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»

Виконала:

студентка групи ІП-84

Шахова Поліна Миколаївна
номер залікової книжки: 8424

Перевірив:

ас. Регіда П. Г.

Київ 2020

Основні теоретичні відомості:

Важливою задачеюяку система реального часу має вирішувати є отримання необхідних для обчислень параметрів, її обробка та виведення результату у встановлений дедлайн. З цього постає проблема отримання водночас точних та швидких результатів. Модель Перцпептрон дозволяє покроково наближати початкові значення.

Розглянемо приклад: дано дві точки A(1,5), B(2,4), поріг спрацювання P = 4, швидкість навчання $\delta = 0.1$. Початкові значення ваги візьмемо нульовими $W1 = 0$, $W2 = 0$. Розрахунок вихідного сигналу у виконується за наступною формулою:

$$x_1 * W_1 + x_2 * W_2 = y$$

Для кожного кроку потрібно застосувати дельта-правило, формула для розрахунку похибки:

$$\Delta = P - y$$

де y – значення на виході.

Для розрахунку ваги, використовується наступна формули:

$$W_1(i+1) = W_1(i) + W_2 * x_{11}$$

$$W_2(i+1) = W_1(i) + W_2 * x_{12}$$

де i – крок, або ітерація алгоритму.

Розпочнемо обробку:

1 ітерація:

Використовуємо формулу обрахунку вихідного сигналу:

$0 = 0 * 1 + 0 * 5$ значення не підходить, оскільки воно менше зазначеного порогу. Вихідний сигнал повинен бути строго більша за поріг.

Далі, рахуємо Δ:

$\Delta = 4 - 0 = 4$

За допомогою швидкості навчання δ та минулих значень ваги, розрахуємо нові значення ваги:

$W_1 = 0 + 4 * 1 * 0,1 = 0,4$

$W_2 = 0 + 4 * 5 * 0.1 = 2$

Таким чином ми отримали нові значення ваги. Можна побачити, що результат змінюється при зміні порогу.

2 ітерація:

Виконуємо ті самі операції, але з новими значеннями ваги та для іншої точки.

$8,8 = 0,4 * 2 + 2 * 4$, не підходить, значення повинно бути менше порогу.

$\Delta = -5$, спрощуємо результат для прикладу.

$W_1 = 0,4 + 5 * 2 * 0,1 = -0,6$

$W_2 = 2 - 5 * 4 * 0.1 = 0$

3 ітерація:

Дано тільки дві точки, тому повертаємось до першої точки та нові значення ваги розраховуємо для неї.

$-0,6 = -0,6 * 1 + 0 * 5$, не підходить, значення повинно бути більше порогу.

$\Delta = 5$, спрощуємо результат для прикладу.

$W_1 = -0,6 + 5 * 1 * 0,1 = -0,1$

$W_2 = 0 + 5 * 5 * 0.1 = 2,5$

По такому самому принципу рахуємо значення ваги для наступних ітерацій, поки не отримаємо значення, які задовольняють вхідним даним.

На восьмій ітерації отримуємо значення ваги $W_1 = -1,8$ та $W_2 = 1,5$.

$5,7 = -1,8 * 1 + 1,5 * 5$, більше за поріг, задовольняє

$2,4 = -1,8 * 2 + 1,5 * 4$, менше за поріг, задовольняє

Отже, бачимо, що для заданого прикладу, отримано значення ваги за 8 ітерацій.

При розрахунку значень, потрібно враховувати дедлайн. Дедлайн може бути в вигляді максимальної кількості ітерацій або часовий.

Завдання за варіантом:

Варіант 24

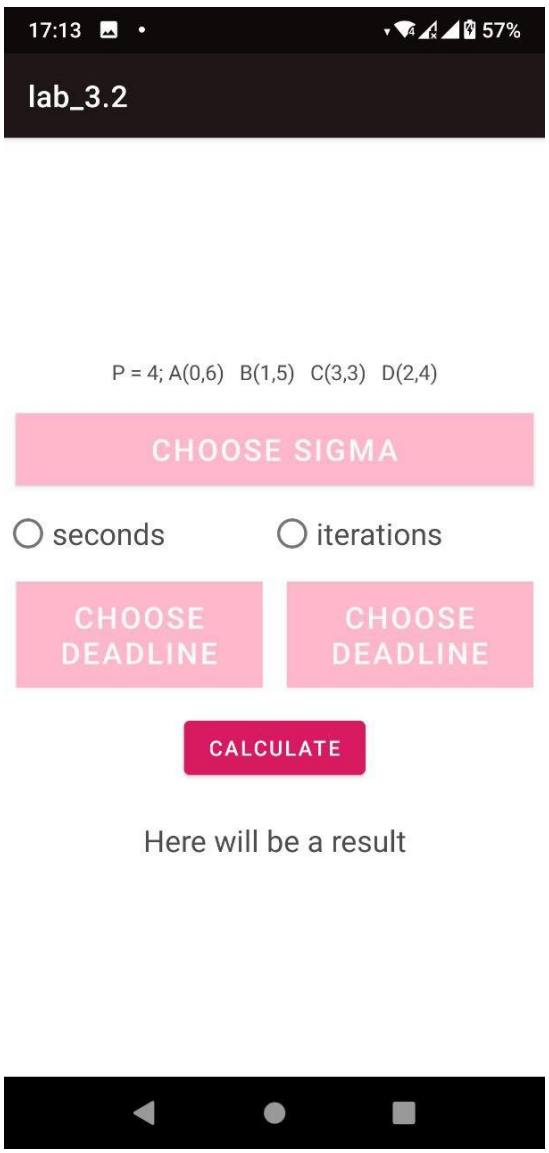
Лістинг програми:

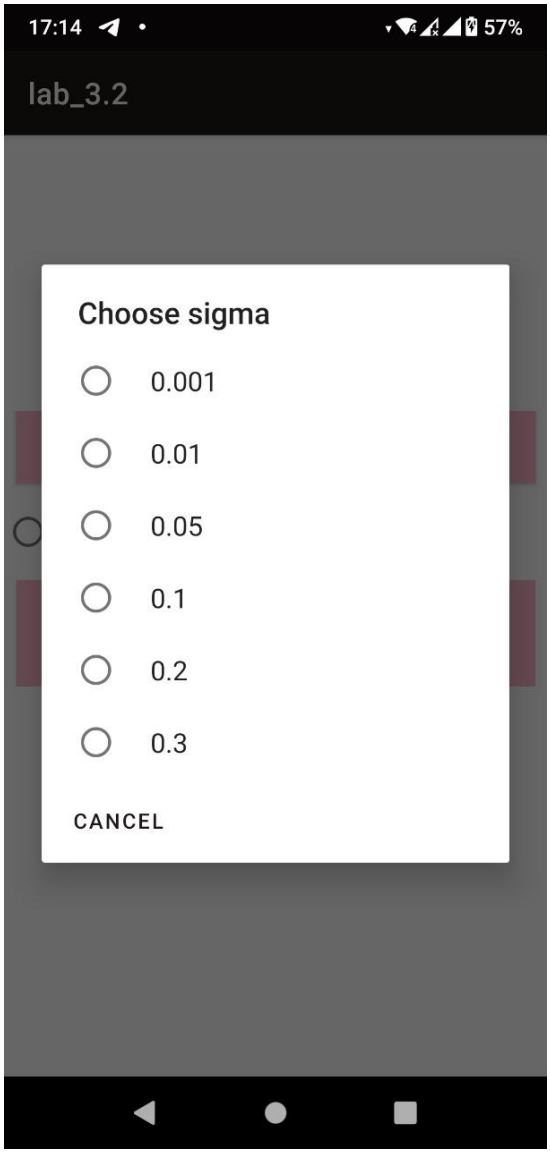
```
package com.example.lab_32

import kotlinx.android.synthetic.main.activity_main.*

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.RadioButton
import androidx.appcompat.app.AlertDialog

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        var sigma: Double = 0.0
        var deadline_seconds: Double = 0.0
        var deadline_iters = 0

        choose_sigma.setOnClickListener{
            val list_of_sigmas = arrayOf("0.001", "0.01", "0.05", "0.1",
"0.2", "0.3")
            val mBuilder = AlertDialog.Builder(this@MainActivity)
            mBuilder.setTitle("Choose sigma")
            mBuilder.setSingleChoiceItems(list_of_sigmas, -1) {
dialogInterface, i ->
                sigma = list_of_sigmas[i].toDouble()
                dialogInterface.dismiss()
            }

            mBuilder.setNeutralButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }
            val mDialog = mBuilder.create()
            mDialog.show()
        }
```

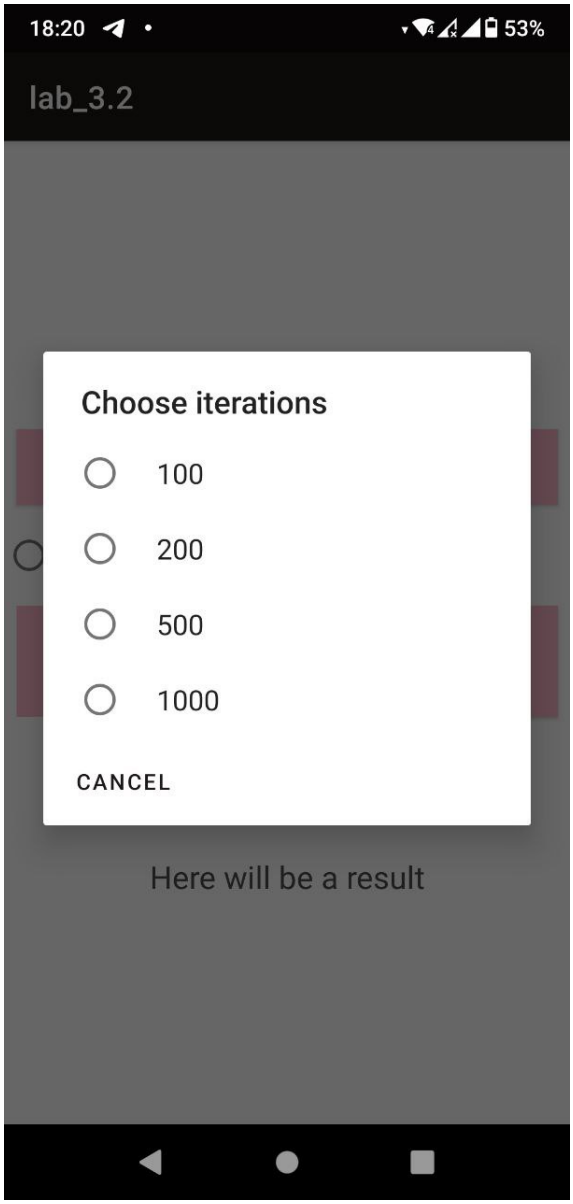
```
        choose_deadline_time.setOnClickListener{
            val list_of_times = arrayOf("0.5", "1", "2", "5")
            val mBuilder = AlertDialog.Builder(this@MainActivity)
            mBuilder.setTitle("Choose seconds")
            mBuilder.setSingleChoiceItems(list_of_times, -1) {
dialogInterface, i ->
                deadline_seconds = list_of_times[i].toDouble()
                dialogInterface.dismiss()
            }

            mBuilder.setNeutralButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }
            val mDialog = mBuilder.create()
            mDialog.show()
        }

        choose_deadline_iters.setOnClickListener{
            val list_of_times = arrayOf("100", "200", "500", "1000")
            val mBuilder = AlertDialog.Builder(this@MainActivity)
            mBuilder.setTitle("Choose iterations")
            mBuilder.setSingleChoiceItems(list_of_times, -1) {
dialogInterface, i ->
                deadline_iters = list_of_times[i].toInt()
                dialogInterface.dismiss()
            }

            mBuilder.setNeutralButton("Cancel") { dialog, _ ->
                dialog.cancel()
            }
            val mDialog = mBuilder.create()
            mDialog.show()
        }

//        radio_group.setOnCheckedChangeListener { group, checkedId ->
//            when(checkedId){
//                R.id.chosen_deadline -> {
//                    choose_deadline_time.isEnabled = true
//                    choose_deadline_iters.isEnabled = false
//                }
//                R.id.chosen_iters -> {
//                    choose_deadline_time.isEnabled = false
//                    choose_deadline_iters.isEnabled = true
//                }
//            }
//        }
        tv_calc.setOnClickListener{Perceptron(sigma,deadline_seconds,
deadline_iters)}
    }


    fun onRadioButtonClicked(view: View) {
        if (view is RadioButton) {
            val checked = view.isChecked

            when (view.getId()) {
                R.id.chosen_deadline ->
                    choose_deadline_time.isEnabled = checked
                R.id.chosen_iters ->
                    choose_deadline_iters.isEnabled = checked
            }
        }
    }

    private fun Perceptron(speed: Double, time: Double, iterations: Int){
        var W1 = 0.00
        var W2 = 0.00
        val P = 4.00
```

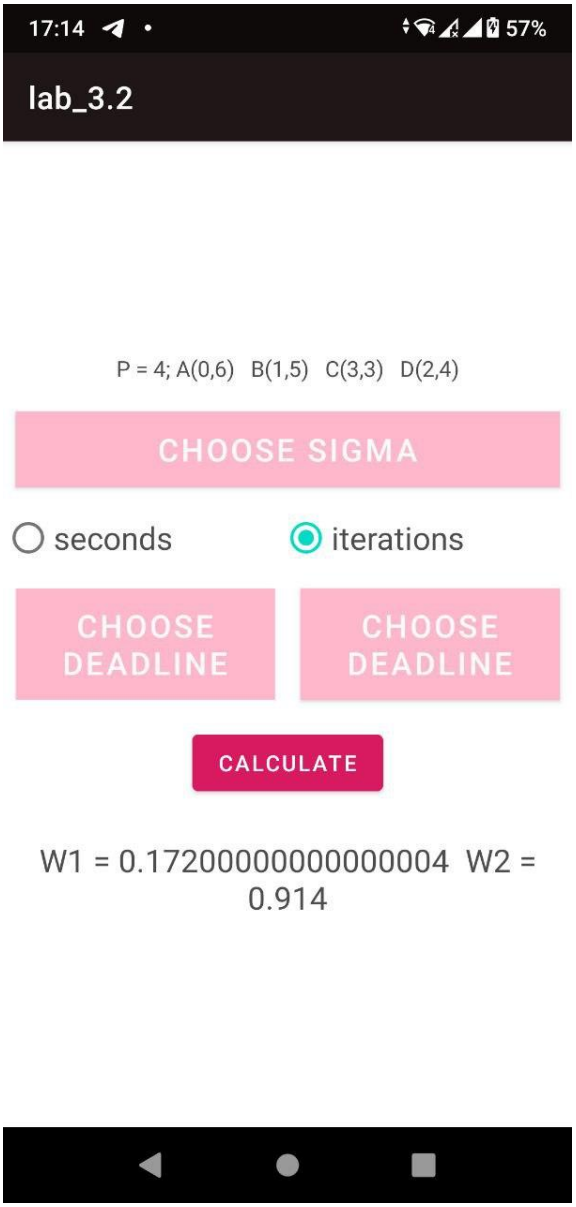
```
        val points = arrayListOf(Pair(0.00, 6.00), Pair(1.00, 5.00),
Pair(3.00, 3.00), Pair(2.00, 4.00))

        fun check(): Boolean {
            for (i in 0 .. 3){
                var y = W1 * points[i].first + W2 * points[i].second
                if ((i < 2 && y < P) || (i >= 2 && y > P) ) return false
            }
            return true
        }

        fun result(): Pair<Double, Double> {
            val startTime = System.currentTimeMillis()
            for (i in 0..iterations) {
                if ((System.currentTimeMillis() - startTime) <= time *
1000) {
                    for (k in 0 until points.size) {
                        val y = W1 * points[k].first + W2 *
points[k].second
                        val delta = P - y
                        W1 += delta * points[k].first * speed
                        W2 += delta * points[k].second * speed
                        if (check()) {
                            return Pair(W1, W2)
                        }
                    }
                }
            }
            return Pair(W1, W2)
        }

        val res =  result()
        tv_result.text = "W1 = ${res.first}  W2 = ${res.second}"
    }
}
```

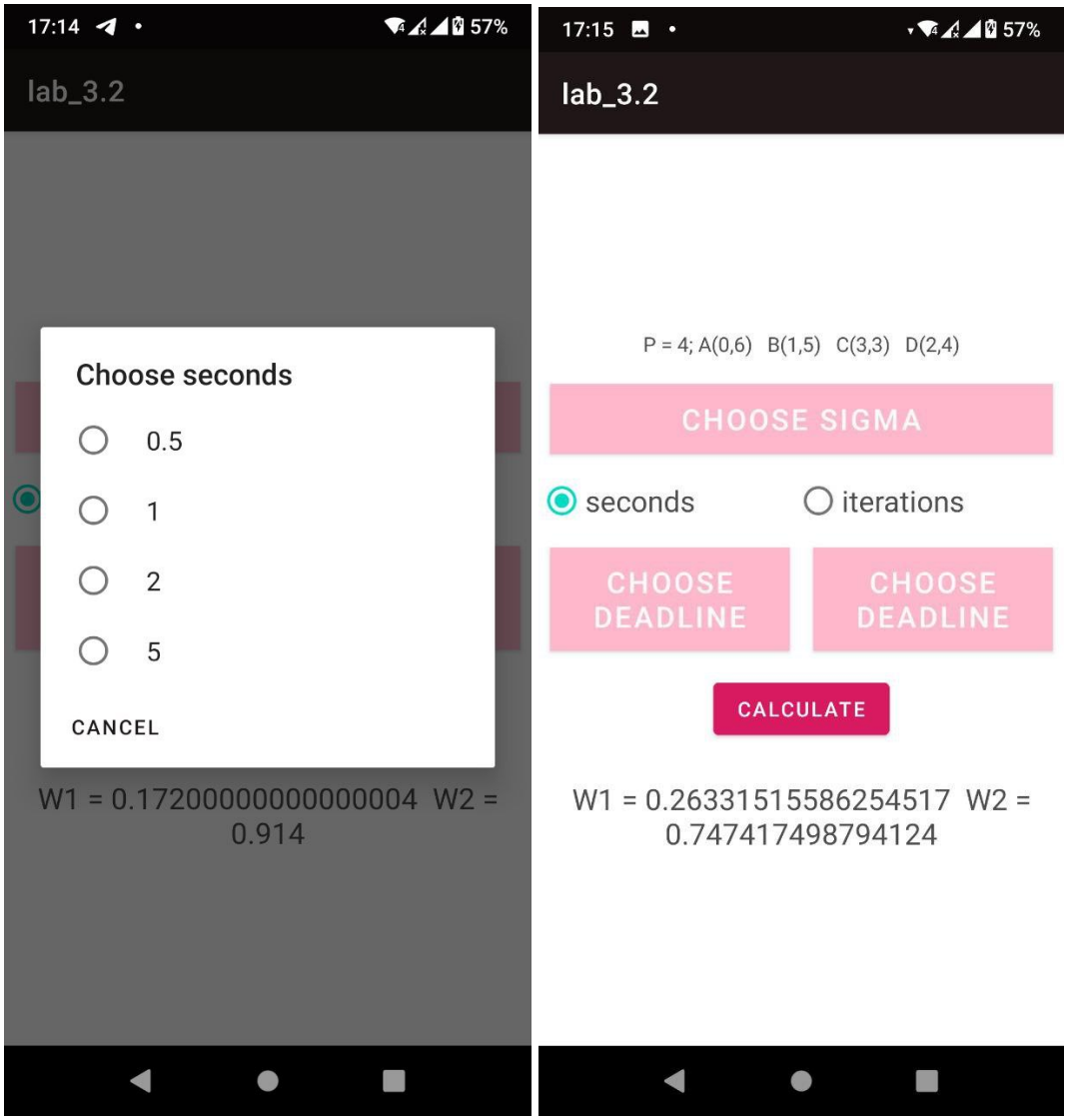
Приклад роботи програми:


17:13
57%
lab_3.2
P = 4; A(0,6) B(1,5) C(3,3) D(2,4)
CHOOSE SIGMA
seconds
iterations
CHOOSE DEADLINE
CHOOSE DEADLINE
CALCULATE
Here will be a result


17:14
57%
lab_3.2
Choose sigma
0.001
0.01
0.05
0.1
0.2
0.3
CANCEL


18:20
53%
lab_3.2
Choose iterations
100
200
500
1000
CANCEL
Here will be a result


17:14
57%
lab_3.2
P = 4; A(0,6) B(1,5) C(3,3) D(2,4)
CHOOSE SIGMA
seconds
iterations
CHOOSE DEADLINE
CHOOSE DEADLINE
CALCULATE
W1 = 0.17200000000000004 W2 = 0.914



Висновки:

Під час виконання лабораторної роботи я ознайомилась з основними принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрон(Perceptron). Змоделювала роботу нейронної мережі та дослідила вплив параметрів на час виконання та точність результату.

Я розробила відповідну програму та реалізувала користувацький інтерфейс з можливістю вводу даних за допомогою Android.