МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования

информационных систем

Направление подготовки «Математическое обеспечение и

администрирование информационных систем»

Форма обучения – очная

**Отчёт**

**о выполнении лабораторной работы № 5**

«Алгоритм Видроу-Хоффа обучения нейронных сетей»

###### Дисциплина «Основы теории нейронных сетей»

|  |  |
| --- | --- |
| Выполнил: | студент группы 413  Мусонда Салиму |
| Проверил: | проф.  кафедры ПОиАИС  Добрица В.П. |

Курск, 2021

**Цель работы:** освоить методы обучения нейронных сетей по правилу Видроу-Хоффа с постоянным и адаптивным коэффициентом скорости обучения.

**Задача № 1.** Провести обучение однослойной нейронной сети логической функции конъюнкции в биполярном случае функции по алгоритму Видроу-Хоффа при  и .

**Задача № 2**. Провести обучение однослойной нейронной сети для логической функции дизъюнкции с помощью алгоритма Видроу-Хоффа с адаптивным шагом обучения при .

**Выполнение работы**

**Задача 1.**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| -1 | -1 | -1 |
| -1 | 1 | -1 |
| 1 | -1 | -1 |
| 1 | 1 | 1 |

Для реализации алгоритма Видроу-Хоффа понадобятся следующие формулы:

(1) – выходное значение нейросети, где – весовые коэффициенты, – значение пороговой функции

(2) – весовые коэффициенты, где – предыдущие весовые коэффициенты, – скорость обучения, и – выходное и эталонное значение нейросети, - входное значение нейросети.

(3) – пороговая функция.

(4) – среднеквадратичная ошибка нейросети для входного образа L.

**Блок-схема алгоритма:**



Рисунок 1 – Блок схема алгоритма Видроу-Хоффа без адаптивного шага

**Задача 2.**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Для реализации алгоритма Видроу-Хоффа понадобятся формулы из предыдущей задачи и формула нахождения адаптивного шага обучения:

**Блок-схема алгоритма:**



Рисунок 3 – Блок схема алгоритма Видроу-Хоффа с адаптивным шагом

**Код программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace NN

{

public partial class Form1 : Form

{

private double E = 0.3;

private double Alph = 0.3;

private List<double> OtvetW = new List<double>();

private double OtvetT = 0;

private List<double> wansW = new List<double>();

private double wansT = 0;

public Form1()

{InitializeComponent();

this.dataGridView2.Rows.Add(-1, -1, -1);

this.dataGridView2.Rows.Add(-1, 1, -1);

this.dataGridView2.Rows.Add(1, -1, -1);

this.dataGridView2.Rows.Add(1, 1, 1);

this.dataGridView1.Rows.Add(0, 0, 0);

this.dataGridView1.Rows.Add(0, 1, 1);

this.dataGridView1.Rows.Add(1, 0, 1);

this.dataGridView1.Rows.Add(1, 1, 1);

}

// Найти функцию активации для алг. без адаптивного шага

private void button1\_Click(object sender, EventArgs e)

{

double[,] Nabor = new double[dataGridView2.RowCount - 1, dataGridView2.ColumnCount];

List<double> W = new List<double>();

for (int i = 0; i < dataGridView2.RowCount-1; i++)

//заполнение массива входных данных

for (int j = 0; j < dataGridView2.ColumnCount; j++)

Nabor[i, j] = Convert.ToDouble(dataGridView2.Rows[i].Cells[j].Value);

for (int i = 0; i < dataGridView2.ColumnCount - 1; i++)

W.Add(1);

double T = 1, TempE = 10;

double LastE = 0;

while (TempE >= E)

{

TempE = 0;

for (int row = 0; row < dataGridView2.RowCount-1; row++)

{

double TempY = 0;

for (int i = 0; i < dataGridView2.ColumnCount - 1; i++)

TempY += Nabor[row, i] \* W[i];//высчитываем значение для каждой строки

TempY -= T;

if (TempY > 0)

TempY = 1;

else

TempY = 0;

TempE += Math.Pow((Nabor[row, dataGridView2.ColumnCount - 1] - TempY), 2);//квадратичное отклонение

for (int k = 0; k < dataGridView2.ColumnCount - 1; k++)

W[k] = W[k] + Alph \* (Nabor[row, dataGridView2.ColumnCount - 1] - TempY) \* Nabor[row, k];//(1)

T -= Alph \* (Nabor[row, dataGridView2.ColumnCount - 1] - TempY);//(2)

}

TempE \*= 0.5;

if (Math.Abs(TempE - LastE) < 0.00001)

break;

LastE = TempE;

}

string resultForm = "";

for (int k = 0; k < W.Count; k++)

{

if (W[k] >= 0)

resultForm += "x" + (k + 1).ToString() + " \* " + Math.Round(W[k], 2).ToString() + " + ";

else

resultForm += "x" + (k + 1).ToString() + " \* (" + Math.Round(W[k], 2).ToString() + ") + ";

}

resultForm = resultForm.Substring(0, resultForm.LastIndexOf("+"));

if (T >= 0)

resultForm += "- " + Math.Round(T, 2).ToString();

else

resultForm += "- " + "(" + Math.Round(T, 2).ToString() + ")";

resultForm = "y = sgn(" + resultForm + ")";

OtvetW = W;

OtvetT = T;

label8.Text = resultForm;

}

// Тестирование для алг. без адаптивного шага

private void button2\_Click(object sender, EventArgs e)

{

bool right = true;

string res = "y\* = (";

for (int row = 0; row < dataGridView2.RowCount-1; row++)

{

double y = 0;

for (int i = 0; i < OtvetW.Count; i++)

y += OtvetW[i] \* Convert.ToDouble(dataGridView2.Rows[row].Cells[i].Value);

y -= OtvetT;

if (y > 0)

y = 1;

else

y = -1;

res += y.ToString() + "; ";

if (y != Convert.ToDouble(dataGridView2.Rows[row].Cells[dataGridView2.ColumnCount - 1].Value))

right = false;

}

res = res.Substring(0, res.LastIndexOf(";"));

res += ")";

label10.Text = res;

if (right)

MessageBox.Show("Тестирование проведено без ошибок", "Test");

else

MessageBox.Show("Ошибка при тестировании", "Error");

}

/// Поиск функции с адаптивным шагом

private void button4\_Click(object sender, EventArgs e)

{

double[,] Nabor = new double[dataGridView1.RowCount - 1, dataGridView1.ColumnCount];

List<double> W = new List<double>();

for (int i = 0; i < dataGridView1.RowCount-1; i++)//заполнение массива входных данных

for (int j = 0; j < dataGridView1.ColumnCount; j++)

Nabor[i,j] = Convert.ToDouble(dataGridView1.Rows[i].Cells[j].Value);

for (int i = 0; i < dataGridView1.ColumnCount - 1; i++)

W.Add(1);

double T = 1, TempE = 5;

double LastE = 0;

while (TempE >= E)

{

TempE = 0;

for (int row = 0; row < dataGridView1.RowCount-1; row++)

{

double alph = 0;

for (int col = 0; col < dataGridView1.ColumnCount - 1; col++)

alph += Math.Pow(Nabor[row, col], 2);

alph = 1 / (1 + alph);//(теорема)

double TempY = 0;

for (int i = 0; i < dataGridView1.ColumnCount - 1; i++)

TempY += Nabor[row, i] \* W[i];//высчитываем значение для каждой строки

TempY -= T;

TempE += Math.Pow((Nabor[row, dataGridView1.ColumnCount - 1] - TempY), 2);//квадратичное отклонение

for (int k = 0; k < dataGridView1.ColumnCount - 1; k++)

W[k] = W[k] + alph \* (Nabor[row, dataGridView1.ColumnCount - 1] - TempY) \* Nabor[row, k];//(1)

T -= alph \* (Nabor[row, dataGridView1.ColumnCount - 1] - TempY);//(2)

}

TempE \*= 0.5;

if (Math.Abs(TempE - LastE) < 0.00001)

break;

LastE = TempE;

}

string rez = "";

for (int k = 0; k < W.Count; k++)

{

if (W[k] >= 0)

rez += "x" + (k + 1).ToString() + " \* " + Math.Round(W[k], 2).ToString() + " + ";

else

rez += "x" + (k + 1).ToString() + " \* (" + Math.Round(W[k], 2).ToString() + ") + ";

}

rez = rez.Substring(0, rez.LastIndexOf("+"));

if (T >= 0)

rez += "- " + Math.Round(T, 2).ToString();

else

rez += "- " + "(" + Math.Round(T, 2).ToString() + ")";

rez = "y =" + rez;

wansW = W;

wansT = T;

label9.Text = rez;

}

private void button3\_Click(object sender, EventArgs e)

{

bool right = true;

string res = "y\* = (";

for (int row = 0; row < dataGridView1.RowCount-1; row++)

{

double y = 0;

for (int i = 0; i < wansW.Count; i++)

y += wansW[i] \* Convert.ToDouble(dataGridView1.Rows[row].Cells[i].Value);

y -= wansT;

if (y > 0)

y = 1;

else

y = 0;

res += y.ToString() + "; ";

if (y != Convert.ToDouble(dataGridView1.Rows[row].Cells[dataGridView1.ColumnCount - 1].Value))

right = false;

}

res = res.Substring(0, res.LastIndexOf(";"));

res += ")";

label11.Text = res;

if (right)

MessageBox.Show("Тестирование проведено без ошибок", "Test");

else

MessageBox.Show("Ошибка при тестировании", "Error");

}

}

}

В результате выполнения программы были получены следующие формулы

Задача 1: y=sgn (1.6\*x1+1.6\*x2-2.8)

Задача 2: y=sgn (0.25\*x1+0.5\*x2-0.25)