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:
студент групи ІП-84
Гудь В.В.
№ залікової книжки: ІП-8405

Перевірив:
викладач
Регіда П.Г.

Київ 2021

Теоретичні відомості

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 y 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 на лабораторну роботу

Поріг спрацювання: P = 4

Дано точки: A(0,6), B(1,5), C(3,3), D(2,4).

Швидкості навчання: $\delta = \{0,001; 0,01; 0,05; 0.1; 0.2; 0,3\}$

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.

Вихідний код

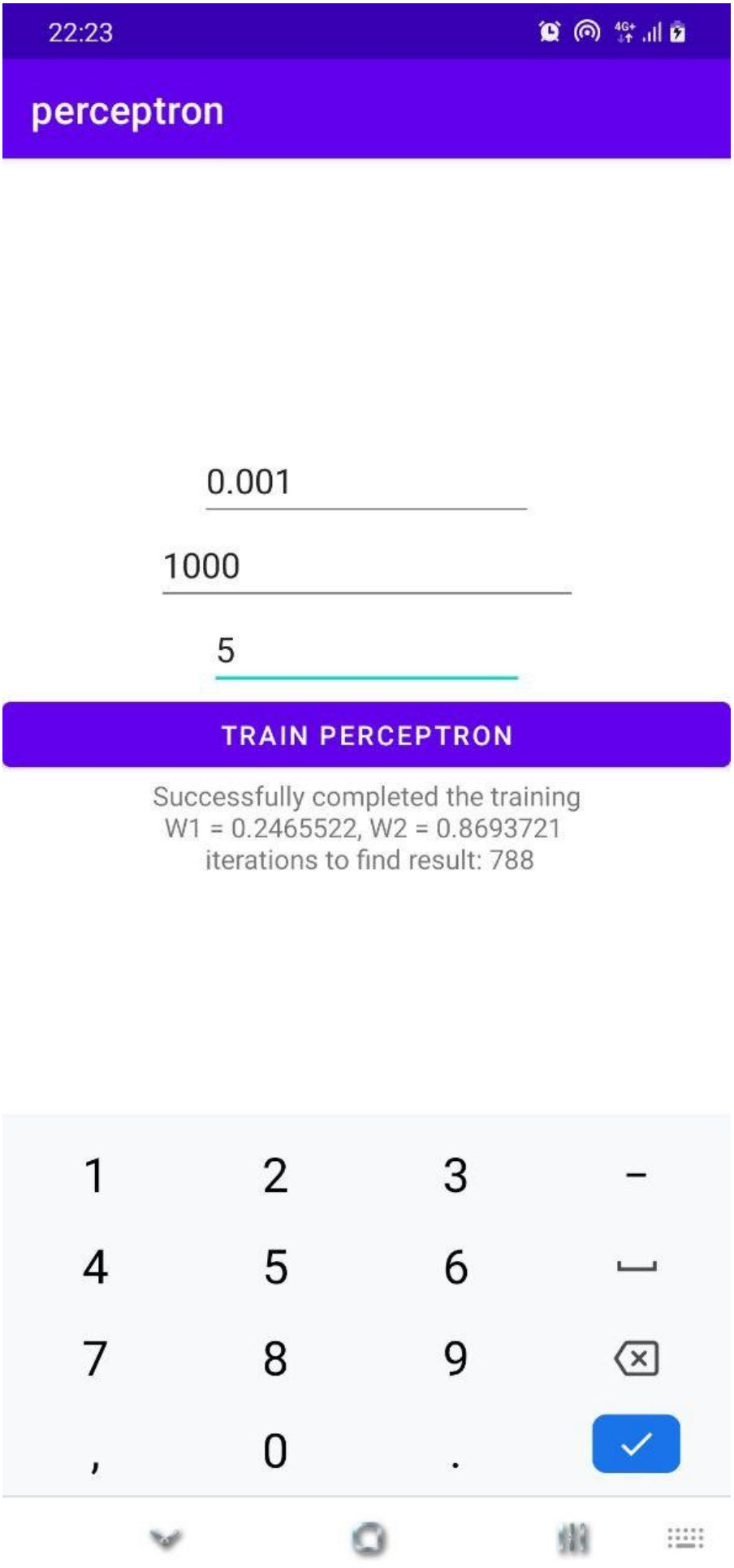
```
private fun train(): String {
    var w1 = 0F
    var w2 = 0F
    var i = 0
    var resultNum = 0
    val n = points.size
    val startTime = System.currentTimeMillis()
    while (i < iterations && resultNum < n) {
        val time = System.currentTimeMillis()
        if (time - startTime > timeDeadline) return "Time deadline exceeded\nW1 = $w1, W2
= $w2 \n iterations to find result: $i"
        val currentPoint = points[i % n]
        val diff = getDiff(currentPoint, w1, w2)
```

```
            if (currentPoint.second == threshold) {
                if (abs(diff - threshold) > 0.03f) {
                    val d = threshold - diff
                    w1 += d * currentPoint.first * learningSpeed
                    w2 += d * currentPoint.second * learningSpeed
                    resultNum = 0
                    if (w1 in illegalValues || w2 in illegalValues) {
                        error("Solution could not be found\n"
                                + "Number of iterations: $i")
                    }
                } else {
                    ++resultNum
                }
            } else if (currentPoint.second < threshold) {
                if (diff > threshold) {
                    val d = threshold - diff
                    w1 += d * currentPoint.first * learningSpeed
                    w2 += d * currentPoint.second * learningSpeed
                    resultNum = 0
                    if (w1 in illegalValues || w2 in illegalValues) {
                        error("Solution could not be found\n" +
                                "Number of iterations: $i")
                    }
                } else {
                    ++resultNum
                }
            } else {
                if (diff < threshold) {
                    val d = threshold - diff
                    w1 += d * currentPoint.first * learningSpeed
                    w2 += d * currentPoint.second * learningSpeed
                    resultNum = 0
                    if (w1 in illegalValues || w2 in illegalValues) {
                        error("Solution could not be found"
                                + "Number of iterations: $i")
                    }
                } else {
                    ++resultNum
                }
            }
            ++i
        }
        return "Successfully completed the training\nW1 = $w1, W2 = $w2 \n iterations to find
result: $i"
    }

    private fun getDiff(point: Pair<Float, Float>, w1: Float, w2: Float): Float {
        return point.first * w1 + point.second * w2
```

```
}
```

Результати роботи програми


22:23
perceptron
0.001
1000
5
TRAIN PERCEPTRON
Successfully completed the training
W1 = 0.2465522, W2 = 0.8693721
iterations to find result: 788
1
2
3
-
4
5
6
7
8
9
,
0
.

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.

.