Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: Приложение нейросетевых алгоритмов**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Иванов

Направление подготовки: 02.03.03 Математическое обеспечение и администрирование информационных систем

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Шиян

**Цель работы**

Получение знаний и практических навыков построения

однослойных нейронных сетей.

***Задача 1***

**Описание решения**

*Задача 1*: Нейроны типа WTA.

Для решения этой задачи были созданы 2 класса - Neuron\_WTA и Neural\_Network. Класс Neuron\_WTA отвечает за создание объекта нейрона, у которого имеются поля весов, входных сигналов, ответа (победил/проиграл) и количества побед. Также этот класс содержит методы сумматора и изменения весов нейрона-победителя по правилу Гроссберга. Класс Neural\_Network отвечает за обучение нейронов. Он содержит методы прогонки (1 круг обучения нейронов) и подсчёта списка выходных сигналов нейронов.

Сам алгоритм решения задачи выглядит следующим образом: имеется список списков входных сигналов и скорость обучения. Создаются объекты типа Neuron\_WTA, у которых случайным образом генерируются веса и передаются одинаковые входные параметры. Далее, создаётся объект типа Neural\_Network, который несколько раз «прогоняет» нейроны. В процессе каждой «прогонки» идёт подсчёт списка выходных сигналов, определения нейрона-победителя и изменение его весов, а также подсчёт побед каждого нейрона.

**Реализация программы на языке Python**

*# Нужно реализовать нейронную сеть из 4 нейронов типа WTA, предназначенных для классификации ->*

*# -> входных двухкомпонентных векторов*

*# х - список векторов входных параметров*

*# n - коэффициент скорости обучения*

*# w - веса*

*# out - ответ нейрона (победил/проиграл)*

from random import \*

class Neuron\_WTA:

    w = []

    x = []

    out = 0

    kol\_win = 0

    def \_\_init\_\_(*self*, *x\_in*):

        """Инициализация нейрона (создаём веса и передаём входные параметры)"""

*self*.w = [uniform(-1.00, 1.00) for i in range(2)]

*self*.x = *x\_in*

    def summat(*self*):

        """Сумматор нейрона (возвращает выходной сигнал)"""

        u = 0

        for i in range(len(*self*.x)):

            u += *self*.w[i] \* *self*.x[i]

        return u

    def change\_w(*self*, *n*):

        """Изменение весов нейрона-победителя по правилу Гроссберга"""

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

*self*.w[i] = *self*.w[i] + *n* \* (*self*.x[i] - *self*.w[i])

*self*.out = 1

*self*.kol\_win += 1

class Neural\_Network:

    def loop(*self*, *neurons*):

        """Прогонка нейронов (1 круг обучения, возвращает массив выходных сигналов)"""

        u = []

        for neuron in *neurons*:

            u.append(neuron.summat())

        return u

    def definition(*self*, *u*):

        """Определение нейрона-победителя на кругу обучения (возвращает номер нейрона)"""

        return *u*.index(max(*u*))

x = [[0.97, 0.20], [1.00, 0.00], [-0.72, 0.70], [-0.67, 0.74], [-0.80, 0.60], [0.00, -1.00],

     [0.20, -0.97], [-0.30, -0.95]]

n = 0.5

for i in range(len(x)):

    print("-" \* 50)

    neurons = [Neuron\_WTA(x[i]) for j in range(4)]

    print("Стартовые веса нейронов для входного вектора:")

    for neuron in neurons:

        print(f"{neuron.w},   {neuron.x}")

    nn = Neural\_Network()

    for l in range(10):

        out\_mas = nn.loop(neurons)

        print(f"Выходные сигналы нейронов: {out\_mas}")

        ind\_win = nn.definition(out\_mas)

        print(f"Индекс нейрона-победителя: {ind\_win}")

        neurons[ind\_win].change\_w(n)

        print(f"Изменённые веса нейрона-победителя: {neurons[ind\_win].w}")

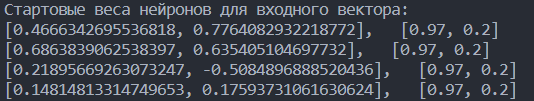
        print("Кол-во побед каждого нейрона:")

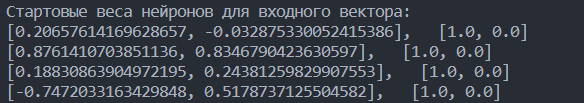
        for k in range(len(neurons)):

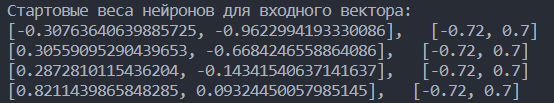
            print(f"Нейрон {k}, побед: {neurons[k].kol\_win}")

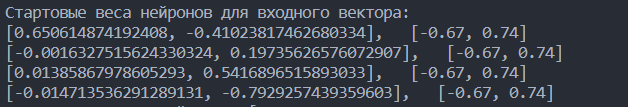
        print(f"Это был {l + 1} круг обучения")

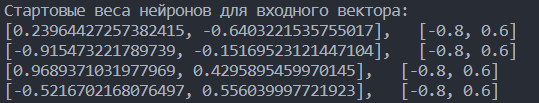
**Примеры запусков**

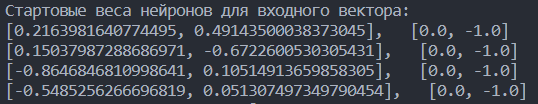


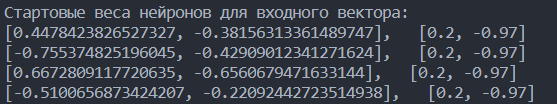


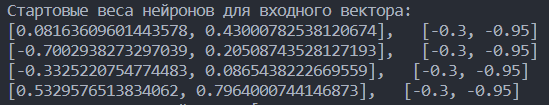




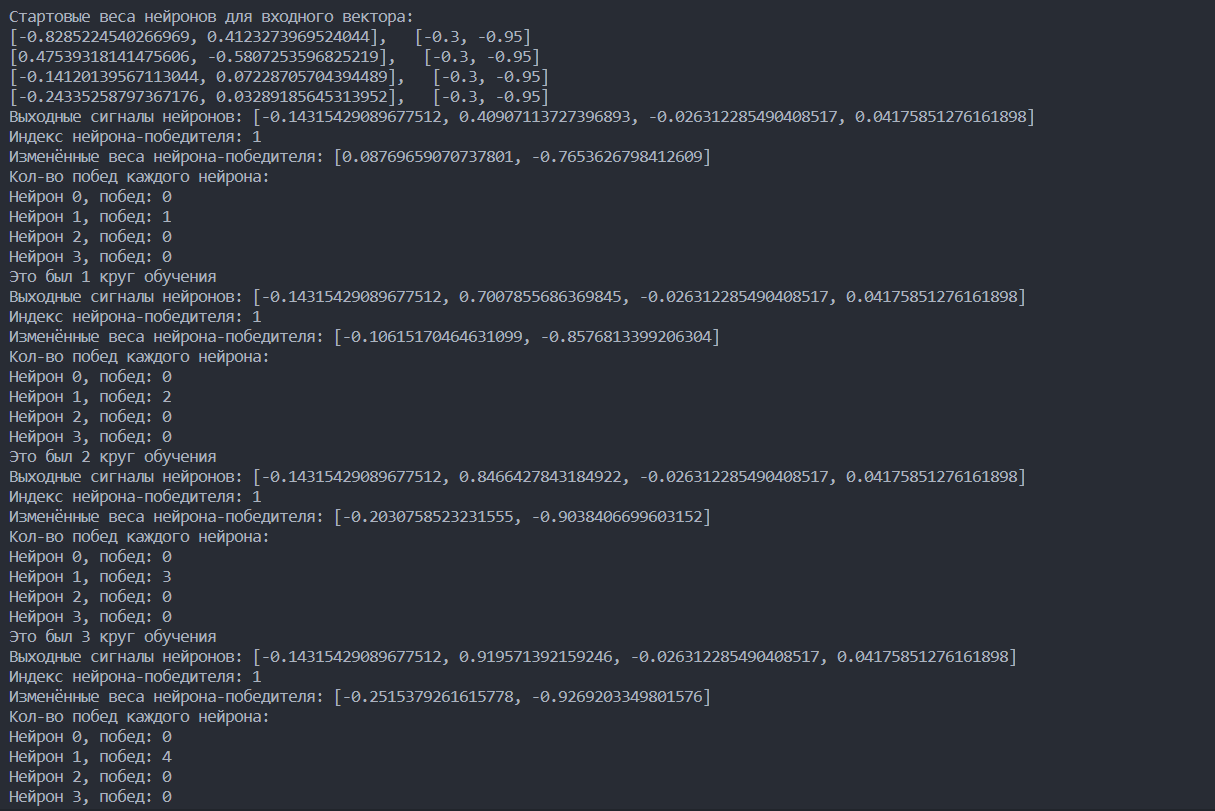


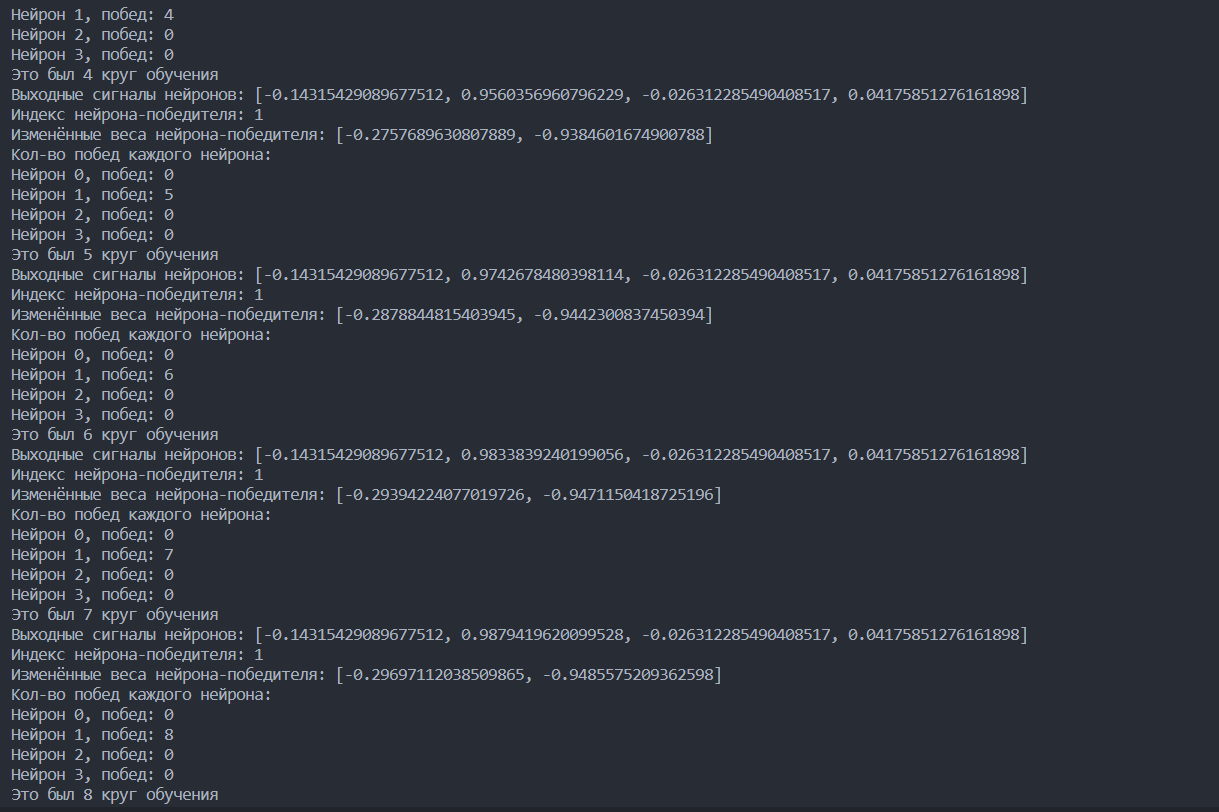


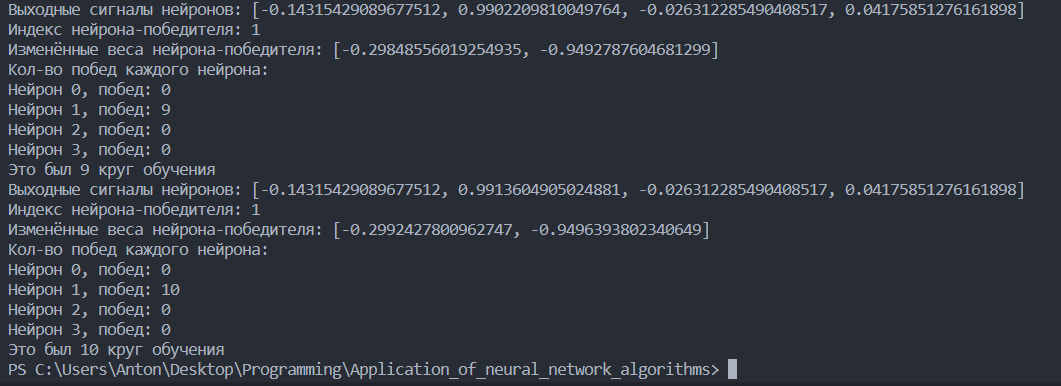




**Выводы программы**







***Задача 2***

**Описание решения**

*Задача 2*: Проблема мёртвых нейронов.

Решение и алгоритм задачи почти не отличаются от Задачи 1. Единственное отличие - изменение метода изменения весов нейронов. Теперь в нём присутствует проверка на количество побед каждого нейрона. Если оно превышает некоторое пороговое значение - изменение весов не происходит (пауза в обучении нейрона).

**Реализация программы на языке Python**

*# Нужно реализовать нейронную сеть из 4 нейронов типа WTA, предназначенных для классификации ->*

*# -> входных двухкомпонентных векторов*

*# х - список векторов входных параметров*

*# n - коэффициент скорости обучения*

*# w - веса*

*# out - ответ нейрона (победил/проиграл)*

*# threshold\_number\_win - пороговое число побед нейрона (для системы штрафов)*

*# Я выбрал систему с паузой при достижении порогового числа побед*

from random import \*

class Neuron\_WTA:

    w = []

    x = []

    out = 0

    kol\_win = 0

    def \_\_init\_\_(*self*, *x\_in*):

        """Инициализация нейрона (создаём веса и передаём входные параметры)"""

*self*.w = [uniform(-1.00, 1.00) for i in range(2)]

*self*.x = *x\_in*

    def summat(*self*):

        """Сумматор нейрона (возвращает выходной сигнал)"""

        u = 0

        for i in range(len(*self*.x)):

            u += *self*.w[i] \* *self*.x[i]

        return u

    def change\_w(*self*, *n*, *tnw*):

        """Изменение весов нейрона-победителя по правилу Гроссберга (ввод пороговой системы с паузой)"""

        if *self*.kol\_win >= *tnw*:

            pass

        else:

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

*self*.w[i] = *self*.w[i] + *n* \* (*self*.x[i] - *self*.w[i])

*self*.out = 1

*self*.kol\_win += 1

class Neural\_Network:

    def loop(*self*, *neurons*):

        """Прогонка нейронов (1 круг обучения, возвращает массив выходных сигналов)"""

        u = []

        for neuron in *neurons*:

            u.append(neuron.summat())

        return u

    def definition(*self*, *u*):

        """Определение нейрона-победителя на кругу обучения (возвращает номер нейрона)"""

        return *u*.index(max(*u*))

x = [[0.97, 0.20], [1.00, 0.00], [-0.72, 0.70], [-0.67, 0.74], [-0.80, 0.60], [0.00, -1.00],

     [0.20, -0.97], [-0.30, -0.95]]

n = 0.5

threshold\_number\_win = 5

for i in range(len(x)):

    print("-" \* 50)

    neurons = [Neuron\_WTA(x[i]) for j in range(4)]

    print("Стартовые веса нейронов для входного вектора:")

    for neuron in neurons:

        print(f"{neuron.w},   {neuron.x}")

    nn = Neural\_Network()

    for l in range(10):

        out\_mas = nn.loop(neurons)

        print(f"Выходные сигналы нейронов: {out\_mas}")

        ind\_win = nn.definition(out\_mas)

        print(f"Индекс нейрона-победителя: {ind\_win}")

        neurons[ind\_win].change\_w(n, threshold\_number\_win)

        print(f"Изменённые веса нейрона-победителя: {neurons[ind\_win].w}")

        print("Кол-во побед каждого нейрона:")

        for k in range(len(neurons)):

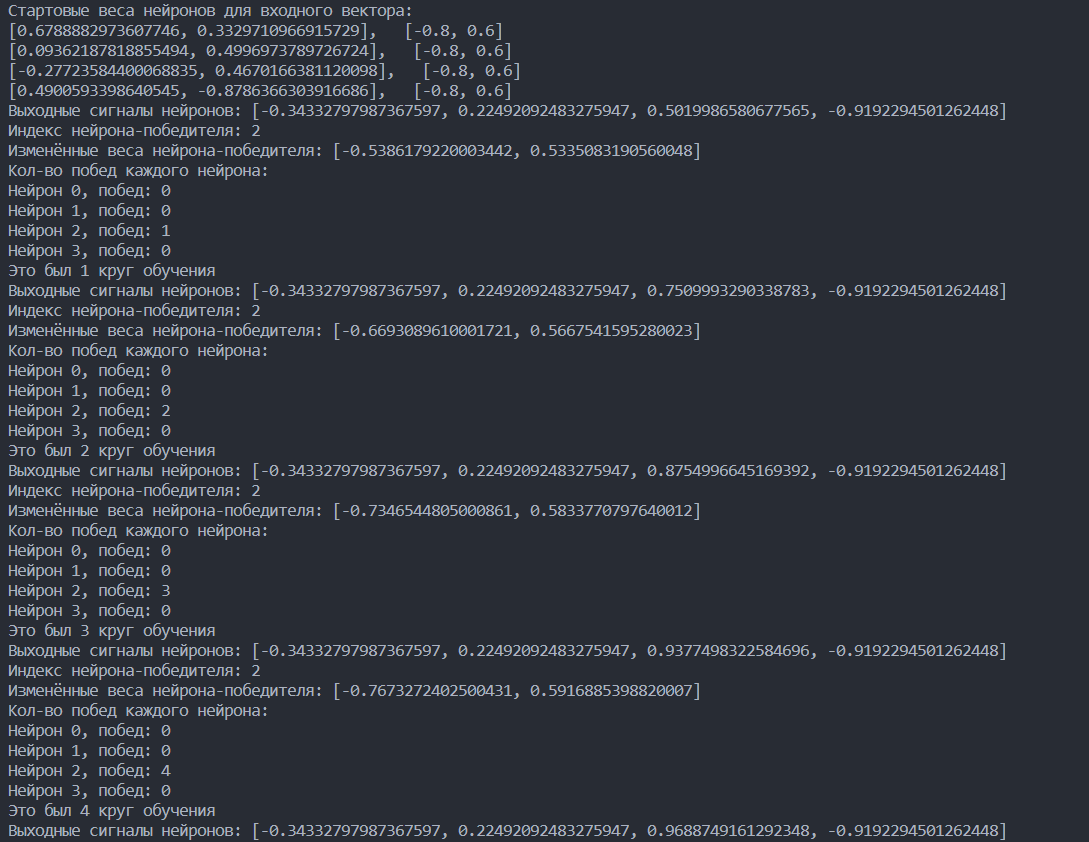
            print(f"Нейрон {k}, побед: {neurons[k].kol\_win}")

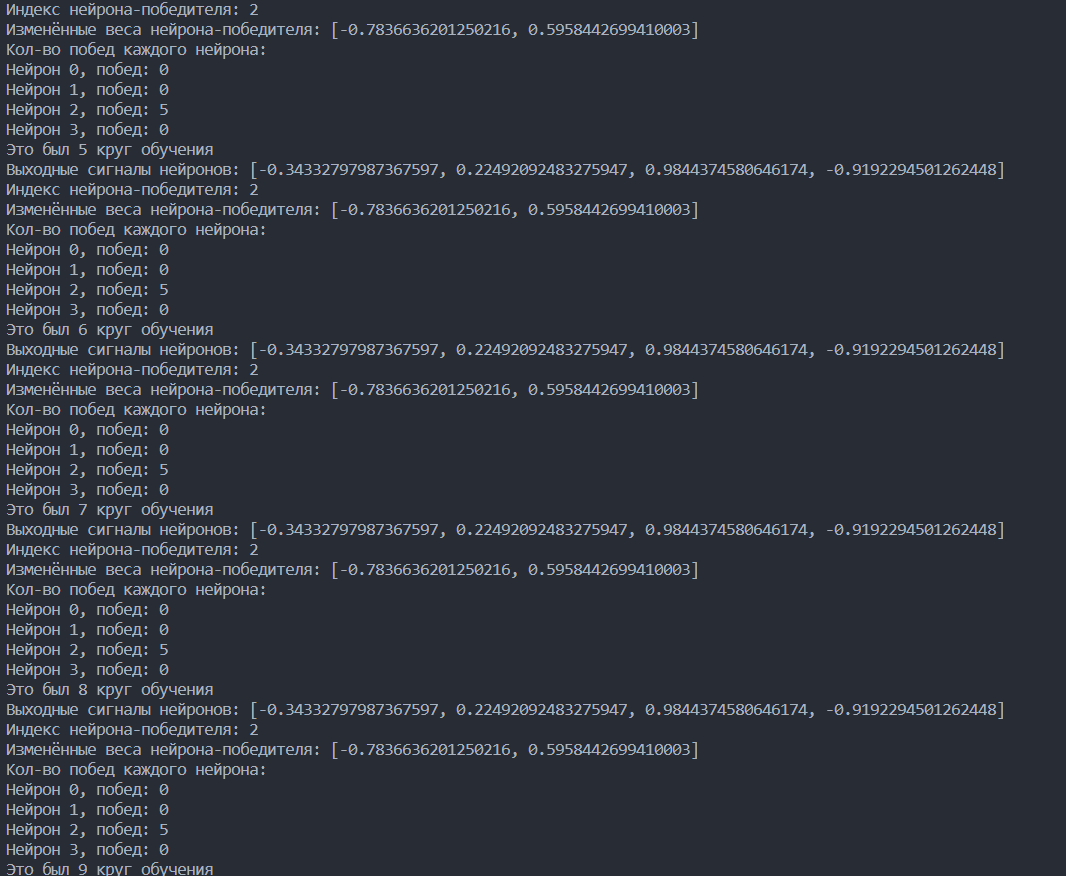
        print(f"Это был {l + 1} круг обучения")

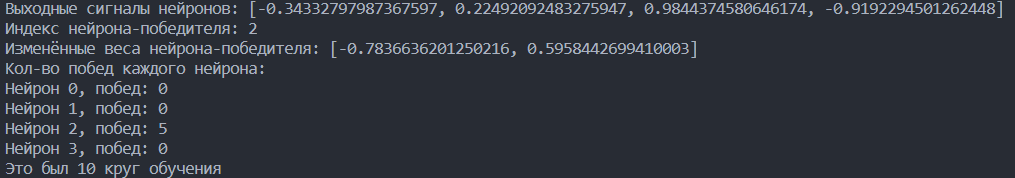
**Примеры запусков**

Примеры запусков аналогичны Задаче 1. Отличия в выводе программы.

**Выводы программы**







***Задача 3***

**Описание решения**

*Задача 3*: Модель нейрона Хебба.

Решание и алгоритм почти не отличаются от Задачи 1. Отличие заключается в новом способе изменения весов нейронов и рандомизации скорости обучения. Теперь программа не считает количество побед нейрона, а изменяет веса по правилу Хебба (через приращение).

**Реализация программы на языке Python**

*# Нужно реализовать нейронную сеть из 4 нейронов типа WTA, предназначенных для классификации ->*

*# -> входных двухкомпонентных векторов*

*# х - список векторов входных параметров*

*# n - коэффициент скорости обучения*

*# w - веса*

*# out - ответ нейрона (победил/проиграл)*

*# threshold\_number\_win - пороговое число побед нейрона (для системы штрафов)*

*# Я выбрал систему с паузой при достижении порогового числа побед*

*# Для обучения Нейронов Хебба нужно: рандомизировать коэффициент обучение n(0, 1); ->*

*# -> изменить метод изменения весов нейронов (сделать по правилу Хебба)*

from random import \*

class Neuron\_WTA:

    w = []

    x = []

    out = 0

    def \_\_init\_\_(*self*, *x\_in*):

        """Инициализация нейрона (создаём веса и передаём входные параметры)"""

*self*.w = [uniform(-1.00, 1.00) for i in range(2)]

*self*.x = *x\_in*

    def summat(*self*):

        """Сумматор нейрона (возвращает выходной сигнал)"""

        u = 0

        for i in range(len(*self*.x)):

            u += *self*.w[i] \* *self*.x[i]

        return u

    def return\_w(*self*):

        """Вспомогательная функция, возвращает массив весов нейрона"""

        return *self*.w

    def save\_change\_w(*self*, *n*, *u*, *x\_in*):

        """Вспомогательная функция, изменяет веса нейрона (через приращение)"""

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

*self*.w[i] += *n* \* *x\_in*[i] \* *u*

class Neural\_Network:

    def loop(*self*, *neurons*):

        """Прогонка нейронов (1 круг обучения, возвращает массив выходных сигналов)"""

        u = []

        for neuron in *neurons*:

            u.append(neuron.summat())

        return u

    def change\_w(*self*, *neurons*, *n*, *u*, *x\_in*):

        """Изменение весов нейронов по правилу Хебба"""

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

*neurons*[i].save\_change\_w(*n*, *u*[i], *x\_in*)

x = [[0.97, 0.20], [1.00, 0.00], [-0.72, 0.70], [-0.67, 0.74], [-0.80, 0.60], [0.00, -1.00],

     [0.20, -0.97], [-0.30, -0.95]]

n = uniform(0.01, 0.99)

for i in range(len(x)):

    print("-" \* 50)

    neurons = [Neuron\_WTA(x[i]) for j in range(2)]

    print("Стартовые веса нейронов для входного вектора:")

    for neuron in neurons:

        print(f"{neuron.w},   {neuron.x}")

    nn = Neural\_Network()

    for l in range(10):

        out\_mas = nn.loop(neurons)

        print(f"Выходные сигналы нейронов: {out\_mas}")

        nn.change\_w(neurons, n, out\_mas, x[i])

        print("Изменение весов нейронов:")

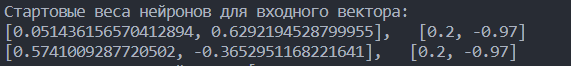
        for k in range(len(neurons)):

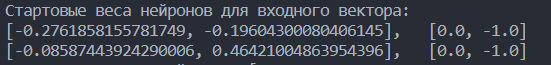
            print(f"Нейрон {k}: {neurons[k].w}")

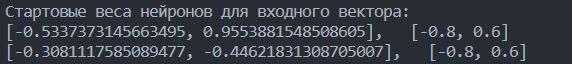
        print(f"Это был {l + 1} круг обучения")

**Примеры запусков**

Примеры запусков почти не отличаются от Задачи 1:







**Выводы программы**

