SHUBHAM NARKHEDE

20011019

Concept questions

Q1 a) True.

Q1. B ~~False~~ True

Q1.2 Bias - Variance trade off is a factor that is seen in ML models which is used to reduce the variance by increasing the bias. It's a combination ~~by increasing the~~ of two errors, bias and variance. Bias can be caused by the learning algorithms. The higher the bias, the more likely it is to miss the relevant features and desired outputs. Variance is caused when noisy data in training set is also considered to operate on for the analysis.

To address the bias and reduce it, we can increase the complexity of the model as well as perform feature engineering to not miss out on relevant features.

One can also change the model parameters to address the variance, the model needs to be simplified. The size of training data can also be increased and the λ parameters can be increased too.

Q1-3 overfitting occurs when a good fit of the model is achieved on training data but not well on any other unseen data (when the model is complex, it tries to fit the training data exactly and predict results based on the same. Thus, it won't be able to generalize better.

Several techniques can reduce overfitting

1 Having a larger data set for training

2 Cross-Validation :- i.e partitioning the available dataset into subsets for training and validation. Repeating this training and testing on different subsets.

3 Regularization :- using methods like L1 (Lasso), L2 (Ridge) regularization add penalty terms to the model's loss function.

4 Feature selection :- using the most important features can prevent model from learning noise.

5 Dimensionality reduction.

6 Early stopping :- in models like neural network, stopping the training process once the model performance on a validation set begins to degrade prevents it from overfitting.

7 Ensembling :- i.e techniques like bagging and boosting involve combining multiple models to create a more accurate and robust predictive

Q2.1 Logistic Regression.

Given a linear regression model:-

$$h(x) = w_0 + w_1 x_1 + \dots + w_n x_n$$

In logistic regression the model output $h(x)$, represents the linear combination of the input features multiplied by their corresponding weights.

∴ using the sigmoid function we can transform continuous output into a valid probability denoted as

$$p(x) = \frac{1}{1 + e^{-h(x)}}$$

Sigmoid function maps the linear combination of features to a value between 0 and 1. Replacing $h(x)$ in the formula, we get:

$$p(x) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \dots + w_n x_n)}}$$

This function ensures that the predicted probability is within the range of 0 to 1.

Q2.2 The sigmoid is ideal for modeling probabilities in logistic regression due to its properties. It takes any real-valued number and squashes it into the range 0 to 1. This property makes it perfect for binary classification, where the goal is to estimate the probability of a ~~binary~~ Binary outcome.

The logistic regression model predicts the probability of a sample belonging to a certain class, and the sigmoid function effectively converts the linear combination of features into a probability between 0 and 1, allowing for easy interpretation as a probability of belonging to a particular class.

Q2.3 The log loss function, often referred to as binary cross-entropy, is a crucial metric used in logistic regression for binary classification problems.

This function quantifies the difference between predicted probabilities and actual binary outcomes.

The log loss function is defined as follows:-

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

where :-
$\hat{y}$ is the predicted probability.
y is the actual binary label (0 or 1).

when $y = 1$, log loss function becomes :-
$$L(\hat{y}, 1) = -\log(\hat{y})$$
when $y = 0$, the log loss function becomes:
$$L(\hat{y}, 0) = -\log(1-\hat{y})$$

The goal during training is to minimize this log loss function by adjusting the model's weight.

Stochastic Gradient Descent can be used for this purpose.
We can use the SGD to train the logistic regression model with the log loss function.

1 initializing the weights $(w_0, w_1, \ldots w_n)$ of the logistic regression model.
2 forward pass :- compute the output of the model for each training sample using the current weights and then calculate the predicted probability $\hat{y}$ using the sigmoid function
$$\hat{y} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_n x_n)}}$$
3 compute log loss :-
using the predicted probability $\hat{y}$ and the actual label y, calculate the log loss for

the current sample.

4 Backward pass (Gradient Descent):-
compute the gradient of the log loss with respect to each weight.
update the weight in the opposite direction of the gradient to minimize the loss.

5 Repeat steps 2-4 each training sample in the dataset.

6 Convergence:-
The training process stops when the loss converges to a minimum, or after a predetermined number of epochs.

The iterative process with stochastic gradient Descent aims to minimize the log loss by adjusting the model's weights, allowing the logistic regression model to learn the relation between the features and the binary target variable for accurate predictions on unseen data.

3 => Decision Tree.

Q3.01. To calculate the information gain for each attribute, we'll use the formula.

$$\text{Information Gain} = \text{Entropy}(Y) - \sum \frac{N_i}{N} \text{Entropy}(Y|X_i)$$

where

entropy (Y) is entropy of target variable Y

N is the total number of instances.

N_i is the number of instances in the subset X_i

Entropy $(Y|X_i)$ is entropy of Y given X_i.

calculating entropy of Y.

Total instances $(N) = 6$

$P(\text{Prime}) = \frac{3}{6}$ $P(\text{Walk}) = \frac{2}{6}$ $P(\text{Bus}) = \frac{1}{6}$

$$\text{Entropy}(Y) = -\left(\frac{3}{6} \times \log_2\left(\frac{3}{6}\right)\right) + \frac{2}{6} \times \log_2\left(\frac{2}{6}\right) + \frac{1}{6} \times \log_2\left(\frac{1}{6}\right)$$

$$\text{Entropy}(Y) \approx -(0.5 \times \log_2(0.5) + 0.333 \log_2(0.333) + 0.167 \log_2(0.167))$$

$$\text{Entropy}(Y) \approx 1.459$$

Now, let's calculate the information gain for each attribute.

=> Attribute x_1 = "owns a car"

Subset 1: owns a car (Yes)

Entropy $(Y|x_i = yes) = -\left(\frac{3}{4} \times \log_2\left(\frac{3}{4}\right) + \frac{1}{4} \times \log_2\left(\frac{1}{4}\right) + 0\right)$

Entropy $(Y|x_1 = yes) \approx -(0.75 \times \log_2(0.75) + 0.25 \times \log_2(0.25))$

Entropy $(Y|x_1 = yes) \approx 0.811$

Subset 2:- Dont own a car (No)

Entropy $(Y|x_1 = No) = -(0 \times \log_2(0) + 0.5 \times \log_2(0.5) + 0.5 \times \log_2(0.5))$

$= 1$

Information gain:-

Information gain (x_1) = Entropy $(Y) - \left(\frac{4}{7} \times 0.811 + \frac{3}{7} \times 1\right)$

$= 1.459 - (0.54 + 0.333)$

≈ 0.586

=> Attribute x_2 = "Distance to campus"

Subset 1: Far.

Entropy $(Y|x_2 = far) = -\left(\frac{2}{3} \log_2\left(\frac{2}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + 0\right)$

$\approx -(0.167 \times \log_2(0.167) + 0.333 \times \log_2(0.333))$

≈ 0.918

Subset 2: close.

$$\text{Entropy}(Y|X_2 = \text{close}) = -\left(\frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right)\right)$$

$$\text{Entropy}(Y|X_2 = \text{close}) = 1.585$$

Information Gain.

$$\text{Information Gain}(X_2) = \text{Entropy}(Y) - \left(\frac{3}{6} \times 0.918 + \frac{3}{6} \times 1.585\right)$$

$$= 1.459 - (0.459 + 0.793)$$

$$\approx 0.207$$

∴ The information gain for "Own a car" (X_1) is approximately 0.586, and for Distance to campus" (X_2) is approximately 0.207.

$\approx -(0.667 \times \log_2(0.667) + 0.333 \times \log_2(0.333))$

≈ 0.918

Subset 2: close.

$$\text{Entropy}(Y|X_2 = \text{close}) = -\left(\frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right)\right)$$

$$\text{Entropy}(Y|X_2 = \text{close}) = 1.585$$

Information Gain:

$$\text{Information Gain}(X_2) = \text{Entropy}(Y) - \left(\frac{3}{6} \times 0.918 + \frac{3}{6} \times 1.585\right)$$

$$= 1.459 - (0.459 + 0.793)$$

$$\approx 0.207$$

∴ The information gain for "own a car" (X_1) is approximately 0.586, and for "Distance to campus" (X_2) is approximately 0.207.

$\approx -(0.667 \times \log_2(0.667) + 0.333 \times \log_2(0.333))$

≈ 0.918

Subset 2: close.

$$\text{Entropy}(Y|X_2 = \text{close}) = -\left(\frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right) + \frac{1}{3} \times \log_2\left(\frac{1}{3}\right)\right)$$

$$\text{Entropy}(Y|X_2 = \text{close}) = 1.585$$

Information Gain:

$$\text{Information Gain}(X_2) = \text{Entropy}(Y) - \left(\frac{3}{6} \times 0.918 + \frac{3}{6} \times 1.585\right)$$

$$= 1.459 - (0.459 + 0.793)$$

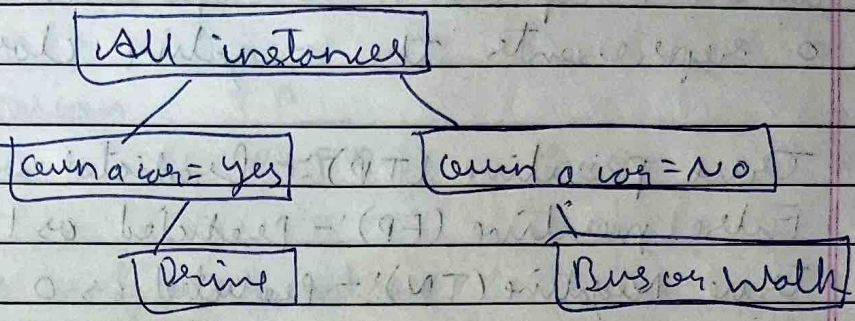
$$\approx 0.207$$

∴ The information gain for "Own a car" (X_1) is approximately 0.586, and for "Distance to campus" (X_2) is approximately 0.207.

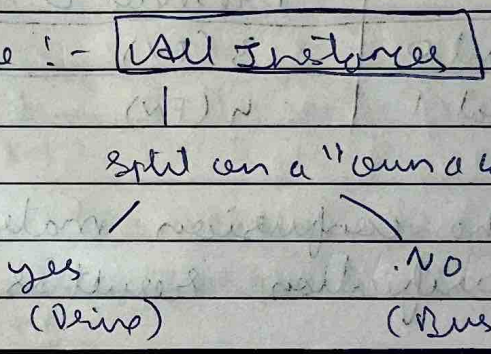
Q3.2

The attribute "own a car" (X_1) has higher information gain than "Decision to campus" (X_2).

∴ The decision tree will split on the attribute "own a car" first



∴ Decision tree structure



Q3.3

Based on the above tree:

1) "Own a car" = No (or the student doesn't own a car)

2) "Distance to campus" = Far

∴ The decision tree predicts that for a student who doesn't own a car and is far from campus the classification will be "Bus or walk".

OM

IV Model Evaluation

Qn.9 Given

$y = [1, 1, 1, 0, 0, 0, 1, 1, 1, 1]$

$y_c = [1, 1, 0, 1, 0, 0, 1, 0, 0, 0]$

where 1 represent the positive class and 0 represents the negative class.

True positive (TP):- Predicted as 1 and is actually 1
False positive (FP):- Predicted as 1 but actually 0.
True Negative (TN):- Predicted as 0 and is actually 0
False Negative (FN):- Predicted as 0 but actually 1

	Predicted 0	Predicted 1
Actual 0	3 (TN)	2 (FP)
Actual 1	4 (FN)	1 (TP)

∴ The confusion matrix for the given classification result is.

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

This matrix summarizes the performance of the binary classifier on the provided dataset.

Q4.2 confusion matrix :- $\begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$

Now,

Precision measures the accuracy of the positive predictions :

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

from the confusion matrix -

$TP = 1$

$FP = 2$.

$FN = 4$.

$$\therefore \text{Precision} = \frac{1}{1+2} = \frac{1}{3} \approx 0.333$$

$\Rightarrow$ Recall measures the proportion of actual positives that were correctly identified.

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$= \frac{1}{1+4} = \frac{1}{5} \approx 0.2$$

$\Rightarrow$ F1 score is the harmonic mean of precision and recall :

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$= 2 \times \frac{0.333 \times 0.2}{0.333 + 0.2} \approx 2 \times \frac{0.0666}{0.533}$$

$F_1 = 2 \times 0.125$

$F_1 = 0.25$

Summary :-

Precision ≈ 0.333

Recall $= 0.2$

F1 score $= 0.25$

These metrics indicate the model's performance in terms of precision, recall, and the balance between precision and recall represented by the F1 score.

Q4-3) The ROC curve generated for the binary classifier shows a representation of the trade-off between the True Positive Rate (sensitivity) and False Positive Rate across different classification thresholds.

The Area under the curve (AUC) is 0.547, which is slightly higher than 0.5, indicating a modest ability of the model to distinguish between the positive and negative classes.

However, it falls relatively closer to a random classifier than to a ideal model. This suggests that the model performs better than random guessing but isn't highly accurate. This suggests that the.

model performs better than random guessing but isn't highly accurate.

AUC reflects the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

In this case, an AUC closer to 1 would indicate a stronger discriminatory ability and better overall performance of the model in distinguishing between the two classes at different thresholds.

Q5 => SVM

Q5.01 To find the weight vector w and bias b for the SVM model, we'll utilize the provided data points. The equation corresponding to the hyperplane for an SVM model can be represented as:

$$W^T \cdot X - b = 0$$

calculating the weight vector and Bias:

for class -1 :-

let the data point be x_1 and x_2

$x_1 = 4,5$

$x_2 = 5,6$

for class +1:

let data point be x_3, x_4

$x_3 = (1,4)$ $x_4 = (2,3)$

weight vector calculation:

calculating the weight vector using the support vectors x_1 and x_3.

$$w = x_1 - x_3$$

$$w = (4,5) - (1,4) = (3,1)$$

Bias calculation:

Bias can be calculated using any of the

support vectors, ~~th~~ here using x_1

$$b = -(w x_1)$$
$$= -((3,1) \cdot (4,5))$$
$$= -(12+5)$$
$$= -17$$

equation of the hyperplane.

The equation of the hyperplane can be represented as

$$(3,1) \cdot x - 17 = 0$$
$$3x_1 + x_2 - 17 = 0$$

(S.2) identifying the support vectors :-

i.e the points that ~~at~~ lie on the margin or are misclassified, playing a crucial role in defining the decision boundary.

~~for class (-1)~~

The decision boundary is defined by

$3x_1 + x_2 - 17 = 0$

i.e $w \cdot x - b = 0$, where $w = (3,1)$ and $b = 17$.

Distance calculations

Let's calculate the distance of each point from decision boundary:-

for class -1 :-

$(4,5) = 3(4) + 5 - 17 = 12 + 5 - 17$
$= 0$ (on the boundary)

$(5,1) = 3(5) + 1 - 17 = 15 + 1 - 17 = 4$
(not on the boundary)

for class +1 :-
$(1,4) = 3(1) + 4 - 17 = 3 + 4 - 17 = -10$
(not on boundary)
$(2,3) = 3(2) + 3 - 17 = 6 + 3 - 17 = -8$
(not on boundary)

∴ from the calculation, the point $(4,5)$ from class -1 is the support vector, as it lies on the decision boundary defined by the hyperplane equation.
$3x_1 + x_2 - 17 = 0$

Q6. => Ensemble Learning:

Q6.1 using Majority Vote fusion function.
we take the most common class prediction among the classifier.

given:
classifier 1 predicts class 1
classifier 2 predicts class 2
classifier 3 predicts class 1

∴ The most common prediction among the classifier is class 1. Therefore the final decision using the Majority vote fusion function would be class 1.

Q6.2) given weight:
classifier 1: 0.2
classifier 2: 0.4
classifier 3: 0.4

Calculating the weighted votes:-

∴ Total weighted votes for class 1
$= 0.2 \times 1 + 0.4 \times 0 + 0.4 \times 1 = 0.2 + 0 + 0.4$
$= 0.6$

Total weighted votes for class 2:-
$= 0.2 \times 0 + 0.4 \times 1 + 0.4 \times 0 = 0 + 0.4 + 0$
$= 0.4$

: The final decision using the weighted majority vote fusion function would be class 1. due to having a higher total weighted vote.

Q6.3)

using the confusion matrix from each classifier to calculate the conditional probabilities for each class.

for classifier 1:-

Total = 80 (class 1) + 60 (class 2) = 140

$$P(\text{class } 1) = \frac{80}{140} = 0.57$$

$$P(\text{class } 2) = \frac{60}{140} = 0.43$$

for classifier 2:-

Total = 70 (class 1) + 100 (class 2) = 160

$$P(\text{class } 1) = \frac{70}{160} = 0.44$$

$$P(\text{class } 2) = \frac{90}{160} = 0.56$$

for classifier 3:-

Total = 120 (class 1) + 80 (class 2) = 200

$$P(\text{class } 1) = \frac{120}{200} = 0.6$$

$$P(\text{class } 2) = \frac{80}{200} = 0.4$$

Calculating the conditional probability.

Based on these probabilities, let's calculate the final class probability given the predicted from 3 class

for class 1 prediction

classifier 1 $P(\text{class } 1 \mid C_1) = \frac{80}{80+40} = 0.67$

classifier 2 $P(\text{class } 1 \mid C_2) = \frac{60}{60+10} = 0.86$

classifier 3 $P(\text{class } 1 \mid C_3) = \frac{70}{70+30} = 0.70$

∴ final probability

$P(\text{class } 1) = P(\text{class } 1) \times P(\text{class } 1 \mid C_1) \times P(\text{class } 1 \mid C_2) \times P(\text{class } 1 \mid C_3)$

$= 0.57 \times 0.67 \times 0.86 \times 0.70$

$= 0.2299038$

$P(\text{class } 2) = 1 - 0.2299038$

$= 0.7700962.$

∴ our final decision is class 2 as it has a higher probability based on the Naive Bayes fusion method.

Q) Naive Bias:

Q.10) Bayes Theorem, a fundamental concept of Probability theory:

describes the Probability of an event based on prior knowledge of conditions that might be related to the event. it is mathematically represented as.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Proof:-

$$P(A \cap B) = P(A|B) \times P(B)$$
$$P(B \cap A) = P(B|A) \times P(A)$$

equating these expressions.

$$P(A|B) \neq P(B) = P(B|A) \times P(A)$$

$$\therefore P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

usefulness in Machine learning:-

Bayes theorem is foundational in machine learning for several reasons:

1. Bayesian inference:- It forms the basis of Bayesian methods, enable the updating of prior beliefs with new data to form

posterior probabilities.

2 Classification Algorithms :- It's used in Naive Bayes classifier, a simple and effective probabilistic classification method that assumes independence among features given the class. These classifiers work well in text classifications, spam filtering, recommendation systems.

3 Handling uncertainty :- Bayes Theorem provides a principled way to update ~~beliefs~~ beliefs in the face of new evidence, making it beneficial for reasoning under uncertainty and making decisions in the presence of incomplete or noisy data.

4 Model flexibility :- It allows incorporation of prior knowledge into models.

∴ Bayes Theorem is crucial for Machine learning as it provides a framework for reasoning about uncertain events, updating beliefs with new evidence, and developing efficient algorithms that can make decisions in uncertain or noisy environment.

07.2)

Total No. of yes = 8.

Total No. of NO = 4.

P(Play Tennis = Yes) = $\frac{8}{12}$ = 0.67

P(Play Tennis = NO) = $\frac{4}{12}$ = 0.34.

Wind

	weak		Strong	
yes	NO	yes = 3	NO	
P(weak/yes) = $\frac{5}{8}$	P(weak/NO) = $\frac{2}{4}$	P(strong/yes) = $\frac{3}{8}$	P(strong/NO) = $\frac{2}{4}$	

Humidity

	High		Normal	
yes = 3	NO	yes	NO	
P(high/yes) = $\frac{3}{8}$	P(high/NO) = $\frac{3}{4}$	P(Normal/yes) = $\frac{5}{8}$	P(Normal/NO) $\frac{1}{4}$	

Temperature.

	Hot		mild		cool.	
yes	NO	yes	NO	yes	NO	
P(hot/yes) = $\frac{1}{8}$	P(hot/NO) = $\frac{2}{4}$	P(mild/yes) = $\frac{4}{8}$	P(mild/NO) = $\frac{1}{4}$	P(cool/yes) = $\frac{3}{8}$	P(cool/NO) $\frac{1}{4}$	

Outlook.

Sunny		Overcast		Rainy	
yes	NO	yes	NO	yes	NO
P(Sunny/yes) = 2/8	P(Sunny/NO) = 3/4	P(over/yes) = 3/8	P(overcast/NO) 0	P(Rainy/yes) = 3/8	P(Rainy/NO) 1/4

X = (Sun, cool, High, strong)

P(X/yes) = P(yes) * P(Sunny/yes) * P(cool/yes) * P(High/yes) * P(strong/yes)

$= \frac{8}{12} \times \frac{2}{8} \times \frac{3}{8} \times \frac{3}{8} \times \frac{3}{8}$

$= \frac{9}{1024}$

$= 0.008784$

P(X/NO) = P(NO) * P(Sunny/NO) * P(cool/NO) * P(High/NO) * P(strong/NO)

$= \frac{4}{12} \times \frac{3}{4} \times \frac{1}{4} \times \frac{3}{4} \times \frac{2}{4}$

$= \frac{3}{128} = 0.023437$

here $P(x|yes) < P(x|No)$

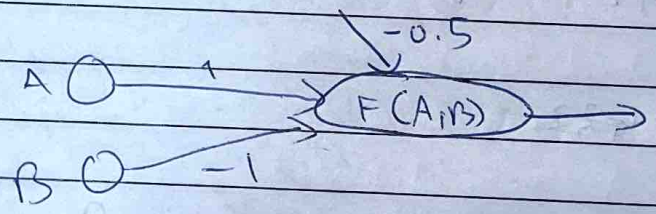
∴ Playtennis = No.

Q8 Neural Network

Q8.1 a) yes, we can represent with a single logistic threshold unit, since it is linearly separable.

$F(A,B) = 1\{A - B - 0.5 > 0\}$

and the single logistic unit for input and output is drawn below:



Q8.2 Given

- 3 layer feedforward neural network
- → two input units x_1, x_2
- → one output unit with y.
- → hidden layer with two units h_1 and h_2.

all functions are linear without Bias.

Parameters :- $\eta = 0.4$

$\alpha = 0.8$, ~~All weights~~

all weights initialized to 0.1

Training sample :

$x_1 = 1, x_2 = 0, y = 1$

calculation of first iteration using Stochastic gradient descent with Backpropogation.

1 Initializing weights :

Hidden layer weights :-

$$W_{input\ to\ hidden} = \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}$$

- Output layer weights.

$$W_{hidden\ to\ output} = \begin{bmatrix} w_{13} \\ w_{23} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}$$

2 Forward Pass.

calculate the predicted output :-

$z_1 = x_1 \cdot w_{11} + x_2 \cdot w_{12} = 1 \times 0.1 + 0 \times 0.1 = 0.1$

$z_2 = x_2 w_{21} + x_2 w_{22} = 1 \times 0.1 + 0 \times 0.1 = 0.1$

hidden layer activation

$h_1 = z_1 = 0.1$

$h_2 = z_2 = 0.1$

- output prediction :-

$y_{predicted} = h_1 w_{13} + h_2 w_{23} = 0.1 \times 0.1 + 0.1 \times 0.1$

$= 0.2$

3 Backword pass:-

Calculate the error:-

$$\text{Error} = \frac{1}{2} \times (y - y_{pred})^2 = \frac{1}{2}(1-0.02)^2$$

$$= 0.9802$$

update weight using gradient descent.

output layer to hidden layer weights updates

$$\Delta w_{13} = \eta \times \text{error} \times h_1$$

$$= 0.4 \times 0.9802 \times 0.1 = 0.039208$$

$$\Delta w_{23} = \eta \times \text{error} \times h_2.$$

$$= 0.4 \times 0.9802 \times 0.1 = 0.039208.$$

update the weights with momentum:-

$$w_{13}^{new} = w_{13} + \Delta w_{13} + \alpha \times \Delta w_{13}$$

$$= 0.1 + 0.039208 + 0$$

$$= 0.139208$$

$$w_{23}^{new} = w_{23} + \Delta w_{23} + \alpha \times w_{23}$$

$$= 0.1 + 0.039208 + 0 = 0.139208$$

1- Input to hidden layer weights remains unchanged as there no direct update coming from the output layer to the hidden layer in this iteration. This concludes the first training iteration using stochastic gradient.

Page No.
Date: / /

Desent with Backpropogation for the given neural network and training sample.