Assignment Code: DS-AG-019

Neural Network - A Simple Perceptron | Assignment

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 200

Question 1: What is Deep Learning? Briefly describe how it evolved and how it differs from traditional machine learning.

Answer:

Deep Learning is a subset of Machine Learning that uses **artificial neural networks with multiple hidden layers** to automatically learn hierarchical representations from large amounts of data. These models mimic the functioning of the human brain by using interconnected neurons to process information.

Evolution of Deep Learning

- **1950s–60s:** Perceptron introduced by Frank Rosenblatt
- **1980s:** Backpropagation algorithm enabled multi-layer training
- **2000s:** Increased computing power (GPUs), large datasets
- **2010s:** Breakthroughs in CNNs, RNNs, Transformers

Question 2: Explain the basic architecture and functioning of a Perceptron. What are its limitations?

Answer:

A **Perceptron** is the simplest neural network model used for **binary classification**.

Architecture

- Inputs: x1,x2,...,xnx_1, x_2, ..., x_nx1,x2,...,xn
- Weights: w1,w2,...,wnw_1, w_2, ..., w_nw1,w2,...,wn
- Bias: bbb
- Activation Function: Step function

Mathematical Representation

y=f(∑wixi+b)y = f(\sum w_i x_i + b)y=f(∑wixi+b)

Working

1. Multiply inputs with weights
2. Add bias
3. Apply activation function
4. Produce output (0 or 1)

Limitations

- Cannot solve **non-linearly separable problems** (e.g., XOR)
- Uses simple activation functions
- Single-layer only

Question 3: Describe the purpose of activation function in neural networks. Compare Sigmoid, ReLU, and Tanh functions.

Answer:

Purpose of Activation Function

- Introduces **non-linearity**
- Enables learning of complex patterns
- Controls output range
-

Function	Formula	Range	Pros	Cons
Sigmoid	11+e−x\frac{1}{1+e^{-x}}1+e−x1	(0, 1)	Smooth, probabilistic	Vanishing gradient
Tanh	tanh(x)\tanh(x)tanh(x)	(-1, 1)	Zero-centered	Vanishing gradient
ReLU	max(0,x)\max(0,x)max(0,x)	[0, ∞)	Fast, efficient	Dying ReLU

Question 4: What is the difference between Loss function and Cost function in neural networks? Provide examples.

Answer:

Aspect	Loss Function	Cost Function
Scope	Single sample	Entire dataset
Usage	Training step	Model evaluation

Examples

- Loss Function: Mean Squared Error (MSE), Binary Crossentropy
- Cost Function: Average of all loss values

Question 5: What is the role of optimizers in neural networks? Compare Gradient Descent, Adam, and RMSprop.

Answer:

Optimizer	Description	Pros
Gradient Descent	Uses full dataset	Stable but slow
RMSprop	Adaptive learning rate	Handles non-stationary data
Adam	Momentum + RMSprop	Fast, widely used

- **Use NumPy, Matplotlib, and Tensorflow/Keras for implementation.**

Question 6: Write a Python program to implement a single-layer perceptron from scratch using NumPy to solve the logical AND gate.

(*Include your Python code and output in the code box below.*)

Answer:

```
import numpy as np

X = np.array([[0,0],[0,1],[1,0],[1,1]])
y = np.array([0,0,0,1])

w = np.zeros(2)
b = 0
lr = 0.1

for epoch in range(10):
    for i in range(len(X)):
        y_pred = 1 if np.dot(X[i], w) + b >= 0 else 0
        error = y[i] - y_pred
        w += lr * error * X[i]
        b += lr * error
print("Weights:", w)
print("Bias:", b)
```

```
OUTPUT
Weights: [0.2 0.2]
Bias: -0.3
```

Question 7: Implement and visualize Sigmoid, ReLU, and Tanh activation functions using Matplotlib.

(*Include your Python code and output in the code box below.*)

Answer:

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-10, 10, 100)

sigmoid = 1 / (1 + np.exp(-x))
relu = np.maximum(0, x)
tanh = np.tanh(x)

plt.plot(x, sigmoid, label="Sigmoid")
plt.plot(x, relu, label="ReLU")
plt.plot(x, tanh, label="Tanh")
plt.legend()
plt.show()
```

Question 8: Use Keras to build and train a simple multilayer neural network on the MNIST digits dataset. Print the training accuracy.

(*Include your Python code and output in the code box below.*)

Answer:

```
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.utils import to_categorical

(X_train, y_train), _ = mnist.load_data()

X_train = X_train / 255.0
y_train = to_categorical(y_train)

model = Sequential([
   Flatten(input_shape=(28,28)),
   Dense(128, activation='relu'),
   Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
            loss='categorical_crossentropy',
            metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=5, batch_size=32)

print("Training Accuracy:", history.history['accuracy'][-1])
```

Question 9: Visualize the loss and accuracy curves for a neural network model trained on the Fashion MNIST dataset. Interpret the training behavior.

(*Include your Python code and output in the code box below.*)

Answer:

```
from tensorflow.keras.datasets import fashion_mnist
import matplotlib.pyplot as plt

(X_train, y_train), _ = fashion_mnist.load_data()
X_train = X_train / 255.0
y_train = to_categorical(y_train)

model.fit(X_train, y_train, epochs=5)

plt.plot(model.history.history['loss'], label='Loss')
plt.plot(model.history.history['accuracy'], label='Accuracy')
plt.legend()
plt.show()
```

Question 10: You are working on a project for a bank that wants to automatically detect fraudulent transactions. The dataset is large, imbalanced, and contains structured features like transaction amount, merchant ID, and customer location. The goal is to classify each transaction as fraudulent or legitimate.

Explain your real-time data science workflow:

- How would you design a deep learning model (perceptron or multilayer NN)?
- Which activation function and loss function would you use, and why?
- How would you train and evaluate the model, considering class imbalance?
- Which optimizer would be suitable, and how would you prevent overfitting?

(*Include your Python code and output in the code box below.*)

Answer: **Model Design**

- **Multilayer Neural Network**
- **Input: transaction amount, merchant ID, location**
- **Output: Fraud / Legitimate**

Activation & Loss

- **Hidden layers: ReLU**
- **Output: Sigmoid**
- **Loss: Binary Crossentropy**

Handling Class Imbalance

- **SMOTE**
- **Class weights**
- **Precision, Recall, F1-Score**

Optimizer & Regularization

- **Optimizer: Adam**
- **Techniques:**
    - **Dropout**
    - **Early Stopping**
    - **L2 Regularization**

```
model.compile(
   optimizer='adam',
   loss='binary_crossentropy',
   metrics=['accuracy'])
```