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
Скрипник Єлена Сергіївна
номер залікової книжки: 8422

Перевірив:

викладач
Регіда Павло Геннадійович

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 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:

Поріг спрацювання: P = 4

Дано точки: A(0,6), B(1,5), C(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 main.dart:

```
import 'package:flutter/material.dart';
import './Perceptron.dart';

void main() {
  runApp(MaterialApp(
    home: HomePage(),
  ));
}
```

```
class HomePage extends StatefulWidget {
  @override
  _HomeAppState createState() => _HomeAppState();
}

class _HomeAppState extends State<HomePage> {
  int page = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: getAppBar(),
      body: getBody(),
    );
  }

  Widget getBody() {
    List<Widget> pages = [
      PerceptionPage(),
    ];
    return IndexedStack(
      index: page,
      children: pages,
    );
  }

  Widget getAppBar() {
    return AppBar(
      backgroundColor: Colors.teal,
      title: Container(
        child: Row(
          children: <Widget>[
            Text(
              "Лабораторна робота 3.2",
              style: TextStyle(fontSize: 20.0),
            ),
          ],
        ),
      ),
    );
  }
}

class PerceptionPage extends StatefulWidget {
  PerceptionPage({Key key}) : super(key: key);

  @override
  State<StatefulWidget> createState() {
    return __PerceptionPageState();
  }
}
```

```
class __PerceptionPageState extends State<PerceptionPage> {
  double w1;
  double w2;
  int iterations;
  double time;

  final iterationsTextController = TextEditingController();
  final maxTimeTextController = TextEditingController();
  final learningRateTextController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(body: getBody());
  }

  Widget getBody() {
    return Center(
      child: ListView
        (children: <Widget>[
        Padding(
          padding: const EdgeInsets.symmetric(vertical: 10.0),
        ),
        Container(
          width: 300,
          child: TextField(
            controller: iterationsTextController,
            decoration: InputDecoration(
              hintText: 'Кількість ітерацій',
              hintStyle: TextStyle(color: Colors.black),
              border: OutlineInputBorder(),
            ),
          ),
        ),
        SizedBox(
          height: 5.0,
        ),
        Container(
          width: 300,
          child: TextField(
            controller: maxTimeTextController,
            decoration: InputDecoration(
              hintText: 'Часовий дедлайн',
              hintStyle: TextStyle(color: Colors.black),
              border: OutlineInputBorder(),
            ),
          ),
        ),
        SizedBox(
          height: 5.0,
        ),
        Container(
          width: 300,
```

```
  child: TextField(
    controller: learningRateTextController,
    decoration: InputDecoration(
      hintText: 'Швидкість навчання',
      hintStyle: TextStyle(color: Colors.black),
      border: OutlineInputBorder(),
    ),
  ),
),
Padding(
  padding: const EdgeInsets.symmetric(vertical: 16.0),
  child: ElevatedButton.icon(
    label: Text('Порахувати',
    style: new TextStyle(
      fontSize: 20.0,
    ),
    ),
    icon: Icon(Icons.calculate),
    style: ElevatedButton.styleFrom(
      primary: Colors.teal,
      shadowColor: Colors.black26,
      elevation: 5,
    ),
    onPressed: () {
      var learningRate = double.parse(learningRateTextController.text);
      var maxTime = double.parse(maxTimeTextController.text);
      var maxIterations = double.parse(iterationsTextController.text);

      var result = new Perceptron(4, learningRate).learn([
        [0, 6],
        [1, 5],
        [3, 3],
        [2, 4]
      ], maxIterations, maxTime);

      setState(() {
        w1 = result[0];
        w2 = result[1];
        time = result[2];
        iterations = result[3];
      });
    },
  ),
),
SizedBox(
  height: 5.0,
),
Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    Padding(
      padding: const EdgeInsets.only(top: 4.0),
```

```
            child:
            Text('W1: ${this.w1 ?? ' '}', style: TextStyle(fontSize: 20)),
          ),
          Padding(
            padding: const EdgeInsets.only(top: 4.0),
            child:
            Text('W2: ${this.w2 ?? ' '}', style: TextStyle(fontSize: 20)),
          ),
          Padding(
            padding: const EdgeInsets.only(top: 4.0),
            child: Text('Час: ${this.time ?? ' '}',
                style: TextStyle(fontSize: 20)),
          ),
          Padding(
            padding: const EdgeInsets.only(top: 4.0),
            child: Text('Кількість ітерацій: ${this.iterations ?? ' '}',
                style: TextStyle(fontSize: 20)),
          ),
        ],
      ),
    ]),
  );
 }
}
```

Лістинг програми Perceptron.dart:

```
class Perceptron {
 int p;
 double r;
 Perceptron(this.p, this.r);

 double w1 = 0;
 double w2 = 0;

 calculateSignal(point) {
  double x1 = point[0].toDouble();
  double x2 = point[1].toDouble();
  return this.w1 * x1 + w2 * x2;
 }

 getDelta(y) {
  double delta = this.p - y;
  if (delta > 0) {
   return delta;
  }
  return 0;
 }

 weightAdjustment(point, delta) {
  double x1 = point[0].toDouble();
  double x2 = point[1].toDouble();
  this.w1 += delta * x1 * this.r;
```

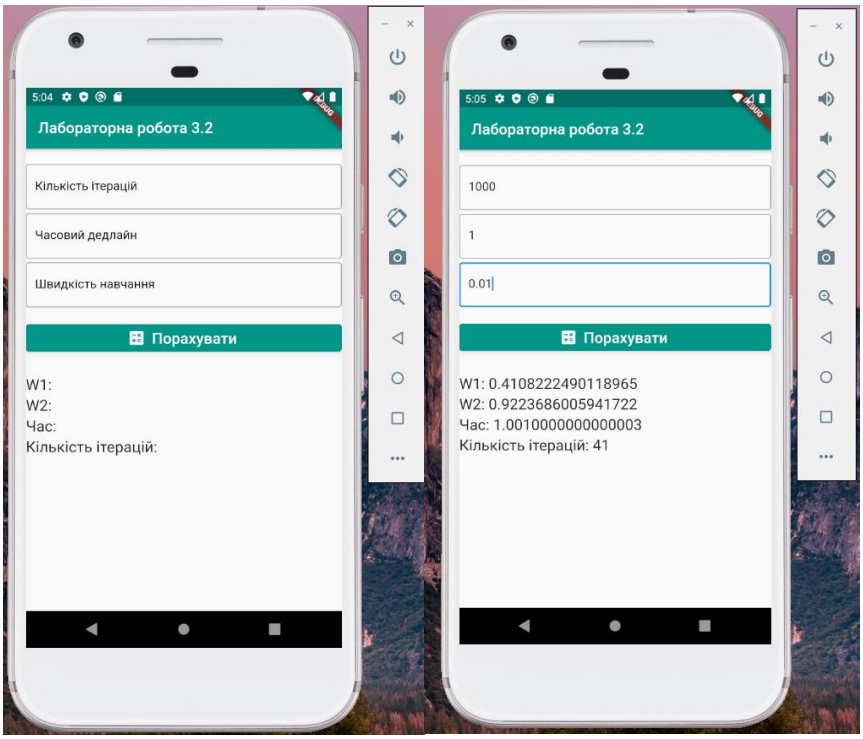
```
    this.w2 += delta * x2 * this.r;
  }

  learn(input, maxIterations, maxTime) {
    double time = 0;
    int iterations = 0;

    while (maxIterations > iterations && maxTime > time) {
      var startDate = DateTime.now().microsecondsSinceEpoch;
      for (final value in input) {
        var y = this.calculateSignal(value);
        var delta = this.getDelta(y);

        this.weightAdjustment(value, delta);
      }
      var endDate = DateTime.now().microsecondsSinceEpoch;
      time += (endDate - startDate) / 1000;
      iterations++;
    }
    return [w1, w2, time, iterations];
  }
}
```

Результат виконання:



Висновки:

На даній лабораторній роботі було здійснено ознайомлення з принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрон. Також було змодельовано роботу нейронної мережі та досліджено вплив параметрів на точність результату.