Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчёт**

**по лабораторной работе №16**

**Дисциплина: НЕЙРОСЕТЕВЫЕ И НЕЧЕТКИЕ МОДЕЛИ**

**Тема: «Алгоритм обратного распространения ошибки»**

Работу выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. А. Корнилов

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии

Направленность (профиль) Математическое и программное обеспечение

компьютерных технологий

Преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Тема:** Алгоритм обратного распространения ошибки.

**Цель:** Изучение процедуры обновления весов матрицы с помощью алгоритма обратного распространения ошибки в системе MATLAB.

**Задания:**

1. Для заданного преподавателем варианта задания выполнить 1 итерацию ручной расчет настройки весов и смещений персептронной нейронной сети.



1. Разработать алгоритм создания и моделирования персептронной нейронной сети
2. Реализовать алгоритм в системе MATLAB
3. Определить параметры созданной нейронной сети и проверить правильность работы сети для последовательности входных векторов
4. Сравнить результаты ручных расчетов и расчетов, выполненных в системе MATLAB

**Ход работы:**

1. Был выполнен ручной расчет настройки весов и смещений двухслойной нейронной сети с весами

Входные данные: x = [ 0 2 4] target = 0

w1 = [0.5 0.5 0.5; 0.5 0.5 0.5; 0.5 0.5 0.5;];

w2 = [0.5 0.5 0.5;];

Прямое распространение

S1 = w1(1,1) \* x1(i) + w1(1,2) \* x2(i) + w1(1,3) \* x3(i) = 3

y1 = sigmoid(S1) = 0.9526

S2 = w1(2,1) \* x1(i) + w1(2,2) \* x2(i) + w1(2,3) \* x3(i) = 3

y2 = sigmoid(S2) = 0.9526

S3 = w1(3,1) \* x1(i) + w1(3,2) \* x2(i) + w1(3,3) \* x3(i) = 3

y3 = sigmoid(S3) = 0.9526

s = w2(1,1) \* y1 + w2(1,2) \* y2 + w2(1,3)\*y3 = 1.4289

y = sigmoid(s) = 0.8067

Вычисление градиентов для каждого из слоев:

gradOutput = (y-target) \* y \* (1-y) = 0.1258

gradOutput21 = y1 \* (1-y1) \* gradOutput \* w2(1,1) = 0.0028

gradOutput22 = y2 \* (1-y2) \* gradOutput \* w2(1,2) = 0.0028

gradOutput23 = y3 \* (1-y3) \* gradOutput \* w2(1,3) = 0.0028

Обновление весов нейронной сети

w1(1,1) = w1(1,1) - learningRage \* x1(i) \* gradOutput21 = 0.5

w1(1,2) = w1(1,2) - learningRage \* x1(i) \* gradOutput22 = 0.5

w1(1,3) = w1(1,3) - learningRage \* x1(i) \* gradOutput23 = 0.5

w1(2,1) = w1(2,1) - learningRage \* x2(i) \* gradOutput21 = 0.4999

w1(2,2) = w1(2,2) - learningRage \* x2(i) \* gradOutput22 = 0.4999

w1(2,3) = w1(2,3) - learningRage \* x2(i) \* gradOutput23 = 0.4999

w1(3,1) = w1(3,1) - learningRage \* x3(i) \* gradOutput21 = 0.4999

w1(3,2) = w1(3,2) - learningRage \* x3(i) \* gradOutput22 = 0.4999

w1(3,3) = w1(3,3) - learningRage \* x3(i) \* gradOutput23 = 0.4999

w2(1,1) = w2(1,1) - learningRage \* y1 \* gradOutput = 0.4988

w2(1,2) = w2(1,2) - learningRage \* y2 \* gradOutput = 0.4988

w2(1,3) = w2(1,3) - learningRage \* y3 \* gradOutput = 0.4988

1. Был реализован алгоритм обучения нейронной сети на базе MATLAB

function target = targetFunction(x1,x2,x3)

target = x1./(x2+x3);

end;

function target = sigmoid(x)

target = 1./(1+exp(-x));

end;

%Задали веса

w1 = [0.5 0.5 0.5; 0.5 0.5 0.5; 0.5 0.5 0.5;];

w2 = [0.5 0.5 0.5;];

%Сделали входные данные

n= 100;

X1Inputs = linspace(0,2,n);

X2Inputs = linspace(2,4,n);

X3Inputs = linspace(4,6,n);

[x1,x2,x3] = meshgrid(X1Inputs,X2Inputs,X3Inputs);

%Задаем параметры обучения

numberOfEpochs = 1000;

learningRage = 0.01;

learningRageDecay = 0.95;

minErrorValue = 0.0001;

previousError = 0;

for j=1:numberOfEpochs

totalError = 0;

for i=1:numel(x1)

target = x1(i)/(x2(i)+x3(i));

%Вычисления значения первого слоя

S1 = w1(1,1) \* x1(i) + w1(1,2) \* x2(i) + w1(1,3) \* x3(i);

y1 = sigmoid(S1);

S2 = w1(2,1) \* x1(i) + w1(2,2) \* x2(i) + w1(2,3) \* x3(i);

y2 = sigmoid(S2);

S3 = w1(3,1) \* x1(i) + w1(3,2) \* x2(i) + w1(3,3) \* x3(i);

y3 = sigmoid(S3);

%Вычисления выходного слоя

s = w2(1,1) \* y1 + w2(1,2) \* y2 + w2(1,3)\*y3;

y = sigmoid(s);

error = (y-target)^2;

%Значение градиента выходного слоя

gradOutput = (y-target) \* y \* (1-y);

%Значения для предпоследнего

gradOutput21 = y1 \* (1-y1) \* gradOutput \* w2(1,1);

gradOutput22 = y2 \* (1-y2) \* gradOutput \* w2(1,2);

gradOutput23 = y3 \* (1-y3) \* gradOutput \* w2(1,3);

%Коррекция синапсов

w1(1,1) = w1(1,1) - learningRage \* x1(i) \* gradOutput21;

w1(1,2) = w1(1,2) - learningRage \* x1(i) \* gradOutput22;

w1(1,3) = w1(1,3) - learningRage \* x1(i) \* gradOutput23;

w1(2,1) = w1(2,1) - learningRage \* x2(i) \* gradOutput21;

w1(2,2) = w1(2,2) - learningRage \* x2(i) \* gradOutput22;

w1(2,3) = w1(2,3) - learningRage \* x2(i) \* gradOutput23;

w1(3,1) = w1(3,1) - learningRage \* x3(i) \* gradOutput21;

w1(3,2) = w1(3,2) - learningRage \* x3(i) \* gradOutput22;

w1(3,3) = w1(3,3) - learningRage \* x3(i) \* gradOutput23;

w2(1,1) = w2(1,1) - learningRage \* y1 \* gradOutput;

w2(1,2) = w2(1,2) - learningRage \* y2 \* gradOutput;

w2(1,3) = w2(1,3) - learningRage \* y3 \* gradOutput;

totalError = totalError + error;

end;

totalError = totalError/numel(x1);

if(not (totalError - previousError)<0)

learningRage = learningRage \* learningRageDecay;

end;

previousError = totalError;

if(totalError<minErrorValue)

disp(['Эпоха ' num2str(j) ' значение ошибки ' num2str(totalError) ' обучение завершено'])

break;

else

disp(['Эпоха ' num2str(j) ' значение ошибки ' num2str(totalError)])

end;

end;

disp(w1)

disp(w2)

x1test = linspace(0,2,10);

x2test = linspace(2,4,10);

x3test = linspace(4,6,10);

[x1,x2,x3] = meshgrid(x1test,x2test,x3test);

y1 = sigmoid(x1\*w1(1,1)+x2\*w1(1,2)+x3\*w1(1,3));

y2 = sigmoid(x1\*w1(2,1)+x2\*w1(2,2)+x3\*w1(2,3));

y3 = sigmoid(x1\*w1(3,1)+x2\*w1(3,2)+x3\*w1(3,3));

y = sigmoid(w2(1,1) \* y1 + w2(1,2) \* y2 + w2(1,3) \* y3);

figure;

target = targetFunction(x1,x2,x3);

plot(1:10^3,y(:)-target(:));

hold off;

3. Было проведено тестирование нейронной сети на входных векторах. Нейронная сеть полностю аппроксимировала функцию заданную входными обучающими значениями.

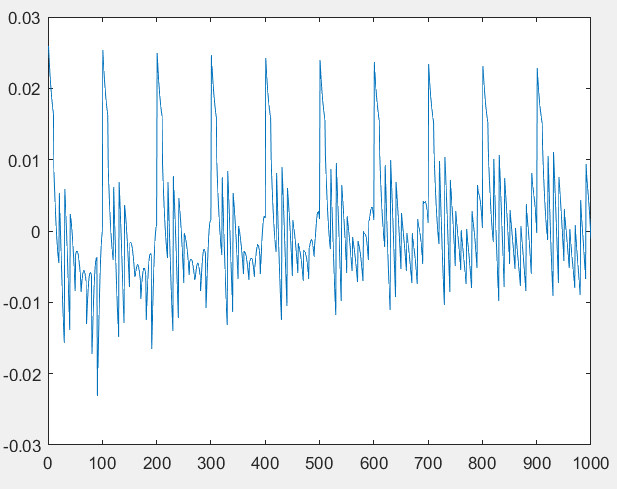


Рисунок 1 – График разности между верным значением функции и выходом нейронной сети для каждого примера.

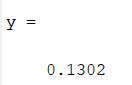


Рисунок 2 – Пример выходного значения для вектора [1,2,6]

**Вывод:** Был изучен и применен для обучения нейронной сети алгоритм обратного распространения ошибки.