МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. ІГОРЯ СІКОРСЬКОГО»

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №1

з дисципліни: “Нейронні мережі”

Виконали:

студенти групи КА-83

Самошин А.О.

Цепа О.Ю.

Перевірив:

Данилов В.Я.

КИЇВ 2020

Завдання

1. Реалізувати одношаровий персептрон, використовуючи такі мови програмування як Python, C++, Java.

2. За допомогою реалізованого персептрона розв’язати задачу згідно з номером варіанту. (Номер варіанту визначається за номером у списку групи.) Для цього на основі відповідного файлу необхідно випадковим чином сформувати навчальну та тестову вибірки (у співвідношенні 4:1). Навчити нейронну мережу на навчальній вибірці, використовуючи різні активаційні функції. Порівняти результати.

3. Перевірити роботу персептрона на тестових даних.

Мета: отримати навички розв’язання практичних задач за допомогою одношарового персептрона.

Розв’язання

Коли ми навчаємо керовану модельвона повинна якось фіксувати інформацію, яку ви їй надаєте. Персептрон робить це за допомогою вектора зважування, або **w**, який визначає точне положення межі рішення і дізнається з даних. Якщо ми вводимо нові дані у вхідному векторі **x**, нам просто доведеться точно визначити цей вектор стосовно вивчених ваг, щоб визначитися з класом. Математично це представлено так:

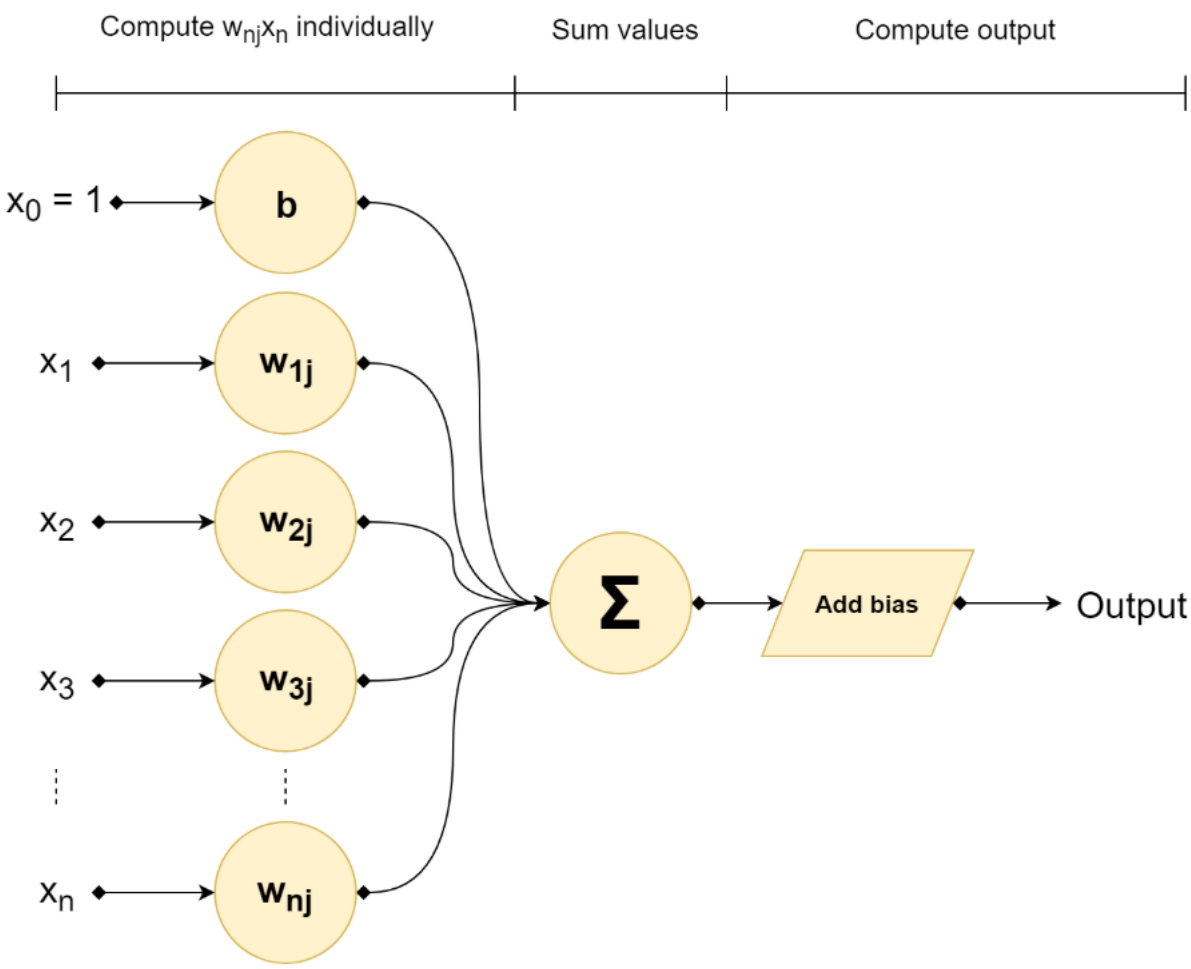
$$f(x) = \begin{cases} 1, & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0, & \text{otherwise} \end{cases}$$

Де $\mathbf{w} \cdot \mathbf{x} + b$ можна записати у вигляді:

$$\begin{aligned} z &= \sum_{i=1}^{n} w_n x_n + b \\ &= w_1 x_1 + \ldots + w_n x_n + b \end{aligned}$$

Також додаємо вектор зміщення.

$$\begin{aligned} z &= \sum_{i=0}^{n} w_n x_n \\ &= w_0 x_0 + w_1 x_1 + \ldots + w_n x_n \\ &= w_0 x_0 + w_1 x_1 + \ldots + w_n x_n \\ &= 1b + w_1 x_1 + \ldots + w_n x_n \\ &= w_1 x_1 + \ldots + w_n x_n + b \end{aligned}$$



Для тренування моделі використовуємо навчання з учителем.

1. Існує вектор ваг, який на початку ітерацій не ініціалізований.
2. Маємо набір значень $T = \{(x_1, d_1), (x_2, d_2), \ldots, (x_n, d_n)\}$
3. Навчальну та тестову вибірки формуємо випадковим чином з даного набору значень у співвідношенні 4:1.
4. Вводимо вектор зсуву.
5. Налаштовуємо коефіцієнт навчання (число від 0 до 1).

Ваги оновлюються для кожного зразка з набору даних. Цей процес може повторюватися до тих пір, поки не буде досягнуто якогось критерію, наприклад, певної кількості помилок, або повної конвергенції (тобто кількість помилок дорівнює 0).

Лістинг програми

```
import pandas as pd
import numpy as np
import random as rm
import math
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

data = pd.read_csv('data23.csv', names=['info'])
```

```
res = []
s = []

for i in range(0, 100):
    strk = data['info'][i]
    strk = strk.split(';')

    s.append([])
    s[i].append(strk[0])
    s[i].append(strk[1])
    s[i].append(strk[2])

rm.shuffle(s)

df = []
res = []
for i in range(0,100):
    df.append([])
    df[i].append(s[i][0])
    df[i].append(s[i][1])
    res.append(s[i][2])

df = np.array(df).astype(float)
res = np.array(res).astype(int)

X_train, X_test, y_train, y_test = train_test_split(df, res, test_size=0.2, random_state=0)

def lin(outcome):
    return outcome

def binary(outcome):
    return np.where(outcome > 0, 1, 0)

def relu(outcome):
    return np.where(outcome > 0, outcome, 0)

def sigmoid1(outcome):
    return 1/(math.exp(-outcome) + 1)

def erf(outcome):
    return math.tanh(outcome)

def bent(outcome):
    return (math.sqrt(outcome ** 2 + 1) - 1)/2 + outcome


class RBPerceptron:



  def __init__(self, number_of_epochs = 100, learning_rate = 0.1):
    self.number_of_epochs = number_of_epochs
```

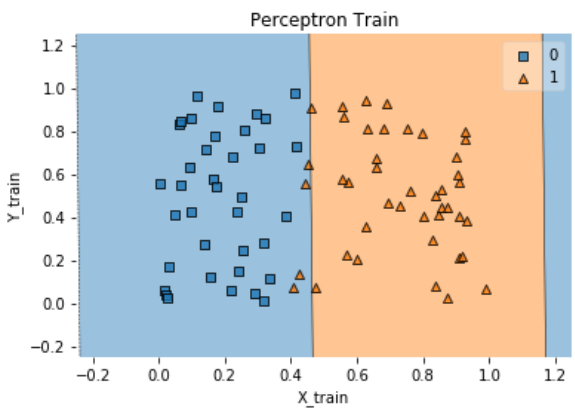
```
    self.learning_rate = learning_rate

  def train(self, X, D):
    # Initialize weights vector with zeroes
    num_features = X.shape[1]
    err = []
    self.w = np.zeros(num_features + 1)
    # Perform the epochs
    for i in range(self.number_of_epochs):
      # For every combination of (X_i, D_i)
      aver = 0
      iter = 0
      for sample, desired_outcome in zip(X, D):
        # Generate prediction and compare with desired outcome
        prediction    = self.predict(sample)
        difference    = (desired_outcome - prediction)
        aver = aver + difference
        # Compute weight update via Perceptron Learning Rule
        weight_update = self.learning_rate * difference
        self.w[1:]    += weight_update * sample
        self.w[0]     += weight_update
        iter = iter + 1
      err.append(math.fabs(aver)/ iter)
    #print(err)
    plt.plot([x for x in range(self.number_of_epochs)], err)
    plt.title('Errors of epochs')
    plt.xlabel('Epochs')
    plt.ylabel('Error')
    plt.show()
    print ('Weights vector: w1 =', round(self.w[1], 4), 'w2 =', round(self.w[2], 4), 'b =',
round(self.w[0], 4))
    return self


  # Generate prediction
  def predict(self, sample):
    outcome = np.dot(sample, self.w[1:]) + self.w[0]
    func_vect = np.vectorize(relu)
    return func_vect(outcome)

rbp = RBPerceptron(800, 0.05)

train_model = rbp.train(X_train, y_train)

from mlxtend.plotting import plot_decision_regions

plot_decision_regions(X_train, y_train, clf=train_model)
plt.title('Perceptron Train')
plt.xlabel('X_train')
plt.ylabel('Y_train')
plt.xlim(-0.25,1.25)
plt.ylim(-0.25,1.25)
```

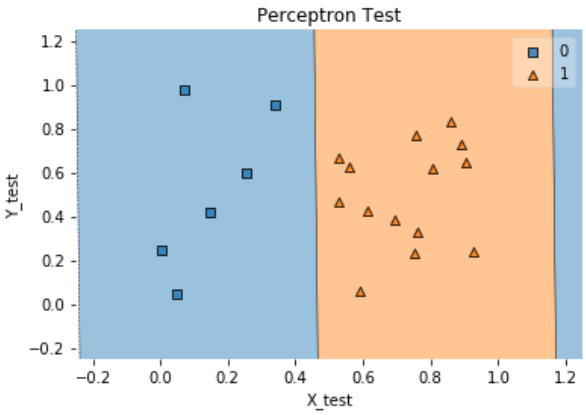
```
plt.show()

plot_decision_regions(X_test, y_test, clf=train_model)
plt.title('Perceptron Test')
plt.xlabel('X_test')
plt.ylabel('Y_test')
plt.xlim(-0.25,1.25)
plt.ylim(-0.25,1.25)
plt.show()
```

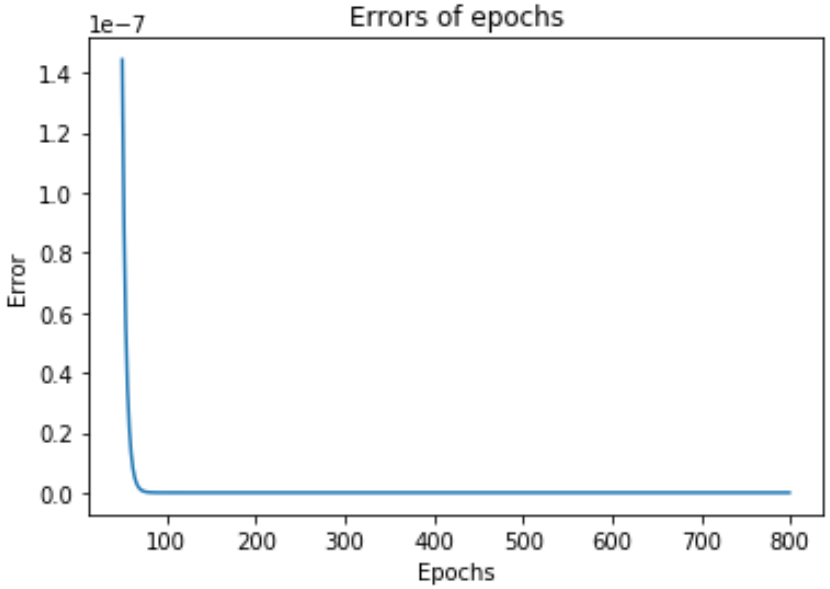
Результати роботи програми

Лінійна функція:

Weights vector: w1 = 1.4194 w2 = 0.0099 b = -0.1595

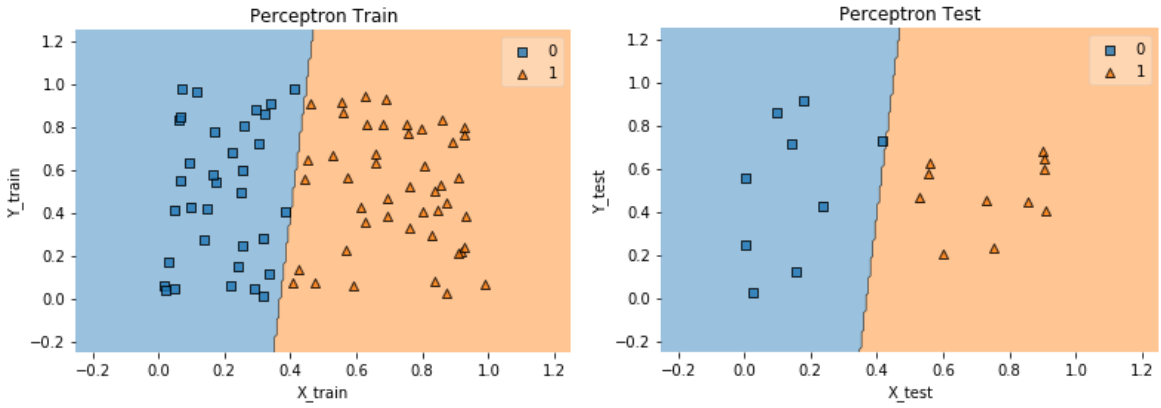


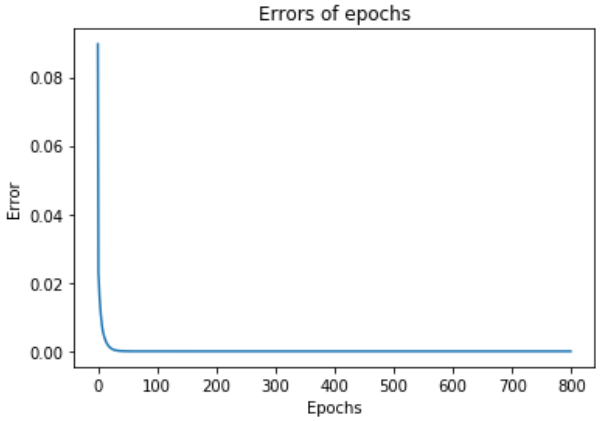




Функція Гефісайда:

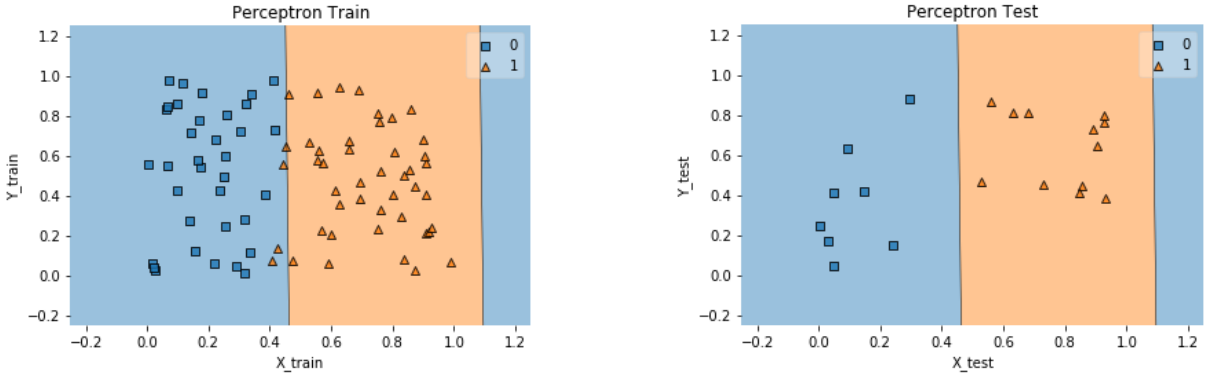
Weights vector: w1 = 0.135 w2 = -0.0147 b = -0.05

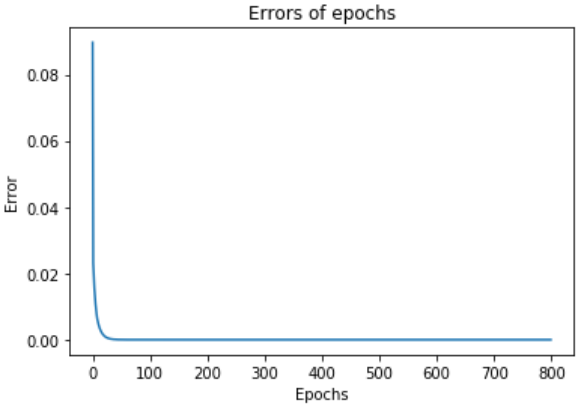




Функція ReLu:

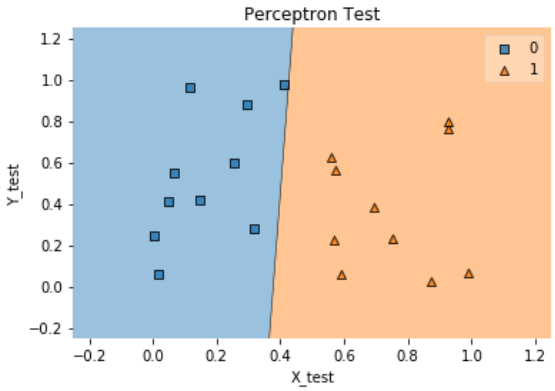
Weights vector: w1 = 1.5793 w2 = 0.0124 b = -0.2279

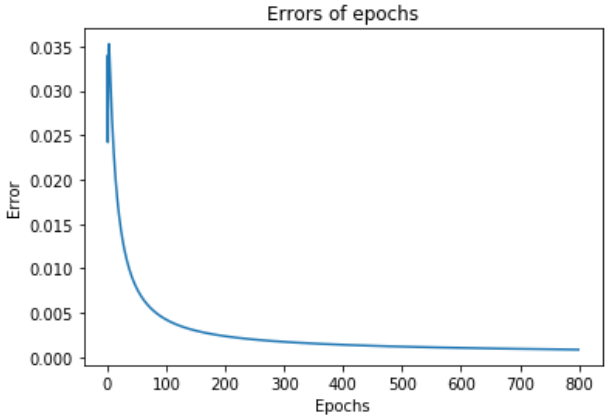




Функція Сігмоїда:

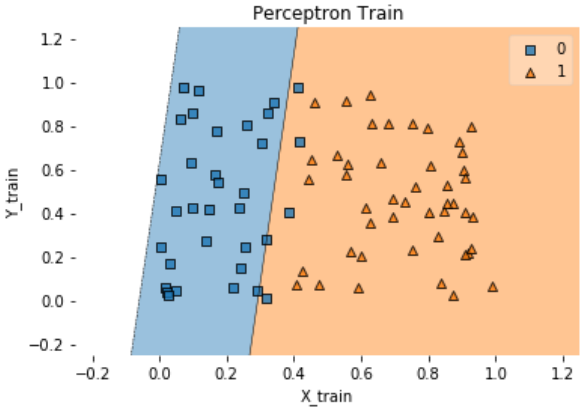
Weights vector: w1 = 22.8233 w2 = -1.1257 b = -8.62

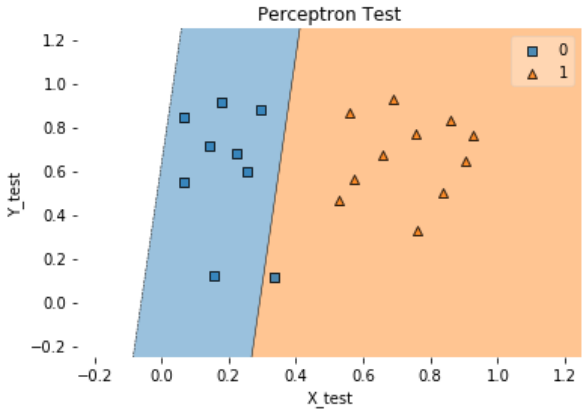


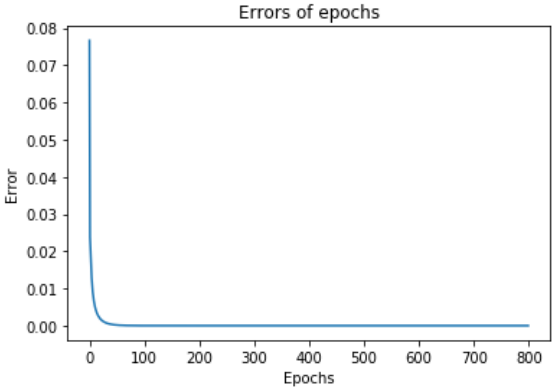


Функція тангенс гіперболічний:

Weights vector: w1 = 3.1182 w2 = -0.2978 b = -0.3617

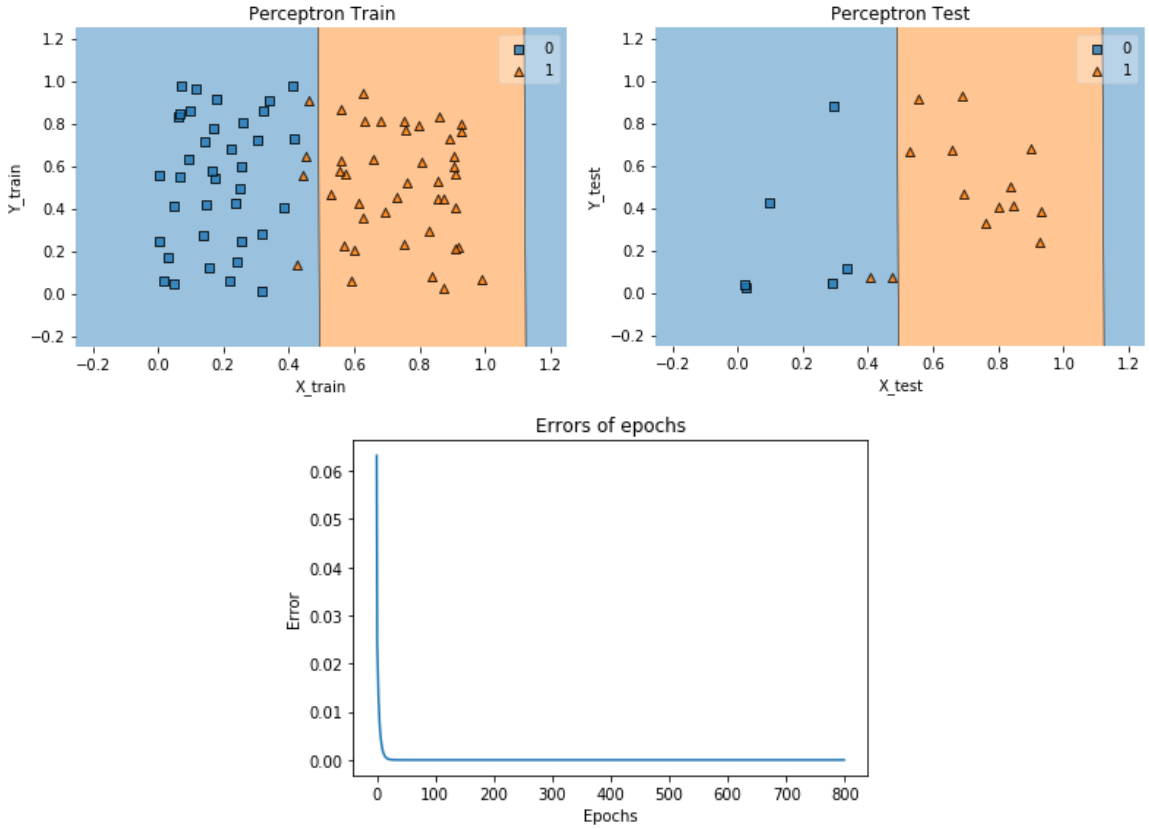






Функція ввігнута тотожня:

Weights vector: w1 = 1.2094 w2 = 0.004 b = -0.145



Висновки

У даній лабораторній роботі я навчився працювати з одношаровим персептроном, використовувати різні активаційні функції та застосовувати отримані навички на практиці.