Міністерство освіти і науки України Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

# Лабораторна робота №3-2

З дисципліни «Методи оптимізації та планування»

# ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ. МОДЕЛЬ PERCEPTRON

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи IO-93 Ященко Є.В. - 9331

ПЕРЕВІРИВ:

Регіда П.Г.

Київ 2021 р.

**Мета роботи -** ознайомлення з принципами машинного навчання за допомогою

математичної моделі сприйняття інформації Перцептрон(Perceptron).

Змоделювати

роботу нейронної мережі та дослідити вплив параметрів на час виконання та точність

результату

# Завдання на лабораторну роботу

## Поріг спрацювання: P = 4

Дано точки: А(0,6), В(1,5), С(3,3), D(2,4).

## Швидкості навчання: δ = {0,001; 0,01; 0,05; 0.1; 0.2; 0,3}

Дедлайн: часовий = {0.5с; 1с; 2с; 5с}, кількість ітерацій = {100;200;500;1000} Обрати швидкість навчання та дедлайн. Налаштувати Перцептрон для даних точок. Розробити відповідний мобільний додаток і вивести отримані значення.

## Провести аналіз витрати часу та точності результату за різних параметрах навчання.

**Код програми:**

package com.lab2a.execution;

import com.lab2a.utils.exception.ItersExceededException; import com.lab2a.utils.exception.LabException;

import com.lab2a.utils.exception.TimeExceededException;

import java.util.ArrayList; import java.util.List;

public class Perceptron {

private final boolean timeLimited, itersLimited; double deadline;

int max\_iters;

private double w1; private double w2;

private final double sigma, P;

private final List<double[]> points = new ArrayList<>(); private final List<Boolean> pointIsMoreP = new ArrayList<>(); private boolean trained = false;

private double time; private int iters;

public Perceptron(double w1, double w2, double sigma, double P) {

this.timeLimited = false; this.itersLimited = false;

this.w1 = w1; this.w2 = w2; this.sigma = sigma; this.P = P;

}

public Perceptron(double w1, double w2, double sigma, double P, int max\_iters) {

this.timeLimited = false; this.itersLimited = true;

this.max\_iters = max\_iters;

this.w1 = w1; this.w2 = w2; this.sigma = sigma; this.P = P;

}

public Perceptron(double w1, double w2, double sigma, double P, double deadline)

{

this.timeLimited = true; this.itersLimited = false;

this.deadline = deadline;

this.w1 = w1; this.w2 = w2; this.sigma = sigma; this.P = P;

}

public Perceptron(double w1, double w2, double sigma, double P, double deadline, int max\_iters) {

this.timeLimited = true; this.itersLimited = true;

this.deadline = deadline; this.max\_iters = max\_iters;

this.w1 = w1; this.w2 = w2; this.sigma = sigma; this.P = P;

}

private void correctWeights(double delta, double x1, double x2) {

this.w1 = this.w1 + delta \* x1 \* this.sigma; this.w2 = this.w2 + delta \* x2 \* this.sigma;

}

public void addPoint(double x1, double x2, boolean isMoreP) {

this.points.add(new double[]{x1, x2}); this.pointIsMoreP.add(isMoreP);

}

public void train() throws LabException {

double time0 = System.*nanoTime*();

this.iters = 0;

while (!this.trained) {

boolean noMistakes = true;

for (int i = 0; i < this.points.size(); i++) {

this.w2;

double y = this.points.get(i)[0] \* this.w1 + this.points.get(i)[1] \*

if (y > this.P != this.pointIsMoreP.get(i)) { double delta = this.P - y;

this.correctWeights(delta, this.points.get(i)[0],

this.points.get(i)[1]);

noMistakes = false;

}

}

this.time = System.*nanoTime*() - time0;

this.iters++;

if (this.timeLimited && this.time >= this.deadline) throw new TimeExceededException();

if (this.itersLimited && this.iters >= this.max\_iters) throw new ItersExceededException();

if (noMistakes) { this.trained = true;

}

}

}

public boolean isPointMoreThanP(double x1, double x2) {

return (this.w1 \* x1 + this.w2 \* x2 > this.P);

}

public boolean isTrained() {

return this.trained;

}

public double getSigma() {

return this.sigma;

}

public double getTime() {

return this.time;

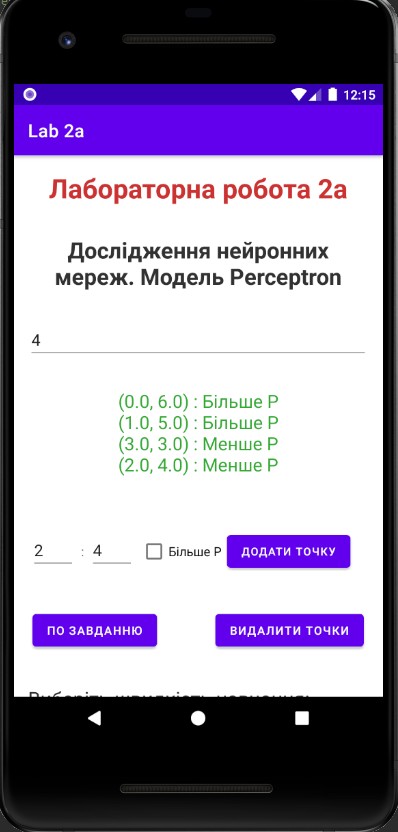
}

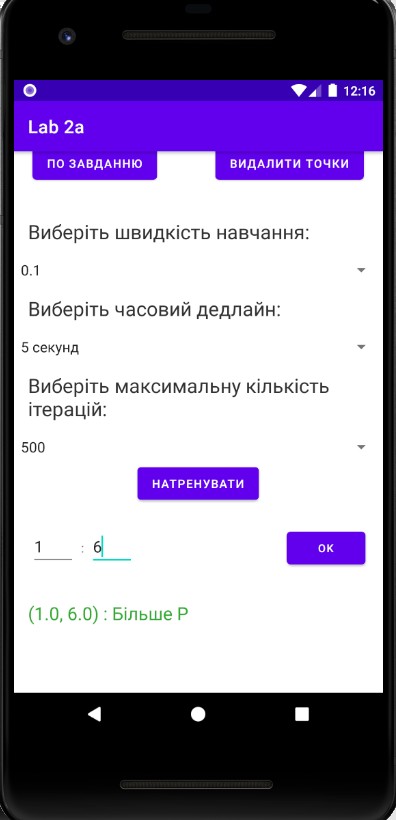
public int getIters() { return this.iters;

}

}

# Результат роботи програми:





**Висновок:**

При виконанні даної лабораторної роботи було вивчено основні принципи розкладання числа на прості множники з використанням різних алгоритмів факторизації. У ході роботи було розроблено програму для факторизації заданого числа методом Ферма.