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-83 викладач  
Фібрук Руслан Сергійович Регіда Павло Геннадійович  
номер залікової книжки: 8320

Київ 2021

**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 А(1,5), В(2,4), поріг спрацювання Р = 4, швидкість навчання δ = 0.1. Початкові значення ваги візьмемо нульовими W1 = 0, W2 = 0. Розрахунок вихідного сигналу y виконується за наступною формулою: x1 \* W1 + x2 \* W2 = y Для кожного кроку потрібно застосувати дельта-правило, формула для розрахунку похибки: Δ = P – y де y – значення на виході. Для розрахунку ваги, використовується наступна формули: W1(i+1) = W1(i) + W2 \* x11 W2(i+1) = W1(i) + W2 \* x12 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: Δ = 4 – 0 = 4 За допомогою швидкості навчання δ та минулих значень ваги, розрахуємо нові значення ваги: W1 = 0 + 4 \* 1 \* 0,1 = 0,4 W2 = 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. Δ = -5 , спрощуємо результат для прикладу. W1 = 0,4 + 5 \* 2 \* 0,1 = -0,6 W2 = 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. Δ = 5 , спрощуємо результат для прикладу. W1 = -0,6 + 5 \* 1 \* 0,1 = -0,1 W2 = 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 W1 = -1,8 та W2 = 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.

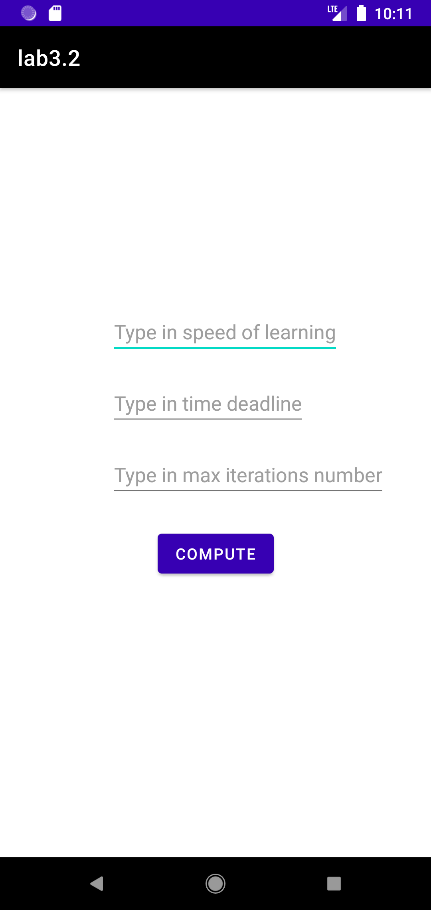
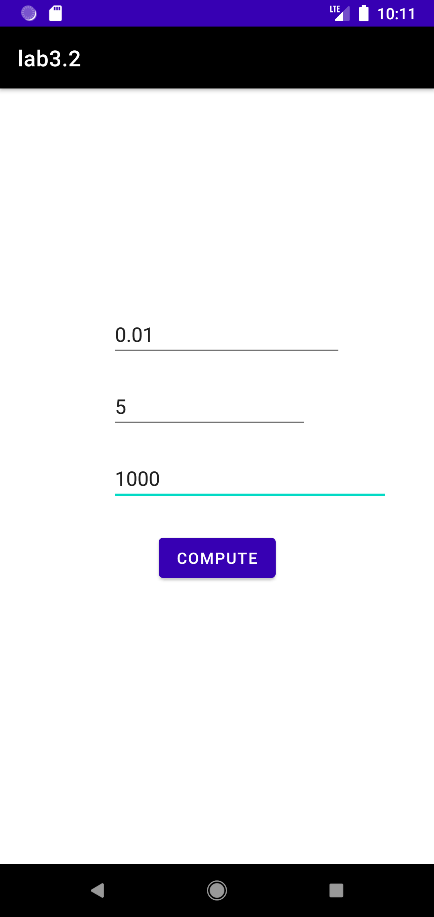
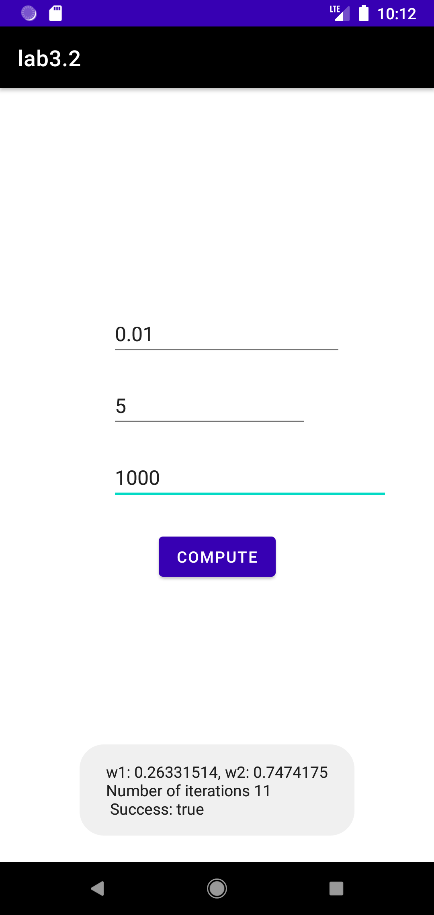
**Завдання на лабораторну роботу**

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.

**Лістинг програми**

package ua.kpi.comsys.ip8313.lab31  
  
import android.content.Context  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.view.inputmethod.InputMethodManager  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
  
class MainActivity : AppCompatActivity() {  
 private val threshold = 4  
 private var speed = 0f  
 private var deadline = 0f  
 private var maxIterations = 0  
 private val points = *listOf*(Pair(0f, 6f), Pair(1f, 5f), Pair(3f, 3f), Pair(2f, 4f))  
  
 private var iteration = 0  
 private var w1 = 0f  
 private var w2 = 0f  
  
 private fun success(): Boolean {  
 points.*forEachIndexed* **{** i, point **->** val multiplier = if (i < points.size / 2) -1 else 1  
 val signal = point.first \* w1 + point.second \* w2  
 if (multiplier \* threshold - multiplier \* signal <= 0) return false  
 **}** return true  
 }  
  
 private fun learn() {  
 val start = System.currentTimeMillis()  
 while (!success() && iteration < maxIterations) {  
 val end = System.currentTimeMillis()  
 if (end - start >= deadline) {  
 Toast.makeText(this, "Time's up!", Toast.*LENGTH\_LONG*).show()  
 return  
 }  
 val point = points[iteration % 4]  
 val signal = point.first \* w1 + point.second \* w2  
 val delta = threshold - signal  
 w1 += delta \* point.first \* speed  
 w2 += delta \* point.second \* speed  
 iteration++  
 }  
 }  
  
 private fun hideKeyboard() {  
 val imm = this.getSystemService(Context.*INPUT\_METHOD\_SERVICE*) as InputMethodManager  
 imm.hideSoftInputFromWindow(*currentFocus*!!.*windowToken*, 0)  
 }  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 val speedInputView = findViewById<EditText>(R.id.*speed\_input*)  
 val deadlineInputView = findViewById<EditText>(R.id.*deadline\_input*)  
 val maxIterationInputView = findViewById<EditText>(R.id.*max\_iteration\_input*)  
 val submitButton = findViewById<Button>(R.id.*compute\_button*)  
  
 submitButton.setOnClickListener **{** speed = speedInputView.*text*.toString().*toFloatOrNull*() ?: 0f  
 deadline = deadlineInputView.*text*.toString().*toFloatOrNull*() ?: 0f  
 maxIterations = maxIterationInputView.*text*.toString().*toIntOrNull*() ?: 0  
 if (speed == 0f || deadline == 0f || maxIterations == 0) {  
 Toast.makeText(this, "Inputs shouldn't be empty", Toast.*LENGTH\_LONG*).show()  
 } else {  
 learn()  
 hideKeyboard()  
 Toast.makeText(this,"w1: $w1, w2: $w2\nNumber of iterations $iteration\n Success: ${success()}", Toast.*LENGTH\_LONG*).show()  
 }  
 **}** }  
}

**Результати роботи програми**

**Висновки**

При виконанні цієї лабораторної роботи ми реалізували алгоритм машинного навчання за допомогою моделі Перцептрон та створили користувацький інтерфейс для введення швидкості навчання, часового дедлайну й максимальної кількості ітерацій та відображення результату.