Міністерство освіти і науки України Державний університет «Одеська політехніка»

Кафедра прикладної математики

Лабораторна робота №1
з курсу “Нейроні мережі”
Варіант 1
Класифікація за допомогою персептрону

Виконала:
студентка 4 курсу
групи НАВ-191
Матиченко А.Д.

Прийняла:
Доктор техн. наук, доцент.
Полякова М.В.

Одеса 2022

Завдання для виконання роботи

1. Побудувати навчальну вибірку. Координати точок навчальної вибірки задані у табл. 1.1. Показати навчальну вибірку на графіку.
2. Побудувати персептрон для класифікації навчальної вибірки на 2 класи, використовуючи приклад 1.1.
3. Провести навчання персептрона на складеній навчальній вибірці.
4. Використовуючи навчений персептрон, зробити класифікацію індивідуального варіанта перевірочної множини. Навести графік результатів класифікації точок. Для графіка застосувати різні кольори для точок різних класів.

Згідно до варіанту:

Навчальна вибірка		Тестова вибірка
0.49 0.89 0	0.55 0.40 1	0.05 0.15
0.34 0.81 0	0.66 0.32 1	0.09 0.39
0.36 0.67 0	0.74 0.49 1	0.13 0.51
0.47 0.49 0	0.89 0.30 1	0.25 0.34
0.52 0.53 0	0.77 0.20 1	0.15 0.36

Код програми

1. Реалізуемо наш персептрон у фигляді класу:

```
import numpy as np


class Perceptron():
    def __init__(self, num_features = 2):
        self.num_features = num_features
        self.weights = np.zeros((num_features, 1), dtype=float)
        self.bias = np.zeros(1, dtype=np.float)

    def forward(self, x):
        linear = np.dot(x, self.weights)+self.bias
        predictions = np.where(linear > 0, 1, 0)
        return predictions

    def backward(self, x, y):
        predictions = self.forward(x)
        errors = y - predictions
        return errors

    def train(self, x, y, epochs=1000):
        for e in range(epochs):
            for i in range(y.shape[0]):
                errors = self.backward(x[i].reshape(1, self.num_features),
y[i]).reshape(-1)
                self.weights += (errors * x[i]).reshape(self.num_features,
1)
                self.bias += errors

    def evaluate(self, x, y):
        predictions = self.forward(x).reshape(-1)
        accuracy = np.sum(predictions == y)/  y.shape[0]
        return accuracy
```

2. Реалізуємо завдання згідно з варіантом:

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

#Dataset
data = np.genfromtxt('data/dataset.txt', delimiter=' ')
X_train, y_train = data[:, :2], data[:, 2]
y_train = y_train.astype(np.int)

#Graphic plotting
plt.scatter(X_train[y_train==0, 0], X_train[y_train==0, 1], label='class
0', marker='o')
plt.scatter(X_train[y_train==1, 0], X_train[y_train==1, 1], label='class
1', marker='s')
plt.title('Training set')
```

```
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.legend()
plt.grid()
plt.show()

#Training the Perceptron
ppn = prc.Perceptron()
ppn.train(X_train, y_train, epochs=50)
print("Parameters of model:\t")
print('Weights:%s\n' % ppn.weights)
print('Bias: %s\n' % ppn.bias)

#Evaluating the model
train_acc = ppn.evaluate(X_train, y_train)
print('Accuracy: %s' % (train_acc*100))


#Testing set
data = np.genfromtxt('data/train_set.txt', delimiter=' ')
```

3. Для реалізації класифікації скористуємося деякими математичними властивостями. А саме:

Якщо точка $x(x_0,...,x_n)$ лежить на гіперплощині, то $\omega^T x+b=0$

Гіперплощина ділить гіперпростір на два гіперпідпростори. Так ось точки, що знаходяться в одному з цих підпросторів (умовно кажучи «вище» гіперплощини), і точки, що знаходяться в іншому з цих підпросторів (умовно кажучи «нижче» гіперплощини), будуть у цій сумі давати різний знак:

$\omega^T x+b>0$ — точка лежить «вище» за гіперплощину

$\omega^T x+b<0$ — точка лежить «нижче» за гіперплощину

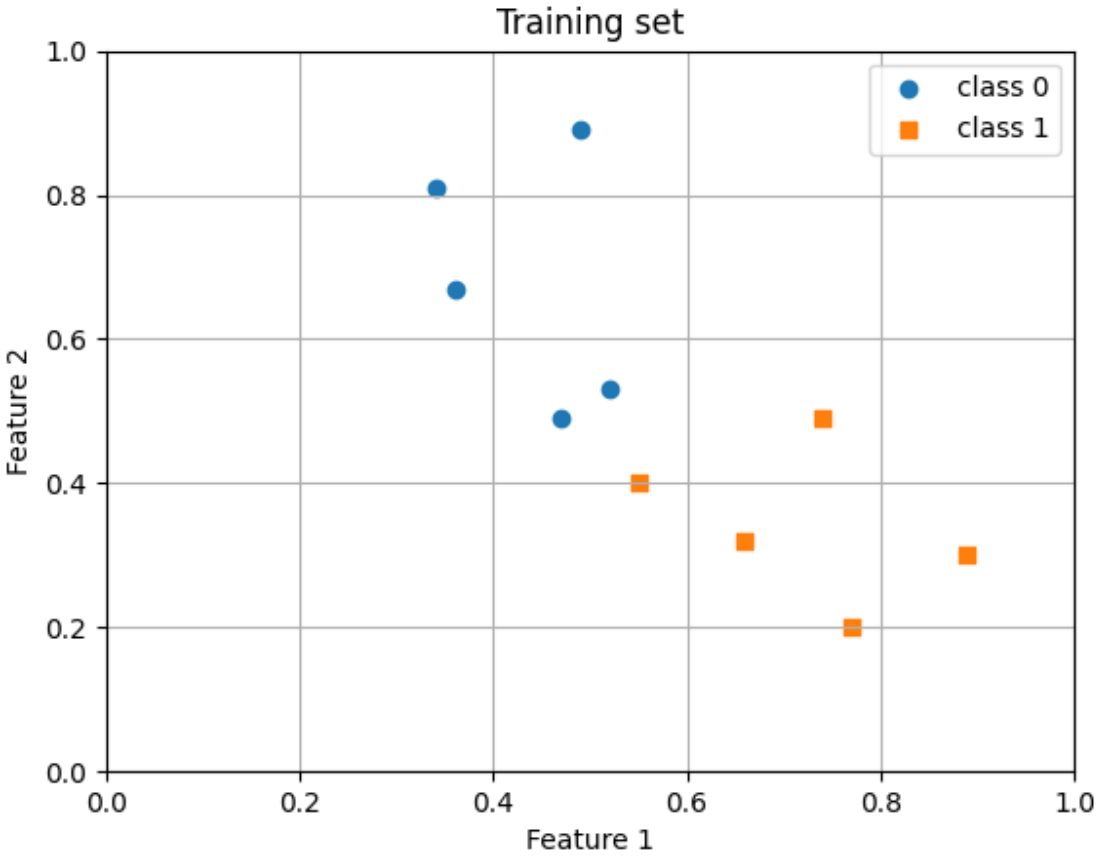
```
#Plotling training and testing sets
fig, ax = plt.subplots(1, 2, sharex=True, figsize=(7, 3))
x_min, x_max = 0, 1
ax[1].plot([x_min, x_max], [x_max, x_max])
ax[1].set_title('Testing set')
ax[1].get_legend()
ax[1].grid()
for x, y in data:
    if (np.dot((ppn.weights).T, np.array([x, y]))+ppn.bias) >0:
        ax[1].scatter(x, y, label='class 1', marker='s', color='orange')
    else:
        ax[1].scatter(x, y, label='class 0', marker='o', color='blue')


ax[0].plot([x_min, x_max], [x_max, x_max])
```

```
ax[0].set_title('Training set')
ax[0].get_legend()
ax[0].scatter(X_train[y_train == 0, 0], X_train[y_train == 0, 1],
label='class 0', marker='o', color='blue')
ax[0].scatter(X_train[y_train == 1, 0], X_train[y_train == 1, 1],
label='class 1', marker='s', color='orange')
ax[0].grid()

plt.show()
```

Результати роботи програми




```
Parameters of model:
Weights:[[ 4.63] [-6.11]]
Bias: [1.]
Accuracy: 80.0
```