**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**)

**Институт управления и цифровых технологий**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**Отчет по Лабораторной работе №3**

**«Обучение однослойного персептрона»**

**По дисциплине «Нейроинформатика»**

*Направление:* 10.03.01*Информационная безопасность*

*Профиль:**Безопасность компьютерных систем*

Выполнил:   
студент группы УИБ-311

Москаленко Виталий Александрович

Проверил:

Доцент Малинский С.В.

(должность, ФИО)

**МОСКВА 2024**

**СОДЕРЖАНИЕ**

[1 ЗАДАНИЕ 3](#_Toc178106761)

[2 ОСНОВНЫЕ ПОНЯТИЯ ОБ ОДНОСЛОЙНОМ ПЕРСЕПТРОНЕ 6](#_Toc178106762)

[3 ФОРМИРОВАНИЕ ОБУЧАЮЩЕЙ ВЫБОРКИ 8](#_Toc178106763)

[4 ОБУЧЕНИЕ ОДНОСЛОЙНОГО ПЕРСЕПТРОНА 9](#_Toc178106764)

[5 РЕЗУЛЬТАТ ОБУЧЕНИЯ ОДНОСЛОЙНОГО ПЕРСЕПТРОНА 11](#_Toc178106766)

[ЗАКЛЮЧЕНИЕ 13](#_Toc178106767)

[ПРИЛОЖЕНИЕ 14](#_Toc178106768)

1 ЗАДАНИЕ

Необходимо разработать программу, которая реализует обучение однослойного персептрона для распознавания цифр от 0 до 9 на основе их графических представлений, состоящих из сегментов шаблона индекс.

На рисунке 1 представлен шаблон, в котором каждая цифра изображается набором отрезков, расположенных в 9 сегментах.

Например:

- Признак "1" обозначает наличие сегмента в соответствующем ячейке.

- Признак "0" обозначает её отсутствие.

Изображение выглядит как линия, диаграмма

Автоматически созданное описание

Рисунок 1 – Шаблон графического изображения цифр

В таблице 1 представлен полный набор цифр, каждая из которых соответствует своему набору признаков. Эти данные используются для формирования обучающей выборки, на которой персептрон будет обучаться распознавать цифры.

Таблица 1 – Цифры и их представления

|  |  |  |
| --- | --- | --- |
| Цифра | Графическое представление | Двоичное представление |
| 0 |  | 1, 1, 0, 1, 0, 1, 0, 1, 1 |
| 1 |  | 0, 0, 1, 1, 0, 0, 0, 1, 0 |
| 2 |  | 0, 1, 0, 1, 0, 0, 1, 0, 1 |
| 3 |  | 0, 1, 1, 0, 1, 0, 1, 0, 0 |
| 4 |  | 1, 0, 0, 1, 1, 0, 0, 1, 0 |

Продолжение таблицы 1:

|  |  |  |
| --- | --- | --- |
| 5 |  | 1, 1, 0, 0, 1, 0, 0, 1, 1 |
| 6 |  | 0, 0, 1, 0, 1, 1, 0, 1, 1 |
| 7 |  | 0, 1, 1, 0, 0, 1, 0, 0, 0 |
| 8 |  | 1, 1, 0, 1, 1, 1, 0, 1, 1 |
| 9 |  | 1, 1, 0, 1, 1, 0, 1, 0, 0 |

На основе данного шаблона для каждой цифры были сформированы обучающие данные, которые содержат бинарный вектор признаков (массив из 9 значений). Этот вектор однозначно описывает графическое представление каждой цифры.

2 ОСНОВНЫЕ ПОНЯТИЯ ОБ ОДНОСЛОЙНОМ ПЕРСЕПТРОНЕ

Однослойный персептрон — это простейшая модель нейронной сети. В этой модели искусственные нейроны обрабатывают входные данные и классифицируют их, используя определённые правила.

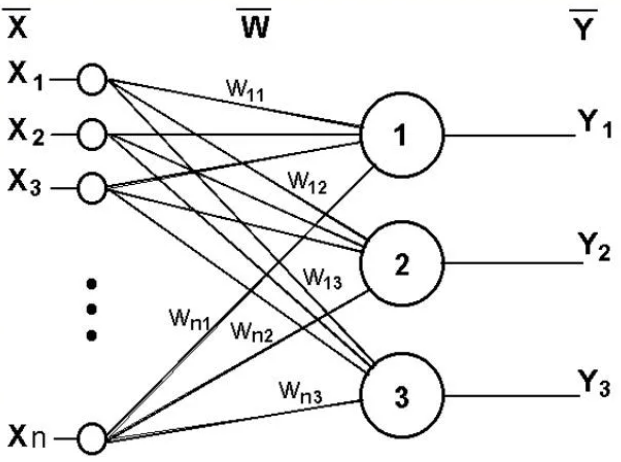


Рисунок 2 – однослойный персептрон

Основные элементы однослойного персептрона:

* Входные данные (признаки):

Персептрон получает набор числовых значений, которые представляют собой признаки (сегменты шаблона индекс). В задаче распознавания цифр каждая цифра от 0 до 9 представлена 9 признаками. Эти признаки поступают на вход каждого нейрона персептрона.

* Веса (коэффициенты):

Каждый входной признак умножается на соответствующий ему вес. Веса определяют важность каждого признака для процесса распознавания модели персептрона и выделяют ключевые признаки цифр.

* Функция активации:

После умножения входных признаков на соответствующие веса, результаты суммируются. Эта сумма передаётся на вход функции активации, которая решает, будет ли персептрон активирован.

* Выход (результат):

Персептрон генерирует выходной сигнал на основе работы функции активации. В контексте задачи распознавания цифр этот сигнал представляет собой классификацию цифр от 0 до 9.

3 ФОРМИРОВАНИЕ ОБУЧАЮЩЕЙ ВЫБОРКИ

Для успешного обучения однослойного персептрона, способного распознавать цифры по их бинарному представлению, необходимо сформировать обучающую выборку. В нашем случае цифры представлены в виде набора признаков. Эти признаки в виде бинарных значений описывают наличие или отсутствие сегментов в шаблоне индекс.

Используя набор данных, персептрон сможет по входным признакам корректно определять, какая цифра перед ним. Обучающая выборка создаётся путём генерации множества примеров этих векторов и передачи их в алгоритм обучения персептрона.

Этот процесс позволяет обучить систему эффективно распознавать цифры од 0 до 9, основываясь на уникальных признаках каждой цифры.

4 ОБУЧЕНИЕ ОДНОСЛОЙНОГО ПЕРСЕПТРОНА

После того, как обучающая выборка была сформирована, запускается алгоритм обучения на распознавание цифр.

Эпохой называется полное прохождение алгоритма по обучающей выборке. Для корректного распознавания программе может потребоваться одна или более эпоха.

На каждой эпохе вычисляются суммы весов для каждого элемента обучающей выборки, а также находятся максимумы среди этих сумм.

Начальная матрица весов выглядит следующим образом:

{0,1,4,7,2,-8,1,9,2}

{7,3,9,4,9,0,1,7,4}

{1,7,9,2,6,0,7,6,0}

{3,9,8,6,-9,6,2,0,0}

{8,7,0,3,-9,3,7,0,0}

{9,0,8,8,4,7,4,9,3}

{2,8,7,4,-1,7,9,1,1}

 {9,8,4,0,8,7,0,8,7}

 {5,7,8,5,7,8,4,0,2}

 {9,4,8,5,7,1,9,-8,5}

В зависимости от того, верно ли классифицируется элемент, происходит корректировка весов:

1. Если максимум относится к распознаваемой цифре, и он один, то веса

не изменяются и алгоритм начинает работать со следующей цифрой в выборке.

1. Если максимум не относится к распознаваемой цифре, то веса всех максимумов, кроме распознаваемой цифры, уменьшаются, а у распознаваемой – увеличиваются.

В конечном итоге это позволяет получить новую матрицу весов. В конце каждой эпохи выводится точность модели, то есть процент правильно классифицированных объектов.

5 РЕЗУЛЬТАТ ОБУЧЕНИЯ ОДНОСЛОЙНОГО ПЕРСЕПТРОНА

В результате выполнения программы были зафиксированы итерации по обучению персептрона для распознавания цифр от 0 до 9 на основе их признаков. Основной целью обучения было создание новой матрицы весов, в которой при подсчете сумм весов максимум этих сумм должен был быть только у распознаваемой цифры.

Программа реализует обучение распознавания цифр от 0 до 9 на трех эпохах. Результаты весов после прохождения каждой из трех эпох представлены на рисунке 3:

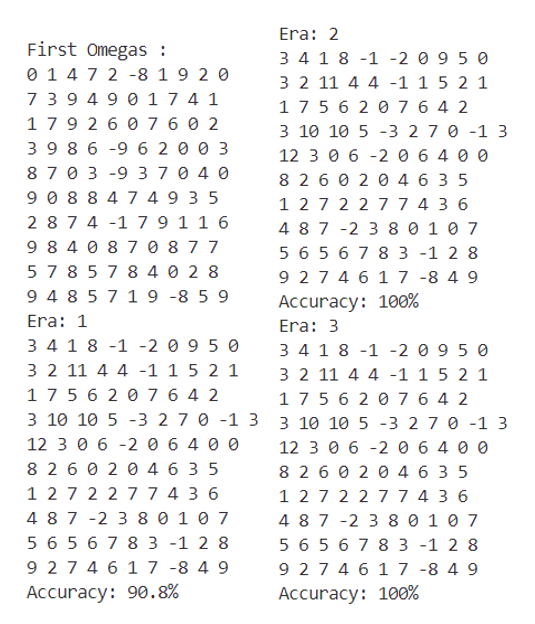


Рисунок 3 – Результат работы программы

Далее была выполнена проверка для каждой цифры, чтобы убедиться в правильности обучения персептрона.

Ниже представлены результаты проверки – как видно, для всех цифр при нахождении сумм весов максимум этих сумм соответствует рассматриваемой цифре:

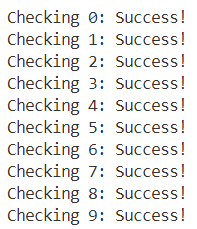


Рисунок 4 – Результаты проверки всех цифр

ЗАКЛЮЧЕНИЕ

По завершении обучения однослойный персептрон успешно обучился распознавать цифры от 0 до 9. Распознавание происходило путём корректировки весов в процессе обучения, что продемонстрировало успешную работу алгоритма.

Этот результат подтверждает, что однослойный персептрон, обучаемый на принципах простых механизмов корректировки весов, может быть обучен распознавать необходимые пользователю символы по заданным признакам.

ПРИЛОЖЕНИЕ

#include <iostream>

#include <algorithm>

#include <ctime>

#include <cstring>

using namespace std;

int main()

{

    int maxeras=3;

    int kolich=0;

    int sums[10]={0};

    int cifri[10][10]=

    {

        {1,1,0,1,0,1,0,1,1,0}, //0

        {0,0,1,1,0,0,0,1,0,1}, //1

        {0,1,0,1,0,0,1,0,1,2}, //2

        {0,1,1,0,1,0,1,0,0,3}, //3

        {1,0,0,1,1,0,0,1,0,4}, //4

        {1,1,0,0,1,0,0,1,1,5}, //5

        {0,0,1,0,1,1,0,1,1,6}, //6

        {0,1,1,0,0,1,0,0,0,7}, //7

        {1,1,0,1,1,1,0,1,1,8}, //8

        {1,1,0,1,1,0,1,0,0,9}  //9

    };

    int Omegi[10][10]=

    {

        {0,1,4,7,2,-8,1,9,2,0},

        {7,3,9,4,9,0,1,7,4,1},

        {1,7,9,2,6,0,7,6,0,2},

        {3,9,8,6,-9,6,2,0,0,3},

        {8,7,0,3,-9,3,7,0,4},

        {9,0,8,8,4,7,4,9,3,5},

        {2,8,7,4,-1,7,9,1,1,6},

        {9,8,4,0,8,7,0,8,7,7},

        {5,7,8,5,7,8,4,0,2,8},

        {9,4,8,5,7,1,9,-8,5,9}

    };

    int randommassiv[500]=

    {

        0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

        1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,

        2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,

        3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,

        4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,

        5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,

        6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,

        7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,

        8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,

        9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9

    };

    random\_shuffle(randommassiv, randommassiv+500);

    int viborka[500][10];

    for (int i=0;i<500;i++)

    {

        for (int j=0;j<10;j++)

        {

            viborka[i][j]=cifri[randommassiv[i]][j];

        }

    }

сout << "Author - Moskalenko Vitaly Alexandrovich UIB-311";

    cout << endl;

    cout << "Training sample: " << endl;

    for (int i=0;i<500;i++)

    {

        cout << i+1 << " | ";

        for (int j=0;j<10;j++)

        {

            cout << viborka[i][j] << ' ';

        }

        cout << "|";

        cout << endl;

    }

    cout << endl;

    cout << "First Omegas :" << endl;

    for (int i=0;i<10;i++)

    {

        for (int j=0;j<10;j++)

        {

            cout << Omegi[i][j] << ' ';

        }

        cout << endl;

    }

    while (kolich<maxeras)

    {

        kolich++;

        int sovpad=0;

        for (int i=0;i<500;i++)

        {

            for (int j=0;j<10;j++)

            {

                sums[j]=0;

                for (int x=0;x<9;x++)

                {

                    sums[j]+=Omegi[j][x]\*viborka[i][x];

                }

            }

            int max=\*max\_element(sums, sums+10);

            int kolvomax=0;

            for (int i=0;i<10;i++)

            {

                if (sums[i]==max)

                {

                    kolvomax++;

                }

            }

            if ((sums[viborka[i][9]]==max)&&(kolvomax==1))

            {

                sovpad++;

            }

            else

            {

                for (int x=0;x<10;x++)

                {

                    if ((sums[x]==max)&&(x!=viborka[i][9]))

                    {

                        for (int j=0;j<9;j++)

                        {

                            Omegi[x][j]-=viborka[i][j];

                        }

                    }

                }

                for (int j=0;j<9;j++)

                {

                    Omegi[viborka[i][9]][j]+=viborka[i][j];

                }

            }

        }

        cout << endl;

        cout <<  "Era: " << kolich << endl;

        for (int i=0;i<10;i++)

        {

            for (int j=0;j<10;j++)

            {

                cout << Omegi[i][j] << ' ';

            }

            cout << endl;

        }

        double tochnost=(double)sovpad/500\*100;

        cout << "Accuracy: " << tochnost << "%" << endl;

    }

    cout << endl;

    for (int i=0;i<10;i++)

    {

        cout << "Checking " << i << ": ";

        bool uspex=true;

        for (int x=0;x<10;x++)

        {

            sums[x]=0;

            for (int j=0;j<9;j++)

            {

                sums[x]+=Omegi[x][j]\*cifri[i][j];

            }

        }

        int max=\*max\_element(sums, sums+10);

        int kolvomax=0;

        for (int i=0;i<10;i++)

        {

            if (sums[i]==max)

            {

                kolvomax++;

            }

        }

        if ((sums[i]!=max)||(kolvomax>1))

        {

            uspex=false;

        }

        if (uspex)

        {

            cout << "Success!" << endl;

        }

        else

        {

            cout << "Fail!" << endl;

        }

    }

    return 0;

}