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 3 курсу

ФІОТ, групи ІП-84

Кришталь Дмитро Вікторович

Номер залікової - 8414

Перевірив:

Регіда П. Г.

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 A(1,5), B(2,4), поріг спрацювання P = 4, швидкість навчання δ = 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.

Умови завдання

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.

Лістинг програми із заданими умовами завдання

```
import UIKit

class SecondViewController: UIViewController {

    @IBOutlet var button: UIButton!
    @IBOutlet var sigmaField: UITextField!
    @IBOutlet var iterField: UITextField!
    @IBOutlet var resutLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(calculateButtonTapped), for: .touchUpInside)

    }

    @objc func calculateButtonTapped() {
        iterField.resignFirstResponder()
        sigmaField.resignFirstResponder()
        let perc = Perceptron(iterations: Int(iterField.text!)!, sigma: Float(sigmaField.text!)!)
        let result = perc.main()
        resutLabel.text = "w1:\(result.0!); w2:\(result.1!)"
    }
}

class Perceptron {
    var points = [(0,6),(1,5),(3,3),(2,4)]
    var learningSpeed:Float? = nil
    var iterations:Int? = nil
    let p: Float = 4
    var y: Float? = nil
    var delta: Float? = nil
    var w1: Float = 0
    var w2: Float = 0
```

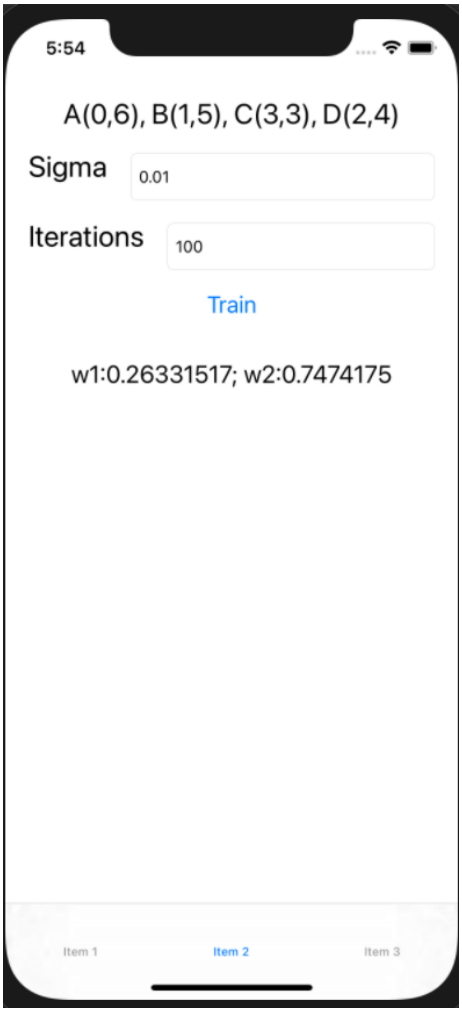
```
    init(iterations: Int, sigma: Float) {
        self.iterations = iterations
        self.learningSpeed = sigma
    }
    func main() -> (Float?, Float?){
        var i = 0
        while (i < iterations!) {
            let currentPoint = points[i % points.count]

            y = Float(currentPoint.0) * w1 + Float(currentPoint.1) * w2
            if((y! > p && (i % 4 == 0 || i % 4 == 1)) || (y! < p && (i % 4 == 2 || i % 4 == 3))) {
                var res = true
                for j in 0..<points.count {
                    if(j < 2) {
                        res = (Float(points[j].0) * w1 + Float(points[j].1) * w2) > p
                    } else {
                        res = (Float(points[j].0) * w1 + Float(points[j].1) * w2) < p
                    }
                    if(!res){
                        break
                    }
                }
                if(res) {
                    return (w1,w2)
                }

            }
            delta = p - y!
            w1 = w1 + delta! * learningSpeed! * Float(currentPoint.0)
            w2 = w2 + delta! * learningSpeed! * Float(currentPoint.1)

            i += 1
        }
        return (nil,nil)
    }
}
```

Результати виконання програми



Висновок

Під час виконня лабораторної роботи №3.2 ознайомився з основними принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрон. Написав програму. Правильність її роботи підтверджують наведені скріншоти.