

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  
Романова Вікторія Андріївна  
номер залікової книжки: 8418

Перевірив:  
ас. кафедри ОТ  
Регіда П. Г.

Київ 2021

## Теоретичні відомості

Важливою задачею яку 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 Перцептрон 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  $W_1 = 0$ ,  $W_2 = 0$ . 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:

$$W_1(i+1) = W_1(i) + \Delta * x_{1i}$$

$$W_2(i+1) = W_2(i) + \Delta * x_{2i}$$

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  $\Delta$ :

$$\Delta = 4 - 0 = 4$$

За допомогою швидкості навчання  $\delta$  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.

## Лістинг коду

### MainActivity.kt

```
package ua.kpi.comsys.lab32
```

```
import androidx.appcompat.app.AppCompatActivity
```

```
import android.os.Bundle
```

```
import kotlinx.coroutines.*
```

```
import ua.kpi.comsys.lab32.databinding.ActivityMainBinding
```

```
import kotlin.system.measureTimeMillis
```

```

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val ls = arrayOf("0.001", "0.01", "0.05", "0.1", "0.2", "0.3")
        val td = arrayOf("0.5c", "1c", "2c", "5c")
        val mi = arrayOf("100", "200", "500", "1000")

        with(binding.learningSpeed) {
            minValue = 0
            maxValue = ls.size - 1
            displayedValues = ls
        }

        with(binding.timeDeadline) {
            minValue = 0
            maxValue = td.size - 1
            displayedValues = td
        }

        with(binding.maxIterations) {
            minValue = 0
            maxValue = mi.size - 1
            displayedValues = mi
        }

        with(binding) {
            btn.setOnClickListener {

```

```
val timer = td[timeDeadline.value]
```

```
var clock: Job? = null
```

```
val train = GlobalScope.launch {
```

```
    val result: Array<String>
```

```
    val time = measureTimeMillis {
```

```
        result = perceptron(
```

```
            ls[learningSpeed.value].toFloat(),
```

```
            mi[maxIterations.value].toInt()
```

```
        )
```

```
    }
```

```
clock?.cancelAndJoin()
```

```
withContext(Dispatchers.Main) {
```

```
    iterations.text = result[0]
```

```
    w1.text = result[1]
```

```
    w2.text = result[2]
```

```
    timeResult.text = ("$time ms")
```

```
}
```

```
}
```

```
clock = GlobalScope.launch {
```

```
    delay((timer.subSequence(0, timer.lastIndex).toString().toFloat() *  
1000).toLong())
```

```
    train.cancelAndJoin()
```

```
withContext(Dispatchers.Main) {
```

```
    iterations.text = "N/A"
```

```
    w1.text = "Not found"
```

```
    w2.text = "Not found"
```

```
    timeResult.text = timer
```

```
}
```

```

    }
  }
}
}

```

```

private fun perceptron(ls: Float, mi: Int): Array<String> {
    val p = 4
    var delta: Float
    val points = arrayOf(0 to 6, 1 to 5, 3 to 3, 2 to 4)
    val more = points.filter { it.second > p }
    val less = points.filter { it.second <= p }

    var w1 = 0f
    var w2 = 0f

    fun y(x1: Int, x2: Int) = x1 * w1 + x2 * w2
    fun next(w: Float, delta: Float, x: Int) = w + delta * x * ls

    for (i in 0..mi) {
        val point = points[i % points.size]
        delta = p - y(point.first, point.second)

        if (more.all { y(it.first, it.second) > p } && less.all { y(it.first, it.second) < p
    })

        return arrayOf(i.toString(), w1.toString(), w2.toString())

        w1 = next(w1, delta, point.first)
        w2 = next(w2, delta, point.second)
    }
    return arrayOf(mi.toString(), "Not found", "Not found")
}
}

```

## **activity\_main.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="horizontal">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="15dp"
            android:text="@string/_0_6"
            android:textColor="@color/black"
            android:textStyle="bold" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="15dp"
            android:text="@string/_1_5"
            android:textColor="@color/black"
            android:textStyle="bold" />

    </LinearLayout>

</LinearLayout>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:text="@string/_3_3"
    android:textColor="@color/black"
    android:textStyle="bold" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="15dp"
    android:text="@string/d_2_4"
    android:textColor="@color/black"
    android:textStyle="bold" />
```

```
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:orientation="horizontal">
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="15dp"
    android:text="@string/learning_speed" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```



```
android:layout_marginHorizontal="15dp"  
android:text="@string/time_deadline" />
```

```
<TextView
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginHorizontal="15dp"  
    android:text="@string/max_iterations" />
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"  
    android:orientation="horizontal">
```

```
<NumberPicker
```

```
    android:id="@+id/learning_speed"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginHorizontal="20dp" />
```

```
<NumberPicker
```

```
    android:id="@+id/time_deadline"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginHorizontal="20dp" />
```

```
<NumberPicker
```

```
    android:id="@+id/max_iterations"  
    android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="20dp" />
```

```
</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/p"
        android:textColor="@color/black"
        android:textStyle="bold" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="5dp"
        android:text="@string/_4"
        android:textColor="@color/black"
        android:textStyle="bold" />
```

```
</LinearLayout>
```

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="15dp">
```

```
<TableRow>
```

```
  <TextView
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="@string/w1" />
```

```
  <TextView
```

```
    android:id="@+id/w1"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginHorizontal="8dp" />
```

```
</TableRow>
```

```
<TableRow>
```

```
  <TextView
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="@string/w2" />
```

```
  <TextView
```

```
    android:id="@+id/w2"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginHorizontal="8dp" />
```

```
</TableRow>
```

```
</TableLayout>
```

```
<TableLayout
```

```
  android:layout_width="match_parent"
```

```
  android:layout_height="wrap_content"
```

```
  android:layout_margin="15dp">
```

```
<TableRow>
```

```
  <TextView
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="@string/time" />
```

```
  <TextView
```

```
    android:id="@+id/time_result"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginHorizontal="8dp" />
```

```
</TableRow>
```

```
<TableRow>
```

```
  <TextView
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="@string/iterations" />
```

```
  <TextView
```

```
    android:id="@+id/iterations"
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_marginHorizontal="8dp" />
```

```
</TableRow>
```

```
</TableLayout>
```

```
<Button
```

```
  android:id="@+id/btn"
```

```
  android:layout_width="match_parent"
```

```
  android:layout_height="wrap_content"
```

```
  android:layout_margin="25dp"
```

```
        android:text="@string/train" />  
</LinearLayout>
```

**Результати виконання**

13:08

lab32

A(0,6)	B(1,5)	C(3,3)	D(2,4)
Learning speed	Time deadline	Max iterations	
0.3	5c	500	
0.001	0.5c	1000	
0.01	1c	100	




P = 4

w1 = 0.32130277  
w2 = 0.7360953

Time: 3 ms  
Iterations: 144

TRAIN

13:08



lab32

A(0,6)	B(1,5)	C(3,3)	D(2,4)
Learning speed	Time deadline	Max iterations	
0.3	5c	1000	
0.001	0.5c	100	
0.01	1c	200	





P = 4

w1 = Not found  
w2 = Not found

Time: 4 ms  
Iterations: 100

TRAIN

13:08



lab32

A(0,6)	B(1,5)	C(3,3)	D(2,4)
Learning speed	Time deadline	Max iterations	
0.001	5c	1000	
0.01	0.5c	100	
0.05	1c	200	

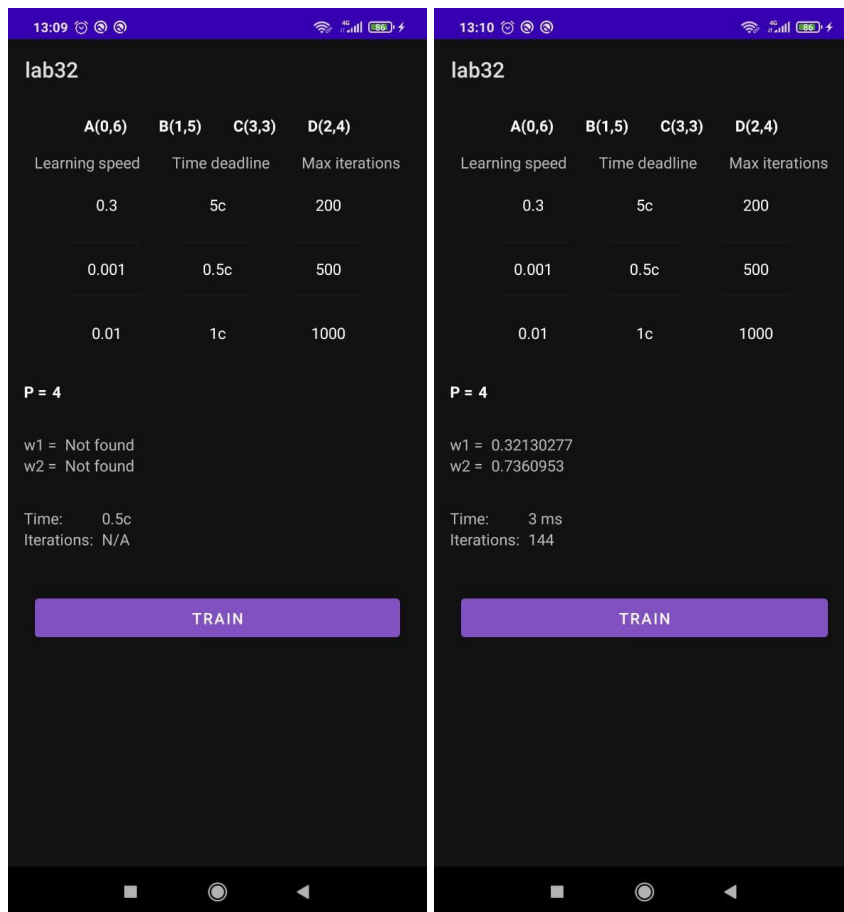
P = 4

w1 = 0.26331514  
w2 = 0.7474175

Time: 1 ms  
Iterations: 11

TRAIN

Через встроєний оптимізатор Kotlin з кожним запуском код виконується швидше, тому за однакових параметрів маємо різні результати:



## Висновок

Було проведено ознайомлення з принципами машинного навчання за допомогою математичної моделі сприйняття інформації Перцептрон(Perceptron). Змоделювати роботу нейронної мережі та дослідити вплив параметрів на час виконання та точність результату.