CSL7670 : Fundamentals of Machine Learning

Lab Report

Name:	**Ganesh Kumar Nagal**
Roll Number:	**M23MEA004**
Program:	**M.Tech Adavance Manufacturing And Design**

Chapter 1

Lab-3

1.1 Objective

Objective of this assignment is to gain familiarity with perceptron.

1.2 Problem-1

1. (Playing with Perceptron) You are given a code for perceptron.Please modify it so that

   1. (a) It works for labels given as (+1,-1).

   2. (b) it works for the data provided in the attached CSV file. For create a CSV file with three features and two classes manually. Read this CSV file and executive the Perceptron code.

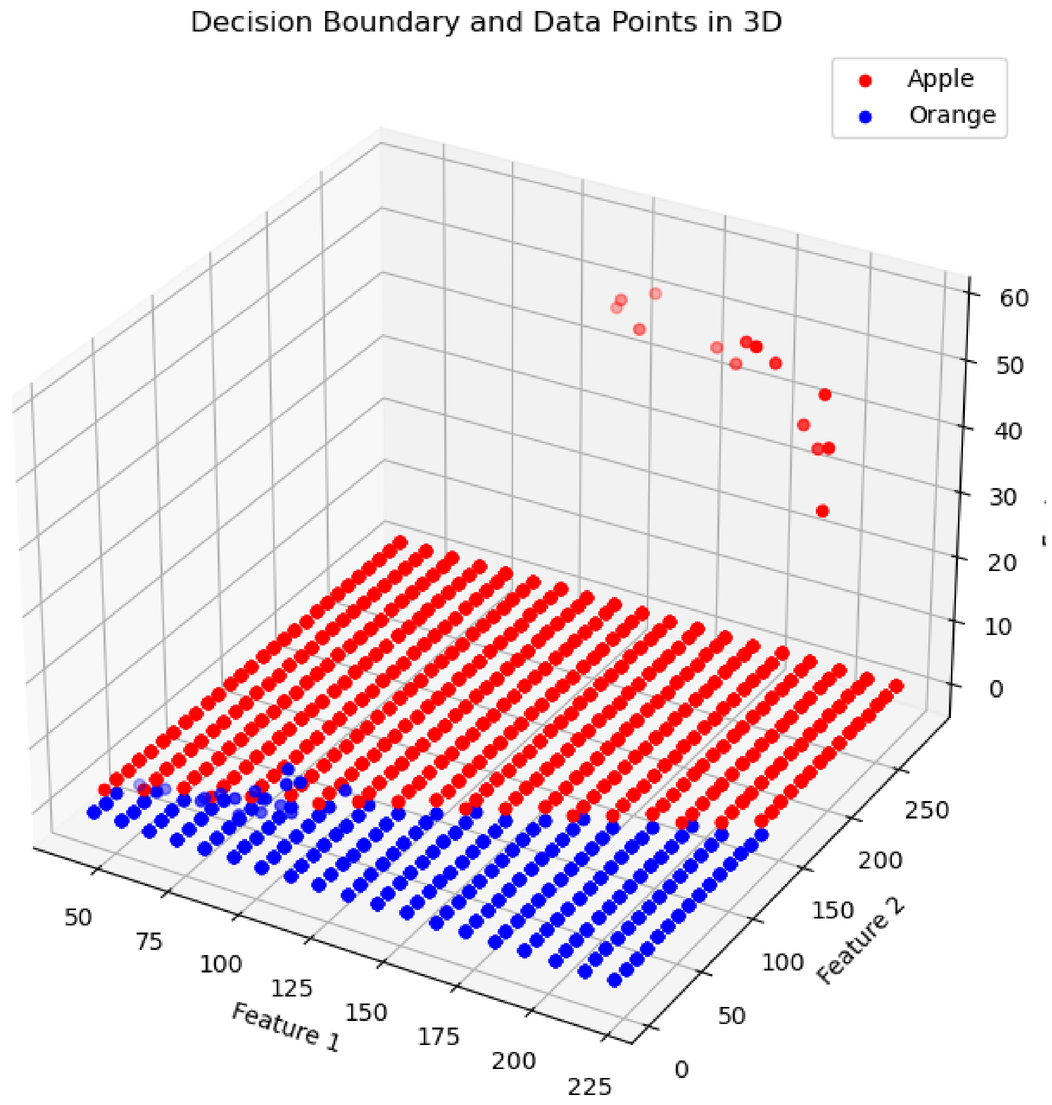
   Solution 1:

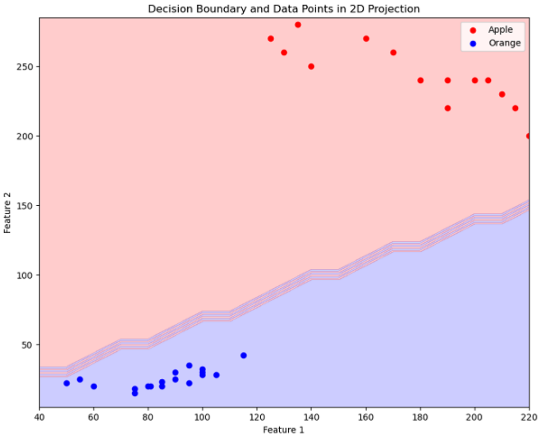
```
# -*- coding: utf-8 -*-
"""Assignment_3 (1).ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1l7LxwZwr35M4rhHiAFZXBIouMLxfUsfL
"""

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
from sklearn.model_selection import train_test_split

class CustomPerceptron:
    def __init__(self, num_features, learning_rate=0.1, epochs=100):
        self.weights = np.random.rand(num_features + 1)  # +1 for the bias term
        self.learning_rate = learning_rate
        self.epochs = epochs

    def activate(self, x):
        return 1 if x >= 0 else -1

    def predict(self, inputs):
        inputs_with_bias = np.insert(inputs, 0, 1)  # Add bias term
        summation = np.dot(inputs_with_bias, self.weights)
        return self.activate(summation)

    def train(self, training_data, labels):
        for _ in range(self.epochs):
            for inputs, label in zip(training_data, labels):
                prediction = self.predict(inputs)
                update = self.learning_rate * (label - prediction)
                inputs_with_bias = np.insert(inputs, 0, 1)  # Add bias term
                self.weights += update * inputs_with_bias  # Update weights
# Load the dataset
data_df = pd.read_csv(r"C:\Users\ganes\Downloads\data.csv")

# Split the data into features (x) and labels (y)
features = data_df.drop(['Label'], axis=1)
labels = data_df['Label']

# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(features, labels, test_size=0.30)

# Convert training data and labels to NumPy arrays
train_features = x_train.values
train_labels = y_train.values

# Create and train the perceptron
num_features = x_train.shape[1]  # Number of features in your data
perceptron = CustomPerceptron(num_features=num_features)
perceptron.train(train_features, train_labels)

# Test the trained perceptron
for test_data in x_test.values:
    prediction = perceptron.predict(test_data)
    if prediction == 1:
        print("Apple")
    else:
        print("Orange")

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Create a 3D scatter plot
fig_3d = plt.figure(figsize=(10, 8))
ax_3d = fig_3d.add_subplot(111, projection='3d')

# Plot data points
ax_3d.scatter(train_features[train_labels == 1][:, 0], train_features[train_labels == 1][:, 1], train_features[train_labels == 1][:, 2], color='red', label='Apple')
```

```
ax_3d.scatter(train_features[train_labels == -1][:, 0], train_features[train_labels == -1][:, 1], train_features[train_labels == -1][:, 2], color=’blue’, label=’Orange’)

# Create a meshgrid for decision boundary in 3D
x_min, x_max = train_features[:, 0].min() - 10, train_features[:, 0].max() + 10
y_min, y_max = train_features[:, 1].min() - 10, train_features[:, 1].max() + 10
z_min, z_max = train_features[:, 2].min() - 10, train_features[:, 2].max() + 10
xx, yy, zz = np.meshgrid(np.arange(x_min, x_max, 10), np.arange(y_min, y_max, 10), np.arange(z_min, z_max, 10))
xyz_points = np.column_stack((xx.ravel(), yy.ravel(), zz.ravel()))
predictions = np.array([perceptron.predict(point) for point in xyz_points])
Z_3d = predictions.reshape(xx.shape)

# Plot the decision boundary in 3D
for x_row, y_row, z_row in zip(xx, yy, Z_3d):
    for x, y, z in zip(x_row, y_row, z_row):
        if z == 1:
            ax_3d.scatter(x, y, z, c=’red’, marker=’o’)
        else:
            ax_3d.scatter(x, y, z, c=’blue’, marker=’o’)

ax_3d.set_title(’Decision␣Boundary␣and␣Data␣Points␣in␣3D’)
ax_3d.set_xlabel(’Feature␣1’)
ax_3d.set_ylabel(’Feature␣2’)
ax_3d.set_zlabel(’Feature␣3’)
ax_3d.legend()

plt.show()

# Create a 2D scatter plot for a specific view
fig_2d = plt.figure(figsize=(10, 8))
ax_2d = fig_2d.add_subplot(111)

# Plot data points
ax_2d.scatter(train_features[train_labels == 1][:, 0], train_features[train_labels == 1][:, 1], color=’red’, label=’Apple’)
ax_2d.scatter(train_features[train_labels == -1][:, 0], train_features[train_labels == -1][:, 1], color=’blue’, label=’Orange’)

# Create a meshgrid for decision boundary
x_min, x_max = features[:, 0].min() - 10, features[:, 0].max() + 10
y_min, y_max = features[:, 1].min() - 10, features[:, 1].max() + 10
xx, yy = np.meshgrid(np.arange(x_min, x_max, 10), np.arange(y_min, y_max, 10))
Z = np.array([perceptron.predict(np.array([x, y, 0])) for x, y in zip(xx.ravel(), yy.ravel())])
Z = Z.reshape(xx.shape)

# Plot the decision boundary as a contour
```

```
plt.contourf(xx, yy, Z, colors=[’blue’, ’red’], alpha=0.2)

ax.set_title(’Decision␣Boundary␣and␣Data␣Points␣in␣2D␣Projection’)
ax.set_xlabel(’Feature␣1’)
ax.set_ylabel(’Feature␣2’)
ax.legend()

plt.show()
```

```
        OUTPUT

Apple
Apple
Apple
Orange
Apple
Apple
Apple
Orange
Apple
Orange
Orange
Apple
Orange
Apple
Apple
```





॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥