БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра информационных систем управления

Чернышева Ксения Юрьевна

**Отчет по заданию 9**

**(“Принятие решений в информационном обществе”)**

студентка 3 курса 12 группы

Преподаватель:

*Железко Борис Александрович*

Минск, 2024

**ОГЛАВЛЕНИЕ**

[Задание 3](#_heading=h.gjdgxs)

[Постановка задания 3](#_heading=h.30j0zll)

[Решение задания 3](#_heading=h.1fob9te)

[Результат работы программы 4](#_heading=h.3znysh7)

# Задание

## Постановка задания

Реализовать алгоритм обратного распространения сети на Python.

## Решение задания

Код со всеми комментариями с объяснением алгоритма по шагам приведен ниже.

import numpy as np

# Функция активации

def sigmoid(x):

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

# Производная функции активации

def sigmoid\_derivative(x):

return x \* (1 - x)

# Функция вычисления ошибки

def calc\_error(y, y\_pred):

return 0.5 \* np.sum((y - y\_pred) \*\* 2)

# Алгоритм обратного распространения

def backpropagation(x, y, learning\_rate, num\_iterations):

# Инициализация случайных весов

np.random.seed(0)

weights1 = np.random.uniform(size=(2, 3))

weights2 = np.random.uniform(size=(3, 1))

for iteration in range(num\_iterations):

# Прямое распространение

hidden\_layer\_input = np.dot(x, weights1)

hidden\_layer\_output = sigmoid(hidden\_layer\_input)

output\_layer\_input = np.dot(hidden\_layer\_output, weights2)

output\_layer\_output = sigmoid(output\_layer\_input)

# Вычисление ошибки

error = calc\_error(y, output\_layer\_output)

# Вычисление градиента на выходном слое

output\_layer\_gradient = (y - output\_layer\_output) \* sigmoid\_derivative(output\_layer\_output)

# Вычисление градиента на скрытом слое

hidden\_layer\_gradient = np.dot(output\_layer\_gradient, weights2.T) \* sigmoid\_derivative(hidden\_layer\_output)

# Обновление весов

weights2 += learning\_rate \* np.dot(hidden\_layer\_output.T, output\_layer\_gradient)

weights1 += learning\_rate \* np.dot(x.T, hidden\_layer\_gradient)

# Вывод ошибки на каждой итерации

print(f"Iteration: {iteration+1}, Error: {error}")

return weights1, weights2

# Входные данные

x = np.array([[0.1, 0.2],

[0.3, 0.4],

[0.5, 0.6]])

y = np.array([[0.3],

[0.4],

[0.5]])

# Параметры обучения

learning\_rate = 0.1

num\_iterations = 1000

# Обучение сети

weights1, weights2 = backpropagation(x, y, learning\_rate, num\_iterations)

# Вывод обученных весов

print("Обученные веса скрытого слоя:")

print(weights1)

print("Обученные веса выходного слоя:")

print(weights2)

## Результат работы программы

