山东大学

信息科学与工程学院

2022－2023 学年第一学期

实 验 报 告

课程名称：信息基础 II

专 业 班 级 崇新学堂

学 生 学 号

学 生 姓 名

课 程 报 告 感知器

1. 问题回答

Consider running the Perceptron algorithm on some sequence of examples *S* (an example is a data point and its label). Let *S′* be the same set of examples as *S*, but presented in a different order.

a) Does the Perceptron algorithm necessarily make the same number of mistakes on *S* as it does on *S′*?

Answer:

The number of mistakes on *S* as it does on *S′* is different.

b) If so, why? If not, show such an *S* and *S′* where the Perceptron algorithm makes a different number of mistakes on *S′* than it does on *S.*

Answer:

S: [(-3, -8), (6, -1), (1, -6), (-10, -2), (-9, -3), (3, 3), (2, 2), (-9, -3), (-3, 9), (-2, 4)]
Label: [-1, 1, 1, -1, -1, 1, 1, -1, -1, -1]

S’ [(-10, -2), (-3, 9), (-3, -8), (3, 3), (2, 2), (-9, -3), (6, -1), (1, -6), (-2, 4), (-9, -3)]
Label [[-1, -1, -1, 1, 1, -1, 1, 1, -1, -1]]

The number of mistakes on S is 3
The number of mistakes on S’ is 2

2.理论部分

在本实验中，为了讨论方便，我们将样本增加了一维常数：

$$x = (1; x_1; \cdots x_m) \tag{1}$$

相应地我们在权矢量中也增加一维度：

$$\theta = (\theta_0; \theta_1; \cdots \theta_m) \tag{2}$$

感知器算法的主要流程为，首先获得 m 个特征的输入，之后将每个输入值求加权和，倘若加权和达到某个阈值，则通过 sign（）函数输出 1，若未达到阈值，则输出 0。经过（1）（2）处理之后，我们提到的阈值，实际上为 $-\theta_0$，由此便可以简化我们的表达：

$$\hat{y}^{(i)} = sign(\theta^T x^{(i)}) \tag{3}$$

具体算法如下：

1. 初始化权向量，将权向量的每一个值赋一个随机值。
2. 对于每个训练样例，求取其预测输出 $\hat{y}$ 。
3. 当预测值不等于实际值时，利用以下公式对权值做出修正：

$$\theta = \theta + y^{(i)} \cdot x^{(i)} \tag{4}$$

反复修正，通过上网查阅相关资料，获取到改良版的修正策略：

$$\theta = \theta + \eta y^{(i)} x^{(i)} \tag{5}$$

其中$\eta > 0$，代表学习速率，在本次实验中，学习率取 1。

本次实验的数据形式如下：

$$Y = \begin{bmatrix} y_1 & \cdots & y_N \end{bmatrix} \qquad X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_M^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_1^{(N)} \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_M \end{bmatrix}$$

3. 代码实现

数据部分使用 random（）函数生成若干组随机数，随后使用 shuffle（）函数打乱输入的顺序，在本次实验中，随机数均为整数，且为二维数据，人为规定，当数据处于 x 轴正半轴时，其 label 为 1；当数据处于 x 轴负半轴时，其 label 为 0.确立了以上规则之后，对数据进行（1）（2）的拓展操作。代码部分如图 1 所示。

算法部分主要依靠 numpy（）库的相关操作，对感知器的算法进行代码复现，最后返回修正后的权数矩阵以及在迭代过程中分类错误的个数。代码部分如图 2 所示。

```
X1 = np.random.randint(-10,10,size=(50,3))   #定义一个50x3的矩阵，作为增广特征矩阵
X2 = np.array(ones((len(X1),3)))             #定义一个和增广特征矩阵维度相同的矩阵，用来储存打乱顺序之前的特征矩阵
for i in range(len(X1)):
    for j in range(len(X1[0])):
        X2[i][j] = X1[i][j]
np.random.shuffle(X1)                        #打乱顺序的过程
for i in range(len(X1)):
    X1[i][0] = 1
    X2[i][0] = 1                             #将增广特征矩阵每个向量的第一维度置1
Y1=np.array(ones((len(X1))))
Y2=np.array(ones((len(X1))))
for i in range(len(X1)):                     #设置label数值
    if X1[i][1] > 0.5:
        Y1[i]=1
    else:
        Y1[i]=-1
for i in range(len(X1)):
    if X2[i][1] > 0.5:
        Y2[i]=1
    else:
        Y2[i]=-1
```

图 1 数据部分

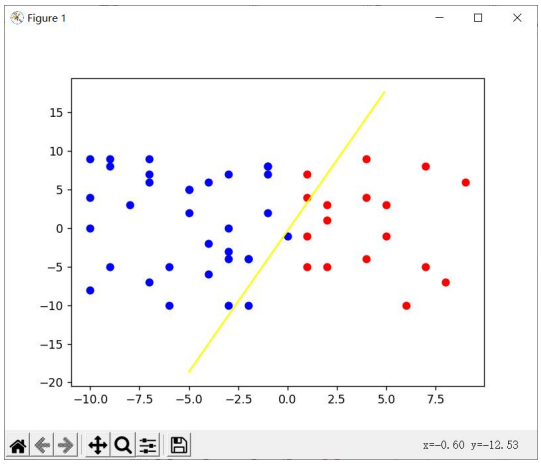
```
def Perceptron(X,label):
    n=1                                      #学习率这里取1
    theta = np.array([[0],[0],[0]])          #theta为2x1矩阵
    wrong_number = 0                         #定义出错数量
    for i in range(len(X)):
        y_hat = np.dot(X[i],theta)           #求取第i个样本的加权和
        y = y_hat
        if y_hat >= 0:
            y_hat = 1                       #此处为sign函数
        else:
            y_hat = -1
        if y_hat != label[i]:                #如果预测值与实际值不相等，做出如下修正
            theta[0][0] += label[i]*n
            theta[1][0] += (label[i])*X[i][1]*n
            theta[2][0] += (label[i])*X[i][2]*n
            wrong_number += 1                #出错数量加1
    return theta,wrong_number
```

图 2 算法部分

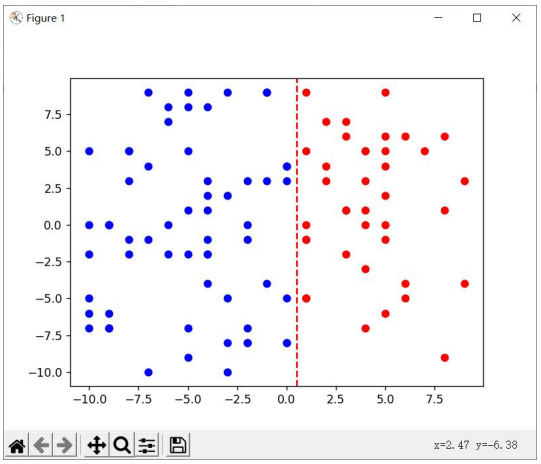
```python
def Plot(X,Y,theta):
    index1 = []
    index0 = []
    for i in range(len(Y)):
        if Y[i] == 1:
            index1.append(i)
        else:
            index0.append(i)
    x1 = []
    x0 = []
    y1 = []
    y0 = []
    for i in index1:
        x1.append(X[i][1])
        y1.append(X[i][2])
    for i in index0:
        x0.append(X[i][1])
        y0.append(X[i][2])
    plt.scatter(x1, y1, color='red')
    plt.scatter(x0, y0, color='blue')
    if theta[2][0] != 0:
        x2 = np.arange(-5, 5, 0.1)
        y2 = (float(theta[0][0]) + float(theta[1][0]) * x2)/float(-theta[2][0])
        plt.plot(x2, y2, color='yellow')
    else:
        plt.axvline(x=float(-(theta[0][0]/theta[1][0])), ls='--', c='red')

    plt.show()
```

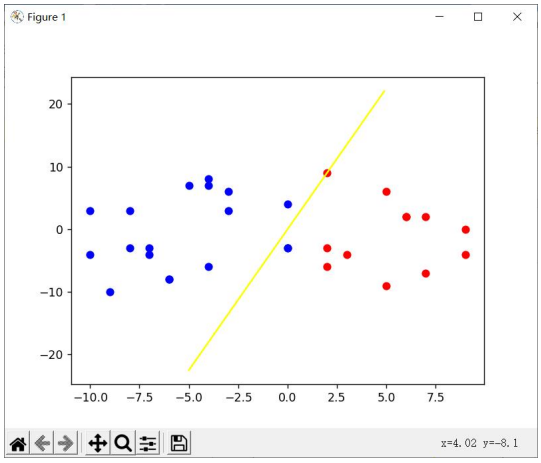
图 3 绘图部分



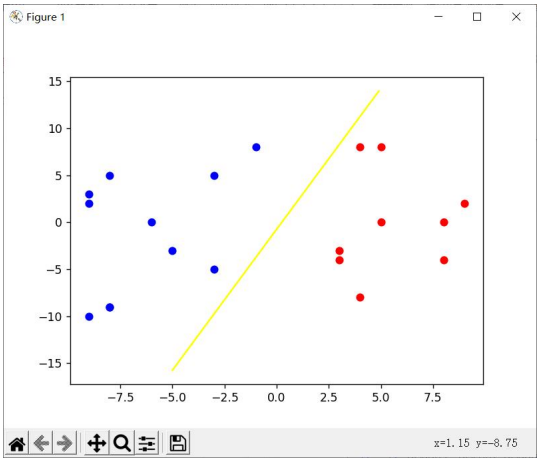
1）50 点



2）100 点



3）30 点



4）20 点

图 5

如图 4 所示结果，随着输入序列点数的逐渐增加，最终的分类器表现逐渐良好，当达到 100 点或更多时，分类器能够找到最佳的权数值。

4. 实验收获

在课堂上老师给的例子当中，没有涉及到常数项和学习速率的问题，因此在一开开始思考问题的时候，出现了一些小问题，之后上网查阅相关资料和教程，将样本和权数矩阵增加了一个维度，同时也接触到了阈值的概念。之后查阅到了学习速率相关的问题，在之前的情况中，如果说不考虑学习速率，那么常数项每次的改变量均为 1 或-1，可能会存在一次变动过大的情况，在引入学习速率之后，这个问题得以解决，此次作业并没有要求编程，但是自己想动手实践一下，通过此次实验，明白了感知器算法更多的一些细节，进行了代码复现，收获比较多。最后还是谢谢老师、学长、学姐！