ABASYN UNIVERSITY

Islamabad Campus
Mid-Term Examination – Spring 2026

Course Code: CS494	Course Title: Special Topics in Computer Science
Teacher's Name: Dr. Sadaf Tanvir	Total Marks: 30
Date & Time: 07.3.26 – 01:00 pm	Total Time: 1.5 hours
Student's Name: _____	Reg. No: _____

1. Follow all the instructions written on the answer booklet.
2. Write your Registration Number and other relevant information on the front page of the answer booklet & sign it.
3. Ask for clarifications by the Course Instructor only within the first 15 minutes of the start of the paper.
4. The marks for each question/part are written in parentheses ( ).
5. Attempt objective and subjective parts on the Answer booklet.
6. Part A and Part B of each question should be written together in the answer book.

QUESTION 1 [5+2.5+2.5 Marks]

PART A: Elaborate the following concepts by giving examples from the code snippets: [5+2.5+2.5][CLO1] **Time: 20 minutes]**

a. Label in deep learning

b. Prediction

c. Error

d. Weights

e. Back Propagation

PART B: What is the output of this code?

```
import numpy as np

# Input features and binary labels for XOR
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0,1,1,0])  # Changed to XOR

# Initialize weights, bias, learning rate
w = np.random.rand(2); b = np.random.rand(1); lr = 0.2
def step(x): return 1 if x >= 0 else 0

# Training loop
for epoch in range(10):
    for xi, yi in zip(X, y):
        y_hat = step(np.dot(xi, w) + b)
        w += lr * (yi - y_hat) * xi
        b += lr * (yi - y_hat)
    print(f"Epoch {epoch+1}: w={w}, b={b}")

# Predictions after training
print("\nPredictions:")
for xi in X:
    print(f"{xi} -> {step(np.dot(xi, w)+b)}")
```

QUESTION 2 [5+5 Marks][CLO2]

PART A: Point out the errors in the code below. Do it on a question paper and attach the question paper with the answer sheet.

```
import numpy as np

# Input features and labels (AND gate)
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0,0,0,1])

# Initialize weights, bias, learning rate
w = np.random.rand(2)
b = np.random.rand(1)
lr = 0.1

print("Initial weights:" w)
print("Initial bias:", b)
print("-" * 40)

# Step activation function
def step(x):
    return 1 if x >= 0 else 0

# Training loop
for epoch in range(1, 4):  # keep epochs small for clarity
    print(f"\nEpoch {epoch}")
    print("-" * 40)

    for xi, yi in zip(X, y):
        net_input = np.dot(xi, w) + b
        y_hat = step(net_input)
        error = yi - y_hat

        print(f"Input: {xi}")
        print(f"Net input (x.w + b): {net_input}")
        print(f"Prediction: {y_hat}, Actual: {yi}")
        print(f"Error (yi - y_hat): {error}")

        # Update weights and bias
        w_old = w.copy()
        b_old = b.copy()

        w += lr * error * xi
        b += lr * error

        print(f"Updated weights: {w_old} -> {w}")
        print(f"Updated bias: {b_old} -> {b}")
        print("-" * 40)

# Final predictions
```

PART B (continuation — Final predictions after training):

```
print("\nFinal predictions after training:")
for xi in X:
    print(f"{xi} -> {step(np.dot(xi, w) + b)}")
```

PART B (sub-part c): Correct the following code. Write the corrected code on the answer sheet.

```
import numpy as np

# Input features and labels (AND gate)
X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0,0,0,1])

# Initialize weights, bias, learning rate
w = np.random.rand(2)
b = np.random.rand(1)
lr = 0.1

print("Initial weights:" w)       # ERROR: missing comma
print("Initial bias:", b)
print("-" * 40)

# Step activation function
def step(x):
    return 1 if x >= 0 else 0     # ERROR: should be x >= 0 (logic issue for perceptron)

# Training loop
for epoch in range(1, 4):
    print(f"\nEpoch {epoch}")
    print("-" * 40)

    for xi yi in zip(X, y):        # ERROR: missing comma between xi, yi
        net_input = np.dot(xi, w) + b
        y_hat = step(net_input)
        error = yi - y_hat

        print(f"Input: {xi}")
        print(f"Net input (x.w + b): {net_input}")
        print(f"Prediction: {y_hat}, Actual: {yi}")
        print(f"Error (yi - y_hat): {error}")

        # Update weights and bias
        w_old = w.copy()
      * b_old = b.copy()           # ERROR: stray '*' character

        w += lr * error * xi
        b += lr * error

        print(f"Updated weights: {w_old} -> {w}")
        print(f"Updated bias: {b_old} -> {b}")
        print("-" * 40)

# Final predictions
print("\nFinal predictions after training:")
for xi in X:
    print(f"{xi} -> {step(np.dot(xi, w) + b)}")
```

QUESTION 3 [5x2=10 Marks][CLO2]

Complete the missing lines of the code

```
import numpy as np
import tensorflow as tf
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.utils import to_categorical

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# One-hot encode labels
y = ____________________________________

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Build 8-16 MLP model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(8, activation='relu', input_shape=(4,)),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense_____________________________________
])

# Compile
model.compile(
    optimizer=_____________________________________,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Train
model.fit_____________________________________

# Evaluate
loss, accuracy = _____________________________________
print("Test Accuracy:", accuracy)
```

★ ★ ★ ★ ★ ★