**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**РУТ(МИИТ)**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**Отчет**

**По лабораторной работе №2  
по дисциплине «Нейроинформатика»**

**НА темУ «Настройка искусственного нейрона В УСЛОВИЯХ ПОМЕХ»**

*Направление: 10.03.01* *Информационная безопасность*

*Профиль: Безопасность компьютерных систем*

Выполнил:

Студент группы УИБ-312

Зайцев О. С.

Проверил:

Малинский С. В.

(должность, ФИО)

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc160037374)

[1 ЗАДАНИЕ 5](#_Toc160037375)

[1 АЛГОРИТМ ОБУЧЕНИЯ ИСКУССТВЕННОГО НЕЙРОНА 6](#_Toc160037376)

[2 ДАННЫЕ ДЛЯ ОБУЧЕНИЯ ИСКУССТВЕННОГО НЕЙРОНА 7](#_Toc160037377)

[3 РЕЗУЛЬТАТ ОБУЧЕНИЯ ИСКУССТВЕННОГО НЕЙРОНА 8](#_Toc160037378)

[ЗАКЛЮЧЕНИЕ 17](#_Toc160037379)

[ПРИЛОЖЕНИЕ А 18](#_Toc160037380)

# ВВЕДЕНИЕ

У каждого нейрона, включая искусственный, должны быть входы, через которые он принимает сигналы. Сигналы, проходящие через связи, умножаются на соответствующие веса, изображенные кружками на картинке.

Полученные сигналы на входах умножаются на соответствующие веса. Для первого входа сигнал x\_1 умножается на вес w\_1, для второго - x\_2 умножается на w\_2 и так далее до n-го входа. В итоге последний вход даёт x\_n умножить на w\_n.

После этого произведения передаются в сумматор. Само название сумматора уже говорит о его функции – он просто складывает все входные сигналы, умноженные на соответствующие веса:

net = x\_1 \* w\_1 + x\_2 \* w\_2 + ⋯ + x\_n \* w\_n

Итоговым значением работы сумматора является число, называемое взвешенной суммой (net).

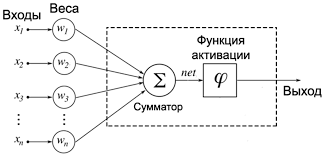


Рис. 1 – искуственный нейрон

Простое присвоение взвешенной суммы net на выходе нейрона бесполезно, поскольку необходимо обработать эту информацию и сформировать соответствующий выходной сигнал. Для решения этой задачи используется функция активации, которая преобразует net в число, являющееся выходом нейрона.

Простейшая функция активации позволяет нейрону выдавать либо 0, либо 1. Если взвешенная сумма превышает определенный порог b, то выход нейрона равен 1, в противном случае - 0.

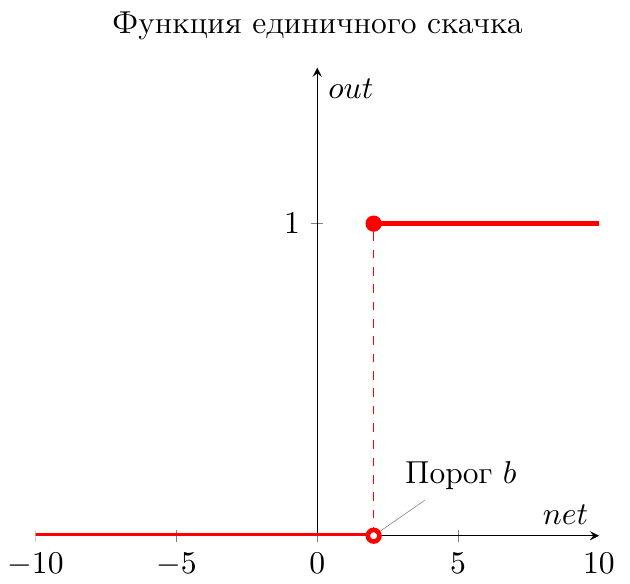


Рис. 2 – графическое представление функции

# 1 ЗАДАНИЕ

Настройка искусственного нейрона для распознавания цифры 2 при описании типов «индекс» в условиях помех. Признаки описания типа индекс представлены на рисунке 1.

Изображение выглядит как текст, число, прямоугольный, кроссворд

Автоматически созданное описание  
Рисунок 1 – Признаки цифр описания типа «индекс»

# 1 АЛГОРИТМ ОБУЧЕНИЯ ИСКУССТВЕННОГО НЕЙРОНА

Для применения алгоритма требуется создать выборку из не менее чем 300 элементов, взятых из набора цифр с указанием типа "индекс" в произвольном порядке. Затем нужно определить веса для каждого из 9 признаков цифры и установить порог распознавания R, превышение которого будет сигнализировать о нахождении цифры 3.

Последовательно подавать цифры из обучающей выборки.

Если число соответствует цифре 3 и превышает порог, переходим к следующему числу. Если число отличается от цифры 3, увеличиваем соответствующие веса.

Если число не соответствует цифре 3 и не превышает порог, переходим к следующему числу. Если число соответствует цифре 3, но не превышает порог, уменьшаем соответствующие веса.

Последовательно вносим ошибки от 1 до 9 в выборку, измеряя вероятность ошибки в случайном месте.

# 

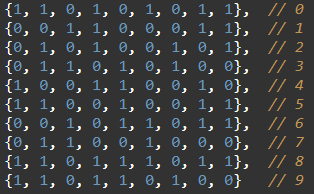
# 2 ДАННЫЕ ДЛЯ ОБУЧЕНИЯ ИСКУССТВЕННОГО НЕЙРОНА

Исходное число для распознавания:

Начальный набор весов: [ 1, 6, 9, 3, 4, 3, 8, 0, 7]

Порог распознавания(R): 26.

Обучающая выборка состоит из 300 цифр с описанием типа «индекс», расположенных в случайном порядке. Часть выборки можно увидеть на рисунке 2.

  
Рисунок 2 – часть обучающей выборки

# 3 РЕЗУЛЬТАТ ОБУЧЕНИЯ ИСКУССТВЕННОГО НЕЙРОНА

В процессе обучения мы изменяли веса исходного вектора w, следовательно, результатом обучения будет являться тот вектор, при котором выборка была пройдена целиком без ошибок, или же количество ошибок оставалось на одном уровне. Конечные вектора весов и вероятность ошибки для разного количества помех можно увидеть на рисунке 3.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание  
Рисунок 3 – Результат работы программы

На графике, изображенном на рисунке 4, отображена зависимость вероятности ошибки от количества помех.

Рисунок 4 – График зависимости ошибок от помех.

Изменения весов относительно изначального для разного количества помех и график ошибок при обучении с 1 - 9 помехами можно наблюдать на рисунках 5 - 13.

Рисунок 5 – График изменения для 1 помехи

Рисунок 6 – График изменения для 2 помех

Рисунок 7 – График изменения для 3 помех

Рисунок 8 – График изменения для 4 помех

Рисунок 9 – График изменения для 5 помех

Рисунок 10 – График изменения для 6 помех

Рисунок 11 – График изменения для 7 помех

Рисунок 12 – График изменения для 8 помех

Рисунок 13 – График изменения для 9 помех

# ЗАКЛЮЧЕНИЕ

В результате выполнения данного задания был обучен искусственный нейрон для распознавания цифры 3 с описанием признаков типа «индекс» в условии помех.

# ПРИЛОЖЕНИЕ А

#include <iostream>

#include <vector>

#include <cstdlib>

using namespace std;

int main() {

vector<vector<int>> training\_data = {

{1, 1, 0, 1, 0, 1, 0, 1, 1}, // 0

{0, 0, 1, 1, 0, 0, 0, 1, 1}, // 1

{0, 1, 0, 1, 0, 0, 1, 0, 1}, // 2

{0, 1, 1, 0, 1, 0, 1, 0, 0}, // 3

{1, 0, 0, 1, 1, 0, 0, 1, 0}, // 4

{1, 1, 0, 0, 1, 0, 0, 1, 1}, // 5

{0, 0, 1, 0, 1, 1, 0, 1, 1}, // 6

{0, 1, 1, 0, 0, 1, 0, 0, 0}, // 7

{1, 1, 0, 1, 1, 1, 0, 1, 1}, // 8

{1, 1, 0, 1, 1, 0, 1, 0, 0} // 9

};

vector<int> weights(9);

for (int i = 0; i < 9; ++i) {

weights[i] = rand() % 10; // Инициализация случайными весами от 0 до 9

}

int Q = 26;

bool error = true;

int epochs = 0;

while (error) {

error = false;

for (int num = 0; num < training\_data.size(); ++num) {

const vector<int>& digit = training\_data[num];

int net = 0;

for (int i = 0; i < 9; ++i) {

net += weights[i] \* digit[i];

}

if ((num != 3 && net >= Q) || (num == 3 && net < Q)) {

error = true;

if (num != 3) {

for (int i = 0; i < 9; ++i) {

weights[i] -= digit[i];

}

cout << "Уменьшение весов на признаках для цифры " << num << ": ";

} else {

for (int i = 0; i < 9; ++i) {

weights[i] += digit[i];

}

cout << "Увеличение весов на признаках для цифры " << num << ": ";

}

for (int i = 0; i < 9; ++i) {

cout << weights[i] << " ";

}

cout << endl;

}

}

if (!error) {

break;

}

epochs++;

}

for (int num = 0; num < training\_data.size(); ++num) {

const vector<int>& digit = training\_data[num];

int control\_sum = 0;

for (int i = 0; i < 9; ++i) {

control\_sum += weights[i] \* digit[i];

}

cout << "Контрольная сумма для цифры " << num << ": " << control\_sum << endl;

}

cout << "Конечный вектор весов: ";

for (int i = 0; i < 9; ++i) {

cout << weights[i] << " ";

}

cout << endl;

cout << "Порог Q: " << Q << endl;

// Подсчет вероятности ошибки

int total\_tests = training\_data.size() \* 10; // 10 тестов для каждой цифры с помехами от 1 до 9

for (int i = 1; i <= 9; ++i) {

int num\_errors = 0;

for (int num = 0; num < training\_data.size(); ++num) {

const vector<int>& digit = training\_data[num];

for (int j = 0; j < 10; ++j) {

vector<int> perturbed\_digit;

perturbed\_digit.reserve(9);

for (int k = 0; k < 9; ++k) {

perturbed\_digit.push\_back(digit[k] + rand() % (2 \* i + 1) - i);

if (perturbed\_digit[k] < 0) {

perturbed\_digit[k] = 0;

} else if (perturbed\_digit[k] > 1) {

perturbed\_digit[k] = 1;

}

}

int net = 0;

for (int k = 0; k < 9; ++k) {

net += weights[k] \* perturbed\_digit[k];

}

if ((num != 3 && net >= Q) || (num == 3 && net < Q)) {

num\_errors++;

}

}

}

double probability\_of\_error = static\_cast<double>(num\_errors) / total\_tests;

cout << "Вероятность ошибки после внедрения ошибки " << i << ": " << probability\_of\_error << endl;

}

// Точность данных с помехами от 1 до 9

for (int i = 1; i <= 9; ++i) {

int num\_errors = 0;

for (int j = 0; j < 30; ++j) { // Повторяем тесты для усреднения результатов

vector<vector<int>> perturbed\_data;

perturbed\_data.reserve(training\_data.size());

for (const auto& digit : training\_data) {

vector<int> perturbed\_digit;

perturbed\_digit.reserve(9);

for (int k = 0; k < 9; ++k) {

perturbed\_digit.push\_back(digit[k] + rand() % (2 \* i + 1) - i);

if (perturbed\_digit[k] < 0) {

perturbed\_digit[k] = 0;

} else if (perturbed\_digit[k] > 1) {

perturbed\_digit[k] = 1;

}

}

perturbed\_data.push\_back(perturbed\_digit);

}

for (int num = 0; num < training\_data.size(); ++num) {

const vector<int>& digit = perturbed\_data[num];

int net = 0;

for (int k = 0; k < 9; ++k) {

net += weights[k] \* digit[k];

}

if ((num != 3 && net >= Q) || (num == 3 && net < Q)) {

num\_errors++;

}

}

}

double accuracy = 100.0 - (static\_cast<double>(num\_errors) / (training\_data.size() \* 30 \* 9)) \* 100.0;

cout << "Точность данных с помехой " << i << ": " << accuracy << "%" << endl;

}

return 0;

}