

2022-2023 学年秋季学期《模式识别》作业三

(特征提取、线性和非线性判别函数分类器)

教师：王玮

1. 对下面的四个样本应用感知器学习算法，直到其收敛。从权向量 $(w_0, w_1, w_2, w_3)^T = (1, 0, 0, 0)^T$ 开始，根据给定顺序循环地应用样本更新（每次用一个样本）。对于感知器学习的每一步，记下应用的样本、分类结果和权向量的更新。

$$(4, 3, 6)^T \in \mathcal{N}, \quad (2, -2, 3)^T \in \mathcal{P}, \quad (1, 0, -3)^T \in \mathcal{P}, \quad (4, 2, 3)^T \in \mathcal{N}$$

```
1966 x 1458
homework2-3-(1)(4).ipynb homework2-3-(2)(3).ipynb test.c homework3-1.ipynb x homework3-1.ipynb (output) ewc.py homework2-1.ipynb
+ 代码 + Markdown + 全部运行 清除所有单元格输出 重启 变量表 Outline ... base (Python 3.9.12)

def pla(X, y):
    """
    感知器学习算法实现
    注意：只能处理线性可分的数据集，输入线性不可分的数据集函数将无法停止
    args:
        X - 训练数据集
        y - 目标标签值
    return:
        w - 权重系数
    """
    done = False
    # 初始化权重系数
    w = np.zeros(X.shape[1])
    print(f'当前的w权重系数为:{w}')
    print(f'!'*78)
    # 循环
    while(not done):
        done = True
        # 遍历训练数据集
        for index in range(len(X)):
            x = X[index]
            print(f'当前使用的样本是{x}')
            print(f'当前的分类结果是:{np.sign(x.dot(w) * y)}')
            # 判定是否与目标值不符
            if x.dot(w) * y[index] <= 0:
                done = False
                # 修正权重系数
                w = w + y[index] * x
                print(f'更正后分类结果是:{np.sign(x.dot(w) * y)}')
                print(f'更新后w权重系数为:{w}')
                print(f'!'*78)
    return w

def main():
    x=[
        [4,3,6],
        [2,-2,3],
        [1,0,-3],
        [4,2,3]
    ]
    x=np.array(x)
    y=[-1,1,1,-1]
    y=np.array(y)
    w=pla(x,y)
    main()

[7] ✓ 0.3s Python

[8] ✓ 0.4s Python

... Output exceeds the size limit. Open the full output data in a text editor
当前的w权重系数为:[0. 0. 0.]
=====
当前使用的样本是[4 3 6]
当前的分类结果是:[0. 0. 0.]
更正后分类结果是:[ 1. -1. -1.  1.]
更新后w权重系数为:[-4. -3. -6.]
=====
当前使用的样本是[ 2 -2  3]
当前的分类结果是:[ 1. -1. -1.  1.]
更正后分类结果是:[ 1. -1. -1.  1.]
更新后w权重系数为:[-2. -5. -3.]
=====
当前使用的样本是[ 1  0 -3]
当前的分类结果是:[-1.  1.  1. -1.]
更正后分类结果是:[-1.  1.  1. -1.]
更新后w权重系数为:[-2. -5. -3.]
=====
当前使用的样本是[4 2 3]
当前的分类结果是:[ 1. -1. -1.  1.]
更正后分类结果是:[ 1. -1. -1.  1.]
更新后w权重系数为:[-2. -5. -3.]
=====
底部 输出 代码 JUPYTER 注册 00:02:00:03 Python 告 告 告
```

```

1792 x 1434 100%
homework3-1.ipynb (output) - vswork - Visual Studio Code
homework2-3-(1)(4).ipynb homework2-3-(2)(3).ipynb C test.c homework3-1.ipynb homework3-1.ipynb (output) x ewc.py homework2-1.ipynb

1 当前的权重系数为:[0. 0. 0.]
2 =====
3 当前使用的样本是[4 3 6]
4 当前的分类结果是:[0. 0. 0.]
5 更正后分类结果是:[ 1. -1. -1. 1.]
6 更新后权重系数为:[-4. -3. -6.]
7 =====
8 当前使用的样本是[ 2 -2 3]
9 当前的分类结果是:[ 1. -1. -1. 1.]
10 更正后分类结果是:[ 1. -1. -1. 1.]
11 更新后权重系数为:[-2. -5. -3.]
12 =====
13 当前使用的样本是[ 1 0 -3]
14 当前的分类结果是:[-1. 1. 1. -1.]
15 更正后分类结果是:[-1. 1. 1. -1.]
16 更新后权重系数为:[-2. -5. -3.]
17 =====
18 当前使用的样本是[4 2 3]
19 当前的分类结果是:[ 1. -1. -1. 1.]
20 更正后分类结果是:[ 1. -1. -1. 1.]
21 更新后权重系数为:[-2. -5. -3.]
22 =====
23 当前使用的样本是[4 3 6]
24 当前的分类结果是:[ 1. -1. -1. 1.]
25 更正后分类结果是:[ 1. -1. -1. 1.]
26 更新后权重系数为:[-2. -5. -3.]
27 =====
28 当前使用的样本是[ 2 -2 3]
29 当前的分类结果是:[ 1. -1. -1. 1.]
30 更正后分类结果是:[-1. 1. 1. -1.]
31 更新后权重系数为:[ 0. -7. 0.]
32 =====
33 当前使用的样本是[ 1 0 -3]
34 当前的分类结果是:[0. 0. 0.]
35 更正后分类结果是:[-1. 1. 1. -1.]
36 更新后权重系数为:[ 1. -7. -3.]
37 =====
38 当前使用的样本是[4 2 3]
39 当前的分类结果是:[ 1. -1. -1. 1.]
40 更正后分类结果是:[ 1. -1. -1. 1.]
41 更新后权重系数为:[ 1. -7. -3.]
42 =====
43 当前使用的样本是[4 3 6]
44 当前的分类结果是:[ 1. -1. -1. 1.]
45 更正后分类结果是:[ 1. -1. -1. 1.]
46 更新后权重系数为:[ 1. -7. -3.]
47 =====
48 当前使用的样本是[ 2 -2 3]
49 当前的分类结果是:[-1. 1. 1. -1.]
50 更正后分类结果是:[-1. 1. 1. -1.]
51 更新后权重系数为:[ 1. -7. -3.]
52 =====
53 当前使用的样本是[ 1 0 -3]
54 当前的分类结果是:[-1. 1. 1. -1.]
55 更正后分类结果是:[-1. 1. 1. -1.]
56 更新后权重系数为:[ 1. -7. -3.]
57 =====
58 当前使用的样本是[4 2 3]
59 当前的分类结果是:[ 1. -1. -1. 1.]
60 更正后分类结果是:[ 1. -1. -1. 1.]
61 更新后权重系数为:[ 1. -7. -3.]
62 =====
63

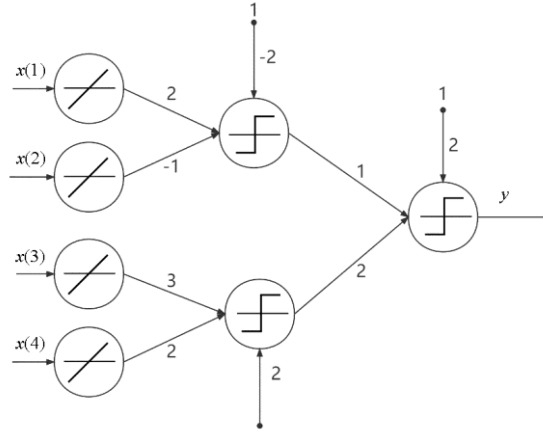
```

3. 有如下图所示的 3 层感知器网络，其中隐含层和输出层均采用符号函数作为激活函数：

$$f(u) = \begin{cases} +1, & u \geq 0 \\ -1, & u < 0 \end{cases}$$

请计算针对下列输入的网络输出，并确定其类别属性： $x(1) \ x(2) \ x(3) \ x(4) \ y$

$$\mathbf{x}_1: (-1, +1, +1, -1)^T, \mathbf{x}_2: (+1, +1, -1, -1)^T, \mathbf{x}_3: (-1, +1, -1, +1)^T$$



第 3 题图

对于 $x_1 = (-1, +1, +1, -1)$,

$$Hid1 = \text{sign}[((-1) * 2 + 1 * (-1)) - 2] = \text{sign}(-5) = -1$$

$$Hid2 = \text{sign}[(1 * 3 + (-1) * 2) + 2] = \text{sign}(3) = 1$$

$$Out1 = \text{sign}[((-1) * 1 + 1 * 2 + 2)] = \text{sign}(3) = 1$$

属于正类

对于 $x_2 = (+1, +1, -1, -1)$,

$$Hid1 = \text{sign}[(1 * 2 + 1 * (-1)) - 2] = \text{sign}(-1) = -1$$

$$Hid2 = \text{sign}[((-1) * 3 + (-1) * 2) + 2] = \text{sign}(-3) = -1$$

$$Out1 = \text{sign}[(-1) * 1 + (-1) * 2 + 2] = \text{sign}(-1) = -1$$

属于负类

对于 $x_3 = (-1, +1, -1, +1)$,

$$Hid1 = \text{sign}[((-1) * 2 + 1 * (-1)) - 2] = \text{sign}(-5) = -1$$

$$Hid2 = \text{sign}[(-1) * 3 + 1 * 2 + 2] = \text{sign}(1) = 1$$

$$Out1 = \text{sign}[(-1) * 1 + 1 * 2 + 2] = \text{sign}(3) = 1$$

属于正类

4. 线性 SVM 分类器在参数 $C = 5$ 的条件下得到如下训练结果:

第一类支持向量: $\mathbf{x}_1 = (1, 1, 1)^T, \alpha_1 = 4$

第二类支持向量: $\mathbf{x}_2 = (1, 2, 1)^T, \alpha_2 = 2; \mathbf{x}_3 = (0, 1, 1)^T, \alpha_3 = 2$

写出对应的线性判别函数, 并判别下列样本的类别属性, 以及相对于支持面和判别界面的位置关系:

$$\mathbf{y}_1 = (0.2, 1, 2)^T, \mathbf{y}_2 = (2, 2, 1)^T, \mathbf{y}_3 = (3, -3, 2)^T$$

$$\mathbf{w} = \sum_{i=1}^3 \alpha_i \mathbf{z}_i \mathbf{x}_i = 4 * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - 2 * \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} - 2 * \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ 0 \end{pmatrix}$$

$$\mathbf{w}_0 = \mathbf{z}_1 - \mathbf{w}^T \mathbf{x}_1 = 1 - (2, -2, 0) * \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 1$$

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \mathbf{w}_0 = 2x_1 - 2x_2 + 1$$

$$g(\mathbf{y}_1) = 2 * 0.2 - 2 * 1 + 1 = -0.6 \text{ 在第二类支持面和分类界面之}$$

间

$$g(y_2) = 2 * 2 - 2 * 2 + 1 = 1, \text{在第一类支持面上}$$

$$g(y_3) = 2 * 3 - 2 * (-3) + 1 = 1, \text{在第一类支持面上}$$

5. 【附加题】编程实现 BP 算法，其中隐含层和输出层的激活函数采用双曲正切型 Sigmoid 函数。学习如下训练样本和期望输出的三层感知器网络。

$$\mathbf{x}_1 = (-1, -1)^T, \mathbf{t}_1 = (-1, -1)^T, \mathbf{x}_2 = (+1, +1)^T, \mathbf{t}_2 = (-1, +1)^T$$

$$\mathbf{x}_3 = (+1, -1)^T, \mathbf{t}_3 = (+1, +1)^T, \mathbf{x}_4 = (-1, +1)^T, \mathbf{t}_4 = (+1, +1)^T$$

```

homework3-5.ipynb > import numpy as np
+ 代码 + Markdown | ▶ 全部运行 | 清除所有单元格输出 | 重启 | 变量表 | Outline ...
> bias1 As 第1项, 共3项 ↑ ↓ ×

import numpy as np

#定义sigmoid函数
def sigmoid(x, deriv = False):
    if(deriv == True):
        return x*(1-x)
    else:
        return 1/(1+np.exp(-x))

#input dataset
X = np.array([[[-1,-1],
               [-1,-1],
               [1,1],
               [-1,1],
               [1,-1],
               [1,1],
               [-1,1],
               [1,1]]])

#output dataset
y = np.array([[1,0,1,0,1,0,1,0]]).T

#初始化权重
weight01 = 2*np.random.random((2,4)) - 1
weight12 = 2*np.random.random((4,2)) - 1
weight23 = 2*np.random.random((2,1)) - 1

#初始化偏倚
b1 = 2*np.random.random((1,4)) - 1
b2 = 2*np.random.random((1,2)) - 1
b3 = 2*np.random.random((1,1)) - 1
bias1=np.array([b1[0],b1[0],b1[0],b1[0],b1[0],b1[0],b1[0],b1[0]])
bias2=np.array([b2[0],b2[0],b2[0],b2[0],b2[0],b2[0],b2[0],b2[0]])
bias3=np.array([b3[0],b3[0],b3[0],b3[0],b3[0],b3[0],b3[0],b3[0]])

#开始训练
for j in range(60000):
    I0 = X
    O0=I0
    I1=np.dot(O0,weight01)+bias1
    O1=sigmoid(I1)
    I2=np.dot(O1,weight12)+bias2
    O2=sigmoid(I2)
    I3=np.dot(O2,weight23)+bias3
    O3=sigmoid(I3)

    f3_error = y-O3

    if(j%10000) == 0:
        print ("Error:"+str(np.mean(f3_error)))

    f3_delta = f3_error*sigmoid(O3,deriv = True)

    f2_error = f3_delta.dot(weight23.T)

    f2_delta = f2_error*sigmoid(O2,deriv = True)

    f1_error = f2_delta.dot(weight12.T)

    f1_delta = f1_error*sigmoid(O1,deriv = True)

    weight23 += O2.T.dot(f3_delta) #调整权重
    weight12 += O1.T.dot(f2_delta)
    weight01 += O0.T.dot(f1_delta)

    bias3 += f3_delta #调整偏倚
    bias2 += f2_delta
    bias1 += f1_delta

print ("outout after Training:")
print (O3)

```

13) ✓ 1.9s Python

```

... Error:0.13119485955513172
Error:-0.0012982018831981643
Error:-0.0009644311240975727
Error:-0.0008070427644909614
Error:-0.00070999194001720683
Error:-0.000642058784949639
outout after Training:
[[0.99842674]
 [0.00273346]
 [0.99842032]
 [0.00276148]
 [0.99845609]
 [0.00277685]
 [0.99837683]
 [0.00277685]]

```