**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине: «Введение в нейронные сети»

на тему: «Простейшие нейронные сети»

Выполнил: студент гр. ИТП-31

Курильчик А.П.

Принял: преподаватель

Гуменников Е. Д.

Гомель 2022

**Цель работы:** спроектировать и обучить нейронную сеть по заданному правилу.

Задание для варианта 12:

Функция: X or Y and not Y

Правило обучения: правило Хэбба.

Листинг программы показан в приложении А.

На рисунке 1 показана структура нейронной сети.

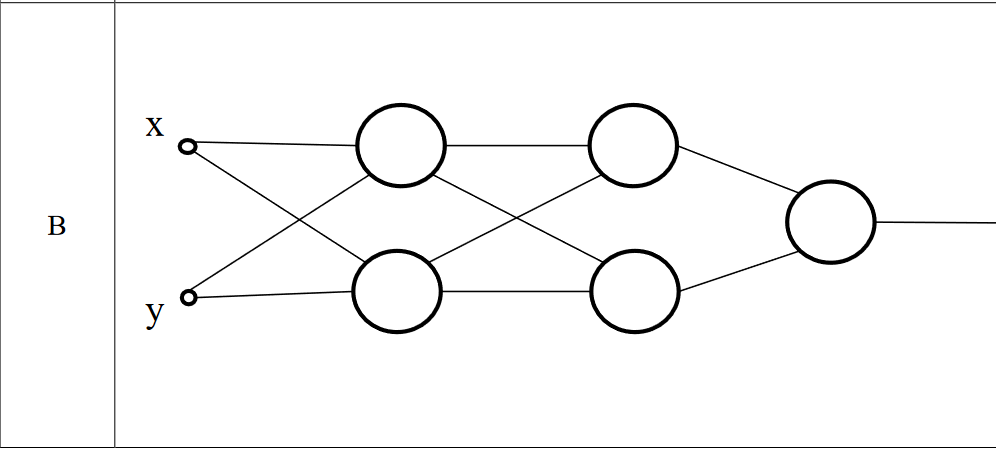


Рисунок 1 – структура нейронной сети

**Практическая часть**

На рисунке 2 показан результат обучения заданной нейронной сети.

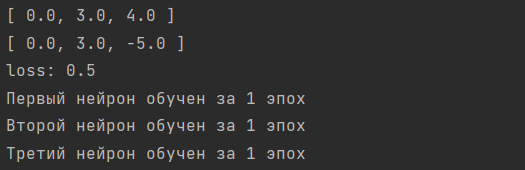


Рисунок 2 – результат обучения нейронной сети

**Вывод:** правило Хэбба не подходит для непосредственного обучения многослойной нейронной сети.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг файла Main.java

import java.util.Arrays;

import static java.lang.Math.\*;

import static java.lang.System.\*;

public class Main {

static double[] startWeights = { 1, 3 };

static double[] biasWeights = {2, 4, -5};

static double[] firstWeights = startWeights.clone();

static double[] secondWeights = startWeights.clone();

static double[] finalWeights = startWeights.clone();

static double[][] xTrain = {

{0, 0},

{1, 0},

{0, 1},

{1, 1}

};

static double[] yTrainFirst = {0, 1, 1, 1};

static double[] yTrainSecond = {0, 0, 1, 1};

static double[] yTrainThird = {0, 0, 0, 1};

public static void main(String[] args) {

fitNetwork();

}

private static double output(double[] x, double[] weights, int neuronId) {

double result = 1;

for (int i = 0; i < x.length; i++) {

result += x[i] \* weights[i];

}

result += biasWeights[neuronId];

return round(1 / (1 + exp(-result)));

}

private static void fitNetwork() {

int age = 1;

boolean firstFitted = false;

boolean secondFitted = false;

boolean finalFitted = false;

double[] epochPredicts = new double[4];

while (!(firstFitted && secondFitted && finalFitted)) {

out.println("age: " + age);

double[] fPr = firstWeights;

double[] sPr = secondWeights;

double[] fnlPr = finalWeights;

for (int i = 0; i < yTrainFirst.length; i++) {

double[] xCurr = xTrain[i];

double[] xFinal = new double[] {

output(xCurr, Main.firstWeights, 0),

output(xCurr, Main.secondWeights, 1)

};

epochPredicts[i] = output(xFinal, finalWeights, 2);

fitNeuron(xCurr, firstWeights, yTrainFirst[i], 0);

fitNeuron(xCurr, secondWeights, yTrainSecond[i], 1);

fitNeuron(xFinal, Main.finalWeights, yTrainThird[i], 2);

}

double lossSum = 0;

for (int j = 0; j < epochPredicts.length; j++) {

lossSum += pow(yTrainThird[j] - epochPredicts[j], 2);

}

out.println("loss: " + lossSum / yTrainThird.length);

firstFitted = fPr == firstWeights;

if (firstFitted) { out.println("РџРµСЂРІС‹Р№ РЅРµР№СЂРѕРЅ РѕР±СѓС‡РµРЅ Р·Р° " + age + " СЌРїРѕС…"); }

secondFitted = sPr == secondWeights;

if (secondFitted) { out.println("Р’С‚РѕСЂРѕР№ РЅРµР№СЂРѕРЅ РѕР±СѓС‡РµРЅ Р·Р° " + age + " СЌРїРѕС…"); }

finalFitted = fnlPr == finalWeights;

if (finalFitted) { out.println("РўСЂРµС‚РёР№ РЅРµР№СЂРѕРЅ РѕР±СѓС‡РµРЅ Р·Р° " + age + " СЌРїРѕС…"); }

age ++;

}

}

private static void fitNeuron(double[] x, double[] weights, double y, int neuronId) {

double yPredicted = output(x, weights, neuronId);

if (yPredicted != y) {

if (yPredicted == 0) {

for (int i = 0; i < weights.length; i++) {

if (weights[i] == 1) {

weights[i] += 1;

}

}

biasWeights[neuronId] = biasWeights[neuronId] == 1

? biasWeights[neuronId] + 1

: biasWeights[neuronId];

}

else {

for (int i = 0; i < weights.length; i++) {

if (weights[i] == 1) {

weights[i] -= 1;

}

}

biasWeights[neuronId] = biasWeights[neuronId] == 1

? biasWeights[neuronId] - 1

: biasWeights[neuronId];

}

}

out.println("[ " + weights[0] + ", " + weights[1] + ", " + Main.biasWeights[neuronId] + " ]");

}

}