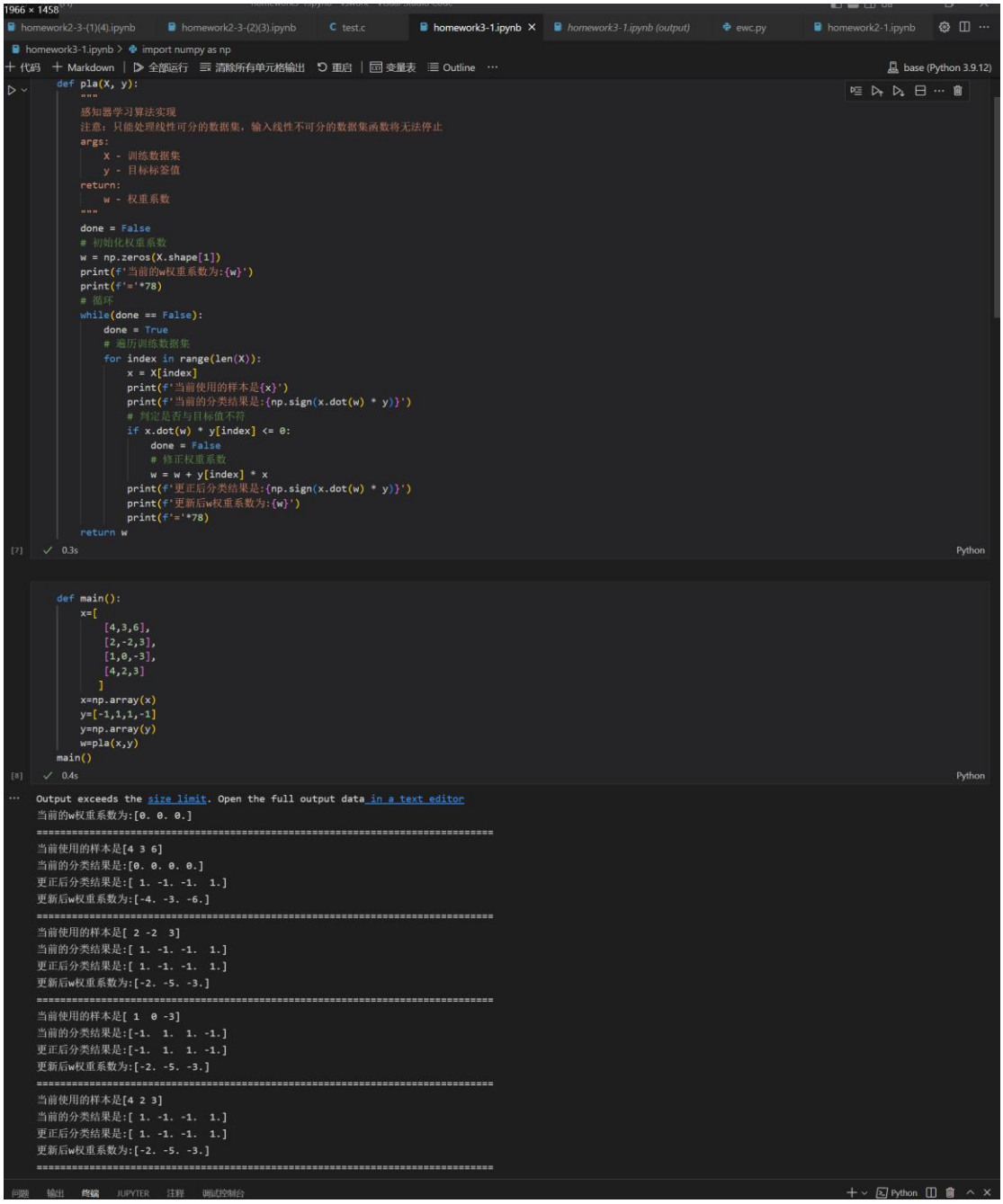
2022-2023 学年秋季学期《模式识别》作业三

（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。

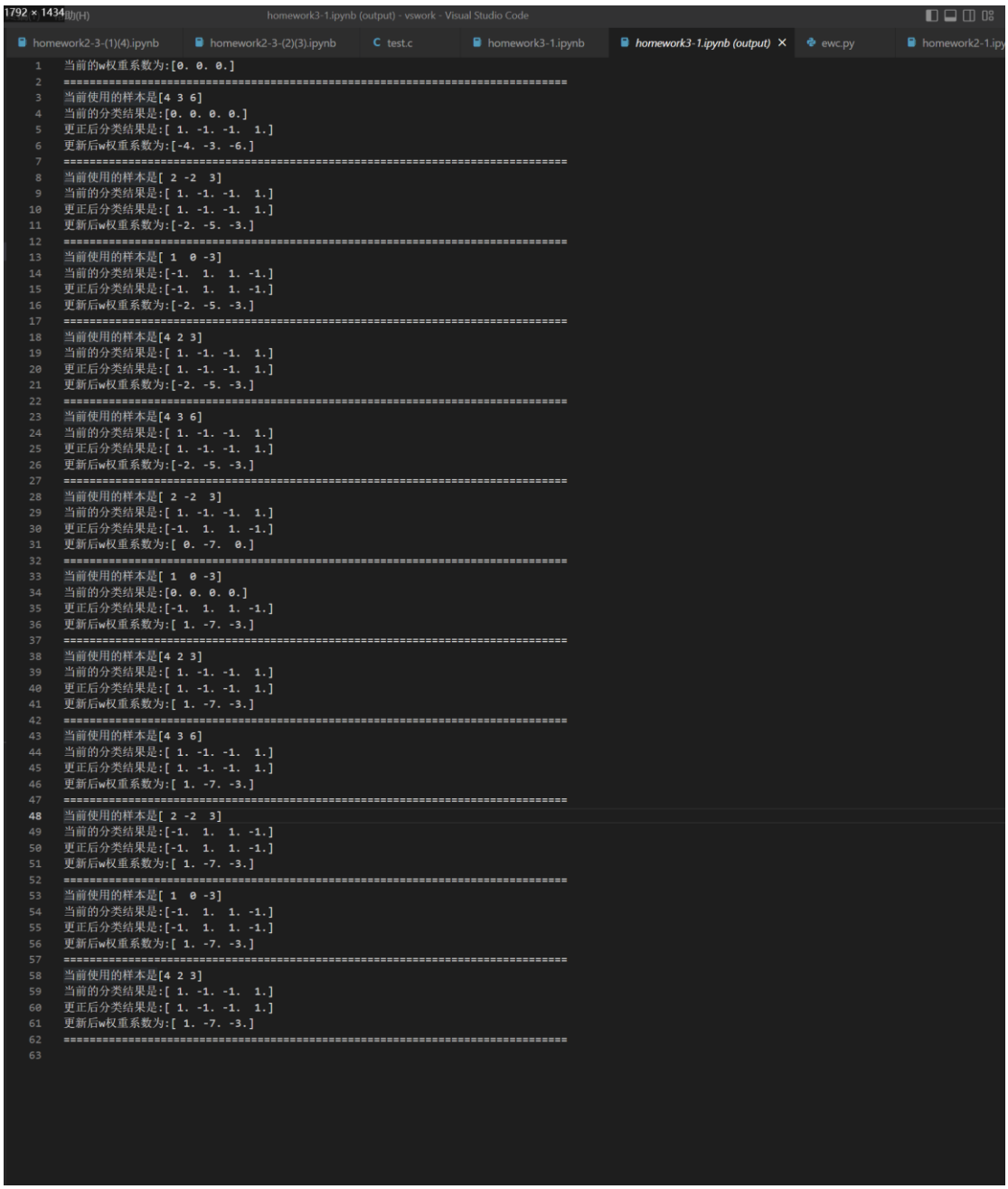
$$(4,3,6)^T \in \mathcal{N}, \qquad (2,-2,3)^T \in \mathcal{P}, \qquad (1,0,-3)^T \in \mathcal{P}, \qquad (4,2,3)^T \in \mathcal{N}$$



```
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(done == False):
        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()
```

```
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. 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.]
==============================================================================
```

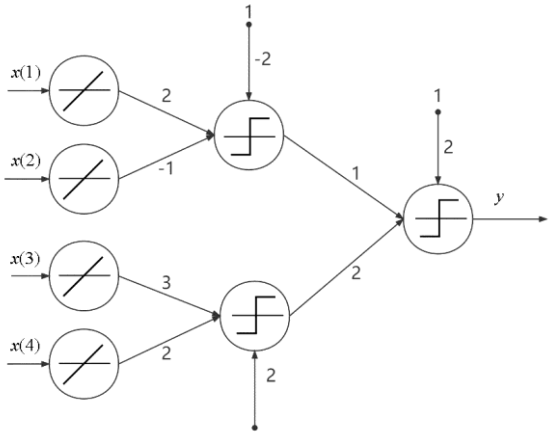


3. 有如下图所示的 3 层感知器网络，其中隐含层和输出层均采用符号函数作为激活函数：

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

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

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



第 3 题图

对于 x1=（-1，+1，+1，-1），
Hid1=sign[（(-1）*2+1*（-1)）-2]=sign(-5)=-1
Hid2=sign[(1*3+(-1)*2)+2]=sign(3)=1
Out1=sign[((-1)*1+1*2+2)]=sign(3)=1
属于正类
对于 x2=（+1，+1，-1，-1），
Hid1=sign[(1*2+1*(-1)-2)]=sign(-1)=-1
Hid2=sign[((-1)*3+(-1)*2+2)]=sign(-3)=-1
Out1=sign[(-1)*1+(-1)*2+2]=sign(-1)=-1
属于负类
对于 x3=（-1，+1，-1，+1），
Hid1=sign[((-1)*2+1*(-1)-2)]=sign(-5)=-1
Hid2=sign[(-1)*3+1*2+2]=sign(1)=1
Out1=sign[(-1)*1+1*2+2]=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：$\boldsymbol{x_1} = (1,1,1)^T, \alpha_1 = 4$
第二类支持向量：$\boldsymbol{x_2} = (1,2,1)^T, \alpha_2 = 2; \boldsymbol{x_3} = (0,1,1)^T, \alpha_3 = 2$
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：

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

$$\boldsymbol{w} = \sum_{i=1}^{3} \boldsymbol{\alpha_i z_i x_i} = \boldsymbol{4} * \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{1} \\ \boldsymbol{1} \end{pmatrix} - \boldsymbol{2} * \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{2} \\ \boldsymbol{1} \end{pmatrix} - \boldsymbol{2} * \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{1} \\ \boldsymbol{1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{2} \\ \boldsymbol{-2} \\ \boldsymbol{0} \end{pmatrix}$$

$$\boldsymbol{w_0} = \boldsymbol{z_1} - \boldsymbol{w^T x1} = \boldsymbol{1} - (\boldsymbol{2}, \boldsymbol{-2}, \boldsymbol{0}) * \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{1} \\ \boldsymbol{1} \end{pmatrix} = \boldsymbol{1}$$

$$\boldsymbol{g(x) = w^T x + w_0 = 2x_1 - 2x_2 + 1}$$

$\boldsymbol{g(y_1) = 2 * 0.2 - 2 * 1 + 1 = -0.6}$在第二类支持面和分类界面之

间

$$\boldsymbol{g(y_2) = 2 * 2 - 2 * 2 + 1 = 1}\text{,在第一类支持面上}$$
$$\boldsymbol{g(y_3) = 2 * 3 - 2 * (-3) + 1 = 1}\text{,在第一类支持面上}$$

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。

$$\boldsymbol{x}_1 = (-1,-1)^T, \boldsymbol{t}_1 = (-1,-1)^T, \boldsymbol{x}_2 = (+1,+1)^T, \boldsymbol{t}_2 = (-1,+1)^T$$
$$\boldsymbol{x}_3 = (+1,-1)^T, \boldsymbol{t}_3 = (+1,+1)^T, \boldsymbol{x}_4 = (-1,+1)^T, \boldsymbol{t}_4 = (+1,+1)^T$$

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 第 1 项，共 3 项

```python
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)
```

[3] ✓ 1.9s Python

```
Error:0.13119485955513172
Error:-0.0012982018831981643
Error:-0.0009644311240975727
Error:-0.0008070427644909614
Error:-0.0007099194001720683
Error:-0.000642058784949639
outout after Training:
[[0.99842674]
 [0.00273346]
 [0.99842032]
 [0.00276148]
 [0.99845609]
 [0.00277685]
 [0.99837683]
 [0.00277685]]
```