Санкт-Петербургский государственный морской технический университет

Отчёт

по практической работе №1

“Алгоритм обучения персептрона”

Выполнил: Студент 1 курса

Факультета Цифровых

Промышленных Технологий

группы 20221 Хохлов Д.Р.

Преподаватель:

Кафедра киберфизических систем

Кайнова Татьяна Денисовна

Санкт-Петербург

2024

**Введение**

Цель работы: доработка программы для реализации демонстрации алгоритма обучения персептрона.

Задача: создании модели, способной предсказывать зависимость на основе предоставленных данных.Вывод графического отображения процесса обучения и итогового результата классификации

**Основная часть**

Для выполнения задания было доработанная программа, обучающая персептрон на данных для предсказания зависимости .

Основные изменения программы включают:

* Обучение персептрона на предоставленных входных данных (Таблица 1)
* Добавление функции вывода графика после обучения (Рисунок 2)
* Добавление функции вывода графика во время обучения (Рисунок 1)

Таблица 1 – Данные для обучения

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x1 | x2 | | | | | | |
|  | -1.7 | -0.8 | 0.7 | 0.8 | 1.2 | 1.7 |
| 1.9 |  |  |  |  |  |  |
| 0.5 |  |  |  |  |  |  |
| -0.6 |  |  |  |  |  |  |
| -0.8 |  |  |  |  |  |  |
| -1.2 |  |  |  |  |  |  |
| -1.9 |  |  |  |  |  |  |

Были добавлены следующие функции:

* draw\_line() - чертит линию на графике, используя формулу
* show\_training() - создает график после финального обучения персептрона, использует функцию draw\_line()
* show\_all\_training() - создает график всего обучения персептрона и показывает как менялся результат, использует функцию draw\_line()

Результатом является график всего обучения(Рисунок 1) нейросети и финальный график с правильным разделением (Рисунок 2)

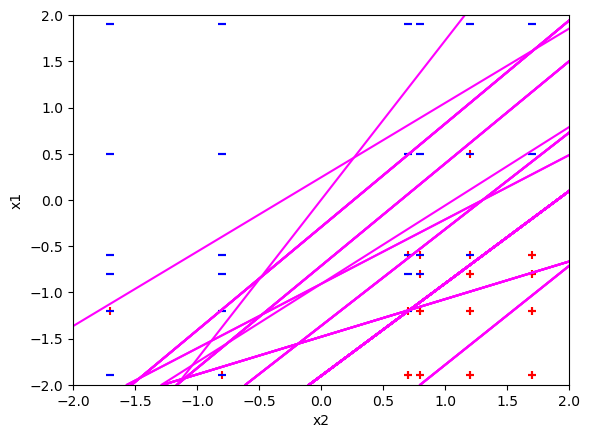


Рисунок 1 - График всего обучения

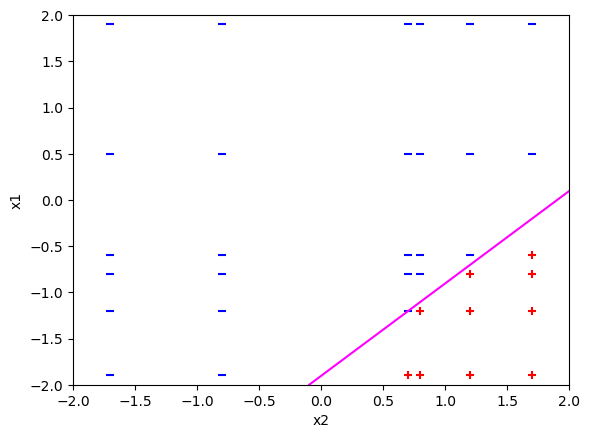


Рисунок 2 - Финальный график

**Заключение**

В ходе выполнения практической работы была успешно реализована модель обучения персептрона. Алгоритм продемонстрировал успешную работу в решении задачи классификации данных на основе заданной зависимости и предоставленных данных. Программа корректно обучалась, что подтверждается графиками, представленными на предыдущей странице.

**Приложение А**

**Листинг кода программы**

**#--УСТАНАВЛИВАЕМ БИБЛИОТЕКИ–#**

**!pip install matplotlib**

**#--ИМПОРТИРУЕМ БИБЛИОТЕКИ –#**

**import random**

**import matplotlib.pyplot as plt**

**def show\_learning(w): # Функция для отображения в консоли весов**

**print('w0 =', '%5.2f' % w[0], ', w1 =', '%5.2f' % w[1], ', w2 =', '%5.2f' % w[2])**

**#-- ОПРЕДЕЛЯЕМ ПЕРЕМЕННЫЕ –#**

**random.seed(7) # Чтобы обеспечить повторяемость**

**LEARNING\_RATE = 0.1**

**# Определяем обучающие примеры**

**x\_train = [(1.0,1.9, -1.7),(1.0, 1.9, -0.8),(1.0, 1.9, 0.7),(1.0, 1.9, 0.8),(1.0, 1.9,1.2),(1.0, 1.9, 1.7),**

**(1.0,0.5, -1.7),(1.0, 0.5, -0.8),(1.0,0.5,0.7),(1.0,0.5,0.8),(1.0,0.5,1.2),(1.0,0.5,1.7),**

**(1.0,-0.6, -1.7),(1.0,-0.6,-0.8),(1.0,-0.6,0.7),(1.0,-0.6,0.8),(1.0,-0.6,1.2),(1.0,-0.6,1.7),**

**(1.0,-0.8,-1.7),(1.0,-0.8,-0.8),(1.0,-0.8,0.7),(1.0,-0.8,0.8),(1.0,-0.8,1.2),(1.0,-0.8, 1.7),**

**(1.0,-1.2,-1.7),(1.0,-1.2,-0.8),(1.0,-1.2,0.7),(1.0,-1.2,0.8),(1.0,-1.2,1.2),(1.0,-1.2, 1.7),**

**(1.0,-1.9,-1.7),(1.0,-1.9,-0.8),(1.0,-1.9,0.7),(1.0,-1.9,0.8),(1.0,-1.9,1.2),(1.0, -1.9, 1.7)] # Входы**

**y\_train = [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0,**

**-1.0, -1.0, -1.0, -1.0, -1.0, -1.0,**

**-1.0, -1.0, -1.0, -1.0, -1.0, 1.0,**

**-1.0, -1.0, -1.0, -1.0, 1.0, 1.0,**

**-1.0, -1.0, -1.0, 1.0, 1.0, 1.0,**

**-1.0, -1.0, 1.0, 1.0, 1.0, 1.0] # Выход (истина)**

**index\_list = list(range(len(y\_train))) # Чтобы сделать порядок случайным**

**w = [0.2, -0.6, 0.25] # Определяем веса персептрона**

**all\_data = {**

**"x":[], # Входные данные**

**"y":[], # Ответы персептрона**

**"w":[] # Веса**

**} # Словарь для хранения всех результатов персептрона**

**show\_learning(w) # Печатаем начальные значения весов**

**# – ФУНКЦИИ – #**

**def compute\_output(w, x): # Функция вычислений персептрона**

**z = 0.0**

**for i in range(len(w)):**

**z += x[i] \* w[i] # Вычисление суммы взвешенных входов(вычисляем скалярное произведение векторов "x" и "w")**

**if z < 0: C**

**return -1**

**else:**

**return 1**

**def draw\_line(w,color = 'magenta'): # Функция черчения линии на графике**

**x1 = [-1.9,1.9] # максимальное и минимальное значение x1**

**x2 = [-w[1]\* x1[i]/w[2] - w[0] / w[2] for i in range(len(x1))] # вычисление x2**

**plt.axline(\*list(zip(x2,x1)), color=color) # черчение линии на графике**

**def show\_training(input\_data,output,w,xlim:list[float,float] = [-1.1,1.1], ylim:list[float,float] = [-1.1,1.1]): # Функция отображения финального графика**

**plt.xlim(xlim) # Лимит оси X**

**plt.ylim(ylim) # Лимит оси Y**

**plt.xlabel("x2") # Подпись оси X**

**plt.ylabel("x1") # Подпись оси Y**

**for i in range(len(output)):**

**if output[i] == -1:**

**# Если выход нейросети == -1, то ставим точку с подписью “-”**

**plt.scatter(input\_data[i][2],input\_data[i][1], color='blue', marker='\_')**

**else:**

**# Если выход нейросети == 1, то ставим точку с подписью “+”**

**plt.scatter(input\_data[i][2],input\_data[i][1], color='red', marker='+')**

**draw\_line(w) # рисуем линию**

**plt.show() # выводим график**

**def show\_all\_training(input\_data,output,all\_w,xlim:list[float,float] = [-1.1,1.1], ylim:list[float,float] = [-1.1,1.1]): # Функция вывода графика всего процесса обучения**

**plt.xlim(xlim) # Лимит оси X**

**plt.ylim(ylim) # Лимит оси Y**

**plt.xlabel("x2") # Подпись оси X**

**plt.ylabel("x1") # Подпись оси Y**

**for i in range(len(output)):**

**if output[i] == -1:**

**# Если выход нейросети == -1, то ставим точку с подписью “-”**

**plt.scatter(input\_data[i][2],input\_data[i][1], color='blue', marker='\_')**

**else:**

**# Если выход нейросети == 1, то ставим точку с подписью “+”**

**plt.scatter(input\_data[i][2],input\_data[i][1], color='red', marker='+')**

**for i in range(len(all\_w)): # Перебираем все значения весов**

**draw\_line(all\_w[i]) # Чертим линии на графике**

**plt.show() # Выводим график**

**# –ПРОЦЕСС ОБУЧЕНИЯ ПЕРСЕПТРОНА –#**

**all\_correct = False # вводим булевую переменную**

**while not all\_correct: # пока all\_correct != False выполняем следующий код**

**all\_correct = True # меняем значение all\_correct на True**

**random.shuffle(index\_list) # Сделать порядок случайным**

**for i in index\_list:**

**x = x\_train[i]**

**y = y\_train[i]**

**p\_out = compute\_output(w, x) # Функция персептрона**

**if y != p\_out: # Обновить веса, когда неправильно**

**for j in range(0, len(w)):**

**w[j] += (y \* LEARNING\_RATE \* x[j]) # вычисляем новые веса**

**all\_correct = False**

**show\_learning(w)**

**all\_data["x"].append(x)**

**all\_data["y"].append(p\_out)**

**all\_data["w"].append(w.copy())**

**if \_\_name\_\_ == '\_\_main\_\_':**

**show\_training(x\_train,y\_train,w,xlim=[-2,2],ylim=[-2,2]) # вызываем функцию создания финального графика**

**show\_all\_training(input\_data=all\_data["x"], output=all\_data["y"],all\_w=all\_data["w"],xlim=[-2,2],ylim=[-2,2]) # вызываем функцию создания графика всего обучения**