notebook

February 28, 2023

1 Phương pháp Toán cho TTNT

1.1 Thông tin học sinh viên

Họ tên: Lê Nhựt Nam

MSHV: 22C11067

[1]:
```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1.2 Phân tách tuyến tính để giải quyết bài toán hàm OR

Dữ liệu

Với a, b là đầu vào, c là kết quả phép OR giữa a và b

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

[3]:
```
X_or = np.array([[0, 0, 1, 1], [0, 1, 0, 1]]).T
y_or = np.array([0, 1, 1, 1]).reshape(-1, 1)
```

[4]:
```
X_or_Aug = np.c_[np.ones(X_or.shape[0]), X_or]
X_or_Aug
```

[4]:
```
array([[1., 0., 0.],
       [1., 0., 1.],
       [1., 1., 0.],
       [1., 1., 1.]])
```

[5]:
```
def sign(s):
    return (0, 1)[s >= 0]
```

[6]:
```
def perceptron_or(x, w, b):
    return sign((np.dot(x, w) - b)[0])
```

[7]:
```
w_or = np.linalg.pinv(X_or_Aug) @ y_or
w_or
```

[7]:
```
array([[0.25],
       [0.5 ],
       [0.5 ]])
```

[8]:
```
print("or(1, 1) = {}".format(perceptron_or(x=[1, 1], w=w_or[1:3], b=w_or[0])))␣
 ↪# 1
print("or(1, 0) = {}".format(perceptron_or(x=[1, 0], w=w_or[1:3], b=w_or[0])))␣
 ↪# 1
print("or(0, 1) = {}".format(perceptron_or(x=[0, 1], w=w_or[1:3], b=w_or[0])))␣
 ↪# 1
print("or(0, 0) = {}".format(perceptron_or(x=[0, 0], w=w_or[1:3], b=w_or[0])))␣
 ↪# 0
```

```
or(1, 1) = 1
or(1, 0) = 1
or(0, 1) = 1
or(0, 0) = 0
```

1.3 Phân tách tuyến tính để giải quyết bài toán hàm AND

Dữ liệu

Với a, b là đầu vào, c là kết quả phép AND giữa a và b

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

[9]:
```
X_and = np.array([[0, 0, 1, 1], [0, 1, 0, 1]]).T
y_and = np.array([0, 0, 0, 1]).reshape(-1, 1)
```

[10]:
```
X_and_Aug = np.c_[np.ones(X_or.shape[0]), X_or]
X_and_Aug
```

[10]:
```
array([[1., 0., 0.],
       [1., 0., 1.],
       [1., 1., 0.],
       [1., 1., 1.]])
```

[11]:
```
w_and = np.linalg.inv(X_and_Aug.T @ X_and_Aug) @ X_and_Aug.T @ y_and

# Thêm bias :3, thực nghiệm em thấy nên trừ thêm 0.5 :D
w_and[0] -= 0.5

w_and
```

[11]:
```
array([[-0.75],
       [ 0.5 ],
       [ 0.5 ]])
```

[12]:
```
def perceptron_and(x, w, b):
    return sign((np.dot(x, w) + b)[0])
```

[13]:
```
print("and(1, 1) = {}".format(perceptron_and(x=[1, 1], w=w_and[1:3],␣
 ↪b=w_and[0]))) # 1
print("and(1, 0) = {}".format(perceptron_and(x=[1, 0], w=w_and[1:3],␣
 ↪b=w_and[0]))) # 1
print("and(0, 1) = {}".format(perceptron_and(x=[0, 1], w=w_and[1:3],␣
 ↪b=w_and[0]))) # 1
print("and(0, 0) = {}".format(perceptron_and(x=[0, 0], w=w_and[1:3],␣
 ↪b=w_and[0]))) # 0
```

```
and(1, 1) = 1
and(1, 0) = 0
and(0, 1) = 0
and(0, 0) = 0
```

1.4 Chơi đùa với hàm XOR bằng nhiều perceptron

1.4.1 Đầu tiên là xử lý hàm NOT

Dữ liệu

Với a, b là đầu vào, c là kết quả phép AND giữa a và b

a	b
0	1
1	0

[14]:
```
X_not = np.array([0, 1]).reshape(-1, 1)
y_not = np.array([1, 0]).reshape(-1, 1)
```

[15]:
```
X_not_Aug = np.c_[np.ones(X_not.shape[0]), X_not]
X_not_Aug
```

[15]:
```
array([[1., 0.],
       [1., 1.]])
```

[16]:
```
w_not = np.linalg.pinv(X_not_Aug) @ y_not
w_not
```

[16]:
```
array([[ 1.],
       [-1.]])
```

[17]:
```
def perceptron_not(x, w, b):
    return sign((np.dot(x, w) + b)[0])
```

[18]:
```
print("not(1) = {}".format(perceptron_not(x=[1], w=w_not[1], b=w_not[0]))) # 0
print("not(0) = {}".format(perceptron_not(x=[0], w=w_not[1], b=w_not[0]))) # 1
```

```
not(1) = 0
not(0) = 1
```

Ta biến đổi

$$\text{XOR}(x, y) = \text{AND}(OR(x, y), \text{NOT}(\text{AND}(x, y)))$$

[19]:
```
def xor_perceptron(x, w_and=w_and, w_not=w_not, w_or=w_or):
    and_x = perceptron_and(x=x, w=w_and[1:3], b=w_and[0])
    not_x = perceptron_not(x=and_x, w=w_not[1], b=w_not[0])
    or_x = perceptron_or(x=x, w=w_or[1:3], b=w_or[0])
    x = perceptron_and(x=[not_x, or_x], w=w_and[1:3], b=w_and[0])
    return x
```

[20]:
```
print("xor(1, 1) = {}".format(xor_perceptron([1, 1]))) # 0
print("xor(1, 0) = {}".format(xor_perceptron([1, 0]))) # 1
print("xor(0, 1) = {}".format(xor_perceptron([0, 1]))) # 1
print("xor(0, 0) = {}".format(xor_perceptron([0, 0]))) # 0
```

```
xor(1, 1) = 0
xor(1, 0) = 1
xor(0, 1) = 1
xor(0, 0) = 0
```