**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 1**

по курсу «Нейроинформатика».

Тема: «Персептроны. Процедура обучения Розенблатта».

Студент: Якимович А.И.

Группа: 80-408Б

Вариант: 20

Оценка:

Москва, 2018

Постановка задачи.

Целью работы является исследование свойств персептрона Розенблатта и его применение для решения задачи распознавания образов.

Описание алгоритма.

Сначала происходит инициализация весов начальными значениями (у меня это нули). Далее для первого примера вычисляется вектор ошибки. Ощибка вычисляется по следующей формуле , где – истинный результат для i-ого примера, а - предсказанное значение сети для i-ого примера (x[i] – i-ый вектор входов, W – текущая матрица весов, b – текущий вектор смещений). Если вектор ошибки ненулевой, проиходит корректировка весов пропорционально компонентам вектора ошибки и все начинается сначала, но уже с новыми весами (). Если же ветор ошибки нулевой, то программа переходит к следующему примеру. Программа завершает свою работу, когда дошла до последнего примера и вычислила на нем нулевую ошибку. Предсказаный ответ представляет собой значение функции Хевисайда от матричного произведения матрицы весов и значений входов.

Ход работы.

1. Реализовать вышеописанный алгоритм (функции train и predict)
2. Подать значения из варианта на вход алгоритма.
3. Отобразить дискриминантную линию.
4. Проверить качество обучения.
5. Изменить обучающее множество так, чтобы классы стали линейно неразделимыми.
6. Убедиться в неспособности решения данной задачи однослойным перцептроном Розенблатта.

Общая информация.

Программа написана на языке Python с применением библиотек numpy (для вычислений) и matplotlib (для графиков).

Данная программа может решать задачи, разрешимые однослойным перцептроном Розенблата. На линейно-неразделимых задачах программа не зацикливается благодаря счетчику итераций, если его значение превышает некоторое заранее выбранное, то программа сообщает пользователю, что задача линейно неразрешима.

Запуск программы.

Чтобы воспользоваться программой, необходимо запустить программу на любом интерпретаторе Python версии 3.x.x.

Входные данные и результаты.

Вариант 20.

|  |  |
| --- | --- |
| **Вход** | **Выход** |
| **В соответствии с вариантом**  [[4.3, 2.3, 3.6, 4.8, 2.8, -3.3],  [2.2, -4.4, 4.3, 3.5, 0.1, -1.1]]  [[-4.4, 0.2, 1.5, -2.1, -4.9, -3.4, -1.3, -0.2], [ -1.1, -0.9, 1.2, -0.7, 4.8, -4, -3.1, -1.7]]  [1, 0, 1, 1, 1, 0]  [[1, 0, 0, 0, 0, 1, 0, 0],  [0, 0, 1, 0, 1, 0, 0, 0]] | **В соответствии с вариантом**  (array([0.2 , 0.66]), 0.0)  (array([[-0.19, -0.05],  [-0.05, 0.59]]),  array([-0.8, 0. ])) C:\Users\now20\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.png |
| **Изменен первый набор**  [[4.3, 2.3, 3.6, 4.8, 2.8, -3.3],  [2.2, -4.4, 4.3, 3.5, 0.1, -1.1]]  [[-4.4, 0.2, 1.5, -2.1, -4.9, -3.4, -1.3, -0.2], [ -1.1, -0.9, 1.2, -0.7, 4.8, -4, -3.1, -1.7]]  [0, 0, 1, 1, 1, 0]  [[1, 0, 0, 0, 0, 1, 0, 0],  [0, 0, 1, 0, 1, 0, 0, 0]] | **Изменен первый набор**  Error! Task is not linearly solvable. **C:\Users\now20\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.png** |
| **Изменен второй набор**  [[4.3, 2.3, 3.6, 4.8, 2.8, -3.3],  [2.2, -4.4, 4.3, 3.5, 0.1, -1.1]]  [[-4.4, 0.2, 1.5, -2.1, -4.9, -3.4, -1.3, -0.2], [ -1.1, -0.9, 1.2, -0.7, 4.8, -4, -3.1, -1.7]]  [1, 0, 1, 1, 1, 0]  [[1, 0, 1, 0, 0, 1, 0, 0],  [0, 0, 1, 0, 1, 1, 0, 0]] | **Изменен второй набор**  (array([0.2 , 0.66]), 0.0)  Error! Task is not linearly solvable. |
| **Проверка предсказания**  [[1, 2, 3, -1, -2, -3],  [4, -4, 2, -2, 0, 0]]  [[1, 1, 2, 2, 3, 4, -4, -3],  [4, -4, 2, -2, -3, 3, -4, 3]] | **Проверка предсказания**  (array([0.2 , 0.66]), 0.0)  (array([[-0.19, -0.05],  [-0.05, 0.59]]), array([-0.8, 0. ])) |

Выводы:

Выполнив лабораторную работу, я научился реализовывать однослойный перцептрон Розенблатта, научился обучать его и предсказывать классы объектов, строить дискриминантные линии, выяснил для каких задач такой перцептрон не подходит.

Программа работает правильно, прошла все тесты, и ее можно легко применять для задач любой размерности без существенного переписывания кода.

Исходный код.

import numpy as np

import matplotlib.pyplot as plt

import sys

samples1 = np.array([[4.3, 2.3, 3.6, 4.8, 2.8, -3.3],

[2.2, -4.4, 4.3, 3.5, 0.1, -1.1]]).T

samples2 = np.array([[-4.4, 0.2, 1.5, -2.1, -4.9, -3.4, -1.3, -0.2],

[ -1.1, -0.9, 1.2, -0.7, 4.8, -4, -3.1, -1.7]]).T

answers1 = np.array([1, 0, 1, 1, 1, 0])

answers2 = np.array([[1, 0, 0, 0, 0, 1, 0, 0],

[0, 0, 1, 0, 1, 0, 0, 0]]).T

w1 = np.array([0, 0], float)

w2 = np.array([[0, 0], [0, 0]], float)

b1 = 0

b2 = np.array([0, 0], float)

learn\_rate = 0.1

max\_iterations = 10000

def f(x, w, b):

return x.dot(w) + b

def train(samples, answers, w, b, learn\_rate, flag, max\_i):

i = 0

while i < samples.T[0].size:

err = answers[i] - np.heaviside(f(samples[i], w, b), 1)

if np.linalg.norm(err) != 0:

if(flag == 1):

w += learn\_rate \* samples[i].reshape(-1, 1).dot(err.reshape(1, -1))

else:

w += learn\_rate \* samples[i] \* err

b += learn\_rate \* err

i = -1

i += 1

max\_i -= 1

if max\_i == 0:

sys.exit("Error! Task is not linearly solvable.")

return w, b

def predict(samples, w, b, answers):

i = 0

while i < samples.T[0].size:

answers[i] = np.heaviside(f(samples[i], w, b), 1)

i += 1

plt.subplot(211)

#sctr1 = plt.scatter(x=samples1.T[0], y=samples1.T[1], s = 100, c=answers1, cmap=plt.cm.RdYlGn)

print(train(samples1, answers1, w1, b1, learn\_rate, 0, max\_iterations))

s1 = np.array([[1, 2, 3, -1, -2, -3], [4, -4, 2, -2, 0, 0]]).T

ans1 = np.array([0, 0, 0, 0, 0, 0])

predict(s1, w1, b1, ans1)

sctr1 = plt.scatter(x=s1.T[0], y=s1.T[1], s = 100, c=ans1, cmap=plt.cm.RdYlGn)

x1 = np.linspace(-5, 5, 100)

x2 = (-x1 \* w1[0] - b1) / w1[1]

plt.plot(x1, x2, color = 'black')

plt.subplot(212)

#sctr1 = plt.scatter(x=samples2.T[0], y=samples2.T[1], s = 100, c=answers2.T[0], cmap=plt.cm.RdYlGn)

#sctr2 = plt.scatter(x=samples2.T[0], y=samples2.T[1], s = 10, c=answers2.T[1], cmap=plt.cm.RdYlGn)

s2 = np.array([[1, 1, 2, 2, 3, 4, -4, -3], [4, -4, 2, -2, -3, 3, -4, 3]]).T

ans2 = np.array([[0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0]]).T

print(train(samples2, answers2, w2, b2, learn\_rate, 1, max\_iterations))

predict(s2, w2, b2, ans2)

sctr1 = plt.scatter(x=s2.T[0], y=s2.T[1], s = 100, c=ans2.T[0], cmap=plt.cm.RdYlGn)

sctr2 = plt.scatter(x=s2.T[0], y=s2.T[1], s = 10, c=ans2.T[1], cmap=plt.cm.RdYlGn)

x1 = np.linspace(-5, 5, 100)

x2 = (-x1 \* w2[0][0] - b2[0]) / w2[1][0]

x4 = (-x1 \* w2[0][1] - b2[1]) / w2[1][1]

plt.plot(x1, x2, color = 'black')

plt.plot(x1, x4, color = 'gray')

plt.show()