Описание разработанной нейронной сети.

Для диагностирования работоспособности винтовых насосов была написана нейронная сеть с обратным распространением ошибки, обучение которой проходило с учителем.

Целью обучения нейронной сети было обнаружение аномального отклонения от нормы, данных, считанных с датчика вибрации.

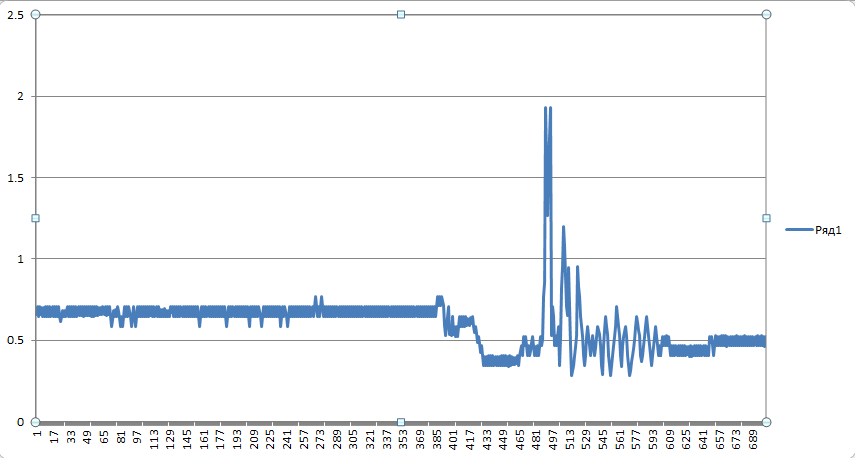


Рис.1. Аномальное отклонение от нормы.

Сеть была обучена для нахождения отклонений от нормального состояния работы ВН (см. Рис.1).

Главная цель написания нейронной сети - заблаговременное обнаружение кавитационных режимов.

Для обучения нейронной сети на вход по отдельности подавались значения, считанные с датчика вибрации ВН. Целевой вектор для обучающей выборки был рассчитан самостоятельно с помощью правила нахождения статистических выбросов, для каждого входного значения был рассчитан выходной, равный либо 1 (соответственно выброс), либо 0 ( нормальное значение вибрации).

Сеть была реализована на языки С++ и содержит 3 нейрона в скрытом слое и 1 выходной нейрон.

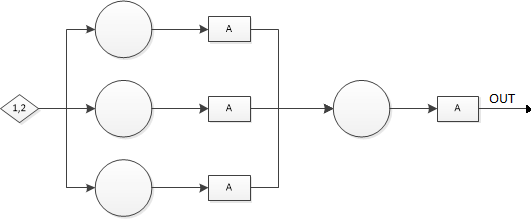


Рис.1. Реализованная нейронная сеть.

Для решения данной проблемы в качестве активационной функции рациональнее было взять ступенчатую, однако в дальнейшем данная нейронная сеть будет использовать данные не только с датчика вибраций и выходным результатом будет не 1 или 0. Поэтому было принято решение в качестве активационной функции использовать сигмоиду.

Процесс обучения заключался в поочередном предоставлении нейронной сети выходных векторов, далее выходное значение сравнивалась с целевым. И на основании данной ошибки происходило корректирование весов. В начале обучения веса вводились вручную.

Данная нейронная сеть была обучена, её квадратичная ошибка не превышает 0.002. Данное условие квадратичной ошибки было выбрано самостоятельно на основе уже реализованных нейронных сетей.

Для обучения использовалась выборка из 702 элементов, данная выборка подавалась на вход несколько раз. Количество эпох было ровно столько, чтобы добиться прописанной квадратичной ошибки.

В качестве тестирования на вход подавалось 230 элементов. На этапе тестирования не было выявлено ошибок, что свидетельствует о том, что сеть была обучена правильно.

С увеличением сложности решаемой задачи количество нейронов в скрытом слое будет увеличиваться.

Код обучения нейронной сети и ее тестирование:

#include "stdafx.h"

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <ctype.h>

#include <iostream>

#include <cmath>

#include <ctime>

#include "StdAfx.h"

#include <fstream>

#include <iostream>

using namespace std;

float activ\_fun(float NET)

{

return 1 / (1 + exp(-NET));

}

float proizvod\_activ\_fun(float OUT)

{

return OUT\*(1 - OUT);

}

int main()

{

float w1\_1 = 0.3273, w1\_2 = 0.2834, w1\_3 = 0.5329;

float w2\_1 = 0.3542, w2\_2 = 0.4374, w2\_3 = 0.3202;

float h = 0.1, cel = 0;

float correctirovka[6];

float delta[4];

float b1\_1 = 0, b1\_2 = 0, b1\_3 = 0, b2\_1 = 0;

ifstream inp("input.txt");

ifstream output("output.txt");

float vxod[702], vyxod[702];

for (int i = 0; i < 702; i++)

{

inp >> vxod[i];

output >> vyxod[i];

//cout << in[i] << " " << out[i] << endl;

}

for (int j = 0; j < 1000000; ++j)

{

cel = 0;

for (int i = 0; i < 702; i++)

{

b1\_1 = vxod[i] \* w1\_1;

b1\_2 = vxod[i] \* w1\_2;

b1\_3 = vxod[i] \* w1\_3;

b1\_1 = activ\_fun(b1\_1);

b1\_2 = activ\_fun(b1\_2);

b1\_3 = activ\_fun(b1\_3);

b2\_1 = b1\_1\*w2\_1 + b1\_2\*w2\_2 + b1\_3\*w2\_3;

b2\_1 = activ\_fun(b2\_1);

cel += (vyxod[i] - b2\_1) \* (vyxod[i] - b2\_1);

//Выходной слой дельта

delta[0] = -2 \* (vyxod[i] - b2\_1) \* proizvod\_activ\_fun(b2\_1);

//w2\_3 = w2\_3 + correctirovka[1];

// Скрытый слой дельта

delta[1] = 2 \* proizvod\_activ\_fun(b1\_1)\*delta[0] \* w2\_1;

delta[3] = 2 \* proizvod\_activ\_fun(b1\_3)\*delta[0] \* w2\_3;

delta[2] = 2 \* proizvod\_activ\_fun(b1\_2)\*delta[0] \* w2\_2;

// рассчитываем на сколько корректируем веса скрытого слоя

correctirovka[3] = -delta[1] \* vxod[i] \* h;

correctirovka[4] = -delta[2] \* vxod[i] \* h;

correctirovka[5] = -delta[3] \* vxod[i] \* h;

// корректировка весов скрытого слоя

w1\_1 = w1\_1 + correctirovka[3];

w1\_2 = w1\_2 + correctirovka[4];

w1\_3 = w1\_3 + correctirovka[5];

// на сколько корректируем веса выходного слоя

correctirovka[0] = -delta[0] \* h\*b1\_1;

correctirovka[1] = -delta[0] \* h\*b1\_2;

correctirovka[2] = -delta[0] \* h\*b1\_3;

//корректируем веса выходного слоя.

w2\_1 = w2\_1 + correctirovka[0];

w2\_2 = w2\_2 + correctirovka[1];

w2\_3 = w2\_3 + correctirovka[2];

}

cout << cel / 702 << endl;

if ((cel / 702) < 0.0002)

break;

}

// тестируем нейросеть

float test[230];

ifstream testirovanie("test.txt");

for (int i = 0; i < 230; i++)

{

testirovanie >> test[i];

cout << test[i] << endl;

}

for (int i = 0; i < 230; i++)

{

b1\_1 = test[i] \* w1\_1;

b1\_2 = test[i] \* w1\_2;

b1\_3 = test[i] \* w1\_3;

b1\_1 = activ\_fun(b1\_1);

b1\_2 = activ\_fun(b1\_2);

b1\_3 = activ\_fun(b1\_3);

b2\_1 = b1\_1\*w2\_1 + b1\_2\*w2\_2 + b1\_3\*w2\_3;

b2\_1 = activ\_fun(b2\_1);

cout << b2\_1 << endl;

}

return 0;

}