****

**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Институт № 3**

**Кафедра 311**

**Интеллектуальные системы**

**Лабораторная работа № 1**

**«Перцептроны»**

**Выполнил студент  
Веденеев Максим Кириллович**

**Группа М3З-501-БК**

**Дата 06.10.2023 г.**

**Принял преподаватель  
Кос Оксана Игоревна**

Оглавление

[Цель лабораторной работы 3](#__RefHeading___1)

[Глава 1. Перцептрон 4](#__RefHeading___2)

[Глава 2. Классификация перцептронов 5](#__RefHeading___3)

[Глава 3. Принцип работы перцептрона 6](#__RefHeading___4)

[Глава 4. Реализация программы 9](#__RefHeading___5)

[Итоги лабораторной работы 14](#__RefHeading___6)

[Список литературы 15](#__RefHeading___7)

# Цель лабораторной работы

Изучить:

1. Принцип работы простейших перцептронов;
2. Реализовать один из них.

# Глава 1. Перцептрон

Нейросети – это компьютерные системы, которые пытаются имитировать работу человеческого мозга. Они состоят из множества соединенных между собой нейронов, которые способны обрабатывать информацию и принимать решения на основе полученных данных. История развития нейросетей началась еще в 1943 году, когда Уоррен Маккаллок и Уолтер Питтс опубликовали статью "Логический калькулятор, использующий нервную систему", в которой они описывали модель искусственного нейрона.

В 1950-х годах Фрэнк Розенблатт разработал первую нейронную сеть, названную "Перцептрон". Перцептрон – это одна из первых разработок в области нейронных сетей. Она представляет собой алгоритм, который пытается имитировать работу нейронов в человеческом мозге. Основным принципом работы “Перцептрона” является обработка входных данных и принятие решений на основе полученной информации.

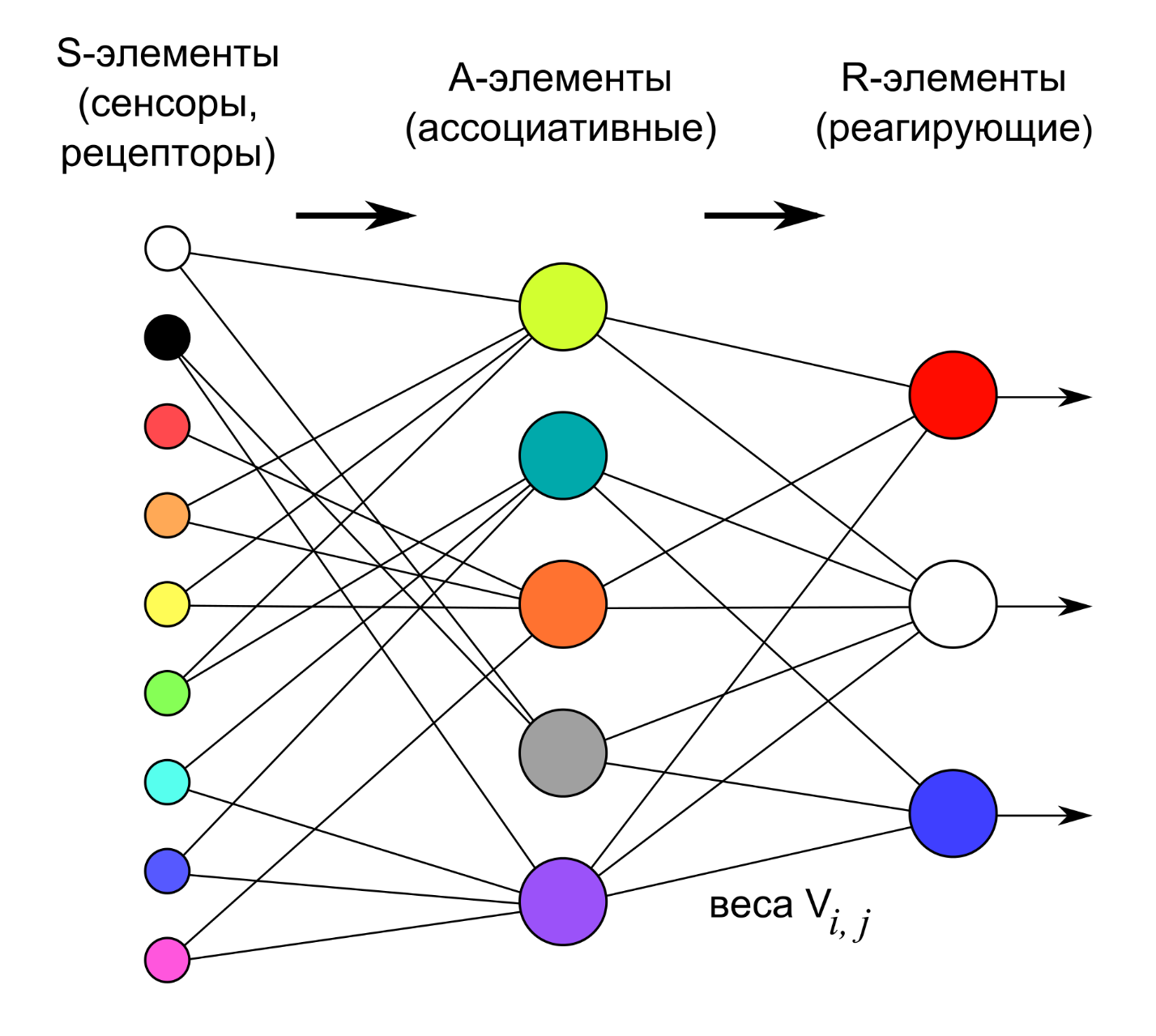


Рисунок 1. Логическая схема перцептрона с тремя выходами

Перцептрон состоит из одного слоя нейронов, каждый из которых имеет несколько входов и один выход. Входные данные подаются на каждый из нейронов, где они умножаются на соответствующие им веса. Затем, полученные произведения суммируются и подаются на функцию активации, которая преобразует их в выходной сигнал.

# Глава 2. Классификация перцептронов

Перцептрон с одним скрытым слоем (элементарный перцептрон) — перцептрон, у которого имеется только по одному слою S, A и R элементов.

Однослойный персептрон — перцептрон, каждый S-элемент которого однозначно соответствует одному А-элементу, S-A связи всегда имеют вес 1, а порог любого А-элемента равен 1. Часть однослойного персептрона соответствует модели искусственного нейрона. Его ключевая особенность состоит в том, что каждый S-элемент однозначно соответствует одному A-элементу, все S-A связи имеют вес, равный +1, а порог A элементов равен 1. Часть однослойного перцептрона, не содержащая входы, соответствует искусственному нейрону, как показано на Рисунке 1. Таким образом, однослойный перцептрон — это искусственный нейрон, который на вход принимает только 0 и 1. Однослойный персептрон также может быть и элементарным персептроном, у которого только по одному слою S,A,R-элементов.

Многослойный перцептрон по Розенблатту — перцептрон, который содержит более 1 слоя А-элементов.

Многослойный перцептрон по Румельхарту — частный случай многослойного персептрона по Розенблатту, с двумя особенностями:

1. S-A связи могут иметь произвольные веса и обучаться наравне с A-R связями;
2. Обучение производится по специальному алгоритму, который называется обучением по методу обратного распространения ошибки.

ы

# Глава 3. Принцип работы перцептрона

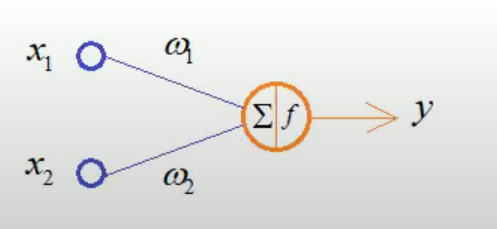
ы

Рисунок 2. – Простейший перцептрон

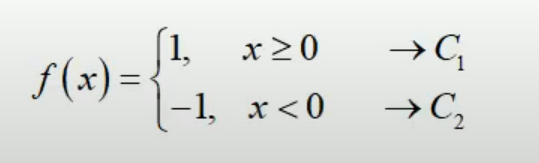
Функция активации может быть различной, но чаще всего используется пороговая функция, которая возвращает 1, если сумма произведений превышает определенный порог, и 0 в противном случае. Таким образом, перцептрон может принимать решения на основе полученной информации.

Рисунок 3 – Активационная функция

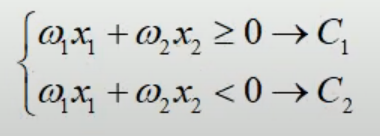


Рисунок 4 – Граница разделения

Рисунок 4 показывает, что граница разделения двух классов C1 и C2 находится на значении ноль. То есть, когда наша сумма больше нуля, это класс C1. Если же значение меньше нуля, то это класс C2. Из этого следует, что, когда выполняется равенство на рисунке 5, мы находимся на границе раздела двух классов.

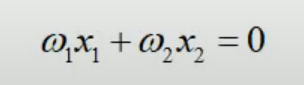


Рисунок 5 – Пороговое равенство

Выражение на рисунке 5 можно записать и в другом виде (рисунок 6). Это есть ничто иное, как прямая с угловым коэффициентом. То есть мы получаем уравнение прямой, которая проходит через начало системы координат (рисунок 6).

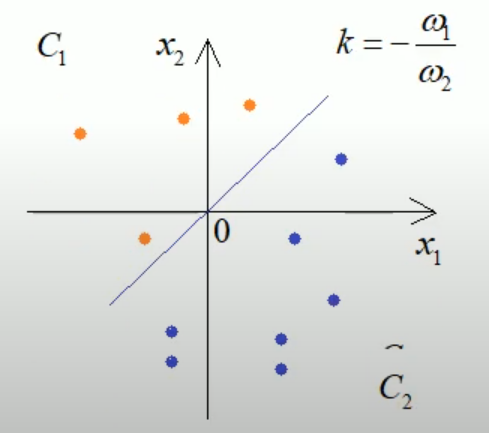


Рисунок 6 – Система координат

Относительно этой прямой можно определить множество точек для класса C1 и множество точек C2. Такая прямая получила название «Разделяющая прямая» (в многомерном случае - «Разделяющая гиперплоскость»).

Обучение перцептрона происходит путем корректировки весов входных данных. Если перцептрон допускает ошибку в принятии решения, то веса корректируются таким образом, чтобы ошибка была уменьшена. Этот процесс повторяется до тех пор, пока перцептрон не станет способен правильно принимать решения на основе входных данных.

# Глава 4. Реализация программы

Ниже представлена реализация обучающейся нейронной сети на языке Python.

import numpy as np

import matplotlib.pyplot as plt

import utils

# Сигмоида

def sigmoid(x):

return 1 / (1 + np.exp(-x))

# Сгенерировать матрицу весов

def generate\_weights(size\_x, size\_y):

return np.random.uniform(-0.5, 0.5, (size\_x, size\_y))

# Алгоритм прямого распространения

def forward\_propagation(input\_data, weights, bias):

raw\_results = (weights @ input\_data) + bias

return sigmoid(raw\_results)

# Собираем изображения и результаты

images, labels = utils.load\_dataset()

# Веса и нейроны смещения

weights\_input\_to\_hidden = generate\_weights(20, 784)

weights\_hidden\_to\_output = generate\_weights(10, 20)

bias\_input\_to\_hidden = np.zeros((20, 1))

bias\_hidden\_to\_output = np.zeros((10, 1))

epochs = 3 # эпохи

e\_loss = 0

e\_correct = 0

learning\_rate = 0.01

for epoch in range(epochs):

print(f"Эпоха №{epoch + 1}")

# По каждому изображению

for image, label in zip(images, labels):

image = np.reshape(image, (-1, 1))

label = np.reshape(label, (-1, 1))

# Проходим до скрытого слоя

hidden = forward\_propagation(image, weights\_input\_to\_hidden, bias\_input\_to\_hidden)

# Проходим до выходного слоя

output = forward\_propagation(hidden, weights\_hidden\_to\_output, bias\_hidden\_to\_output)

# Считаем потери и точность нейросети с ожидаемым результатом (MSE)

e\_loss += 1 / len(output) \* np.sum((output - label) \*\* 2, axis=0)

e\_correct += int(np.argmax(output) == np.argmax(label))

# Обратное распространение (до скрытого слоя)

delta\_output = output - label

weights\_hidden\_to\_output += -learning\_rate \* delta\_output @ np.transpose(hidden)

bias\_hidden\_to\_output += -learning\_rate \* delta\_output

# Обратное распространение (до входного слоя)

delta\_hidden = np.transpose(weights\_hidden\_to\_output) @ delta\_output \* (hidden \* (1 - hidden))

weights\_input\_to\_hidden += -learning\_rate \* delta\_hidden @ np.transpose(image)

bias\_input\_to\_hidden += -learning\_rate \* delta\_hidden

# Выводим потери и точность нейросети между эпохами

print(f"Loss: {round((e\_loss[0] / images.shape[0]) \* 100, 3)}%")

print(f"Accuracy: {round((e\_correct / images.shape[0]) \* 100, 3)}%")

e\_loss = 0.0

e\_correct = 0.0

# Проверка на реальном изображении

img = plt.imread("7.jpg", format="jpeg")

# Перекрашиваем изображение в чёрно-белое

gray = lambda rgb: np.dot(rgb[..., :3], [0.299, 0.587, 0.114])

img = 1 - (gray(img).astype("float32") / 255)

img = np.reshape(img, (img.shape[0] \* img.shape[1]))

image = np.reshape(img, (-1, 1))

# Проходим до скрытого слоя

hidden = forward\_propagation(image, weights\_input\_to\_hidden, bias\_input\_to\_hidden)

# Проходим до выходного слоя

output = forward\_propagation(hidden, weights\_hidden\_to\_output, bias\_hidden\_to\_output)

# Показываем начальное изображение и предположение нейросети

plt.imshow(img.reshape(28, 28), cmap="Greys")

plt.title(f"Я думаю, что число на картинке это: {output.argmax()}")

plt.show()

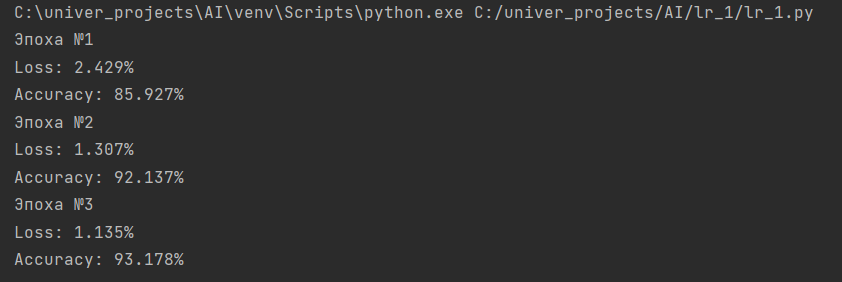


Рисунок 7. Результат выполнения программы



Рисунок 8. Результат выполнения программы

# Итоги лабораторной работы

Мы изучили и научились реализовывать простейшие нейронные сети – перцептроны.

# Список литературы

1. [[wikipedia.org](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%86%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD)] – Перцептрон (Год, старницы, эл. Ресурс, ссылка в работе)