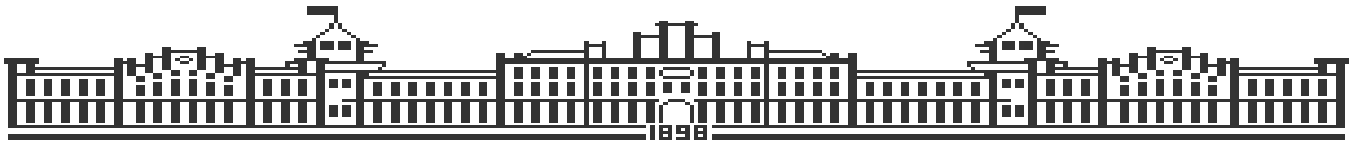


Міністерство освіти і науки України
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЗВІТ

про виконання лабораторної роботи №3.2

з дисципліни

«Інтелектуальні вбудовані системи»

на тему:

«ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ. МОДЕЛЬ PERCEPTRON»

Перевірив:
асистент кафедри ОТ
Регіда П. Г

ВИКОНАВ:
студент 3 курсу
групи ІП-83, ФІОТ
Мінченко В.Ю.
Залікова книжка №8315
Варіант – 18

Київ 2021

Завдання на лабораторну роботу

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 = {0.5c; 1c; 2c; 5c}, 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

Програмний код:

Perceptron.dart

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

String perceptron({
  @required String currentPoints,
  @required int thresholdOperation,
  @required double speed,
  int maxIterations,
  double maxTime
}) {
  double w1 = 0;
  double w2 = 0;
  double time = 0;
  int iterations = 0;
  final parsedStr = currentPoints.replaceAll(new RegExp(r'(\D)'), '').split('');
  final points = List.generate(parsedStr.length ~/ 2,
    (i) => [int.parse(parsedStr[2 * i]), int.parse(parsedStr[2 * i + 1])]
  );

  final stopwatch = Stopwatch()..start();
  while (maxIterations != 0 && maxIterations > iterations ||
    maxTime != 0 && maxTime*1000 > stopwatch.elapsedMilliseconds
  ) {
    points.forEach((point) {
      final currentY = calcY(point, w1, w2);
      final delta = calcDelta(thresholdOperation, currentY);
      w1 += weightCalc(point[0], delta, speed);
      w2 += weightCalc(point[1], delta, speed);
    });
    iterations++;
  }
  time = stopwatch.elapsedMilliseconds / 1000;
  return 'W1: $w1, W2: $w2, time: $time, iterations: $iterations';
}

double calcY(List<int> point, double w1, double w2) =>
  point[0] * w1 + point[1] * w2;

double calcDelta(int thresholdOperation, double y) =>
  thresholdOperation - y ?? 0;
```

```
double weightCalc(
  int point, double delta, double speed
) => point * delta * speed;
```

perceptron_screen.dart

```
import 'package:flutter/material.dart';
import 'package:lab3_mobile/helpers/perceptron.dart';

class Perceptron extends StatefulWidget {
  @override
  _PerceptronState createState() => _PerceptronState();
}

class _PerceptronState extends State<Perceptron> {
  @override
  // ignore: override_on_non_overriding_member
  final _points = 'A(0,6), B(1,5), C(3,3), D(2,4)';
  int _thresholdOperation = 4;
  String _chosenSpeed;
  String _chosenTime;
  bool isSwitched = false;
  String _chosenIteration;
  String resultValue = '';
  bool _offstage = true;

  Widget build(BuildContext context) {
    return Container(
      child: Padding(
        padding: const EdgeInsets.all(40.0),
        child: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Padding(
                padding: const EdgeInsets.symmetric(vertical: 10),
                child: Text(
                  _points,
                  style: TextStyle(
                    color: Colors.indigo, fontWeight: FontWeight.w500),
                ),
              ),
              Padding(
                padding: const EdgeInsets.symmetric(vertical: 10),
                child: Text(
                  'Threshold of operation: $_thresholdOperation',
                  style: TextStyle(
                    color: Colors.purple, fontWeight: FontWeight.w500),
                ),
              ),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
```

```
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 10),
          child: Text('Learning speed'),
        ),
        _buildDropDownElement(
          items: ['0.001', '0.01', '0.05', '0.1', '0.2', '0.3'],
          chosenValue: _chosenSpeed,
          onChanged: (String value) => {
            setState(() {
              _chosenSpeed = value;
            })
          }
        ),
      ],
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 10),
          child: Text('Deadline:'),
        ),
        Row(
          children: [
            Text('Time'),
            Switch(
              value: isSwitched,
              onChanged: (value) {
                setState(() {
                  isSwitched = value;
                });
              },
              activeTrackColor: Colors.yellow,
              inactiveTrackColor: Colors.blue,
              activeColor: Colors.orangeAccent,
            ),
            Text('Iterations')
          ],
        ),
      ],
    ),
    !isSwitched
      ? Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Padding(
              padding: const EdgeInsets.symmetric(horizontal: 10),
              child: Text('Time'),
            ),
            _buildDropDownElement(
              items: ['0.5', '1', '2', '5'],
              chosenValue: _chosenTime,
              onChanged: (String value) => {
                setState(() {
```

```
                    _chosenTime = value;
                  })
                }),
              Text('c'),
            ],
          )
        : Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Padding(
                padding: const EdgeInsets.symmetric(horizontal: 10),
                child: Text('Number of iteration'),
              ),
              _buildDropDownElement(
                items: ['100', '200', '500', '1000'],
                chosenValue: _chosenIteration,
                onChanged: (String value) => {
                  setState(() {
                    _chosenIteration = value;
                  })
                }
              ),
            ],
          ),
      Offstage(
        offstage: _offstage,
        child: Padding(
          padding: const EdgeInsets.all(25.0),
          child: Text(
            resultValue,
            style: TextStyle(
              color: resultValue.contains('*')
                  ? Colors.orange
                  : Colors.red
              ),
          ),
        ),
      ),
      _offstage ? SizedBox(height: 8.0) : SizedBox(height: 0.0),
      ElevatedButton(
        child: const Text('Calculate'),
        style: ElevatedButton.styleFrom(
          primary: Colors.purple,
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.all(Radius.circular(10.0))),
        ),
        onPressed: () {
          setState(() {
            _offstage = false;
            resultValue = perceptron(
              currentPoints: _points,
              thresholdOperation: _thresholdOperation,
              speed: double.parse(_chosenSpeed),
              maxIterations: int.parse(_chosenIteration ?? '0'),
```

```
                    maxTime: double.parse(_chosenTime ?? '0')
                  );
                });
              },
            ),
          ],
        ),
      ),
    ),
  );
 }

 Widget _buildDropDownElement({
   List<String> items,
   String chosenValue,
   ValueChanged<String> onChanged
 }) =>
   DropdownButton<String>(
     value: chosenValue,
     style: TextStyle(color: Colors.black),
     items: items.map<DropdownMenuItem<String>>((String value) {
       return DropdownMenuItem<String>(
         value: value,
         child: Text(value),
       );
     }).toList(),
     onChanged: onChanged
   );
}
```

Bottom_navigation.dart

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

class TabItem {
  TabItem({this.label, this.title, this.icon, this.backgroundColor});
  final String label;
  final String title;
  final icon;
  final Color backgroundColor;
}

List<TabItem> allTabItems = <TabItem>[
  TabItem(
      icon: Icon(Icons.miscellaneous_services),
      label: 'factorization',
      title: 'Fermat`s factorization example',
      backgroundColor: Colors.redAccent[400]),
  TabItem(
      icon: Icon(Icons.mediation),
      label: 'perceptron',
      title: 'Perceptron example',
      backgroundColor: Colors.tealAccent[400]),
```

```
  TabItem(
      icon: Icon(Icons.more_time),
      label: 'lab33',
      title: 'lab33',
      backgroundColor: Colors.deepPurpleAccent[400]),
];

class BottomNavigation extends StatelessWidget {
  BottomNavigation({@required this.currentIndex, @required this.onSelectTab});

  final int currentIndex;
  final ValueChanged<int> onSelectTab;

  @override
  Widget build(BuildContext context) {
    return BottomNavigationBar(
      type: BottomNavigationBarType.shifting,
      items: allTabItems
          .map((TabItem tabItem) => BottomNavigationBarItem(
              icon: tabItem.icon,
              backgroundColor: tabItem.backgroundColor,
              label: tabItem.label))
          .toList(),
      currentIndex: currentIndex,
      selectedItemColor: Colors.white,
      unselectedItemColor: Colors.grey,
      onTap: onSelectTab,
      iconSize: 30,
    );
  }
}
```

Main.dart

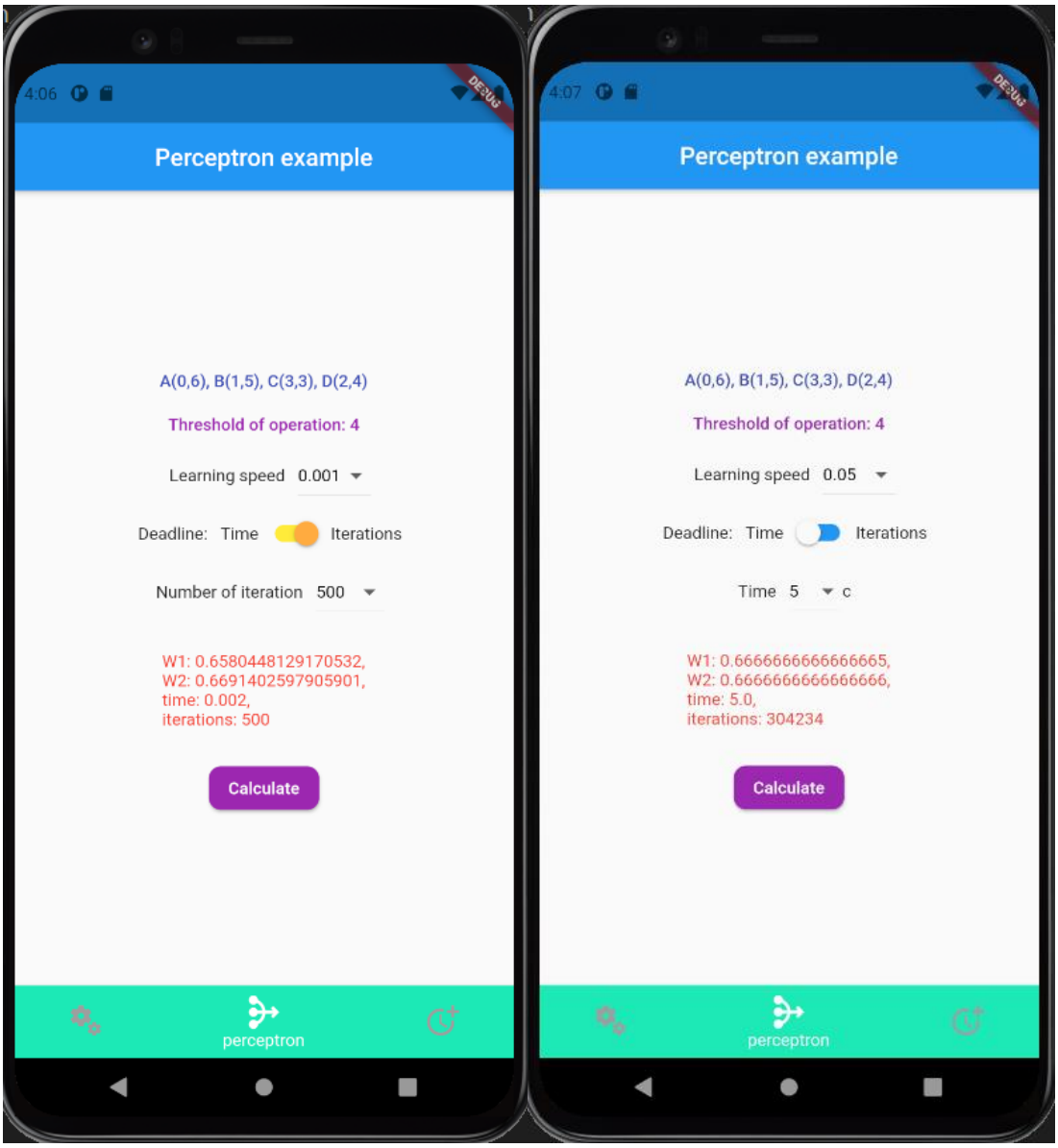
```
import 'package:flutter/material.dart';
import 'screens/main_screen.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MainScreen(),
    );
```

```
  }
}
```

Результати роботи програми:



Висновки:
Отже, в ході лабораторної роботи, ми отримали навички з мащинного навчання на прикладі математичної моделі Перцептрон та змоделювали його роботу у вигляді клієнтського додатку із зручним інтерфейсом.
Результати наведено в звіті та врепозиторії. Кінцеву мету було досягнуто.