|  |  |
| --- | --- |
| Picture 1 | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  «Санкт-Петербургский государственный морской технический университет» (СПбГМТУ) |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет цифровых промышленных технологий

Направление подготовки 09.03.01

"**Интеллектуальные технологии киберфизических систем**"

Лабораторная работа №5

Вариант №14

Студент 2 курса группы 20221

Очного отделения

Руденко Вячеслав Сергеевич

Проверил:

Преподаватель CПбГМТУ

Кайнова Татьяна Денисовна

Санкт-Петербург

2024

ОГЛАВЛЕНИЕ

[ОБЩАЯ ЧАСТЬ 3](#_Toc184209082)

[1.1 Цель работы 3](#_Toc184209083)

[1.2 Формулировка задачи 3](#_Toc184209084)

[ХОД РАБОТЫ 4](#_Toc184209085)

[2.1 Подготовка данных. 4](#_Toc184209086)

[2.2 Реализация функций для нейронной сети 5](#_Toc184209087)

[2.3 Архитектура нейронной сети 5](#_Toc184209088)

[2.4 Процесс обучения 6](#_Toc184209089)

[2.5 Цикл обучения 7](#_Toc184209090)

[2.6 Тестирование модели 8](#_Toc184209091)

[ЗАКЛЮЧЕНИЕ 10](#_Toc184209092)

[ПРИЛОЖЕНИЕ А 11](#_Toc184209093)

# ОБЩАЯ ЧАСТЬ

## 1.1 Цель работы

Исследование работы простейших нейронных сетей на примере задачи бинарной классификации с использованием набора данных MNIST. Освоение методов обработки и нормализации данных, реализации функций активации, обучения нейронной сети и оценки её точности.

## 1.2 Формулировка задачи

Реализовать программу, обучающую нейронную сеть для классификации цифры 4 в наборе данных MNIST. Для этого требуется:

* подготовить данные для обучения и тестирования;
* реализовать нейронную сеть с одним скрытым слоем и функцией активации "сигмоида";
* провести обучение сети методом обратного распространения ошибки;
* оценить точность модели на тестовой выборке.

# ХОД РАБОТЫ

## 2.1 Подготовка данных.

С помощью функции fetch\_openml загружается набор данных MNIST, содержащий изображения цифр и их метки, как показано на рисунке 1:

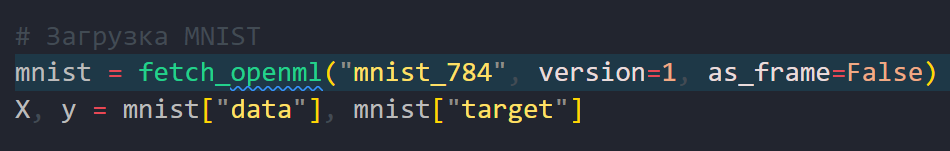


Рисунок 1 – Загрузка данных.

Здесь:

X — массив размерностью (70000, 784), где каждая строка представляет изображение размером 28x28 (развёрнутое в вектор из 784 элементов).

y — метки классов (цифры от 0 до 9).

Далее преобразовываем метки. Метки y преобразуются из строкового формата в целые числа. Это показано на рисунке 2

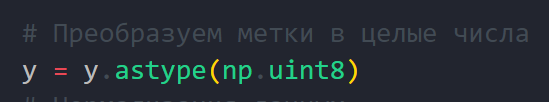


Рисунок 2 – Преобразование меток.

Для ускорения сходимости алгоритма данные нормализуются: значения пикселей (от 0 до 255) приводятся в диапазон [0, 1]. Этот процесс показан на рисунке 3

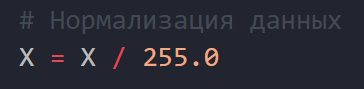


Рисунок 3 – Нормализация данных.

Для упрощения задачи классификации выделяется класс "цифра 4", который получает значение 1, а все остальные цифры — 0. Как происходит классификация показано на рисунке 4.

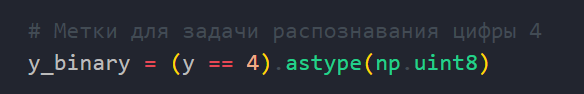


Рисунок 4 – Бинаризация меток.

Для обучения используются первые 5000 примеров, а для тестирования — следующие 7000. Рисунок 5 показывает, как данные разделяются.

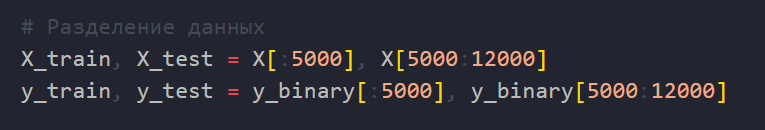


Рисунок 5 – Разделение данных.

Таким образом, данные подготовлены для задачи бинарной классификации: модель должна научиться различать, является ли изображение цифрой "4".

## 2.2 Реализация функций для нейронной сети

На этом этапе реализованы основные функции, необходимые для работы нейронной сети. Их реализации представлены на рисунке 6.

Используется как функция активации для скрытого и выходного слоёв. Она преобразует линейные комбинации входных данных в значения в диапазоне [0,1].

Производная сигмоиды необходима для вычисления градиентов на этапе обратного распространения ошибки.

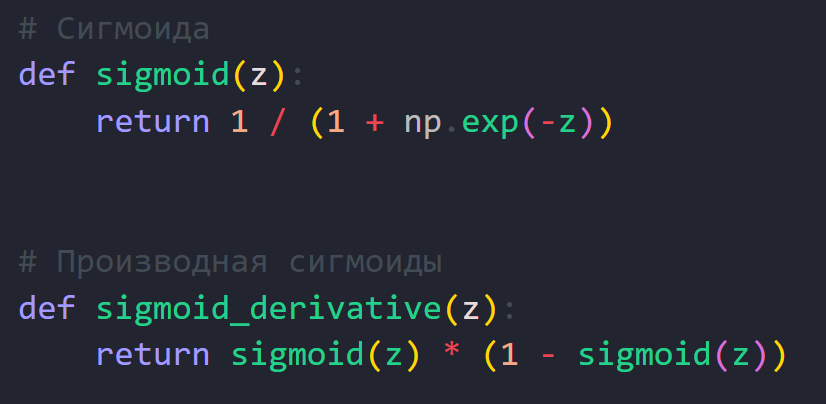


Рисунок 6 – Реализация функции сигмоиды и её производной.

Эти функции обеспечивают основные математические операции, необходимые для работы нейронной сети: активация нейронов, вычисление ошибок и их градиентов.

## 2.3 Архитектура нейронной сети

Нейронная сеть имеет архитектуру с одним скрытым слоем.

Размерность входного слоя соответствует количеству пикселей изображения, т.е. 784 (развёрнутое изображение размером 28×28). На вход подаются нормализованные значения пикселей.

Скрытый слой cодержит 32 нейрона. Каждый нейрон принимает линейную комбинацию входных данных и преобразует её через функцию активации — сигмоиду. Весовая матрица W1 имеет размер 784× 32, смещение b1 — размер 1×32.

На рисунке 7 изображена инициализация параметров. Весовые матрицы W1 и W2 инициализируются случайными числами, умноженными на 0.01, чтобы уменьшить разброс значений. Смещения b1 и b2 инициализируются нулями.

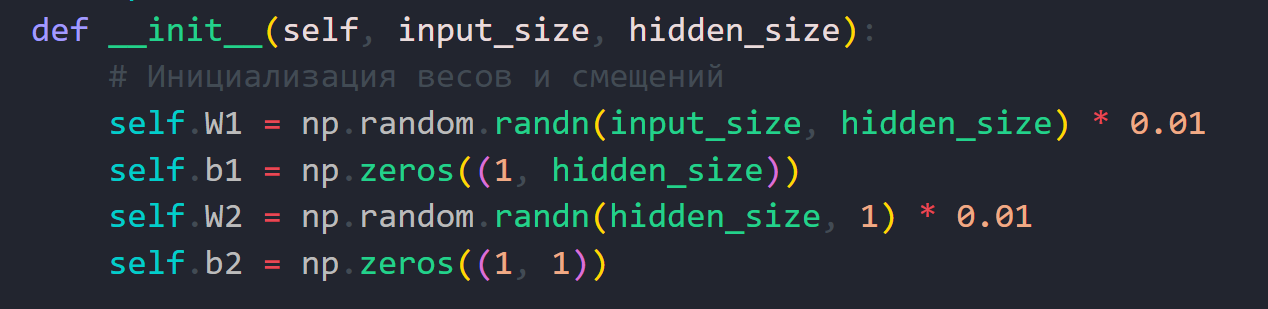


Рисунок 7 – Реализация инициализации класса SimpleNN.

## 2.4 Процесс обучения

Процесс обучения нейронной сети реализован в несколько шагов, которые повторяются на протяжении всех эпох.

На этапе прямого распространения данные проходят через сеть, и на основе текущих весов и смещений вычисляются предсказания. Реализация метода forward представлена на рисунке 8

Сначала рассчитываются линейные комбинации входов и весов для скрытого слоя и кладется в Z1. Затем вычисляется активация скрытого слоя через сигмоиду и кладется в A1. Аналогично происходит для выходного слоя, но уже кладется в Z2. Предсказание (вероятность принадлежности к классу "4") кладется в A2.

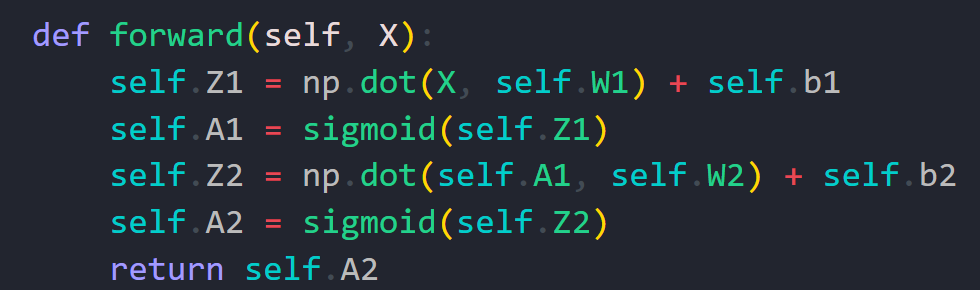


Рисунок 8 – Реализация метода forward.

После прямого распространения вычисляются градиенты для весов и смещений с целью их обновления. Это реализуется в метода backward, который изображен на рисунке 9.

Ошибка выходного слоя считается в dZ2. Градиент для выходного слоя считается в dW2. Ошибка скрытого слоя считается в dZ1. Градиент для скрытого слоя считается в dW1.

После вычисления градиентов параметры сети (веса и смещения) обновляются с использованием правила градиентного спуска.

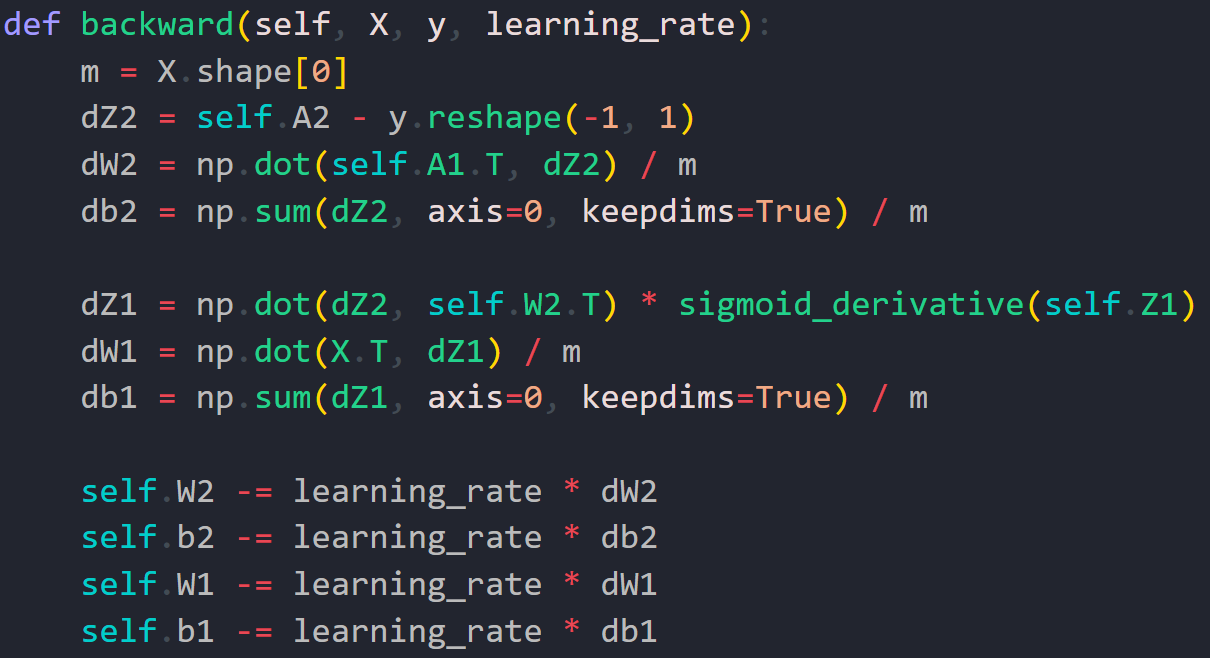


Рисунок 9 – Реализация метода forward.

## 2.5 Цикл обучения

Алгоритм обучения, изображенный на рисунке 10, повторяется в течение заданного числа эпох. На каждой эпохе:

1. Выполняется прямое распространение.
2. Выполняется обратное распространение ошибки.
3. Обновляются параметры.
4. Выводится информация о текущей эпохе, если индекс эпохи кратен 100.

Процесс обучения сети повторяется 1000 раз, что позволяет модели постепенно минимизировать ошибку и корректировать параметры.

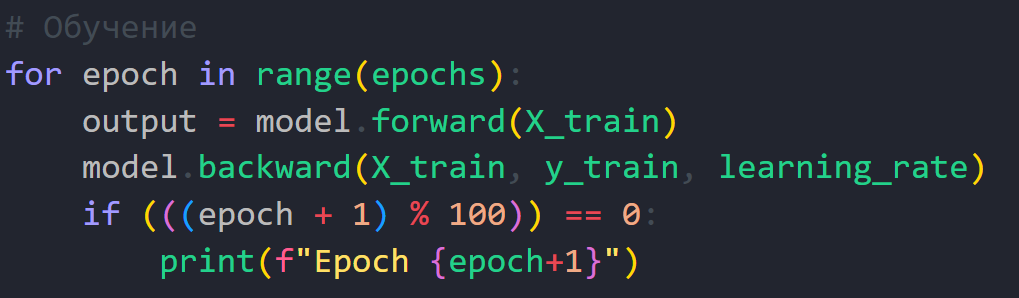


Рисунок 10 – Реализация цикла обучения.

## 2.6 Тестирование модели

На тестовой выборке выполняется прямое распространение данных через обученную сеть. Это изображено на рисунке 11.



Рисунок 11 –Выполнение прямого распространения при тестировании модели.

Вероятности A2 преобразуются в бинарные метки. Если вероятность ≥ 0.5, изображение классифицируется как цифра "4" (метка 1), иначе — как не "4" (метка 0). Это изображено на рисунке 12.



Рисунок 12 – Классификация результатов при тестировании модели

Точность измеряет долю правильно классифицированных изображений среди общего количества. Это происходит по формуле 1.

На рисунке 13 изображено использование метода np.mean, который вычисляет среднее арифметическое значений, полученных после сравнения предсказанных меток y\_pred\_class с истинными метками y\_test.



Рисунок 13 – Нахождение точности предсказаний модели.

Далее выводится итоговая точность модели на тестовой выборке в процентах. Это ключевой показатель успешности обучения модели. На рисунке 12 изображено, как происходит вывод точности



Рисунок 14 – Вывод точности модели.

На рисунке 15 изображен вывод точности модели в консоль после 1000 эпох при learning\_rate = 0.1.

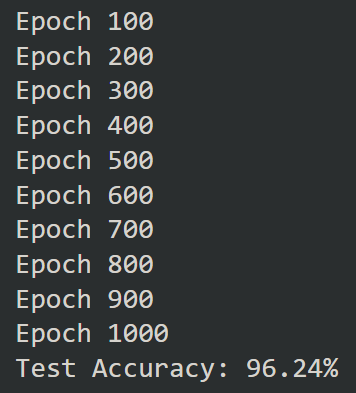


Рисунок 15 – Вывод точности модели в консоль.

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы была разработана и протестирована простая нейронная сеть для бинарной классификации изображений из набора данных MNIST. Целью модели было распознавание цифры "4" среди других цифр.

Работа успешно продемонстрировала, как можно применить базовые знания о нейронных сетях для решения реальных задач классификации.

# ПРИЛОЖЕНИЕ А

Листинг кода

Листинг 1 - index.py

import numpy as np

from sklearn.datasets import fetch\_openml

import matplotlib.pyplot as plt

# Загрузка MNIST

mnist = fetch\_openml("mnist\_784", version=1, as\_frame=False)

X, y = mnist["data"], mnist["target"]

# Преобразуем метки в целые числа

y = y.astype(np.uint8)

# Нормализация данных

X = X / 255.0

# Метки для задачи распознавания цифры 4

y\_binary = (y == 4).astype(np.uint8)

# Разделение данных

X\_train, X\_test = X[:5000], X[5000:12000]

y\_train, y\_test = y\_binary[:5000], y\_binary[5000:12000]

# Сигмоида

def sigmoid(z):

    return 1 / (1 + np.exp(-z))

# Производная сигмоиды

def sigmoid\_derivative(z):

    return sigmoid(z) \* (1 - sigmoid(z))

# Нейронная сеть

class SimpleNN:

    def \_\_init\_\_(self, input\_size, hidden\_size):

        # Инициализация весов и смещений

        self.W1 = np.random.randn(input\_size, hidden\_size) \* 0.01

        self.b1 = np.zeros((1, hidden\_size))

        self.W2 = np.random.randn(hidden\_size, 1) \* 0.01

        self.b2 = np.zeros((1, 1))

    def forward(self, X):

        self.Z1 = np.dot(X, self.W1) + self.b1

        self.A1 = sigmoid(self.Z1)

        self.Z2 = np.dot(self.A1, self.W2) + self.b2

        self.A2 = sigmoid(self.Z2)

        return self.A2

    def backward(self, X, y, learning\_rate):

        m = X.shape[0]

        dZ2 = self.A2 - y.reshape(-1, 1)

        dW2 = np.dot(self.A1.T, dZ2) / m

        db2 = np.sum(dZ2, axis=0, keepdims=True) / m

        dZ1 = np.dot(dZ2, self.W2.T) \* sigmoid\_derivative(self.Z1)

        dW1 = np.dot(X.T, dZ1) / m

        db1 = np.sum(dZ1, axis=0, keepdims=True) / m

        self.W2 -= learning\_rate \* dW2

        self.b2 -= learning\_rate \* db2

        self.W1 -= learning\_rate \* dW1

        self.b1 -= learning\_rate \* db1

# Инициализация сети

model = SimpleNN(input\_size=784, hidden\_size=32)

# Гиперпараметры

epochs = 100\_000

learning\_rate = 0.1

# Обучение

for epoch in range(epochs):

    output = model.forward(X\_train)

    model.backward(X\_train, y\_train, learning\_rate)

    if (((epoch + 1) % 100)) == 0:

        print(f"Epoch {epoch+1}")

# Тестирование

y\_pred = model.forward(X\_test)

y\_pred\_class = (y\_pred >= 0.5).astype(int)

accuracy = np.mean(y\_pred\_class.flatten() == y\_test)

print(f"Test Accuracy: {accuracy \* 100:.2f}%")