Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

*Кафедра програмного забезпечення автоматизованих систем*

**Лабораторна робота №2**

Тема:

**«Нейронна мережа backpropogation»**

***Виконав*:**

ст. групи ПІт-15-3

Свирид О. Б.

***Перевірив:***

Дитко Т.В.

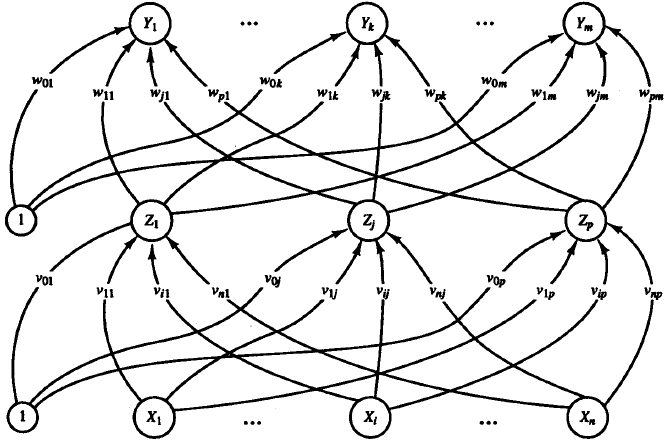
Івано-Франківськ

2015

**Мета:** вивчити один із методів функціонування нейронних мереж і принцип його функціонування.

**Короткі теоретичні відомості**

**Структура нейронної мережі backpropagation з одним прихованим шаром:**



**Алгоритм функціонування мережі:**

**Крок 0:** Ініціалізація ваг ( випадкові значення від -0.5 до 0.5 )

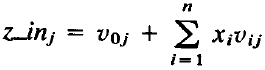
**Крок 1:** Поки умова зупинки = false виконуємо кроки 2-9

**Крок 2:** Для кожного вхідного паттерну виконуємо кроки 3-8

**Крок 3:** Кожен вхідний нейрон Xi отримує вхідні значення і передає їх в наступний шар.

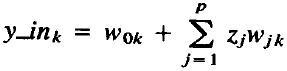
**Крок 4:** Кожен нейрон прихованого шару Zj сумує зважений вхідний сигнал

і застосовує до нього активаційну функцію та відправляє свій сигнал наступному шару (в даному випадку вихідному).



**Крок 5:** Кожен вихідний нейрон Yk сумує зважений вхідний до нього сигнал

і застосовує до суми активаційну функцію для вирахування вихідного сигналу .



**Крок 6:** Кожен вихідний нейрон Yk отримує правильне запропоноване «вчителем» значення для данного вхідного паттерну і вираховує свою помилку .



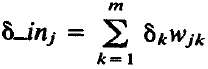
Далі вираховує зміну своїх ваг і зміщення і надсилає попередньому шару.



**Крок 7:** Кожен нейрон прихованого шару Zj сумує від попереднього шару вище



і множить на похідну активаційної функції для визначення своєї помилки. Далі вираховує зміну своїх ваг і зміщення



.



**Крок 8:** Кожен вихідний нейрон оновлює свої ваги і зміщення



Кожен прихований нейрон оновлює свої ваги і зміщення



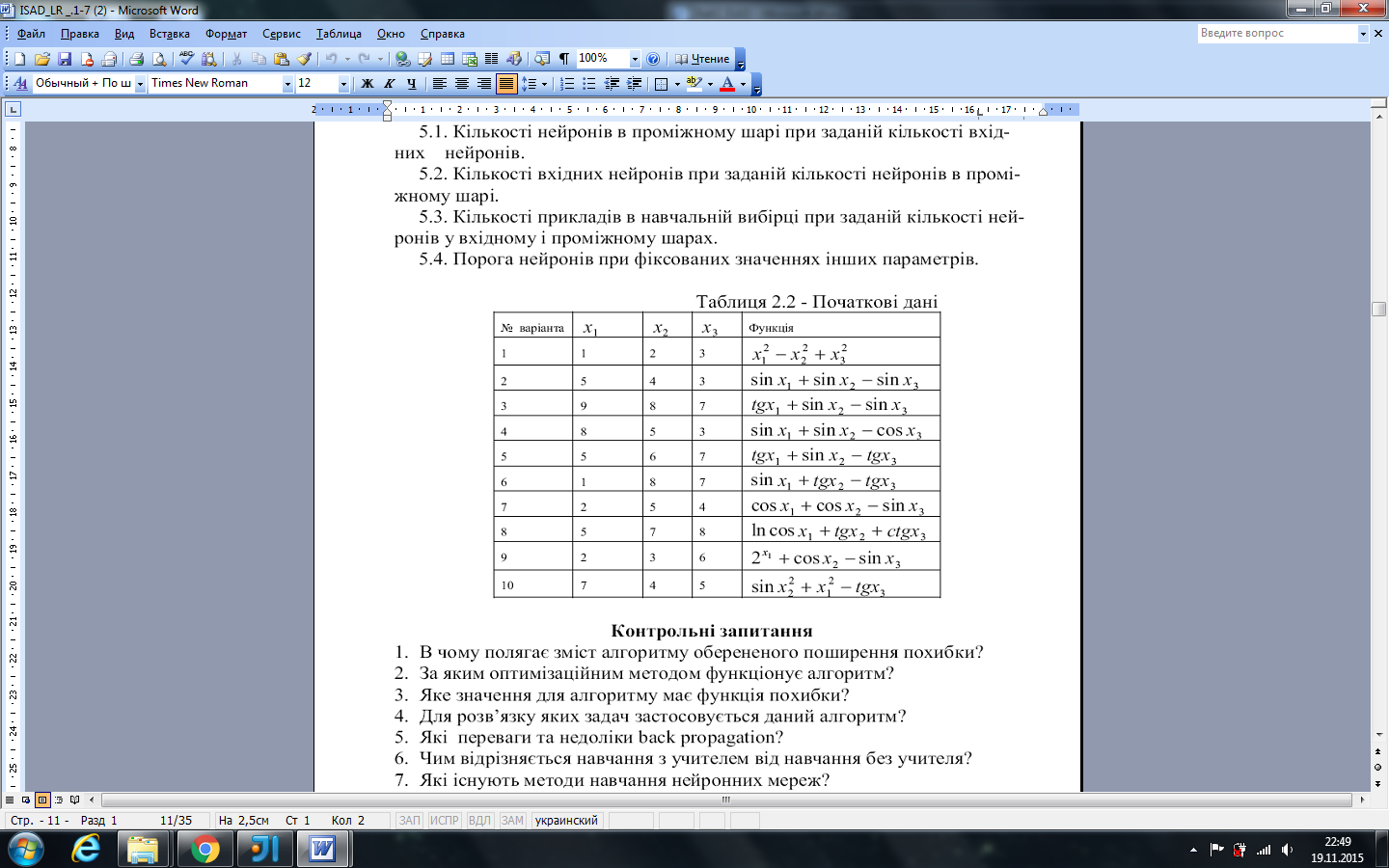
**Крок 9:** Перевірка умови припинення навчання.

В своїй мережі в якості активаційної функції використовував бінарний сигмоїд

з похідною .



**Завдання:** подати на навчання вхідний образ заданий варіантом та декілька інших, значення аргументів яких відрізняються на ±1. В якості виходу (значення запропоноване вчителем) взяти 0 якщо значення функції < 0.5 і 1 якщо > 0.5.



**Хід роботи**

В мому випадку мережа backpropagation на вхід і вихід приймає тільки 0 і 1, тож перетворимо значення аргументів типу INT (будь-які) на масив елементів типу INT який складається з 0 і 1, а значення вихідної функції нормалізуємо

(від 0 до 1) і якщо це значення > 0,5 то вихід = 1, якщо < 0,5 то = 0.

В своїй мережі я задав 10 000 епох, хоча для навчання вистачає менше.

**Код програми:**

**Main:**

package com.Lab2ISAD;

public class MainClass {

public static void main(String[] args)

{

NeuralNetwork nn = new NeuralNetwork();

nn.go();

}

}

**Клас Neural Network:**

package com.Lab2ISAD;

import java.util.Arrays;

import java.util.Random;

import static java.lang.Math.exp;

public class NeuralNetwork {

private final int INP = 6;

private final int HIDE = 6;

private final int OUT = 1;

private final double ALFA = 0.1; // Швидкість навчання

private final int EPOCH = 10000; // Кількість ітерацій навчання

private final double e = 0.000001; // Точність навчання

private double wih[][] = new double[INP+1][HIDE]; // Ваги

private double who[][] = new double[HIDE+1][OUT]; // Ваги

private double[] x = new double[INP]; // Вхід

private double[] z = new double[HIDE]; // Прихований шар

private double[] y = new double[OUT]; // Вихід

private double sumZ; // Сумує сигнал на прихованому шарі

private double sumY; // Сумує сигнал на вихідному шарі

private double dk[] = new double[OUT];

private double dj[] = new double[HIDE];

private double djSum;

private int[][] Xint = { // Для заповнення масиву відповідей T

{1,2,3},

{0,2,3},

{3,2,3},

{1,1,3},

{1,3,3},

{1,2,2},

};

private double[][] X = { // Вхідні дані

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

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

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

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

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

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

};

private double[] T;

public void go()

{

NeuralWork();

}

private void NeuralWork()

{

tInit(); // Ініціалізація відповідей

initWeights(); // Ініціалізація ваг

for (int epoch = 0; epoch < EPOCH; epoch++)

{

for (int pattern = 0; pattern < X.length; pattern++)

{

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

x[i] = X[pattern][i];

feedForward();

backpropagation(pattern);

}

}

System.out.println("\nStudy COMPLETE\n");

System.out.println("Перевіримо роботу мережі після навчання.\nПодамо наступні образи:");

for (int ind = 0; ind < X.length; ind++)

feedForward2(ind);

}

private void feedForward()

{

for (int hid = 0; hid < HIDE; hid++)

{

sumZ = 0;

for (int inp = 0; inp < INP; inp++)

sumZ += x[inp] \* wih[inp][hid];

sumZ += wih[INP][hid];

z[hid] = binarySigmoid(sumZ);

}

for (int out = 0; out < OUT; out++)

{

sumY = 0;

for (int hid = 0; hid < HIDE; hid++)

sumY += z[hid] \* who[hid][out];

sumY += who[HIDE][out];

y[out] = binarySigmoid(sumY);

}

}

private void backpropagation(int pattern)

{

for (int out = 0; out < OUT; out++)

{

dk[out] = (T[pattern] - y[out]) \* binarySigmoidDerivate(y[out]);

}

for (int hid = 0; hid < HIDE; hid++)

{

djSum = 0;

for (int out = 0; out < OUT; out++)

djSum += dk[out] \* who[hid][out];

dj[hid] = djSum \* binarySigmoidDerivate(z[hid]);

}

for (int out = 0; out < OUT; out++)

{

for (int hid = 0; hid < HIDE; hid++)

who[hid][out] += (ALFA \* dk[out] \* z[hid]);

who[HIDE][out] += (ALFA \* dk[out]);

}

for (int hid = 0; hid < HIDE; hid++)

{

for (int inp = 0; inp < INP; inp++)

wih[inp][hid] += (ALFA \* dj[hid] \* x[inp]);

wih[INP][hid] += (ALFA \* dj[hid]);

}

}

private void feedForward2(int index) // Для перевірки навченої мережі

{

for (int col = 0; col < X[index].length; col++)

x[col] = X[index][col];

for (int hid = 0; hid < HIDE; hid++) {

sumZ = 0;

for (int inp = 0; inp < INP; inp++)

sumZ += x[inp] \* wih[inp][hid];

sumZ += wih[INP][hid];

z[hid] = binarySigmoid(sumZ);

}

for (int out = 0; out < OUT; out++) {

sumY = 0;

for (int hid = 0; hid < HIDE; hid++)

sumY += z[hid] \* who[hid][out];

sumY += who[HIDE][out];

y[out] = binarySigmoid(sumY);

feedForwardText(index, out);

}

}

//Виводить результати при перевірці навченої мережі

private void feedForwardText(int index, int out) {

System.out.print("\nВхідний образ:" + Arrays.toString(Xint[index]) + " Вихід: " + Math.round(y[out]));

if (Math.round(y[out]) == T[index])

System.out.print(" ПРАВИЛЬНО");

else System.out.print(" НЕ ПРАВИЛЬНО");

}

private double binarySigmoid(double sum)

{

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

}

private double binarySigmoidDerivate(double sum)

{

return sum \* (1 - sum);

}

private void initWeights()

{

Random rnd = new Random();

for (int input = 0; input <= INP; input++)

for (int hidden = 0; hidden < HIDE; hidden++)

wih[input][hidden] = rnd.nextDouble() - 0.5;

for (int hidden = 0; hidden <= HIDE; hidden++)

for (int output = 0; output < OUT; output++)

who[hidden][output] = rnd.nextDouble() - 0.5;

}

private void tInit() {

T = new double[X.length];

for (int row = 0; row < Xint.length; row++)

if ((f(Xint[row][0],Xint[row][1],Xint[row][2])/14) > 0.5)

T[row] = 1;

else T[row] = 0;

}

private double f(double x1, double x2, double x3)

{

return Math.log10(Math.cos(x1)) + Math.tan(x2) + (1.0 / Math.tan(x3));

}

}

**Результати роботи**

Study COMPLETE

Перевіримо роботу мережі після навчання.

Подамо наступні образи:

Вхідний образ:[1, 2, 3] Вихід: 0 ПРАВИЛЬНО

Вхідний образ:[0, 2, 3] Вихід: 0 ПРАВИЛЬНО

Вхідний образ:[3, 2, 3] Вихід: 1 ПРАВИЛЬНО

Вхідний образ:[1, 1, 3] Вихід: 1 ПРАВИЛЬНО

Вхідний образ:[1, 3, 3] Вихід: 0 ПРАВИЛЬНО

Вхідний образ:[1, 2, 2] Вихід: 0 ПРАВИЛЬНО

***Висновок:***  в ході лабораторної роботи я ознайомився з принципом роботи мережі типу backpropagation і програмно реалізував алгоритм її навчання.