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 информационных технологий**

**ОТЧЕТ**

**о выполнении лабораторной работы №1**

**по дисциплине «Приложение нейросетевых алгоритмов»**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.М.Шадрина

(подпись)

Направление подготовки 02.03.03 — «Математическое обеспечение и

администрирование информационных систем»\_\_\_\_\_\_\_\_\_\_\_курс\_\_\_\_\_3\_\_\_\_\_

Направленность (профиль) Технология программирования

Краснодар

2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc190732236)

[1 Понятие скорости обучения 5](#_Toc190732237)

[1.1 Краткое описание однослойного перцептрона 5](#_Toc190732238)

[1.2 Методы обучения однослойного персептрона, использующие скорость обучения в качестве параметра 6](#_Toc190732239)

[1.3 Проблемы, связанные с выбором скорости обучения 9](#_Toc190732240)

[1.4 Экспериментальное изучение влияния скорости обучения на сходимость и результаты 10](#_Toc190732241)

[ЗАКЛЮЧЕНИЕ 13](#_Toc190732242)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 14](#_Toc190732243)

[ПРИЛОЖЕНИЕ А 15](#_Toc190732244)

# ВВЕДЕНИЕ

Лабораторная работа № 1. Однослойные нейронные сети.

**Задание.** Требуется реализовать алгоритмы на языке программирования Python без использования специализированных библиотек.

**Задание 1**

Нейроны WTA (Winner Takes All)— это группа нейронов, которые конкурируют между собой. Они получают одни и те же входные сигналы, но каждый нейрон имеет свои собственные веса (настройки). Нейрон, который "выигрывает" (то есть его выходной сигнал оказывается самым сильным), активируется (его выход равен 1), а все остальные нейроны "проигрывают" (их выход равен 0).

Обучение без учителя: Нейроны WTA работают в режиме обучения без учителя. Это означает, что они не знают, что именно представляют собой данные (например, "кошки", "собаки", "цветы"). Они просто находят сходства и различия между входными векторами.

Нейросеть WTA группирует данные в некие кластеры на основе их сходства.

**Алгоритм работы:**

1. **Инициализация весов**: Веса нейронов прошлого выбираются случайным образом и нормализуются (приводятся к единичной длине).
2. **Входные сигналы:** Нейроны слоя получают одни и те же входные данные

​.

1. **Сумматор:** Каждый нейрон вычисляет свой выходной сигнал  по формуле:
2. **Конкуренция:** После вычисления выходных сигналов всех нейронов, выбирается нейрон с наибольшим значением . Этот нейрон становится "победителем".
3. **Активация:** Победивший нейрон активируется (его выход равен 1), а все остальные нейроны слоя отключаются (их выход равен 0).
4. **Обновление весов**: Только победивший нейрон обновляет свои веса по правилу Гроссберга.
5. **Нормализация**: Веса победившего нейрона снова нормализуются, чтобы их длина оставалась равной 1.
6. **Далее новый слой и шаг 1**.

**Код:**

import numpy as np

# Функция для нормализации вектора

def normalize\_vector(vector):

    norm = np.linalg.norm(vector)

    if norm == 0:

        return vector

    return vector / norm

# Инициализация весов нейронов случайными значениями и их нормализация

def initialize\_weights(num\_neurons, input\_size):

    weights = np.random.rand(num\_neurons, input\_size)

    for i in range(num\_neurons):

        weights[i] = normalize\_vector(weights[i])

    return weights

# Обучение нейронов WTA

def train\_wta(weights, inputs, learning\_rate=0.5, epochs=1):

    for epoch in range(epochs):

        print(f"\nЭпоха {epoch + 1}:")

        for i, x in enumerate(inputs):

            print(f"\nОбработка входного вектора {i + 1}: {x}")

            # Вычисление выходных сигналов нейронов

            outputs = np.dot(weights, x)

            # Определение нейрона-победителя

            winner = np.argmax(outputs)

            print(f"Нейрон-победитель: {winner + 1}")

            # Обновление весов победившего нейрона

            weights[winner] += learning\_rate \* (x - weights[winner])

            # Нормализация весов победившего нейрона

            weights[winner] = normalize\_vector(weights[winner])

            # Вывод весов после обновления

            print("Веса нейронов после обновления:")

            for j, w in enumerate(weights):

                print(f"Нейрон {j + 1}: {w}")

    return weights

# Входные данные (уже нормализованные)

inputs = np.array([

    [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]

])

# Инициализация весов

num\_neurons = 4

input\_size = 2

weights = initialize\_weights(num\_neurons, input\_size)

# Вывод начальных весов

print("\nНачальные веса нейронов:")

for i, w in enumerate(weights):

    print(f"Нейрон {i + 1}: {w}")

# Обучение нейронов

learning\_rate = 0.5 # Скорость обучения

trained\_weights = train\_wta(weights, inputs, learning\_rate)

# Вывод итоговых весов

print("\nИтоговые веса нейронов после обучения:")

for i, w in enumerate(trained\_weights):

    print(f"Нейрон {i + 1}: {w}")

**Вывод:**

Начальные веса нейронов:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [0.98951165 0.14445312]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [0.23575956 0.97181142]

Эпоха 1:

Обработка входного вектора 1: [0.97 0.2 ]

Нейрон-победитель: 2

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [0.98489887 0.17313063]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [0.23575956 0.97181142]

Обработка входного вектора 2: [1. 0.]

Нейрон-победитель: 2

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [0.99621756 0.08689399]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [0.23575956 0.97181142]

Обработка входного вектора 3: [-0.72 0.7 ]

Нейрон-победитель: 4

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [0.99621756 0.08689399]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.27821448 0.96051898]

Обработка входного вектора 4: [-0.67 0.74]

Нейрон-победитель: 4

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [0.99621756 0.08689399]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.48700881 0.87339705]

Обработка входного вектора 5: [-0.8 0.6]

Нейрон-победитель: 4

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [0.99621756 0.08689399]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.65786352 0.75313717]

Обработка входного вектора 6: [ 0. -1.]

Нейрон-победитель: 2

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [ 0.73718857 -0.67568706]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.65786352 0.75313717]

Обработка входного вектора 7: [ 0.2 -0.97]

Нейрон-победитель: 2

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [ 0.49486303 -0.86897099]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.65786352 0.75313717]

Обработка входного вектора 8: [-0.3 -0.95]

Нейрон-победитель: 2

Веса нейронов после обновления:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [ 0.10651868 -0.9943107 ]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.65786352 0.75313717]

Итоговые веса нейронов после обучения:

Нейрон 1: [0.79269856 0.60961381]

Нейрон 2: [ 0.10651868 -0.9943107 ]

Нейрон 3: [0.73533682 0.67770182]

Нейрон 4: [-0.65786352 0.75313717]

**Задание 2**

**Проблема мертвых нейронов.** Серьезной проблемой при обучении WTA остается проблема мертвых нейронов, которые после инициализации ни одного раза не победили в конкурентной борьбе и остались в состоянии, сформированном в начальный момент времени. Каждый мертвый нейрон уменьшает эффективное количество элементов, прошедших обучение, и соответственно увеличивает общую погрешность распознавания данных. Для разрешения этой проблемы применяется модифицированное обучение, основанное на учете прошлых побед каждого нейрона и штрафовании (временной дисквалификации) тех из них, которые побеждали чаще всего. Дисквалификация слишком активных нейронов может осуществляться либо назначением порогового числа побед, по достижении которого наступает обязательная пауза, либо уменьшением фактического значения при нарастании количества побед i-го нейрона.

**Логика:**

1. **Учет побед нейронов**:
   * Для каждого нейрона храним количество побед.
   * Если нейрон побеждает слишком часто, его "штрафуем".
2. **Штрафование**:
   * Штраф может быть реализован двумя способами:
     + **Пороговое значение побед**: Если нейрон побеждает больше заданного порога, он временно отключается.
     + **Уменьшение функции активации**: Чем больше побед у нейрона, тем меньше его выходной сигнал ​ (реализован в коде).

**Код:**

# задание 2 с промежуточными

import numpy as np

# Функция для нормализации вектора

def normalize\_vector(vector):

    norm = np.linalg.norm(vector)

    if norm == 0:

        return vector

    return vector / norm

# Инициализация весов нейронов случайными значениями и их нормализация

def initialize\_weights(num\_neurons, input\_size):

    weights = np.random.rand(num\_neurons, input\_size)

    for i in range(num\_neurons):

        weights[i] = normalize\_vector(weights[i])

    return weights

# Модифицированное обучение WTA с учетом штрафов

def train\_wta\_modified(weights, inputs, learning\_rate=0.5, epochs=1, penalty\_factor=0.1):

    num\_neurons = weights.shape[0]

    win\_counts = np.zeros(num\_neurons)  # Счетчик побед каждого нейрона

    for epoch in range(epochs):

        print(f"\nЭпоха {epoch + 1}:")

        for i, x in enumerate(inputs):

            print(f"\nОбработка входного вектора {i + 1}: {x}")

            # Вычисление выходных сигналов нейронов без штрафа

            outputs = np.dot(weights, x)

            print("Выходные сигналы до штрафа:")

            for j, output in enumerate(outputs):

                print(f"Нейрон {j + 1}: {output:.4f}")

            # Применение штрафа к выходным сигналам

            penalized\_outputs = outputs / (1 + penalty\_factor \* win\_counts)

            print("Выходные сигналы после штрафа:")

            for j, output in enumerate(penalized\_outputs):

                print(f"Нейрон {j + 1}: {output:.4f} (побед: {win\_counts[j]})")

            # Определение нейрона-победителя

            winner = np.argmax(penalized\_outputs)

            win\_counts[winner] += 1  # Увеличиваем счетчик побед

            print(f"Нейрон-победитель: {winner + 1} (побед: {win\_counts[winner]})")

            # Обновление весов победившего нейрона

            weights[winner] += learning\_rate \* (x - weights[winner])

            # Нормализация весов победившего нейрона

            weights[winner] = normalize\_vector(weights[winner])

            # Вывод весов после обновления

            print("Веса нейронов после обновления:")

            for j, w in enumerate(weights):

                print(f"Нейрон {j + 1}: {w} (побед: {win\_counts[j]})")

    return weights

# Входные данные (уже нормализованные)

inputs = np.array([

    [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]

])

# Инициализация весов

num\_neurons = 4

input\_size = 2

weights = initialize\_weights(num\_neurons, input\_size)

# Вывод начальных весов

print("\nНачальные веса нейронов:")

for i, w in enumerate(weights):

    print(f"Нейрон {i + 1}: {w}")

# Обучение нейронов с модификацией

learning\_rate = 0.5 # Скорость обучения

penalty\_factor = 0.1  # Коэффициент штрафа

trained\_weights = train\_wta\_modified(weights, inputs, learning\_rate, penalty\_factor=penalty\_factor)

# Вывод итоговых весов

print("\nИтоговые веса нейронов после обучения:")

for i, w in enumerate(trained\_weights):

    print(f"Нейрон {i + 1}: {w}")

**Вывод:**

Начальные веса нейронов:

Нейрон 1: [0.9871965 0.15950882]

Нейрон 2: [0.73875605 0.67397292]

Нейрон 3: [0.97656492 0.21522304]

Нейрон 4: [0.92877731 0.37063825]

Эпоха 1:

Обработка входного вектора 1: [0.97 0.2 ]

Выходные сигналы до штрафа:

Нейрон 1: 0.9895

Нейрон 2: 0.8514

Нейрон 3: 0.9903

Нейрон 4: 0.9750

Выходные сигналы после штрафа:

Нейрон 1: 0.9895 (побед: 0.0)

Нейрон 2: 0.8514 (побед: 0.0)

Нейрон 3: 0.9903 (побед: 0.0)

Нейрон 4: 0.9750 (побед: 0.0)

Нейрон-победитель: 3 (побед: 1.0)

Веса нейронов после обновления:

Нейрон 1: [0.9871965 0.15950882] (побед: 0.0)

Нейрон 2: [0.73875605 0.67397292] (побед: 0.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 2: [1. 0.]

Выходные сигналы до штрафа:

Нейрон 1: 0.9872

Нейрон 2: 0.7388

Нейрон 3: 0.9780

Нейрон 4: 0.9288

Выходные сигналы после штрафа:

Нейрон 1: 0.9872 (побед: 0.0)

Нейрон 2: 0.7388 (побед: 0.0)

Нейрон 3: 0.8891 (побед: 1.0)

Нейрон 4: 0.9288 (побед: 0.0)

Нейрон-победитель: 1 (побед: 1.0)

Веса нейронов после обновления:

Нейрон 1: [0.99679399 0.08001093] (побед: 1.0)

Нейрон 2: [0.73875605 0.67397292] (побед: 0.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 3: [-0.72 0.7 ]

Выходные сигналы до штрафа:

Нейрон 1: -0.6617

Нейрон 2: -0.0601

Нейрон 3: -0.5581

Нейрон 4: -0.4093

Выходные сигналы после штрафа:

Нейрон 1: -0.6015 (побед: 1.0)

Нейрон 2: -0.0601 (побед: 0.0)

Нейрон 3: -0.5074 (побед: 1.0)

Нейрон 4: -0.4093 (побед: 0.0)

Нейрон-победитель: 2 (побед: 1.0)

Веса нейронов после обновления:

Нейрон 1: [0.99679399 0.08001093] (побед: 1.0)

Нейрон 2: [0.01364969 0.99990684] (побед: 1.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 4: [-0.67 0.74]

Выходные сигналы до штрафа:

Нейрон 1: -0.6086

Нейрон 2: 0.7308

Нейрон 3: -0.5009

Нейрон 4: -0.3480

Выходные сигналы после штрафа:

Нейрон 1: -0.5533 (побед: 1.0)

Нейрон 2: 0.6644 (побед: 1.0)

Нейрон 3: -0.4553 (побед: 1.0)

Нейрон 4: -0.3480 (побед: 0.0)

Нейрон-победитель: 2 (побед: 2.0)

Веса нейронов после обновления:

Нейрон 1: [0.99679399 0.08001093] (побед: 1.0)

Нейрон 2: [-0.35295449 0.93564049] (побед: 2.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 5: [-0.8 0.6]

Выходные сигналы до штрафа:

Нейрон 1: -0.7494

Нейрон 2: 0.8437

Нейрон 3: -0.6572

Нейрон 4: -0.5206

Выходные сигналы после штрафа:

Нейрон 1: -0.6813 (побед: 1.0)

Нейрон 2: 0.7031 (побед: 2.0)

Нейрон 3: -0.5975 (побед: 1.0)

Нейрон 4: -0.5206 (побед: 0.0)

Нейрон-победитель: 2 (побед: 3.0)

Веса нейронов после обновления:

Нейрон 1: [0.99679399 0.08001093] (побед: 1.0)

Нейрон 2: [-0.6004079 0.79969391] (побед: 3.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 6: [ 0. -1.]

Выходные сигналы до штрафа:

Нейрон 1: -0.0800

Нейрон 2: -0.7997

Нейрон 3: -0.2086

Нейрон 4: -0.3706

Выходные сигналы после штрафа:

Нейрон 1: -0.0727 (побед: 1.0)

Нейрон 2: -0.6151 (побед: 3.0)

Нейрон 3: -0.1897 (побед: 1.0)

Нейрон 4: -0.3706 (побед: 0.0)

Нейрон-победитель: 1 (побед: 2.0)

Веса нейронов после обновления:

Нейрон 1: [ 0.73485064 -0.67822897] (побед: 2.0)

Нейрон 2: [-0.6004079 0.79969391] (побед: 3.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 7: [ 0.2 -0.97]

Выходные сигналы до штрафа:

Нейрон 1: 0.8049

Нейрон 2: -0.8958

Нейрон 3: -0.0068

Нейрон 4: -0.1738

Выходные сигналы после штрафа:

Нейрон 1: 0.6707 (побед: 2.0)

Нейрон 2: -0.6891 (побед: 3.0)

Нейрон 3: -0.0061 (побед: 1.0)

Нейрон 4: -0.1738 (побед: 0.0)

Нейрон-победитель: 1 (побед: 3.0)

Веса нейронов после обновления:

Нейрон 1: [ 0.49335376 -0.86982875] (побед: 3.0)

Нейрон 2: [-0.6004079 0.79969391] (побед: 3.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Обработка входного вектора 8: [-0.3 -0.95]

Выходные сигналы до штрафа:

Нейрон 1: 0.6783

Нейрон 2: -0.5796

Нейрон 3: -0.4916

Нейрон 4: -0.6307

Выходные сигналы после штрафа:

Нейрон 1: 0.5218 (побед: 3.0)

Нейрон 2: -0.4458 (побед: 3.0)

Нейрон 3: -0.4469 (побед: 1.0)

Нейрон 4: -0.6307 (побед: 0.0)

Нейрон-победитель: 1 (побед: 4.0)

Веса нейронов после обновления:

Нейрон 1: [ 0.10565366 -0.99440299] (побед: 4.0)

Нейрон 2: [-0.6004079 0.79969391] (побед: 3.0)

Нейрон 3: [0.97799736 0.20861726] (побед: 1.0)

Нейрон 4: [0.92877731 0.37063825] (побед: 0.0)

Итоговые веса нейронов после обучения:

Нейрон 1: [ 0.10565366 -0.99440299]

Нейрон 2: [-0.6004079 0.79969391]

Нейрон 3: [0.97799736 0.20861726]

Нейрон 4: [0.92877731 0.37063825]

# Задание 3

**Основная идея**

Дональд Хебб заметил, что если два нейрона активируются одновременно, то связь между ними усиливается. Это явление он описал в своем знаменитом **правиле Хебба**, которое можно сформулировать так:

**"Нейроны, которые активируются вместе, связываются вместе."**

То есть, если два нейрона часто активируются одновременно, их связь становится сильнее.

**Как это работает?**

1. **Структура нейрона**:
   * Нейрон Хебба имеет входы yj (выходы других нейронов) и веса wij​, которые связывают входы с нейроном.
   * Нейрон вычисляет выходной сигнал yi на основе входных сигналов и весов.
2. **Правило Хебба**:
   * Вес wij​ обновляется пропорционально произведению входного сигнала yj​ и выходного сигнала yi​:
3. **Обновление весов**:
   * Новый вес вычисляется по формуле:
   * То есть, вес увеличивается, если оба нейрона (j-й и i-й) активируются одновременно.

**Обучение с учителем и без учителя**

1. **Без учителя**:
   * Используется фактический выходной сигнал yi​.
   * Правило Хебба:
2. **С учителем**:
   * Вместо фактического выходного сигнала yi​ используется ожидаемый (целевой) сигнал di​.
   * Правило Хебба:

**Проблема неограниченного роста весов**

Одна из проблем правила Хебба заключается в том, что веса могут неограниченно расти, так как на каждом шаге обучения они увеличиваются. Это может привести к нестабильности сети.

Код:

import numpy as np

# Функция для нормализации вектора

def normalize\_vector(vector):

    norm = np.linalg.norm(vector)

    if norm == 0:

        return vector

    return vector / norm

# Инициализация весов случайными значениями

def initialize\_weights(num\_neurons, input\_size):

    weights = np.random.rand(num\_neurons, input\_size)

    for i in range(num\_neurons):

        weights[i] = normalize\_vector(weights[i])

    return weights

# Функция активации (пороговая функция)

def activation\_function(x):

    return 1 if x > 0 else 0

# Обучение по правилу Хебба

def train\_hebb(weights, inputs, learning\_rate=0.1, epochs=1):

    num\_neurons = weights.shape[0]

    for epoch in range(epochs):

        print(f"\nЭпоха {epoch + 1}:")

        for i, x in enumerate(inputs):

            print(f"\nОбработка входного вектора {i + 1}: {x}")

            # Вычисление выходных сигналов нейронов

            outputs = np.dot(weights, x)

            y = np.array([activation\_function(output) for output in outputs])

            print(f"Выходные сигналы: {y}")

            # Обновление весов по правилу Хебба

            for j in range(num\_neurons):

                weights[j] += learning\_rate \* y[j] \* x

            # Нормализация весов

            for j in range(num\_neurons):

                weights[j] = normalize\_vector(weights[j])

            # Вывод весов после обновления

            print("Веса нейронов после обновления:")

            for j, w in enumerate(weights):

                print(f"Нейрон {j + 1}: {w}")

    return weights

# Входные данные (уже нормализованные)

inputs = np.array([

    [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]

])

# Инициализация весов

num\_neurons = 2

input\_size = 2

weights = initialize\_weights(num\_neurons, input\_size)

# Вывод начальных весов

print("\nНачальные веса нейронов:")

for i, w in enumerate(weights):

    print(f"Нейрон {i + 1}: {w}")

# Обучение по правилу Хебба

learning\_rate = 0.5 # Скорость обучения

trained\_weights = train\_hebb(weights, inputs, learning\_rate)

# Вывод итоговых весов

print("\nИтоговые веса нейронов после обучения:")

for i, w in enumerate(trained\_weights):

    print(f"Нейрон {i + 1}: {w}")

**Вывод:**

Начальные веса нейронов:

Нейрон 1: [0.33146925 0.94346602]

Нейрон 2: [0.04284796 0.9990816 ]

Эпоха 1:

Обработка входного вектора 1: [0.97 0.2 ]

Выходные сигналы: [1 1]

Веса нейронов после обновления:

Нейрон 1: [0.61623508 0.78756227]

Нейрон 2: [0.43292356 0.90143063]

Обработка входного вектора 2: [1. 0.]

Выходные сигналы: [1 1]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [0.7191407 0.69486449]

Обработка входного вектора 3: [-0.72 0.7 ]

Выходные сигналы: [0 0]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [0.7191407 0.69486449]

Обработка входного вектора 4: [-0.67 0.74]

Выходные сигналы: [0 1]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [0.33933678 0.94066495]

Обработка входного вектора 5: [-0.8 0.6]

Выходные сигналы: [0 1]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [-0.04883738 0.99880674]

Обработка входного вектора 6: [ 0. -1.]

Выходные сигналы: [0 0]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [-0.04883738 0.99880674]

Обработка входного вектора 7: [ 0.2 -0.97]

Выходные сигналы: [0 0]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [-0.04883738 0.99880674]

Обработка входного вектора 8: [-0.3 -0.95]

Выходные сигналы: [0 0]

Веса нейронов после обновления:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [-0.04883738 0.99880674]

Итоговые веса нейронов после обучения:

Нейрон 1: [0.81709489 0.57650321]

Нейрон 2: [-0.04883738 0.99880674]