**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №3.2**

з дисципліни  
«Інтелектуальні вбудовані системи»

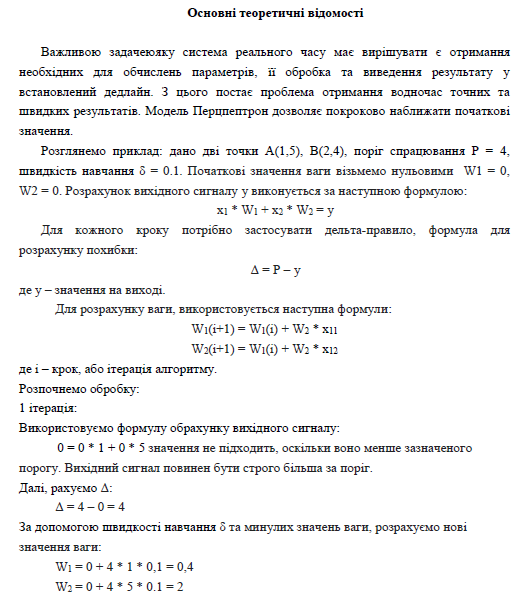
на тему  
«ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ. МОДЕЛЬ PERCEPTRON»

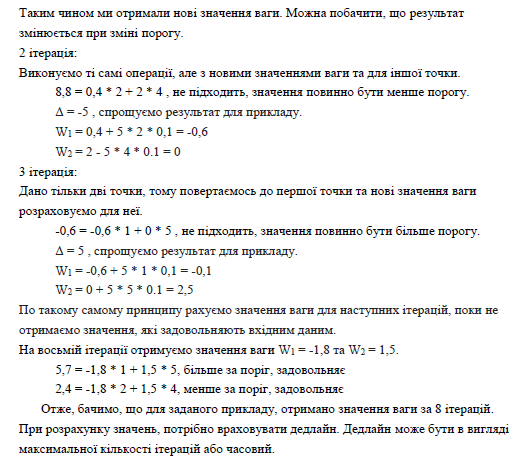
Виконав: Перевірив:

студент групи ІП-84 викладач  
Кучін Владислав Дмитрович Регіда Павло Геннадійович  
номер залікової книжки: 8415

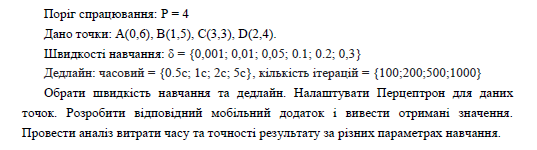
Київ 2021

**Основні теоретичні відомості**



****

**Завдання на лабораторну роботу**

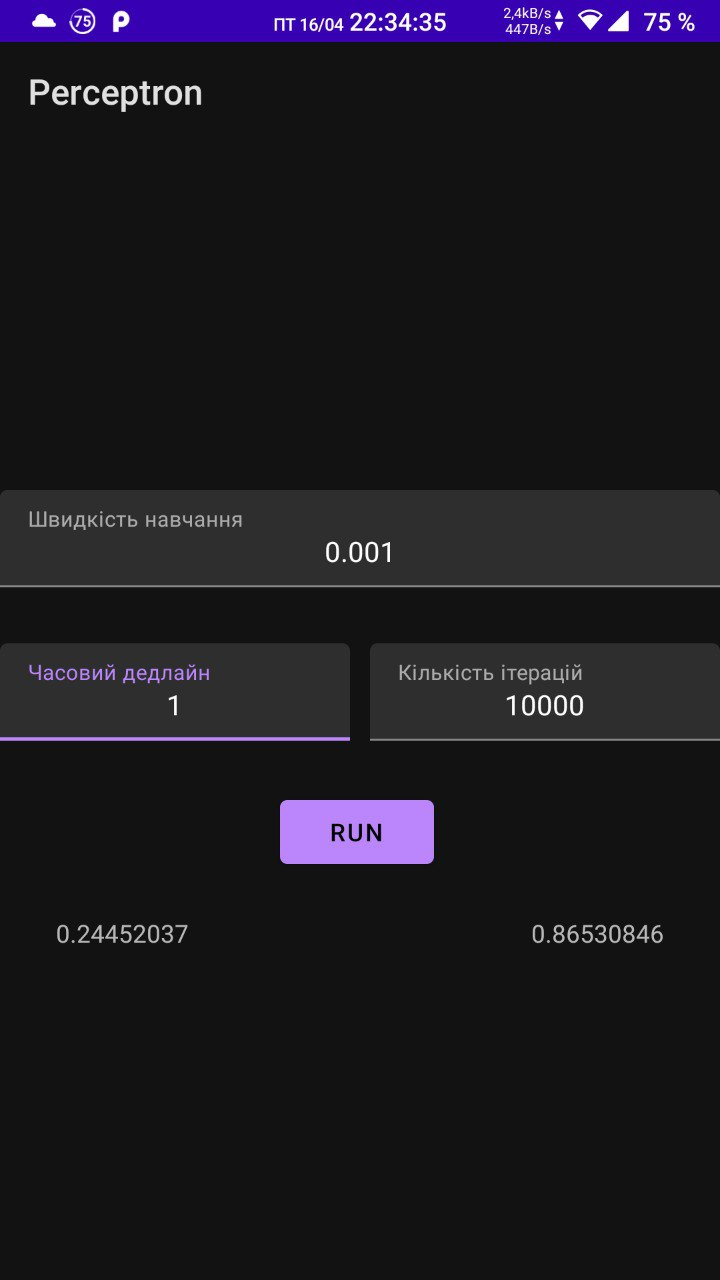


**Варіант-15**

**Лістинг програми**

package ua.kpi.comsys.perceptron  
  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.TextView  
import kotlin.collections.ArrayList  
import kotlin.math.abs  
  
class MainActivity : AppCompatActivity() {  
 var learningSpeed = 0f  
 var timeDeadline = 0f  
 var maxIterations = 0  
 var threshold = 4f  
 var points = *arrayListOf*(Pair(0f, 6f), Pair(1f, 5f), Pair(3f, 3f), Pair(2f, 4f))  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 val learningSpeedView = findViewById<EditText>(R.id.*learning\_speed*)  
 val timeDeadlineView = findViewById<EditText>(R.id.*time\_deadline*)  
 val maxIterationsView = findViewById<EditText>(R.id.*max\_iterations*)  
 val weight1View = findViewById<TextView>(R.id.*weight1*)  
 val weight2View = findViewById<TextView>(R.id.*weight2*)  
 findViewById<Button>(R.id.*run\_button*).setOnClickListener **{** learningSpeed = validateInput(learningSpeedView)  
 if (learningSpeed == 0f) return@setOnClickListener  
  
 timeDeadline = validateInput(timeDeadlineView)  
 if (timeDeadline == 0f) return@setOnClickListener  
 timeDeadline \*= 1000  
  
 maxIterations = validateInput(maxIterationsView).toInt()  
 if (maxIterations == 0) return@setOnClickListener  
  
 val weights = learn()  
 weight1View.*text* = weights[0].toString()  
 weight2View.*text* = weights[1].toString()  
 **}** }  
  
 private fun learn(): ArrayList<Float> {  
 val weights = *arrayListOf*(0f, 0f)  
 var correctResultsCounter = 0  
 var iterationsCounter = 0  
 val startTime = System.currentTimeMillis()  
 while (correctResultsCounter < points.size && iterationsCounter < maxIterations ) {  
 val time = System.currentTimeMillis()  
 if (time - startTime > timeDeadline) return weights  
 val point = points[iterationsCounter++ % points.size]  
 val signal = calculateSignal(point, weights)  
 if (point.second > threshold) {  
 if (signal < threshold) {  
 updateWeights(weights, signal, point)  
 correctResultsCounter = 0  
 } else {  
 ++correctResultsCounter  
 }  
 } else if (point.second < threshold) {  
 if (signal > threshold) {  
 updateWeights(weights, signal, point)  
 correctResultsCounter = 0  
 } else {  
 ++correctResultsCounter  
 }  
 } else {  
 if (*abs*(signal - threshold) > 0.05f) {  
 updateWeights(weights, signal, point)  
 correctResultsCounter = 0  
 } else {  
 ++correctResultsCounter  
 }  
 }  
 }  
 return weights  
 }  
  
 fun calculateSignal(point: Pair<Float, Float>, weights: ArrayList<Float>): Float {  
 return point.first \* weights[0] + point.second \* weights[1]  
 }  
  
 fun updateWeights(weights: ArrayList<Float>, signal: Float, point: Pair<Float, Float>) {  
 val delta = threshold - signal  
 weights[0] += delta \* point.first \* learningSpeed  
 weights[1] += delta \* point.second \* learningSpeed  
 }  
  
 fun validateInput(input: EditText): Float {  
 val string = input.*text*.toString()  
 if (string.*isNotEmpty*()) {  
 val number = string.*toFloat*()  
 if (number != 0f) {  
 return number  
 }  
 }  
 input.*error* = "Введіть додатне значення"  
 return 0f  
 }  
}

**Результати роботи програми**



**Висновки**

Під час даної лабораторної роботи ми вивчили, як за допомогою моделі Perceptron можна розбивати простір на частини і визначати, до якої з частин належить задана точка.