider a binary classification task where you have implemented three models, i.e., Model A, Model B, and Model C, to classify positive and negative classes. The performances of these three models on a test dataset are shown below. [9]

Performance Analysis

	Model A	*Model B*	*Model C*
Precision	*0.92*	*0.72*	*0.80*
Recall	*0.78*	*0.90*	*0.80*
	0.844	0.8	0.8

Now, answer the following questions:

a) Which model do you select if the ratio of the positive class to the negative class is 1?

b) Can you figure out the most accurate model among the three? If not, what is the minimal information you need to figure out the most accurate model?

c) Which model do you think is the better performer between Model B and Model C?

Solution:

D : 1 set

Quiz 2 -Qubits45

Set A

1. Describe the role of a ReLU layer in a convolutional neural network. [3]

era-038

Ans: The role of a ReLU layer in a CNN includes the following:

a. Non-Linearity: The ReLU layer adds a touch of complexity to the network by making sure that, if the input is positive, it stays as it is, but if it's negative, it turns into zero. This introduces non-linearity, allowing the network to learn and understand more complicated patterns in data.

b. Feature Map Sparsity: ReLU helps create sparse feature maps by zeroing out negative values. This sparsity can be beneficial as it encourages the network to focus on more important features, improving generalization and reducing the risk of overfitting.

c. Gradient Descent Optimization: ReLU facilitates gradient-based optimization during backpropagation. Its derivative is either 0 or 1, making it computationally efficient for calculating gradients

"it is important to note that ReLU can suffer from the "dying ReLU" problem, where neurons may become inactive during training and stop learning."

d. Improved Training Convergence: The non-saturation property of ReLU (unlike some other activation functions that saturate for large positive or negative inputs) can lead to faster convergence during training, as it mitigates the vanishing gradient problem in deep networks.

Set B

1. Describe the role of a convolution layer in a convolutional neural network. [3]

era 38

In a Convolutional Neural Network (CNN), a convolutional layer plays a crucial role in capturing patterns and features from input data, such as images. Here's a simple description:

1. Feature Extraction:

- The convolutional layer scans the input image with small filters (also known as kernels or convolutions).

- These filters capture local patterns or features, like edges, corners, or textures, in the image.

2. Spatial Hierarchy:

- By using multiple filters, the convolutional layer creates a spatial hierarchy of features, **learning to recognize more complex patterns as it goes deeper into the network.**

3. Parameter Sharing:

- Parameters (weights) of the filters are shared across the entire image, reducing the number of parameters and making the network more efficient.

4. Translation Invariance:

- Convolutional layers introduce translation invariance, meaning the network can recognize patterns regardless of their exact position in the image.

In simpler terms, think of a convolutional layer as a feature detector that slides over an image, recognizing different shapes and patterns. This helps the neural network understand and learn hierarchical representations of visual features, making it effective for tasks like image recognition.

Quiz 3 -Qubits45

Set A

Q3. Analyze the following architecture and figure out the number of parameters for the input shape of (442, 442).

input→ cnn _1→ dropout→ relu→ maxpool→ cnn _2→ relu→ maxpool

output ←linear_3 ←relu ←linear_2 ←relu ←linear_1 ←flatten ←

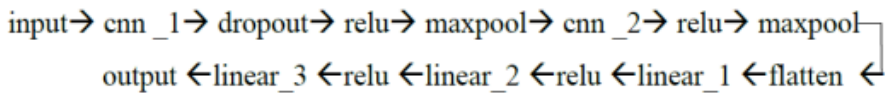
Here,
cnn _1: Conv2d(in_feature=1, out_feature=4, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
cnn _2: Conv2d(in_feature=4, out_feature=8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
maxpool: MaxPool2d(kernel_size=2, stride=2, padding=0)
linear_1: Linear(in_features=95920, out_features=512, bias=True)
linear_2: Linear(in_features=512, out_features=128, bias=True)
linear_3: Linear(in_features=128, out_features=10, bias=True)
flatten: Flatten(start_dim=1, end_dim=-1)
dropout: Dropout(p=0.2, inplace=False)
relu: ReLU()

PARVEZ 129

Input Layer	Output Shae	Parameter
input	442 , 442	0
cnn_1	(442- 5 + (2*2) / 1) + 1 442 , 442 , 4	((5*5*1) + 1) * 4 104
dropout	442 , 442 , 4	0
relu	442 , 442 , 4	0
maxpool	(442-2 + 0) /2 +1 221 , 221 , 4	0
cnn_2	(221-5 + 2*2) / 1 + 1 221 , 221 , 8	((5*5*4) + 1) * 8 808
relu	221 , 221 , 8	0
maxpool	110 , 110 , 8	0
flatten	110 * 110 * 8 96800	0
linear_1	512	(96800 +1) * 512 49562112
relu	512	0
linear_2	128	(512+1) * 128 65664
relu	128	0
linear_3	10	(128 +1) * 10 1290
output	10	0
		49629978

Set B

Q3. Analyze the following architecture and figure out the number of parameters for the input shape of (445, 445).



Here,

cnn _1: Conv2d(in_feature=1, out_feature=4, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
cnn _2: Conv2d(in_feature=4, out_feature=8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
maxpool: MaxPool2d(kernel_size=2, stride=2, padding=0)
linear_1: Linear(in_features=95920, out_features=512, bias=True)
linear_2: Linear(in_features=512, out_features=128, bias=True)
linear_3: Linear(in_features=128, out_features=10, bias=True)
flatten: Flatten(start_dim=1, end_dim=-1)
dropout: Dropout(p=0.2, inplace=False)
relu: ReLU()

set B

input (438, 438)

input → conv (3,2,16) → meanpool(4) → ReLU → maxpool(3) → FCwB (256) → FC(5) → output

conv(x,y,z) x → kernel size; y = stride, z = output channel

mean/maxpool(x) → both window size & stride = x.

FCwB → with bias FC → without bias.

layer	output shape	parameter
Input	(438, 438)	0
conv	$\left\lfloor \frac{438-3+0\times2}{2} \right\rfloor + 1 = 218$ (16, 218, 218)	((3*3*1)+1)*16 = 160
meanpool	$\left\lfloor \frac{218-4+2\times0}{4} \right\rfloor + 1 = 54$ (16, 54, 54)	0
Relu	16, 54, 54	0
maxpool	$\left\lfloor \frac{54-3+2\times0}{3} \right\rfloor + 1 = 18$ (16, 18, 18)	0
FCwB	256	flatten করতে (16×18×18)×256 + 256 = 1327360
FC	5	256×5 = 1280
output	5	0

Total = 1328800

Set C

Q3. Analyze the following architecture and figure out the number of parameters for the input shape of (427, 427).

input→ cnn _1→ dropout→ relu→ maxpool→ cnn _2→ relu→ maxpool

output ←linear_3 ←relu ←linear_2 ←relu ←linear_1 ←flatten ←

Here,

cnn _1: Conv2d(in_feature=1, out_feature=4, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
cnn _2: Conv2d(in_feature=4, out_feature=8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
maxpool: MaxPool2d(kernel_size=2, stride=2, padding=0)
linear_1: Linear(in_features=95920, out_features=512, bias=True)
linear_2: Linear(in_features=512, out_features=128, bias=True)
linear_3: Linear(in_features=128, out_features=10, bias=True)
flatten: Flatten(start_dim=1, end_dim=-1)
dropout: Dropout(p=0.2, inplace=False)
relu: ReLU()

Set D

Q3. Calculate the number of parameters for the following architecture for the input shape of (438,438).

input→Conv(3,2,16)→Minpool(4)→ReLU→Maxpool(3)→FCwB(256)→FC(5)→output

Here, Conv(x,y,z) denotes a convolutional layer that has the kernel size of x, stride size of y, and output channel of z.
Minpool(x) denotes a Minpooling layer that has both window size and stride size of x,
Maxpool(x) denotes a Maxpooling layer that has both window size and stride size of x,
FCwB(x) denotes a Fully Connected Layer of x neurons with Bias, and
FC(x) denotes a Fully Connected Layer of x neurons without Bias.

Parvez 129 [CHECK CALCULATION]

Output Shape = [floor(W/H - K + 2P) / S] + 1 ---------- [K = kernel , P = Padding , S = Stride ,W/H = Width or Height]
Parameter = (m * n * d +1) * K

Layer	Output Shape	Parameter
input	438 ,438	0

conv(3 , 2 , 16)	floor((438 - 3 + (2* 0)) / 3) + 1 = 217 + 1 = 218 218 ,218 ,16	((3 * 3 * 1) + 1) * 16 = (9 +1) * 16 = 160 160
Minpool(4)	floor((218- 4 + (2* 0)) / 4) + 1 = 53 + 1 = 54 54 , 54 ,16	0
Relu	**[ACTIVATION FUNCTION DOES NO CHANGE]** 54 ,54 ,16	0
Maxpool(3)	floor((54- 3 + (2* 0)) / 3) + 1 = 17 + 1 = 18 18 , 18 , 16	0
FCwB(256)	256	((18 * 18 * 16) + 1) * 256 = 1327360 1327360
FC	5	(256 + 0) * 5 = 1280 1280
Output	5	0
		132880

Set E

Q3. Calculate the number of parameters for the following architecture for the input shape of (428,428).

$$\text{input} \rightarrow \text{Conv}(3,2,8) \rightarrow \text{Minpool}(2) \rightarrow \text{ReLU} \rightarrow \text{Maxpool}(5) \rightarrow \text{FCwB}(256) \rightarrow \text{FC}(5) \rightarrow \text{output}$$

Here, Conv(x,y,z) denotes a convolutional layer that has the kernel size of x, stride size of y, and output channel of z.
Minpool(x) denotes a Minpooling layer that has both window size and stride size of x,
Maxpool(x) denotes a Maxpooling layer that has both window size and stride size of x,
FCwB(x) denotes a Fully Connected Layer of x neurons with Bias, and
FC(x) denotes a Fully Connected Layer of x neurons without Bias.

Layer	Output Shape	Parameter
input	428 , 428	0
Conv(3 , 2 ,8)	floor((428 - 3 + 2*0)/2) + 1 = 212 + 1 = 213 213 , 213 , 8	((3 * 3 *1) + 1) * 8 = 80 80
Minpool(2)	106 ,106 , 8	0
Relu	106 , 106 ,8	0
Maxpool(5)	21 , 21 , 8	0
FCwB(256)	256	((21 * 21 * 8) + 1)*256 903424
FC(5)	5	1280
Output	5	0
Total		904784

Quiz-1 - Integer43

Set-A

Q1. A. What is the role of an optimizer in a neural network?

Solution:45

The optimizer plays a crucial role in training a neural network. Its primary responsibility is to adjust the model's internal parameters (weights and biases) during the training process to minimize the loss function, which measures the difference between the model's predictions and the actual target values. The role of the optimizer can be summarized as follows:

1. ****Minimizing the Loss Function**:**
   - The primary goal of training a neural network is to find the model parameters (weights and biases) that minimize a specified loss function. This loss function quantifies how well the

model's predictions match the true target values.
- The optimizer is responsible for iteratively updating the model's parameters in a way that reduces the loss. It does this by computing the gradients of the loss with respect to the parameters and making parameter updates in the direction that reduces the loss.

2. **Gradient Descent:**
- Most optimizers, including popular ones like Stochastic Gradient Descent (SGD), Adam, RMSprop, and others, **use gradient information to update model parameters**.
- The optimizer calculates the gradient of the loss function with respect to each parameter, indicating how the loss changes as each parameter is adjusted. This gradient points in the direction of steepest increase in the loss.

3. **Learning Rate Control:**
- **The optimizer also manages the learning rate, which determines the step size for parameter updates**

5. **Convergence and Stability:**
- The choice of optimizer can significantly impact the convergence speed and stability of the training process.
- **A well-chosen optimizer helps the neural network converge to an optimal or near-optimal solution more quickly and reliably.**

6. **Regularization and Other Techniques:**
- **Some optimizers incorporate regularization techniques, such as L1 or L2 regularization, weight decay, or dropout, to improve the generalization of the model and prevent overfitting**.

B. How does learning rate impact the performance of a model? Explain briefly.

Solution:45
The learning rate is a crucial hyperparameter in training machine learning models, especially neural networks. It determines the size of steps that the model takes during the optimization process, such as gradient descent, when updating its parameters (like weights and biases). Here's how the learning rate impacts model performance:

1. **Too High Learning Rate**:
- If the learning rate is too high, the model might converge quickly, but it can overshoot the optimal solution and fail to converge to a good solution. This can lead to instability and divergence.

2. **Too Low Learning Rate**:
- Conversely, if the learning rate is too low, the model's training process will progress very slowly. It may get stuck in local minima or take an excessively long time to converge to the optimal solution.

3. **Appropriate Learning Rate**:
- The key is to find an appropriate learning rate. A well-chosen learning rate allows the

model to converge to a good solution efficiently. This often involves experimentation and adjusting the learning rate during training (e.g., learning rate schedules or adaptive methods like Adam) to strike the right balance between convergence speed and accuracy.

In summary, the learning rate directly impacts how quickly a model learns and the quality of the solution it converges to. It's essential to choose an appropriate learning rate to achieve the best performance in training your machine learning models.

Quiz-3 - Integer43

Set-A

Q1. Which of the following is not true as a characteristic of parameters of neural network?

A. Required by the model when making predictions
B. Learned from data
C. Often not set manually by the practitioner
D. Determine how the network is trained

Solution: By Hussain-060
D
Hyperparameters determine how the network is trained [Slide Hyperparameters Parameters, P. 5], not parameters.

Q2. CNN is mostly used when there is an -

A. structured data
B. unstructured data
C. both A and B
D. none of the above

Solution: Sujon 49
B. unstructured data
CNN is mostly used when there is an unstructured data set (e.g., images) and the practitioners need to extract information from it.

Q3. Calculate the number of parameters for the following architecture for the input shape of (428,428).

input→Conv(3,2)→Minpool(2)→ReLU→Conv(5,3)→Maxpool(5)→FCwB(256)→FC(5)→output

Here, Conv(x,y) denotes a convolutional layer that has the kernel size of x and stride size of y,
Minpool(x) denotes a Minpooling layer that has both window size and stride size of x,
Maxpool(x) denotes a Maxpooling layer that has both window size and stride size of x,
FCwB(x) denotes a Fully Connected Layer of x neurons with Bias, and
FC(x) denotes a Fully Connected Layer of x neurons without Bias.

Solution: Rabab 039

Layer	Output size	Parameter#
input	478, 464	0
Conv(3,2)	floor(W-F+2P)/S + 1 (478-3 + 2*0)/2 +1 = 238, (464-3 + 2*0)/2 +1 = 231	((filter_dim*filters_prev)+1)*filters_current ((3*3*1)+1)*1 = 10
Minpool(4)	(W-F+2P)/S + 1 (238-4 + 2*0)/4 +1 = 59, (231-4 + 2*0)/4 +1 = 57	0
ReLU	59, 57	0
Conv(5,3)	(W-F+2P)/S + 1 (59-5 + 2*0)/3 +1 = 19, (57-5 + 2*0)/3 +1 = 18	((filter_dim*d)+1)*k ((5*5*1)+1)*1 = 26
Maxpool(3)	(W-F+2P)/S + 1 (19-3 + 2*0)/3 +1 = 6, (18-3 + 2*0)/3 +1 = 6	0
FCwB(256)	256	(inputs*nodes)+nodes (6*6 * 256) + 256 = 9472
ReLU	256	0
Dropout	256	0
FC(128)	128	(inputs*nodes) (256*128) = 32768
tanh	128	0
FC(5)	5	128*5 = 640
output	5	0
	Total	42916

***Formulae*:**
Convolution: output size = (W-F+2P)/S + 1, parameters = ((m*n*d)+1)*k
Pooling: output size = (W-F+2P)/S + 1, parameters = 0
Activation functions: output size = same as inputs, parameters = 0
Dropout: output size = same as inputs, parameters = 0
FCwB: output size = no. of nodes, parameters = (inputs * no. of nodes) + no. of nodes
FC: output size = no. of nodes, parameters = (inputs * no. of nodes)

Set-B

Q1. Which of the following is not true as a characteristic of hyperparameters?

A. Set before training
B. Determines the network structure
C. Determine how the network is trained
D. Often saved as part of the learned model

Solution: ***Added by Younus-131***

D. Often saved as part of the learned model

This statement is not true as a characteristic of hyperparameters. Hyperparameters are not typically saved as part of the learned model. Instead, they are set before training and determine various aspects of the training process and network structure, but they are separate from the learned model's parameters, which are the weights and biases that are adjusted during training based on the data. Hyperparameters are external to the model and are typically specified by the machine learning engineer or researcher before the training process begins.

Q2. An input image has been converted into a matrix of size 12 X 12 along with a filter of size 3 X 3 with a Stride of 2 and padding of 0. Determine the size of the convoluted matrix.

A. 10*10
B. 8*8
C. 5*5
D. 3*3

Solution: ***Added by Younus-131***

To determine the size of the convoluted matrix after applying a 3x3 filter with a stride of 2 to a 12x12 input matrix with no padding, you can use the following formula:

Output size = ((Input size - Filter size) / Stride) + 1

In this case:

Input size = 12x12
Filter size = 3x3
Stride = 2
Padding = 0

So, plugging these values into the formula:

Output size = ((12 - 3) / 2) + 1
Output size = (9 / 2) + 1
Output size = 4.5 + 1
Output size = 5.5

Since the output size should be a whole number, you typically round down to the nearest integer. Therefore, the size of the convoluted matrix is 5x5.

So, the correct answer is C. 5x5.

Q3. Calculate the number of parameters for the following architecture for the input shape of (438,438).

input→Conv(3,2)→Minpool(4)→ReLU→Conv(5,3)→Maxpool(3)→FCwB(256)→FC(5)→output

Here, Conv(x,y) denotes a convolutional layer that has the kernel size of x and stride size of y,
Minpool(x) denotes a Minpooling layer that has both window size and stride size of x,
Maxpool(x) denotes a Maxpooling layer that has both window size and stride size of x,
FCwB(x) denotes a Fully Connected Layer of x neurons with Bias, and
FC(x) denotes a Fully Connected Layer of x neurons without Bias.

Solution: Added by Hussain-060
[Please correct if any mistake is found]

Layer	Input Shape	Output Shape	# Params
1. Conv(3,2)	(438, 438)	floor((438-3+2*0)/2) + 1 = 218 [Assuming padding = 0] (218, 218)	3*3 + 1 = 10 [assuming no. of in and out channel = 1]
2. MinPool(4)	(218, 218) ?- output shape of previous layer - yep	floor((218-4+2*0)/4) + 1 = 54 (54, 54)	0
3. ReLU	(54, 54)	(54, 54)	0
4. Conv(5, 3)	(54, 54)	floor((54-5+2*0)/3)+1 = 17 (17, 17)	5*5 + 1 = 26
5. MaxPool(3)	(17, 17)	floor((17-3+2*0)/3) + 1 = 5 (5, 5)	0
Flatten	**(5,5)**	**25**	**0**
6. FCwB(256)	25	256	25*256+256=6656

7. FC(5)	256	5	256*5=1280
Total Params			**7972**

Note: “Input shape” column could be discarded altogether.

CNN

Integer43

a) “In Convolutional Neural Network, Pooling always decreases the parameters.”- Is this statement true? Explain your answer with proper justification. [3]

In Convolutional Neural Networks (CNNs), pooling layers, such as max pooling or average pooling, indeed contribute to a reduction in the number of parameters. Here's a justification for this statement:

Justification:

Spatial Resolution Reduction:

Pooling operations, especially max pooling and average pooling, involve aggregating information from local neighborhoods in the input feature maps. By taking the maximum or average value within a pooling window, the spatial resolution of the feature maps is effectively reduced.
As the pooling window slides over the input, only the most significant or representative information is retained, leading to a downsampling of the spatial dimensions.

Parameter Sharing:

Pooling layers do not introduce additional parameters to be learned. Unlike convolutional layers that use learnable filters, pooling layers apply fixed operations (max or average) across local regions without introducing any new parameters.
The absence of learnable parameters in pooling layers contributes to a reduction in the overall parameter count of the network.

Translation Invariance:

Pooling layers provide a degree of translation invariance by selecting the most relevant features within local regions. This means that the exact spatial location of a feature becomes less crucial, further reducing the impact of spatial resolution on the overall network.

Computational Efficiency:
The reduction in spatial resolution achieved through pooling leads to computational efficiency during both training and inference. With fewer parameters to process, the network can operate more

efficiently while still preserving essential features.

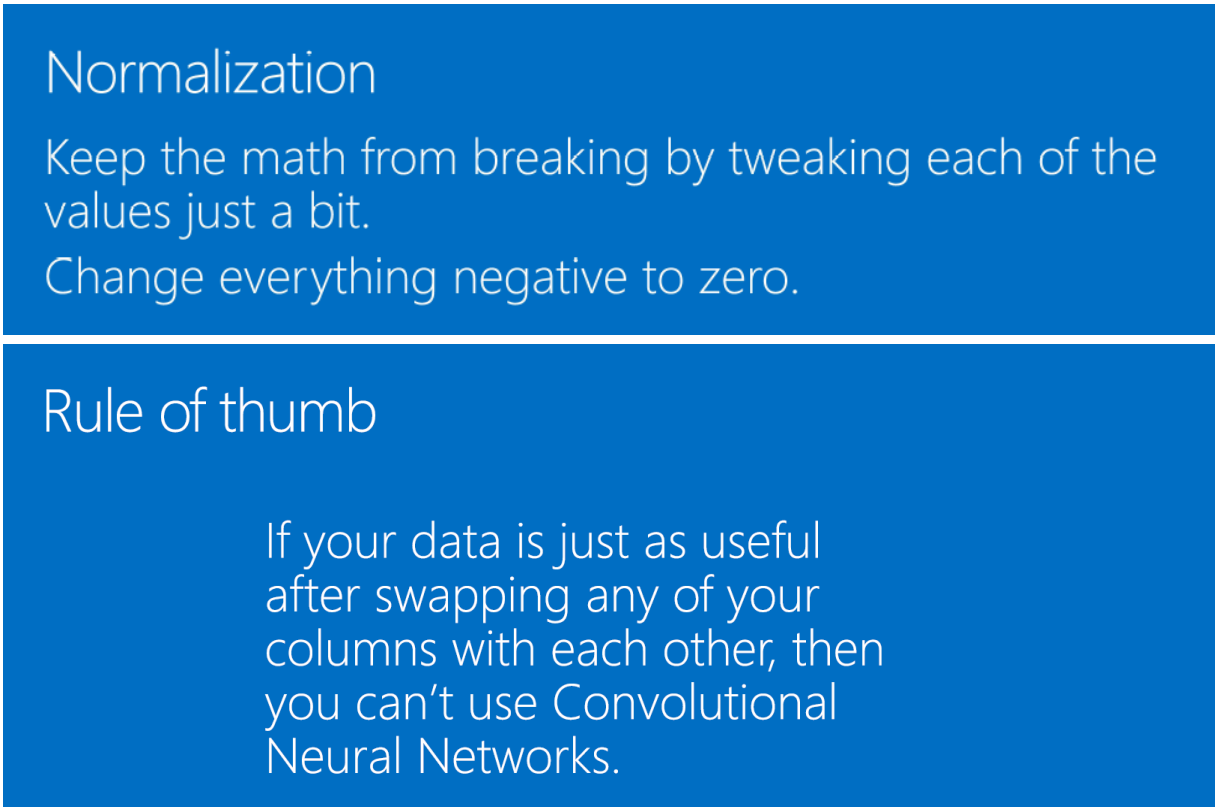
In Convolutional Neural Networks, pooling layers, such as max pooling or average pooling, consistently contribute to a reduction in the number of parameters. By aggregating information within local regions and downsampling spatial dimensions, pooling layers effectively decrease the spatial resolution of the feature maps without introducing additional learnable parameters. This spatial reduction not only enhances computational efficiency but also promotes translation invariance, making pooling a key component in the parameter-efficient design of CNN architectures

Origin42

Question 5. [Marks: 14]

a) How normalization can be achieved in Convolutional Neural Network? What is the rule of thumb to use a CNN architecture? CNN [6]

Solution:



Alternative: Sujon 49

Normalization in Convolutional Neural Networks (CNNs) is typically achieved using techniques like Batch Normalization (BatchNorm) and Layer Normalization. These techniques help stabilize and accelerate the training of deep neural networks by normalizing the inputs to each layer. Here's a brief explanation of Batch Normalization:

Batch Normalization (BatchNorm):

BatchNorm is applied to the activations of a layer within a mini-batch during training. It works as follows:

Compute Mean and Variance: For each mini-batch, calculate the mean and variance of the activations across the mini-batch for each channel.

Normalize: Normalize the activations by subtracting the mean and dividing by the standard deviation (with a small epsilon added for numerical stability).

Scale and Shift: After normalization, scale the normalized activations by a learnable parameter (gamma) and shift them by another learnable parameter (beta). These parameters allow the model to adapt and learn the best scaling and shifting for each layer.

Backpropagation: During backpropagation, gradients are computed for gamma and beta, allowing the network to learn how to adjust the normalization.

BatchNorm helps with the following:

Stability: It stabilizes training by reducing internal covariate shift, making it easier to train deep networks.
Regularization: It acts as a form of regularization by introducing noise in the training process.
Faster Training: It often accelerates training since it allows for more aggressive learning rates.

Enigma41

1

ii) Suppose you want to build an automated Image Bot that generates the caption for the image. You have created a complex deep model consisting of a Long Short-Time Memory (LSTM) followed by a Convolutional Neural Network (CNN). Your friend suggests a Recurrent Neural Network (RNN) instead of LSTM. RNN, CNN, LSTM Do you think your friend's suggestion would improve the performance? Why or why not? What are the possible corners of improvement for this model? [8]

Solution:

4. i) "In Convolutional Neural Network, Pooling always decreases the parameters". Is this statement true? Explain your answer with proper justification. [6] CNN

Solution: 024

The statement is false.

Generally pooling decreases parameters. But if the pool size is set to 1, it essentially means that no pooling is being applied. In other words, each individual value in the input feature map remains unchanged. Consequently, there is no reduction in the spatial dimensions, and the number of parameters also remains the same after this "pooling" operation (which isn't really pooling at all in this case).

Recurssive40

2. ii) Briefly explain the slowness of Recurrent Neural Network (RNN) compared to other deep networks, such as Convolutional Neural Network (CNN). [4] RNN-CNN

Solution:

	Convolutional neural network (CNN)	Recurrent neural network (RNN)
ARCHITECTURE	Feed-forward neural networks using filters and pooling	Recurring network that feeds the results back into the network
INPUT/OUTPUT	The size of the input and the resulting output are fixed (i.e., receives images of fixed size and outputs them to the appropriate category along with the confidence level of its prediction)	The size of the input and the resulting output may vary (i.e., receives different text and output translations—the resulting sentences can have more or fewer words)
IDEAL USAGE SCENARIO	Spatial data (such as images)	Temporal/sequential data (such as text or video)
USE CASES	Image recognition and classification, face detection, medical analysis, drug discovery and image analysis	Text translation, natural language processing, language translation, entity extraction, conversational intelligence, sentiment analysis, speech analysis

Parameter Calculation

Video By Paul 023: Parameter Calculation & TF IDF Math

Integer43

c) Analyze the following architecture and figure out the number of parameters for the input shape of (437, 442). [7]

input→ cnn _1→ dropout→ relu→ maxpool→ cnn _2→ relu→ maxpool
output ←linear_3 ←relu ←linear_2 ←relu ←linear_1 ←flatten

Here,
cnn _1: Conv2d(in_feature=1, out_feature=4, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
cnn _2: Conv2d(in_feature=4, out_feature=8, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
maxpool: MaxPool2d(kernel_size=2, stride=2, padding=0)
linear_1: Linear(in_features=95920, out_features=512, bias=True)
linear_2: Linear(in_features=512, out_features=128, bias=True)
linear_3: Linear(in_features=128, out_features=10, bias=True)
flatten: Flatten(start_dim=1, end_dim=-1)
dropout: Dropout(p=0.2, inplace=False)
relu: ReLU()

Input Layer	Output Shae	Parameter
input	437 , 442	0
cnn_1	(437- 5 + (2*2) / 1) + 1 (442- 5 + (2*2) / 1) + 1 437 , 442 , 4	((5*5*1) + 1) * 4 104
dropout	437 , 442 , 4	0
relu	437 , 442 , 4	0
maxpool	(437-2 + 0) /2 +1 (442-2 + 0) /2 +1 218 , 221 , 4	0
cnn_2	(218-5 + 2*2) / 1 + 1 (221-5 + 2*2) / 1 + 1 218 , 221 , 8	((5*5*4) + 1) * 8 808
relu	218 , 221 , 8	0
maxpool	109 , 110 , 8	0
flatten	109 * 110 * 8	0

	95920	
linear_1	512	(95920*512) + 512 49111552
relu	512	0
linear_2	128	(512+1) * 128 65664
relu	128	0
linear_3	10	(128 +1) * 10 1290
output	10	0
		49179418

Origin42

5

b) Calculate the number of parameters for the following architecture for the input shape of (478,464). [8]

Parameter Calculation

input→Conv(3,2)→Minpool(4)→ReLU→Conv(5,3)→Maxpool(3)

output←FC(5)←tanh←FC(128)←Dropout(0.2)←ReLU←FCwB(256)

Here, Conv(x,y) denotes a convolutional layer that has the kernel size of x and stride size of y,
Minpool(x) denotes a Minpooling layer that has both window size and stride size of x,
Maxpool(x) denotes a Maxpooling layer that has both window size and stride size of x,
FCwB(x) denotes a Fully Connected Layer of x neurons with Bias,
Dropout (y) denotes a Dropout Layer that drops input at a rate of y, and
FC(x) denotes a Fully Connected Layer of x neurons without Bias.

Solution:Toufique 116

	Output Shape	Parameters
input	478,464	0
Conv(3,2)	238,231	10
Min Pool(4)	59,57	0
ReLU	59,57	0

Conv(5,3)	19,18	26
Maxpool(3)	6,6	0
FCwB(256)	256	9472
ReLU	256	0
Dropout	256	0
FC(128)	128	32768
tanh	128	0
FC(5)	5	640
Output	5	0
	Total	42916

Enigma41

4

ii) Calculate the number of parameters for the following architecture for the input shape of 4237. [8]

input→FCwB(256)→Dropout(0.1)→FC(128)→tanh→Maxpool(3)→FC(3)→output

Parameter Calculation

Here, FCwB(x) denotes a Fully Connected Layer of x neurons with Bias,
FC(x) denotes a Fully Connected Layer of x neurons without Bias,
Dropout (y) denotes a Dropout Layer that drops input at a rate of y, and
Maxpool(x) denotes a flat Maxpooling layer that has both window size and stride size of x.

Solution: 024

Layer	Output Shape	No. of Parameters
input(4237)	4237	0
FCwB(256)	256	(input_shape x output_shape) + bias (4237x256)+256=1084928
Dropout(0.1)	256	0
FC(128)	128	(input_shape x output_shape) (256x128)=32768
tanh	128	0
Maxpool(3)	floor(((W-F+2P)/S)+1) floor(((128-3+2x0)/3)+1)=42	0

FC(3)	3	(input_shape x output_shape) (42x3)=126
output	3	0
	Total:	1117822

So, number of total parameters = 1117822

Recurssive40

6

ii) Consider the following architecture that takes a (256, 256, 3) dimensional image and outputs a (56,56,56) dimensional image. The architecture consists of a convolutional layer (CNN) followed by a pooling layer (POOL) and another convolutional layer (CNN). Now, find a possible set of hyperparameters for the architecture. [9]

Parameter Calculation

image (256, 256, 3)→ **CNN → POOL →CNN** → *image* (56, 56, 56)

Your answer must include the following hyperparameters:

CNN → *number of filters, filter size, stride, padding*

POOL → *stride, pool size, padding*

Solution: **By Hussain-060**
[Updates are welcome]

CNN1:
Input shape = (256, 256, 3)

No. of filters = 28 # 28 kivabe? - No particular reason, could've been 1 as well
Filter size = 3
Stride = 2
Padding = 0

floor((256-3+2*0)/2) + 1 = 127

Output shape:(127, 127, 28)

Pool:
Input shape = (127, 127, 28)

Filter size = 5
Stride = 2
Padding = 0

floor((127-5+2*0)/2) + 1 = 62
Output shape:(62, 62, 28)

CNN2:
Input shape = (62, 62, 28)

No. of filters = 56
Filter size = 7
Stride = 1
Padding = 0

floor((62-7+2*0)/1) + 1 = 56
Output shape:(56, 56, 56)

Summary: (No of filters, filter size, stride, padding)
CNN1 -> 28, 3, 2, 0
Pool -> 1, 5, 2, 0
CNN2 -> 56, 7, 1, 0

Note: Most of the values are random or educated guesses with the goal of reaching the final shape from the initial shape. Keu jodi guesswork chara onno kono technique janos then please update koris.

Genetic Algorithm, TSP

Integer43

b) Apply crossover and mutation on the following two parents to produce a single offspring. [4]
You must employ Davis Order Crossover as your crossover strategy.

1	3	5	7	8	4	6	2

3	4	1	5	2	7	6	8

Solution:By Paul 023

Integer-43 (3b)

Population:

R_1 =

1	3	5	7	8	4	6	2

R_2 =

3	4	1	5	2	7	6	8

Crossover:

R_1 =

1	3	5	7	8	4	6	2

R_2 =

3	4	1	5	2	7	6	8

$C1$ =

3	1	5	7	8	4	2	6

In this crossover, we select the genes of one parent and maintain the order of the genes in child. The rest are taken from another parent and excluding the genes of Parent 1, the rest are filled from parent 2.

Davis order crossover

Mutation:

Before:

$C1$ =

3	1	5	7	8	4	2	6

After:

$C1$ =

3	2	5	7	8	4	1	6

(Swap mutation)

is our required single offspring.

Recurssive40

7. i) What do you mean by mutation in Genetic Algorithms (GA)? Provide an example of mutation. [5]

Genetic Algo

Solution: 024

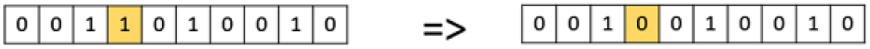
Mutation:

Mutation is a natural process that occurs due to an error in replication or copying of genes.

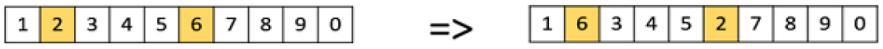
By mixing and matching the genes from both parents, it is possible to reproduce the parent chromosomes during crossover. There is no guarantee that the copying of the parent gene is 100% accurate. There always occurs an error, which leads to the scope of exploration. Mutating the chromosome in the genetic algorithm is necessary because it may result in revolutionary results that will help solve problems more efficiently.

Example:

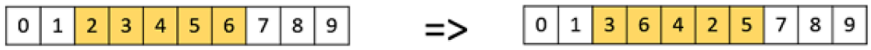
Bit Flip Mutation



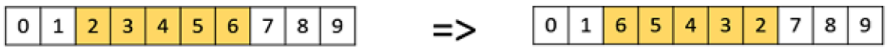
Swap Mutation



Scramble Mutation



Inversion Mutation



Here 4 examples are given. According to the question, discuss only one.

E : 1/1.5 set

Fuzzy (All the topics)

Recommended by sir:

1. Membership Function (Graph to Equation, Equation to Graph)
2. CrispSet VS FuzzySet
3. Fuzzy, NeuroFuzzy system importance
4. Alpha Cut / Extension Principle

Video By Paul 023: Fuzzy Graph Math

Fuzzy By Paul & Ira

Fuzzy

Fuzzy

Defⁿ: A fuzzy set is a set which allow partial belonging to a set, that is defined by a degree of membership denoted by 'μ', that can take any value from 0 to 1.

Crisp set: If we remove all the values of belonging except 0 & 1 the set will collapes into a crisp set.

Membership function: The membership function of the set is the relationship between the elements of the set & their degree of belonging.

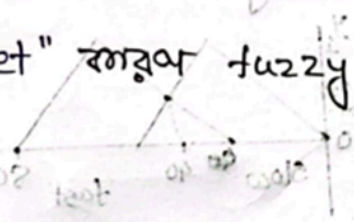
(2,0) . (2,0.1)

Example of Crisp set:

$F = \{i | i$ is an integer and $4 \leq i \leq 12$

If $i = 4$ to 12 then it will be 1 otherwise 0

"Fuzzy is one kind of crisp set" মানে fuzzy set-এ 0, 1 র combination ও থাকতে পারে।



Advantages and Disadvantages of Fuzzy Logic System:

Advantages:

Tolerance for Imprecise Data: Fuzzy logic is good at handling imprecise, noisy, or incomplete data, making it valuable in real-world applications where such data are common.

Modeling Human Reasoning: It mimics human decision-making logic, which is not always black or white but often involves shades of gray. This makes fuzzy logic systems more intuitive and effective for certain types of decision-making processes.

Flexibility: Fuzzy logic can be combined with other methodologies, such as neural networks (forming neuro fuzzy systems), to enhance their capabilities, particularly in learning and adapting to new data.

Simplicity: In many cases, fuzzy logic can simplify complex problems, reducing the need for precise mathematical formulations. This can make the development of control and decision-making systems more straightforward.

Disadvantages:

Computational Complexity: The process of converting input data into fuzzy values, processing them through fuzzy rules, and then defuzzifying the results can be computationally intensive, especially for systems with a large number of rules or variables.

Difficulty in Designing Fuzzy Rules: The effectiveness of a fuzzy logic system heavily depends on the quality and comprehensiveness of the fuzzy rules set. Designing these rules requires expertise and a deep understanding of the application domain, which can be challenging.

Lack of Standardization: There are no universally accepted standards for the design and implementation of fuzzy logic systems, which can lead to inconsistency and difficulty in comparing different systems or transferring knowledge between them.

Subjectivity in Membership Functions: The design of membership functions (which define how input values are mapped to fuzzy values) is somewhat subjective and relies on the designer's intuition and experience. This subjectivity can lead to variability in system performance.

Fuzzy Logic: It's like a smart system that can handle uncertainty and make decisions based on "maybe" instead of just "yes" or "no." It works well when things aren't black or white.

Neurofuzzy System: Imagine combining that smart system (fuzzy logic) with a learning brain (neural network) that gets smarter over time. It not only handles uncertainty but also learns from new information to make even better decisions.

So, while fuzzy logic deals with uncertainty, a neurofuzzy system learns from it and adapts, making it a smarter, self-improving system.

Importance of Neuro Fuzzy System:

Make Better Decisions with Unclear Data: They are really good at making decisions when the information isn't clear or complete, much like how humans do, by understanding shades of meanings rather than just yes or no.

Learn and Adapt: These systems learn from experiences and can adjust to new situations on their own. This means they get better over time at whatever they're doing, from recognizing patterns to predicting outcomes.

Solve Complex Problems: They can handle very complicated issues, including those that change in unpredictable ways, by combining the strengths of fuzzy logic and neural networks.

Better Control for Machines and Systems: They offer more precise control in situations that are uncertain or keep changing, making them ideal for things like cars, planes, and factories.

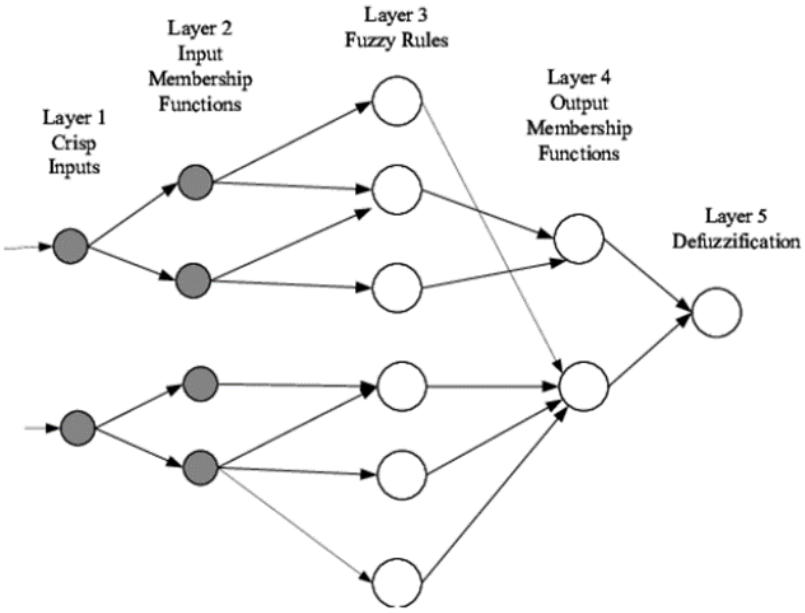
Fit to Your Needs and Grow: They can be tailored to specific tasks and are capable of dealing with larger and more complex situations as needed, making them versatile for a wide range of uses.

How can a neuro fuzzy system be built?

Neuro Fuzzy System:

- A neuro-fuzzy system is based on a fuzzy system which is trained by a learning algorithm derived from neural network theory.
- A neuro-fuzzy system can be built as a 3-layer feedforward neural network.
    - First layer represents input variables
    - Middle (hidden) layer represents fuzzy rules
    - Third layer represents output variables

  Fuzzy sets are encoded as (fuzzy) connection weights. It represents the data flow of input processing and learning within the model. Sometimes a 5-layer architecture is used, where the fuzzy sets are represented in the units of the second and fourth layer.
- A neuro-fuzzy system can be always interpreted as a system of fuzzy rules. It is also possible to create the system out of training data from scratch, as it is possible to initialize it by prior knowledge in the form of fuzzy rules.



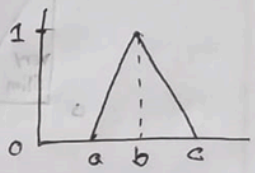
Graph math:

Fuzzy

Remember:

① For Positive slope (/); $\frac{x-a}{b-a}$ for $a \le x \le b$

② For negative slope (\); $\frac{c-x}{c-b}$ for $b \le x < c$



Type-1: (Mathematical expression to Graph)

$$\mu_{cold}(x) = \frac{-x}{30} + 1 \quad ; \text{if } 0 \le x < 30$$

$$\mu_{Mid\ Hot}(x) = \begin{cases} \frac{x-30}{20} & ; \text{if } 30 \le x < 50 \\ \frac{x-60}{-10} & ; \text{if } 50 \le x < 60 \end{cases}$$

$$\mu_{Hot}(x) = \frac{x-60}{40} \quad ; \text{if } 60 \le x < 100$$

$$\mu_A(x) = \begin{cases} 0 & \text{if } x \le a \\ \frac{x-a}{b-a} & \text{if } a < x \le b \\ \frac{c-x}{c-b} & \text{if } b \le x < c \\ 0 & \text{if } x \ge c \end{cases}$$

Ans:

① $\frac{-x}{30} + 1$

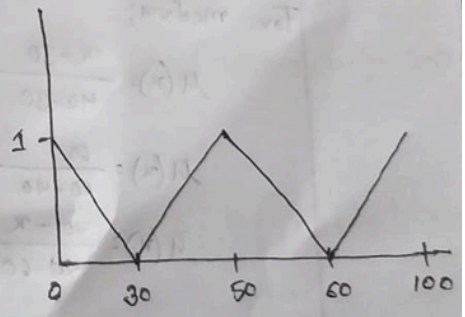
$= \frac{-x+30}{30}$

$= \frac{30-x}{30-0}$ (negative slope format)

② $\frac{x-30}{20} = \frac{x-30}{50-30}$ (positive)

③ $\frac{x-60}{-10} = \frac{x-60}{50-60} = \frac{60-x}{60-50}$ (negative)

④ $\frac{x-60}{40} = \frac{x-60}{100-60}$ (positive)



Extension Principle math:

Fuzzy Addition using Extension Principle

From slide:

$A = 3 = 0.3/1 + 0.7/2 + 1.0/3 + 0.7/4 + 0.3/5 + 0/6$

$B = 7 = 0.2/5 + 0.6/6 + 1.0/7 + 0.6/8 + 0.2/9 + 0/10$

A \ B	y=1	y=2	y=3	y=4	y=5	y=6	y=7	y=8	y=9	y=10
x=1	0.0 0.3	0.0 0.3	0.0 0.3	0 0 0.3	0.2 (0.2) 0.3	0.6 (0.3) 0.3	1.0 0.3 0.3	0.6 0.3 0.3	0.2 0.2 0.3	0.0 0 0.3
x=2	0.0 0.7	0.0 0.7	0.0 0 0.7	0 0 0.7	0.2 0.2 0.7	0.6 0.6 0.7	1.0 (0.7) 0.7	0.6 0.6 0.7	0.2 0.2 0.7	0.0 0 0.7
x=3	0.0 1.0	0.0 0 1.0	0 0 1.0	0 0 1.0	0.2 0.2 1.0	0.6 (0.6) 1.0	1.0 (1) 1.0	0.6 0.6 1.0	0.2 0.2 1.0	0.0 0 1.0
x=4	0.0 0 0.7	0.0 0 0.7	0 0 0.7	0 0 0.7	0.2 0.2 0.7	0.6 0.6 0.7	1.0 (0.7) 0.7	0.6 (0.6) 0.7	0.2 0.2 0.7	0.0 0 0.7
x=5	0.0 0 0.3	0.0 0 0.3	0 0 0.3	0 0 0.3	0.2 0.2 0.3	0.6 0.3 0.3	1.0 0.3 0.3	0.6 (0.3) 0.3	0.2 (0.2) 0.3	0.0 0 0.3
x=6	0.0 0 0	0.0 0 0	0 0 0	0 0 0	0.2 0 0	0.6 0 0	1.0 0 0	0.6 0 0	0.2 0 0	0.0 0
x=7	0.0 0 0	0.0 0 0	0 0 0	0 0 0	0.2 0 0	0.6 0 0	1.0 0 0	0.6 0 0	0.2 0	0.0 0
x=8	0.0 0 0	0.0 0 0	0 0 0	0 0 0	0.2 0 0	0.6 0 0	1.0 0 0	0.6 0	0.2 0	0.0 0
x=9	0.0 0 0	0.0 0 0	0 0 0	0 0 0	0.2 0 0	0.6 0 0	1.0 0	0.6 0	0.2 0	0.0 0
x=10	0.0 0 0	0.0 0 0	0 0 0	0 0 0	0.2 0 0	0.6 0	1.0 0	0.6 0	0.2 0	0.0 0

max 0 ← বৃ আসাবো না

↑ 12 (10+2) ↑ 13 (10+3) ↑ 14 (10+4) ↑ 15

↳ এগুলো যাবে ০০২ max'0 এসেছে

diagonally check করব(x,y) র value বদ আসবে।

Suppose for addition of 5

$(x=4, y=1)$, $(x=3, y=2)$ $(x=2, y=3)$ $(x=1, y=4)$

it creates a diagonal from $(x=4$ to $y=4)$

** Similarly for 6 → $(x=5$ to $y=5)$; 7 → $(x=6$ to $y=6)$

Output of C

= 0/5 + 0.2/6 + 0.3/7 + 0.6/8 + 0.7/9 + 1.0/10 + 0.7/11

+ 0.6/12 + 0.3/13 + 0.2/14 + 0/15

For addition of 12 → $(x=10, y=2)$ - এই box থেকে diagonally আগাবো। same for 13 → $(x=10, y=3)$ ....

& rest of all.

Alpha cut:

Return - 38

Alpha - Cut Principle.

$A = 0.33/6 + 0.67/7 + 1.00/8 + 0.67/9 + 0.33/10$

$B = 0.33/1 + 0.67/2 + 1.00/3 + 0.67/4 + 0.33/5$

For A = 8;

1.0 → $\alpha_1 : 8$
0.9
0.8 $\alpha_2 : 8$
0.7

0.6 → $\alpha_1 : 7$
0.5 $\alpha_2 : 9$
0.4

0.3 → $\alpha_1 : 6$
0.2 $\alpha_2 : 10$
0.1

	0	0	0	0	0	0.33	0.67	1.00	0.67	0.33
α = 1.0								1		
α = 0.9								1		
α = 0.8								1		
α = 0.7								1		
α = 0.6							1	1	1	
α = 0.5							1	1	1	
α = 0.4							1	1	1	
α = 0.3						1	1	1	1	1
α = 0.2						1	1	1	1	1
α = 0.1						1	1	1	1	1
α = 0.0	1	1	1	1	1	1	1	1	1	1
	1	2	3	4	5	6	7	8	9	10

Here, no. of col = 10 because in A we have values upto 10. So, start from 1 and end in 10.

For B = 3;

1.00 → $b_1 : 3$
0.9 $b_2 : 3$
0.8
0.7

0.6 → $b_1 : 2$
0.5 $b_2 : 4$
0.4

0.3 → $b_1 : 1$
0.2 $b_2 : 5$
0.1

	0.33	0.67	1.00	0.67	0.33
α = 1.00			1		
α = 0.9			1		
α = 0.8			1		
α = 0.7			1		
α = 0.6		1	1	1	
α = 0.5		1	1	1	
α = 0.4		1	1	1	
α = 0.3	1	1	1	1	1
α = 0.2	1	1	1	1	1
α = 0.1	1	1	1	1	1
α = 0.0	1	1	1	1	1
	1	2	3	4	5

Here no. of col = 5 because in B we have value upto 5. So, start from 1 and end in 5

α-cut:

For 0.1; $C_{0.1} = (6,10) + (1,5) = (7,15)$

0.2; $C_{0.2} = (6,10) + (1,5) = (7,15)$

0.3; $C_{0.3} = (6,10) + (1,5) = (7,15)$

0.4; $C_{0.4} = (7,9) + (2,4) = (9,13)$

0.5; $C_{0.5} = (7,9) + (2,4) = (9,13)$

0.6; $C_{0.6} = (7,9) + (2,4) = (9,13)$

0.7, 0.8, 0.9, 1.0; $C_{0.7}, C_{0.8}, C_{0.9}, C_{1.0} = (8,8) + (3,3) = (11,11)$

							0.3	0.3	0.6	0.6	1.0	0.6	0.6	0.3	0.3
α=1.0											1				
α=0.9											1				
α=0.8											1				
α=0.7											1				
α=0.6									1	1	1	1	1		
α=0.5									1	1	1	1	1		
α=0.4									1	1	1	1	1		
α=0.3							1	1	1	1	1	1	1	1	1
α=0.2							1	1	1	1	1	1	1	1	1
α=0.1							1	1	1	1	1	1	1	1	1
α=0.0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$C = 0/6 + 0.3/7 + 0.3/8 + 0.6/9 + 0.6/10 + 1/11 + 0.6/12 + 0.6/13 + 0.3/14 + 0.3/15$

Integer43

Solved By Paul 023 & Ira 038

Question 7. [Marks: 14]

a) List the differences between Crisp Set and Fuzzy Set. [3]

Defination:

Crisp set: A crisp set is a collection of discrete value.

Fuzzy set: 'Already given'

Membership:

Crisp set: Elements either belong to the set (membership value 1) or do not belong ~~check~~ m. value is 0.

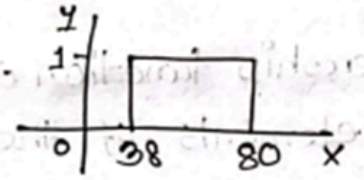


Fig: Crisp set

So if x is less than 38 or $80 < x$ then it will be zero.
x will be 1 if it remains between 38 & 80.

More precisely. A set of red apples{apples, not apples}

Fuzzy: Elements have degree of membership between 0 and 1, indicating the extent to which they belong to the set.

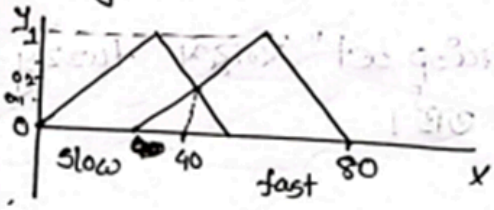


Fig: fuzzy set

So, here starting from zero it gradually speed up not directly goes to 1. For 40, there is an equal possibility of being slow or fast. It's a continuos value.

Example: The set of ripe apple, where the degree of membership varies for each apple based on its ripeness.

Boundary:

Crisp set has a clear, well defined boundary but fuzzy set lacks a precise boundary and elements can have partial membership.

S.No	Crisp Set	Fuzzy Set
1	Crisp set defines the value is either 0 or 1.	Fuzzy set defines the value between 0 and 1 including both 0 and 1.
2	It is also called a classical set.	It specifies the degree to which something is true.
3	It shows full membership	It shows partial membership.
4	Eg1. She is 18 years old. Eg2. Rahul is 1.6m tall	Eg1. She is about 18 years old. Eg2. Rahul is about 1.6m tall.
5	Crisp set application used for digital design.	Fuzzy set used in the fuzzy controller.
6	It is bi-valued function logic.	It is infinite valued function logic
7	Full membership means totally true/false, yes/no, 0/1.	Partial membership means true to false, yes to no, 0 to 1.

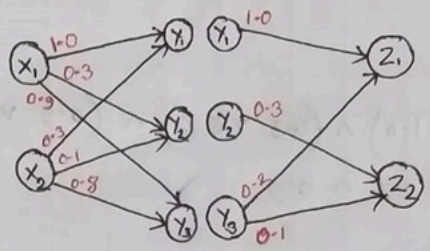
b) Illustrate max-average composition of fuzzy relation with a suitable example. [4]

Remember: $\wedge$ = min

$\vee$ = max

Max - Average:

The composition of fuzzy relations is an operation that combines two fuzzy relations to obtain a new fuzzy relation.



X × Y Y × Z

Here, two fuzzy relations can be observed seperately, one is in between x and y, other one y and z. Now we will apply Max-average composition to obtain a fuzzy relation between x and z.

R_1:

	Y_1	Y_2	Y_3
x_1	1·0	0·3	0·9
x_2	0·3	0·1	0·8

R_2:

	Z_1	Z_2
Y_1	1·0	0·0
Y_2	0·0	0·3
Y_3	0·2	0·1

Final matrix:

	Z_1	Z_2
x_1	1·0	0·5
x_2	0·5	0·45

$\mu_{R_1}(x_1,Y_1) + \mu_{R_2}(Y_1,Z_1) = 1·0 + 1·0 = 2$

$\mu_{R_1}(x_1,Y_2) + \mu_{R_2}(Y_2,Z_1) = 0·3 + 0·0 = 0·3$

$\mu_{R_1}(x_1,Y_3) + \mu_{R_2}(Y_3,Z_1) = 0·9 + 0·2 = 1·1$

$\mu_{R_1 \leftrightarrow R_2}(x_1,Z_1) = \frac{1}{2}(2 \vee 0·3 \vee 1·1) = 1·0$

$\mu_{R_1}(x_1,Y_1) + \mu_{R_2}(Y_1,Z_2) = 1·0 + 0·0 = 1·0$

$\mu_{R_1}(x_1,Y_2) + \mu_{R_2}(Y_2,Z_2) = 0·3 + 0·3 = 0·6$

$\mu_{R_1}(x_1,Y_3) + \mu_{R_2}(Y_3,Z_3) = 0·9 + 0·1 = 1·0$

$\mu_{R_1 \leftrightarrow R_2}(x_1,Z_2) = \frac{1}{2}(1 \vee 0·6 \vee 1) = 0·5$

$\mu_{R_1}(x_2,Y_1) + \mu_{R_2}(Y_1,Z_1) = 0·3 + 0·1 = 0·4$

$\mu_{R_1}(x_2,Y_2) + \mu_{R_2}(Y_2,Z_1) = 0·1 + 0·0 = 0·1$

$\mu_{R_1}(x_2,Y_3) + \mu_{R_2}(Y_3,Z_1) = 0·8 + 0·2 = 1·0$

$\mu_{R_1 \leftrightarrow R_2}(x_2,Z_1) = \frac{1}{2}(0·4 \vee 0·1 \vee 1·0) = 0·5$

$\mu_{R_1}(x_2,Y_1) + \mu_{R_2}(Y_1,Z_2) = 0·3 + 0·0 = 0·3$

$\mu_{R_1}(x_2,Y_2) + \mu_{R_2}(Y_2,Z_2) = 0·1 + 0·3 = 0·4$

$\mu_{R_1}(x_2,Y_3) + \mu_{R_3}(Y_3,Z_2) = 0·8 + 0·1 = 0·9$

$\mu_{R_1 \leftrightarrow R_2}(x_2,Z_2) = \frac{1}{2}(0·3 \vee 0·4 \vee 0·9) = 0·45$

* If the question's structure changes do the following processes for the respective questions.

For max-min:

$$\mu_{R_1 \circ R_2}(x_1, z_1) = (1.0 \wedge 1.0) \vee (0.3 \wedge 0.0) \vee (0.9 \wedge 0.2)$$
$$= 1.0 \vee 0.0 \vee 0.2$$
$$= 1.0$$

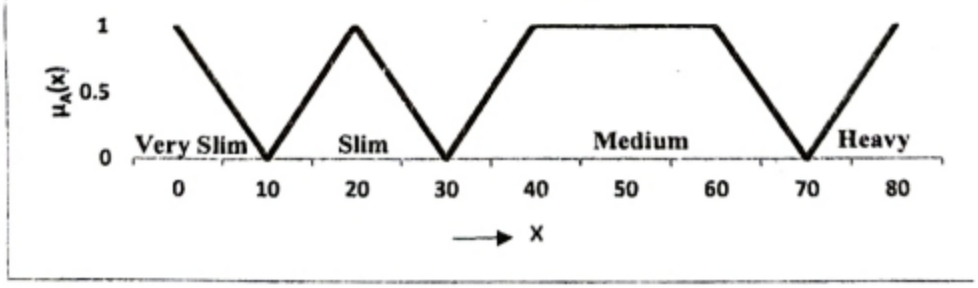
For min-max:

$$\mu_{R_1 \circ R_2}(x_1, z_1) = (1.0 \vee 1.0) \wedge (0.3 \vee 0.0) \wedge (0.9 \vee 0.2)$$
$$= 1.0 \wedge 0.3 \wedge 0.9$$
$$= 0.3$$

Max-product:

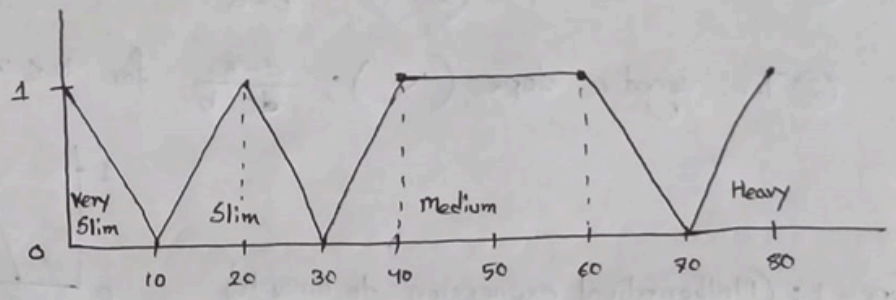
$$\mu_{R_1 \circ R_2}(x_1, z_1) = (1.0 \times 1.0) \vee (0.3 \times 0.0) \vee (0.9 \times 0.2)$$
$$= 1.0 \vee 0.0 \vee 0.18$$
$$= 1.0$$

c) Assume you want to calculate the health of a person based on his/her weight. A graphical representation of the membership function for this task is given in the following figure. Analyze the graph and write down the mathematical expression of the membership function. [7]



Integer-43 | 7(c)

Type-2 (Graph to Mathematical expression)



For very Slim;

$$\mu(x) = \frac{10-x}{10-0} \quad ; \quad 0 \le x < 10 \quad \text{(negative)}$$

For Slim;

$$\mu(x) = \frac{x-10}{20-10} \quad ; \quad 10 \le x < 20 \quad \text{(positive)}$$

$$\mu(x) = \frac{30-x}{30-20} \quad ; \quad 20 \le x < 30 \quad \text{(negative)}$$

For medium;

$$\mu(x) = \frac{x-30}{40-30} \quad ; \quad 30 \le x < 40 \quad \text{(positive)}$$

$$\mu(x) = 1 \quad ; \quad 40 \le x < 60 \quad \text{(zero)}$$

$$\mu(x) = \frac{70-x}{70-60} \quad ; \quad 60 \le x < 70 \quad \text{(negative)}$$

For Heavy;

$$\mu(x) = \frac{x-70}{80-70} \quad ; \quad 70 \le x < 80 \quad \text{(positive)}$$

Origin42

Question 6. [Marks: 14] Fuzzy

a) List the advantages of a fuzzy system. How a Neuro-Fuzzy system can be built? [6]

Solution: **O24**

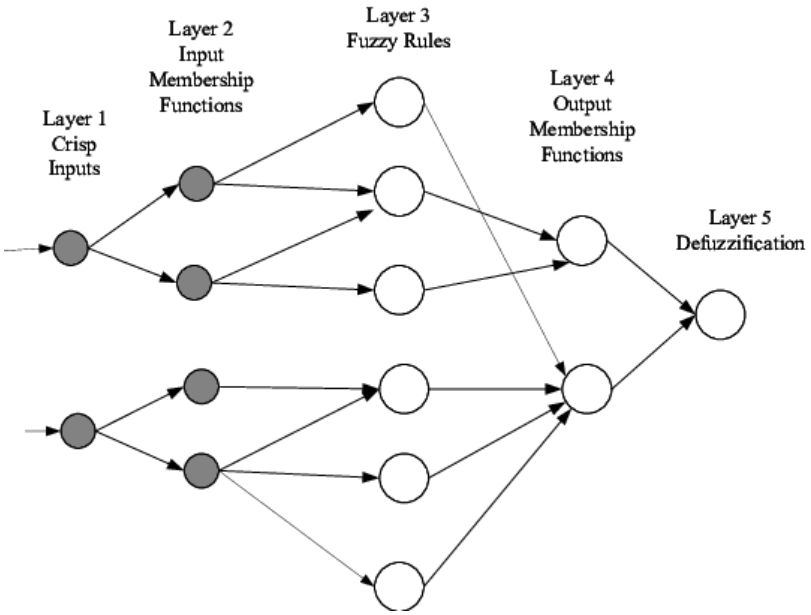
Advantages of Fuzzy Logic System:

- This system can work with any type of input whether it is imprecise, distorted or noisy input information.
- The construction of Fuzzy Logic Systems is easy and understandable.
- Fuzzy logic comes with mathematical concepts of set theory and the reasoning of that is quite simple.
- It provides a very efficient solution to complex problems in all fields of life as it resembles human reasoning and decision-making.
- The algorithms can be described with little data, so little memory is required.

Neuro Fuzzy System:

- A neuro-fuzzy system is based on a fuzzy system which is trained by a learning algorithm derived from neural network theory.
- A neuro-fuzzy system can be built as a 3-layer feedforward neural network.
    - First layer represents input variables
    - Middle (hidden) layer represents fuzzy rules
    - Third layer represents output variables

  Fuzzy sets are encoded as (fuzzy) connection weights. It represents the data flow of input processing and learning within the model. Sometimes a 5-layer architecture is used, where the fuzzy sets are represented in the units of the second and fourth layer.
- A neuro-fuzzy system can be always interpreted as a system of fuzzy rules. It is also possible to create the system out of training data from scratch, as it is possible to initialize it by prior knowledge in the form of fuzzy rules.



Fuzzy

Addition

#Fuzzy Addition with extension principle :

Origin 42 : (6) (b) :

$A = 0.2/1 + 0.5/2 + 0.7/3 + 1/4 + 0.7/5 + 0.5/6 + 0.2/7$

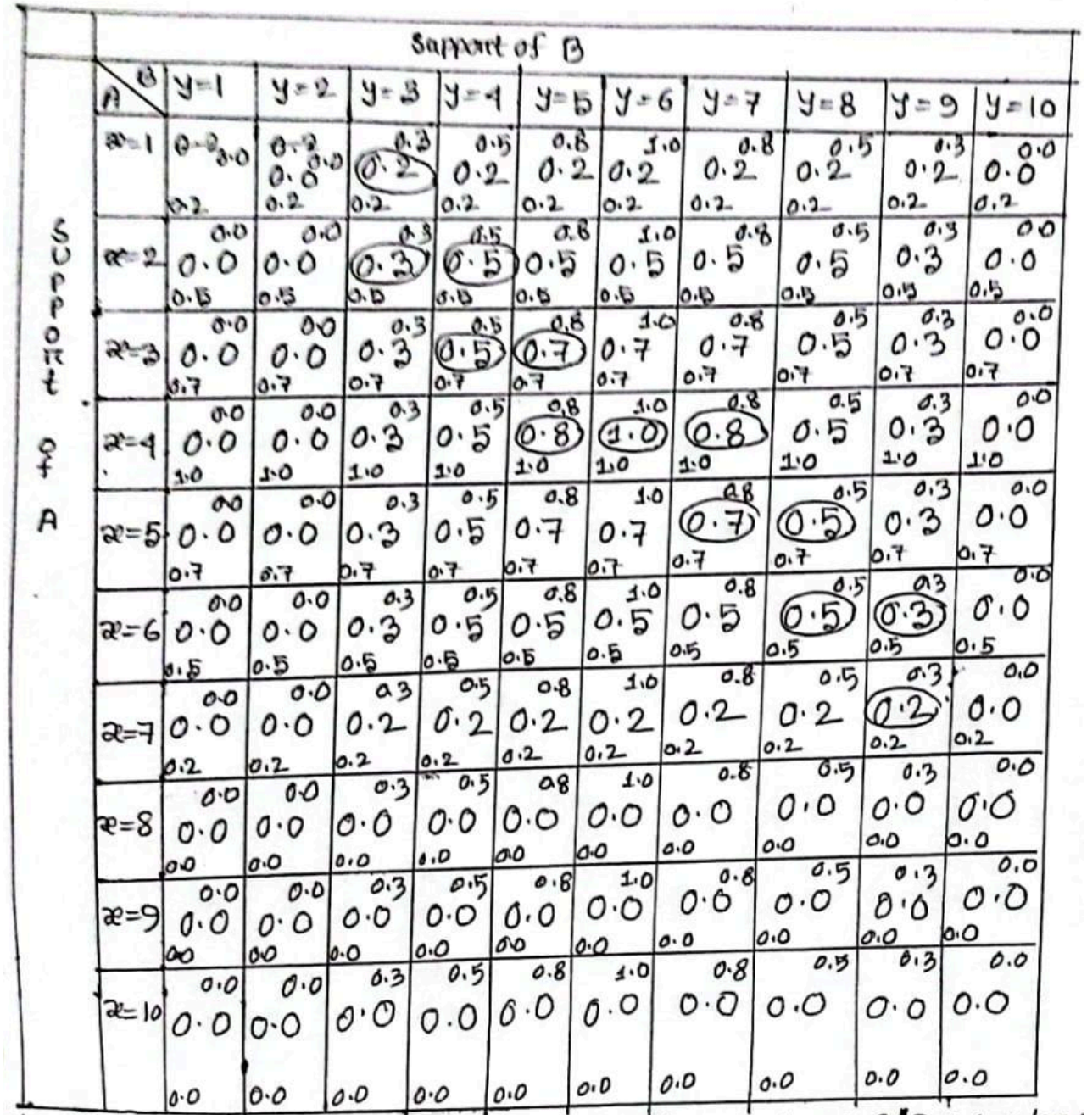
$B = 0.3/3 + 0.5/4 + 0.8/5 + 1/6 + 0.8/7 + 0.5/8 + 0.3/9$

b) Two fuzzy numbers A and B are given below. Add the numbers using the extension principle. [8]

A = 0.2/1 + 0.5/2 + 0.7/3 + 1/4 + 0.7/5 + 0.5/6 +0.2/7

B = 0.3/3 + 0.5/4 + 0.8/5 + 1/6 + 0.8/7 + 0.5/8 +0.3/9

Solution: **035 - Tuli**



∴ Final output, $C = 0.2/4 + 0.3/5 + 0.5/6 + 0.5/7 + 0.7/8 + 0.8/9 + 1.0/10 + 0.8/11 + 0.7/12 + 0.5/13 + 0.5/14 + 0.3/15 + 0.2/16$

Enigma41

7. i) What do you mean by Fuzzy Set? List the differences between Crisp Set and Fuzzy Set. [6]
Fuzzy

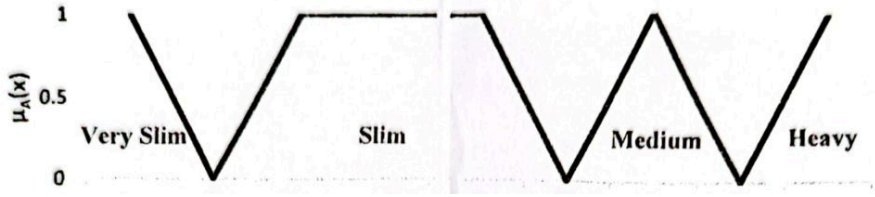
Solution: ***By - Rubayat***

A fuzzy set is a set where each element has a degree of membership between 0 and 1. It is an extension of the classical notion of set. For example, if you have a set of fruits, you can assign a membership value to each fruit based on how sweet it is. A fuzzy set allows you to capture uncertainty and vagueness in your data.

The main difference between crisp and fuzzy sets is that crisp sets have elements with binary membership (either 0 or 1), while fuzzy sets have elements with degrees of membership between 0 and 1. Crisp sets use bi-valued logic, while fuzzy sets use infinite-valued logic. Crisp sets have precise boundaries, while fuzzy sets have indeterminate boundaries

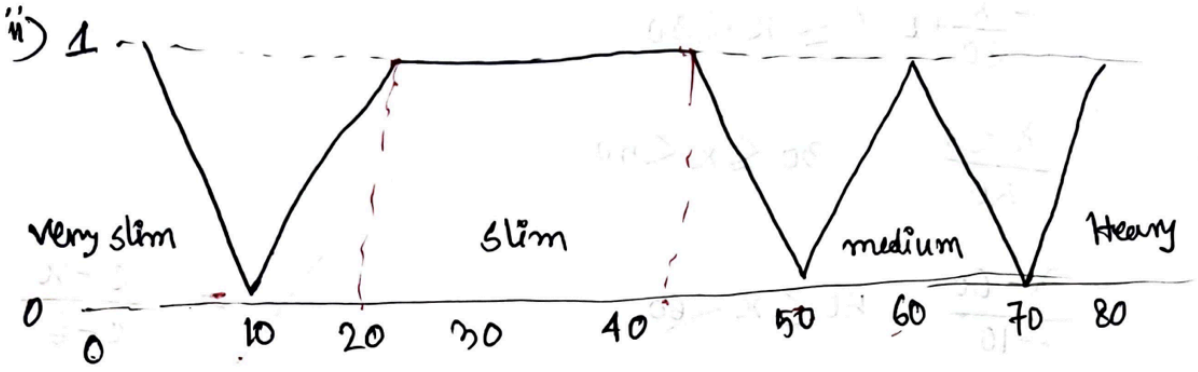
7. i) What do you mean by Fuzzy Set? List the differences between Crisp Set and Fuzzy Set. [6]
Fuzzy

ii) Assume, we want to calculate the hei lth of a person based on his/her weight. A graphical representation of the memb ership function for this task is given in the following figure. Write down the n athematical expression of the membership function. Describe each step briefly. [8]



Solution: ***By - Rubayat***
#Ehane maybe vul aache –swarna(061)

ii)



$$\mu_{\text{very slim}} = \frac{10 - x}{10 - 0} \qquad 10 < x < 20$$

$$\mu_{\text{slim}} = \frac{x - 10}{20 - 10} \quad (x = 10 \to 20)$$

$$\mu_{\text{slim}} = 1 \qquad 20 \le x \le 40.$$

$$\mu_{\text{slim}} = \frac{50 - x}{50 - 40} \qquad 40 \le x < 50$$

→ slim equation

$$\mu_{\text{medium}} = \frac{x - 50}{60 - 50}$$

$$= \frac{c - x}{c - b}$$

$$\mu_{\text{medium}} = \frac{70 - x}{70 - 60}$$

$$= \frac{x - a}{b - a}$$

$$\mu_{\text{Heavy}} = \frac{x - 70}{80 - 70}$$

Return 38

b)	The fuzzy numbers **A** and **B** are given by **A = 0.33/6 + 0.67/7 + 1.00/8 + 0.67/9 + 0.33/10** **B = 0.33/1 + 0.67/2 + 1.00/3 + 0.67/4 + 0.33/5** Draw a sketch of fuzzy number C, where C results from adding A and B by applying the alpha-cut principle.	**[3]**

Solution: Rafi-148

$A = 0.33/6 + 0.67/7 + 1.00/8 + 0.67/9 + 0.33/10$

α-cut/A	0	0	0	0	0	0.33	0.67	1.00	0.67	0.33
$\alpha = 1.0$				1				1		
$\alpha = 0.9$				1				1		
$\alpha = 0.8$				1				1		
$\alpha = 0.7$				1				1		
$\alpha = 0.6$				1			1	1	1	
$\alpha = 0.5$				1			1	1	1	
$\alpha = 0.4$				1			1	1	1	
$\alpha = 0.3$				1		1	1	1	1	1
$\alpha = 0.2$				1		1	1	1	1	1
$\alpha = 0.1$				1		1	1	1	1	1
	1	2	3	4	5	6	7	8	9	10

for, 1.0
$a_1 = 8$
$a_2 = 8$

for 0.9,
$a_1 = 8$
$a_2 = 8$

for 0.8,
$a_1 = 8$
$a_2 = 8$

for 0.7,
$a_1 = 8$
$a_2 = 8$

for 0.6, 0.5, 0.4,
$a_1 = 7$
$a_2 = 9$

for, 0.3, 0.2, 0.1,
$a_1 = 6$
$a_2 = 10$

$B = 0.33/1 + 0.67/2 + 1.00/3 + 0.67/4 + 0.33/5$

	0.33	0.67	1.00	0.67	0.33	0
$\alpha = 1.0$			1			
$\alpha = 0.9$			1			
$\alpha = 0.8$			1			
$\alpha = 0.7$			1			
$\alpha = 0.6$		1	1	1		
$\alpha = 0.5$		1	1	1		
$\alpha = 0.4$		1	1	1		
$\alpha = 0.3$	1	1	1	1	1	
$\alpha = 0.2$	1	1	1	1	1	
$\alpha = 0.1$	1	1	1	1	1	
	1	2	3	4	5	6

for, 1.0, 0.9, 0.8, 0.7:

$b_1 = 3$

$b_2 = 3$

for, 0.6, 0.5, 0.4:

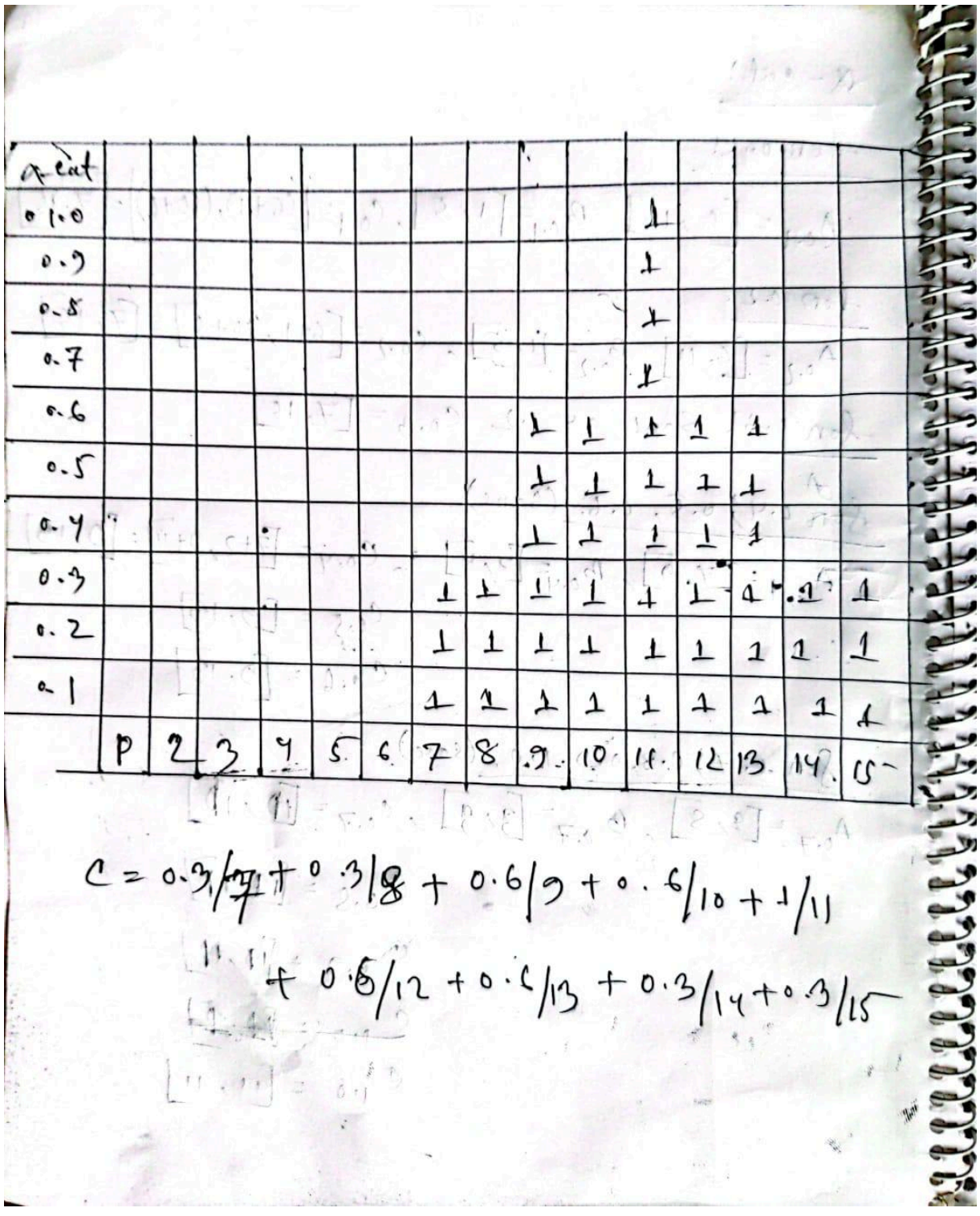
$b_1 = 2$

$b_2 = 4$

for 0.3, 0.2, 0.1

$b_1 = 1$

$b_2 = 5$



α cut															
1.0											1				
0.9											1				
0.8											1				
0.7											1				
0.6									1	1	1	1	1		
0.5									1	1	1	1	1		
0.4									1	1	1	1	1		
0.3							1	1	1	1	1	1	1	1	1
0.2							1	1	1	1	1	1	1	1	1
0.1							1	1	1	1	1	1	1	1	1
	P	2	3	4	5	6	7	8	9	10	11	12	13	14	15

$$C = 0.3/7 + 0.3/8 + 0.6/9 + 0.6/10 + 1/11$$
$$+ 0.6/12 + 0.6/13 + 0.3/14 + 0.3/15$$

α - cut:

for 0.1:

$A_{0.1} = [6, 10]$ $B_{0.1} = [1, 5]$, $C_{0.1} = [(6+1), (10+5)] = [7, 15]$

for 0.2:

$A_{0.2} = [6, 11]$ $B_{0.2} = [1, 5]$, $C_{0.2} = [6+1, 10+5] = [7, 15]$

for 0.3: Same as 0.2, $C_{0.3} = [7, 15]$

for 0.4, 0.5, 0.6: (same)

$A_{0.4} = [7, 9]$, $B_{0.4} = [2, 4]$, $C_{0.4} = [7+2, 9+4] = [9, 13]$

$C_{0.5} = [9, 13]$

$C_{0.6} = [9, 13]$

for 0.7, 0.8, 0.9, 1.0: (same)

$A_{0.7} = [8, 8]$, $B_{0.7} = [3, 3]$, $C_{0.7} = [11, 11]$

$C_{0.8} = [11, 11]$

$C_{0.9} = [11, 11]$

$C_{1.0} = [11, 11]$