МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №1**

**по курсу «Нейроинформатика»**

Выполнил: Кузьмичев А. Н.

Группа: 8О-406Б

Преподаватели: Н. П. Аносова

Москва, 2021

**Персептроны. Процедура обучения Розенблатта.**

*Цель работы:* исследование свойств персептрона Розенблатта и его применение для решения задачи распознавания образов

*Этапы работы:*

1. Для первой обучающей выборки построить и обучить сеть, которая будет правильно относить точки к двум классам. Отобразить дискриминантную линию и проверить качество обучения.

2. Изменить обучающее множество так, чтобы классы стали линейно неразделимыми. Проверить возможности обучения по правилу Розенблатта.

3. Для второй обучающей выборки построить и обучить сеть, которая будет правильно относить точки к четырем классам. Отобразить дискриминантную линию и проверить качество обучения.

*Вариант 12:*

1. [(2,7; 4, 3), (-3,8; 0,6), (-0,4; -4,9), (-1,7; -3,4), (2,9; -1,9), (0,2; -3,4)]

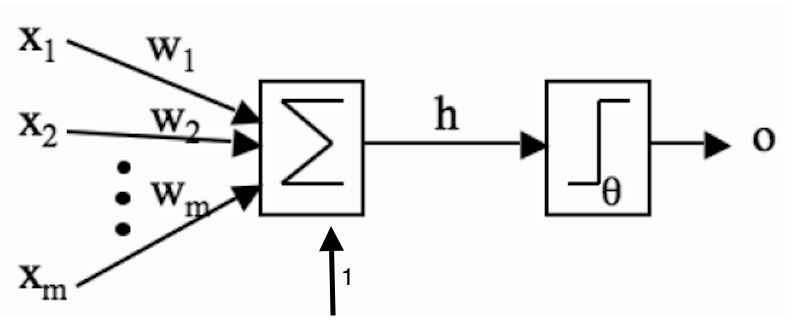
[0, 0, 1, 1, 1, 1]

1. [(-1,5; -0,6), (4,6; -4,6), (4,7; -3,2), (1,6; 0,8), (1,7; -1,4), (1,2; 3,1), (-4,9; -4,2), (4,7; 1,5)]

[(0;0), (0;1), (0;1), (1;0), (0;0), (1;0), (0;1), (1;1)]

**Ход работы**

Для решения этой задачи необходимо воспользоваться Перцептроном Розенблата, который имеет следующую структуру:



Чтобы реализовать слой таких перцептронов можно воспользоваться представлением весов и смещений перцептронов как матрицу (n+1)×m, где n - число входов, а m - число выходов. При этом в качестве выходов я использую функцию net, а ошибку измеряю помощи метрики MAE. Для удобства классификации и обучения, я заменяю метки негативных классов с 0 на −1. Реализация слоя из перцептронов Розенблата:

Реализация слоя из персептрона Розенблатта:

class RosenblattLayer:

    def \_\_init\_\_(self, steps = 50, early\_stop = False):

        self.steps = steps

        self.w = None

        self.negative\_is\_zero = False

        self.early\_stop = early\_stop

    def fit(self, X, y):

        # add column for bias and transpose data for comphort operations:

        X\_t = np.append(X, np.ones((1, X.shape[1])), axis = 0).T

        y\_t = np.array(y.T)

        if (y\_t == 0).sum():

            self.negative\_is\_zero = True

            y\_t[y\_t == 0] = -1

        #init weights

        if self.w is None:

            self.w = np.random.random((X\_t.shape[1], y\_t.shape[1]))

        # main loop

        for step in tqdm(range(self.steps)):

            stop = True

            for i in range(X\_t.shape[0]):

                # compute error of all perceptrons

                predict = X\_t[i].dot(self.w)

                if np.sum(predict\*y\_t[i] < 0):

                    stop = False

                    e = y\_t[i] - X\_t[i].dot(self.w)

                    e[predict\*y\_t[i] >= 0] = 0.0

                    self.w += X\_t[i].reshape(X\_t.shape[1], 1).dot(e.reshape(1, y\_t.shape[1]))

            if stop and self.early\_stop:

                break

        return self

    def set\_steps(self, steps):

        self.steps = steps

    def set\_early\_stop(self, stop):

        self.early\_stop = stop

    # Predict answers

    def predict(self, X):

        X\_t = np.append(X, np.ones((1, X.shape[1])), axis = 0).T

        return X\_t.dot(self.w).T

    def predict\_classes(self, X):

        X\_t = np.append(X, np.ones((1, X.shape[1])), axis = 0).T

        ans = X\_t.dot(self.w)

        a\_t =  ans < 0

        if self.negative\_is\_zero:

            ans[a\_t] = 0

        else:

            ans[a\_t] = -1

        ans[np.logical\_not(a\_t)] = 1

        return ans.T

    def display(self):

        ans = " Input(n," + str(self.w.shape[0] - 1) + ")  -->  "

        ans += "Rosenblat Perceptrons(" + str(self.w.shape[1]) + ")  -->  "

        ans += "Output(n, " + str(self.w.shape[1]) + ")"

        return ans

    def weights(self):

        return self.w[:-1]

    def bias(self):

        return self.w[-1]

    # MAE

    def score(self, X, y):

        X\_t = np.append(X, np.ones((1, X.shape[1])), axis = 0).T

        y\_t = np.array(y.T)

        y\_t[y\_t == 0] = -1

        return np.abs(y\_t - X\_t.dot(self.w)).mean()

Затем я создал модель со случайными коэффициентами для первых обучающих данных и вывел ее схему на экран:

P = np.array([

    [2.7, -3.8, -0.4, -1.7, 2.9, 0.2],

    [4.3, 0.6, -4.9, -3.4, -1.9, -3.4]

])

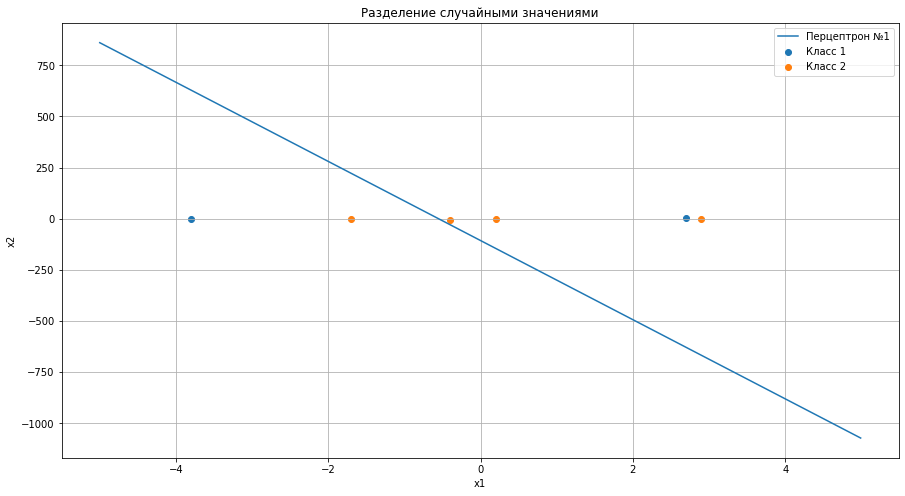
T = np.array([[0, 0, 1, 1, 1, 1]])

model = RosenblattLayer(0).fit(P, T)

print(model.display())

0it [00:00, ?it/s] Input(n,2) --> Rosenblat Perceptrons(1) --> Output(n, 1)

Модель со случайными коэффицентами разделяет выборку следующим образом:



Я обучил модель и построил разделяющую прямую, добавив 3 случайные точки и классифицировав их:

model.set\_steps(50)

model.set\_early\_stop(True)

model.fit(P, T)

plt.figure(figsize=(15, 8))

plot\_line(model.weights(), model.bias())

plt.scatter(P.T[T.ravel() == 0].T[0], P.T[T.ravel() == 0].T[1], label="Класс 1")

plt.scatter(P.T[T.ravel() == 1].T[0], P.T[T.ravel() == 1].T[1], label="Класс 2")

x = np.random.randint(-5, 4, 3) + np.random.random(3)

y = np.random.randint(-5, 4, 3) + np.random.random(3)

plt.scatter(x, y, color="red", label="3 случайные точки")

plt.xlabel("x1")

plt.ylabel("x2")

plt.title("Разделение выборки обученным перцептроном")

plt.grid()

plt.legend()

test = np.append(x.reshape(1, 3), y.reshape(1, 3), axis=0)

print("Test Data:")

print(test)

print("Predict:")

print(model.predict\_classes(test))

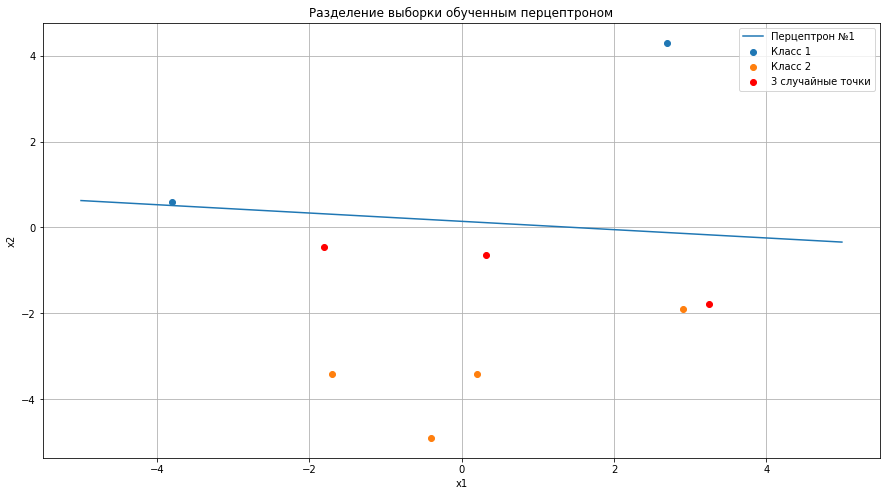
Test Data:

[[ 0.32279564 -1.80453877 3.24492738]

[-0.63238154 -0.4657158 -1.79242583]]

Predict:

[[1. 1. 1.]]



Полученная модель обучилась классифицировать данные:

model.predict\_classes(P)

array([[0., 0., 1., 1., 1., 1.]])

Веса модели:

print("Веса:")

print(model.weights())

print("Смещения:")

print(model.bias())

Веса:

[[ -23.02834153]

[-238.06899665]]

Смещения:

[33.94086248]

Добавив в датасет точку так, что выборка стала линейно неразделима, я увидел что даже после нескольких циклов обучения модель все еще не научилась разделять данные, выдавая ужасный результат классификации:

P = np.append(P, np.array([[0.2], [3.4]]), axis=1)

T = np.append(T, np.array([[1]]), axis=1)

model = RosenblattLayer(50, True)

model.fit(P, T)

plt.figure(figsize=(15, 8))

plot\_line(model.weights(), model.bias())

plt.scatter(P.T[T.ravel() == 0].T[0], P.T[T.ravel() == 0].T[1], label="Класс 1")

plt.scatter(P.T[T.ravel() == 1].T[0], P.T[T.ravel() == 1].T[1], label="Класс 2")

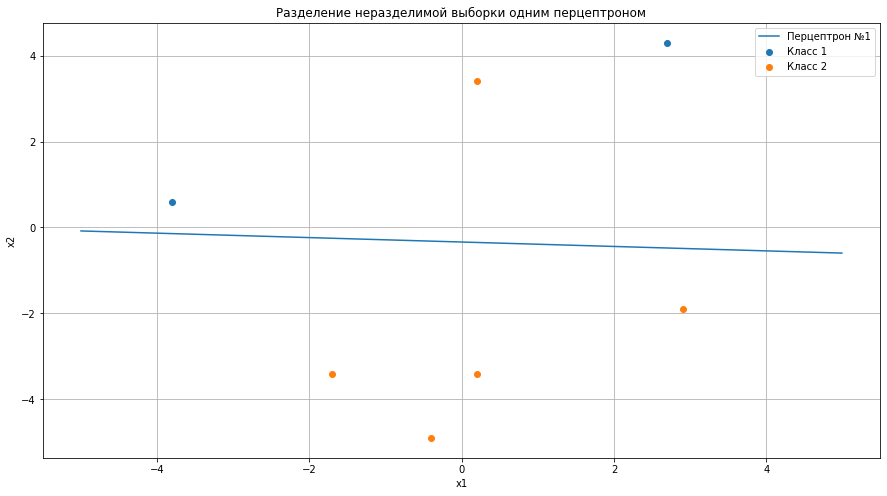
plt.xlabel("x1")

plt.ylabel("x2")

plt.title("Разделение неразделимой выборки одним перцептроном")

plt.grid()

plt.legend()



model.predict\_classes(P) == T

array([[False, False, False, False, False, False, True]])

Для того, чтобы классифицировать данные на несколько классов, недостаточно одного перцептрона, а необходим целый слой. Так как мой класс реализует целый слой, интерфейс работы никак не поменялся. Создание и обучение сети с последующей классификацией 5 случайных точек:

P = np.array([

    [-1.5, 4.6, 4.7, 1.6, 1.7, 1.2, -4.9, 4.7],

    [-0.6, -4.6, -3.2, 0.8, -1.4, 3.1, -4.2, 1.5]

])

T = np.array([

    [0, 0, 0, 1, 0, 1, 0, 1],

    [0, 1, 1, 0, 0, 0, 1, 1]

])

model = RosenblattLayer(0).fit(P, T)

print(model.display())

0it [00:00, ?it/s] Input(n,2) --> Rosenblat Perceptrons(2) --> Output(n, 2)

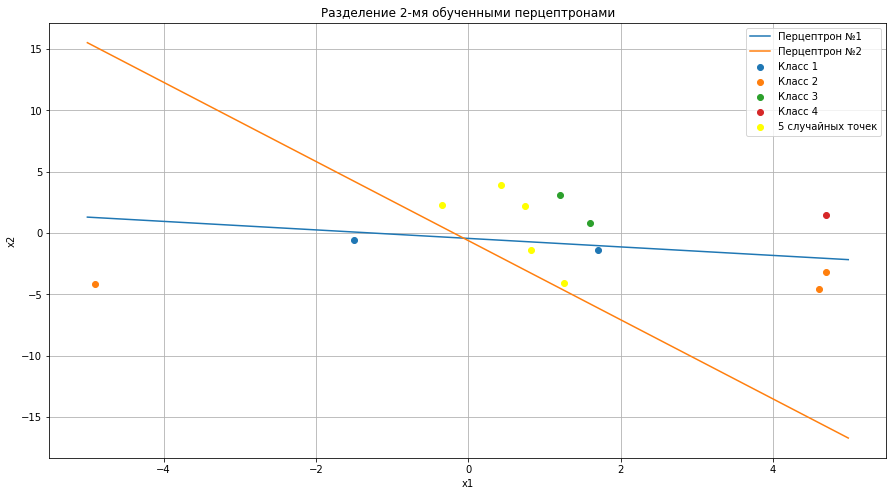
import numpy as np

np.seterr(divide='ignore', invalid='ignore')

model.set\_steps(50)

model.set\_early\_stop(True)

model.fit(P, T)



Веса:

print("Веса:")

print(model.weights())

print("Смещения:")

print(model.bias())

Веса:

[[6.78928959e+00 7.00305513e+91]

[1.95760726e+01 2.17117716e+91]]

Смещения:

[8.67319022e+00 1.34449309e+91]

**Выводы**

Выполнив первую лабораторную работу по курсу «Нейроинформатика», я узнал о перцептроне Розенблата, который имеет историческое значение для всей науки. Перцептрон Розенблата несмотря на свою простоту и умение классифицировать объекты, имеет ряд очевидных недостатков, таких, как:

• Невозможность классификации на неразделяемых данных.

• Неустойчивость.

Эти недостатки решают более современные архитектуры, с которыми мне предстоит познакомиться. Эта работа была одной из самых простых для выполнения, однако это только вводная работа, и в будущем я ожидаю более трудных и интересных задач.