National University of Computer and Emerging Sciences

FAST School of Computing **Spring-2024** **Islamabad Campus**

CS-4090: Machine Learning for Robotics Final Exam

Total Time: 3 Hour
Total Marks: 108

Date: 31st May, 2024

Course Instructor(s)

Dr. Imran Ashraf

Student Name	Roll No.	Course Section	Student Signature

Do not write anything on the question paper except the information required above.

Instructions:

1. Read the question carefully, understand the question, and then attempt your answers in the provided answer booklet.
2. Verify that you have **Nine (9)** printed pages of the question paper including this page and a bubble sheet for MCQs at then end. There are **Eight (8)** questions. Solve all questions.
3. Calculator sharing is strictly prohibited.
4. Properly indent your code and use meaningful names for variables.
5. For the important parts, use comments to explain your code.
6. Solve all parts of a questions together and in order as the next part builds upon the previous one.
7. In any question if a calculation is involved, do not forget to provide the formula that you used for calculation.
8. Fill MCQ answers on the bubble sheet.
9. Return the answer sheet as well as the question paper to the invigilator.

Q1: Basics and Terminology **[Marks = 2x5]**

Compare and Contrast

a) Stochastic Gradient Descent vs Mini-batch Gradient Descent
b) Bias vs Variance
c) Logistic Regression vs Softmax Regression
d) Grid Search vs Randomized Search
e) Soft voting vs Hard voting

Stochastic Gradient Descent	Mini-batch Gradient Descent
After each training example updates weights	Updates weights after each mini-batch of training examples
Suitable for very large datasets or online learning	Commonly used in deep learning and practical settings
Less efficient for modern hardware	More efficient, leverages matrix operations and GPUs

Bias	**Variance**
Bias refers to errors introduced by approximating a real-world problem (which may be complex) by a much simpler model. High bias can cause underfitting.	Variance refers to errors introduced by the model's sensitivity to small fluctuations in the training set. High variance can cause overfitting.
Causes the model to miss relevant relations between features and target outputs (underfitting)	Causes the model to model the random noise in the training data rather than the intended outputs (overfitting)
Low flexibility	High flexibility
Logistic Regression	**Softmax Regression**
A binary classification algorithm that predicts the probability of one of two classes	A generalization of logistic regression for multiclass classification problems
Sigmoid function is used as activation function	Softmax function is used as activation function
Grid Search	**Randomized Search**
Systematically searches through a specified subset of the hyperparameter space	Randomly samples hyperparameters from specified distributions
Exhaustive search over a predefined grid of hyperparameters	Random search within predefined distributions for each hyperparameter
Inefficient with large hyperparameter spaces	More efficient with large hyperparameter spaces
Soft voting	**Hard voting**
Combines the predicted probabilities from multiple models and averages them to make the final prediction	Combines the predicted class labels from multiple models and takes a majority vote to make the final prediction
Utilizes the probability estimates from each model	Does not use probability estimates, only final class labels
Can potentially be more accurate if probability estimates are reliable	May be less accurate if individual models vary widely in their confidence levels

Q2: Multilayer Perceptron **[Marks = 3x4]**

Suppose you have an MLP composed of one input layer with 10 pass-through neurons, followed by one hidden layer with 50 artificial neurons, and finally one output layer with 3 artificial neurons. All artificial neurons use the ReLU activation function.

a) What is the shape of the input matrix X?

The shape of X will be (m x 10) where m is the batch size.

b) What about the shape of the hidden layer's weight vector Wh, and the shape of its bias vector bh?

Since the hidden layer has 50 neurons and there are 10 features, its weight vector will have the shape of 10 x 50. Because each of the 10 features will need to be multiplied by a weight which is connected to each of the 50 hidden layer neurons.

The bias vector will have a length of 50. This is because one bias unit is added to the output of the passthrough layer. That unit is then multiplied once for each of the 50 artificial neurons.

c) What is the shape of the output layer's weight vector Wo, and its bias vector bo?
The output layer has 3 neurons. The input to the output layer is the output of the hidden layer. The output of the hidden layer has 50 neurons, therefore the shape of the output layer's weight vector is 50 x 3.

The bias vector will have a length of 3 because one bias unit is added to the output of the hidden layer. That unit is then multiplied once for each of the 3 neurons in the output layer.

d) What is the shape of the network's output matrix Y?
The shape of the output matrix is going to be m x 3, where m is the batch size, and 3 because each example will compute a probability that it belongs to one of three classes.

Q3) Regression Performance Metrics [Marks = 12]
A machine learning model is trained to predict house prices based on square footage. The model generates the following predictions for a set of houses and their corresponding actual prices:

House No	Predicted Price	Actual Price
1	\$400,000	\$425,000
2	\$320,000	\$300,000
3	\$550,000	\$575,000
4	\$280,000	\$290,000
5	\$600,000	\$620,000

A) Calculate Mean Absolute Error (MAE) for this model's predictions.
20000

B) Calculate Root Mean Squared Error (RMSE) for this model's predictions.
20736

C) Briefly explain which metric (MAE or RMSE) might be more suitable for evaluating house price predictions in this scenario and why.

RMSE squares the errors before averaging them. This disproportionately penalizes larger errors, which can be significant in house prices. A slight overestimation for a very expensive house can significantly inflate the RMSE compared to MAE. Therefore, MAE is more suitable.

Q4) Classification Performance Metrics [Marks = 12]
A classification model is evaluated on a test dataset with 100 positive examples and 200 negative examples. The model correctly classifies 90 positive examples and 180 negative examples, but misclassifies 10 positive examples as negative and 20 negative examples as positive. Calculate:

A) Accuracy: 0.9
B) Precision: 0.8182
C) Recall: 0.9
D) F1 score: 0.8571

E) Briefly describe a scenario where high precision might be more critical than high recall?
While missing some spam emails (false negatives) might be inconvenient, it's generally less detrimental than mistakenly marking important emails as spam (false positives). A flooded inbox with spam is annoying, but accidentally deleting or overlooking a critical email due to a false positive can have serious consequences.

F) Briefly describe a scenario where high recall might be more critical than high precision?
High recall is crucial in medical screening tests to ensure that as many cases of the disease as possible are detected early, even if it means accepting a higher rate of false positives.

Q5) Decision Trees [Marks = 10]
Consider the following trading dataset where we have three features (Past Trend, Open Interest and Trading Volume) and one target variable (Return). For the construction of the Decision Tree classifier using Gini Index, which feature should be used as root node and why?

Past Trend	Open Interest	Trading Volume	Return
Positive	Low	High	Up
Negative	High	Low	Down
Positive	Low	High	Up
Positive	High	High	Up
Negative	Low	High	Down
Positive	Low	Low	Down
Negative	High	High	Down
Negative	Low	High	Down
Positive	Low	Low	Down
Positive	High	High	Up

Calculating the Gini Index for past trend

P(Past Trend=Positive): 6/10
P(Past Trend=Negative): 4/10

If (Past Trend = Positive & Return = Up), probability = 4/6
If (Past Trend = Positive & Return = Down), probability = 2/6

Gini Index = 1 - ((4/6)^2 + (2/6)^2) = 0.45

If (Past Trend = Negative & Return = Up), probability = 0
If (Past Trend = Negative & Return = Down), probability = 4/4

Gini Index = 1 - ((0)^2 + (4/4)^2) = 0

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index for Past Trend = (6/10)0.45 + (4/10)0 = <u>0.27</u>

Calculating the Gini Index for open interest

P(Open Interest=High): 4/10
P(Open Interest=Low): 6/10

If (Open Interest = High & Return = Up), probability = 2/4
If (Open Interest = High & Return = Down), probability = 2/4

Gini Index = 1 - ((2/4)^2 + (2/4)^2) = 0.5

If (Open Interest = Low & Return = Up), probability = 2/6
If (Open Interest = Low & Return = Down), probability = 4/6

Gini Index = 1 - ((2/6)^2 + (4/6)^2) = 0.45

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index for Open Interest = (4/10)0.5 + (6/10)0.45 = <u>0.47</u>

Calculating the Gini Index for trading volume
P(Trading Volume=High): 7/10
P(Trading Volume=Low): 3/10

If (Trading Volume = High & Return = Up), probability = 4/7
If (Trading Volume = High & Return = Down), probability = 3/7

Gini Index = 1 - ((4/7)^2 + (3/7)^2) = 0.49

If (Trading Volume = Low & Return = Up), probability = 0
If (Trading Volume = Low & Return = Down), probability = 3/3

Gini Index = 1 - ((0)^2 + (1)^2) = 0

Weighted sum of the Gini Indices can be calculated as follows:

Gini Index for Trading Volume = (7/10)0.49 + (3/10)0 = <u>0.34</u>

‘past trend’ has the lowest Gini Index and hence, it will be chosen as the root node for how the decision tree works.

Q6) Dimensionality Reduction [Marks = 10]

A) Briefly explain the term Curse of Dimensionality?

The "Curse of Dimensionality" refers to a set of challenges that arise when dealing with high-dimensional data. As the number of dimensions (features) in a dataset increases, several problems become more pronounced. These problems include:

- increase in sparsity
- over-fitting
- growth of computational complexity

B) List two advantages and two disadvantages of dimensionality reduction.

Advantages:

1. By reducing the number of features, the risk of overfitting can be decreased because the model becomes simpler and less likely to fit noise in the training data.
2. With fewer features, models can often be trained faster and require less computational resources, which can lead to improved performance, especially with large datasets.

Disadvantages:

1. Dimensionality reduction involves transforming the original data into a lower-dimensional space, which can result in the loss of some information. This loss can affect the performance and accuracy of models trained on the reduced data.
2. Some dimensionality reduction techniques, like PCA, produce transformed features (principal components) that are linear combinations of the original features. These new features can be difficult to interpret and may not have a straightforward meaning.

C) PCA relies on Singular Value Decomposition (SVD) to compute Eigen values. What will happen when Eigenvalues are roughly equal?

PCA aims to capture as much variance in the data as possible using fewer dimensions. When eigenvalues are roughly equal, it suggests that the data doesn't exhibit clear disparities in variance along different dimensions. As a result, it might be challenging for PCA to determine which dimensions are more important for capturing the underlying structure of the data.

Q7) Python Implementation [Marks = 12]

The following class implements Linear Regression. Complete the missing code (mentioned as TODO 1 to 3). No need to reproduce the complete source code on the answer sheet. Only produce the missing source code with its TODO number.

```
class LinearRegression:
   def __init__(self, lr=0.001, epochs=30):
      """
      Fits a linear regression model on a given dataset.
```

```
        Args:
            lr: learning rate
            epochs: number of iterations over the dataset
        """
        self.lr = lr
        self.epochs = epochs
        self.weights = None
        self.bias = None


    def train(self, X, y):
        """
        Initialize weights. Iterate through the dataset and update weights once every epoch.

        Args:
            X: features
            y: target
        """
        n_samples, n_features = X.shape

        # TODO 1: Initialize weights and bias
        self.weights = np.zeros(n_features)
        self.bias = 0

        for _ in range(self.epochs):
            self.update_weights(X, y)

    def update_weights(self, X, y):
        """
        Helper function to calculate the gradients and update weights using batch gradient
descent.

        Args:
            X: features
            y: target
        """
        n_samples = X.shape[0]

        # Predictions
        y_pred = self.predict(X)


        # TODO 2: Compute gradients
       dw = (1 / n_samples) * np.dot(X.T, (y_pred – y))
       db = (1 / n_samples) * np.sum(y_pred – y)

        # TODO 3: Update weights and bias
        self.weights = self.weights – self.lr * dw
        self.bias =  self.bias – self.lr * db
```

```
def predict(self, X):
    """
    Predict values using the weights.

    Args:
        X: features

    Returns:
        The predicted value.
    """
    return np.dot(X, self.weights) + self.bias
```

Q8: Multiple Choice Question **[Marks = 30]**

MCQ No	Answer	MCQ No	Answer
1	A	16	C
2	A	17	A
3	B	18	B
4	D	19	B
5	C	20	B
6	C	21	B
7	C	22	C
8	B	23	B
9	A	24	A
10	B	25	C
11	B	26	A
12	B	27	C
13	B	28	C
14	A	29	B
15	A	30	B