**北京邮电大学软件学院**

**2018－2019学年第一学期实验报告**

**课程名称： 云计算数据中心**

**项目名称： 机器学习基础**

**项目完成人：**

**姓名：\_\_\_\_白烨淞\_\_\_\_学号：\_\_\_\_2016522050\_\_\_\_**

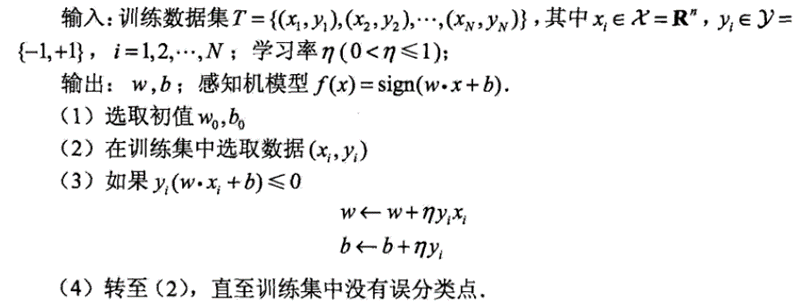
**指导教师： \_ \_管皓\_\_\_\_\_\_\_\_\_\_\_**

**日 期： 2018 年 12 月 28 日**

1. **实验目的**

学生通过编程，初步掌握机器学习中的感知机算法原理，为进一步研究更高级的智能算法（神经网络与深度学习）打下基础。

1. **实验内容**

感知机算法是近代神经网络的重要基础，它是一种二类分类的线性分类模型。其基础的算法流程如下：

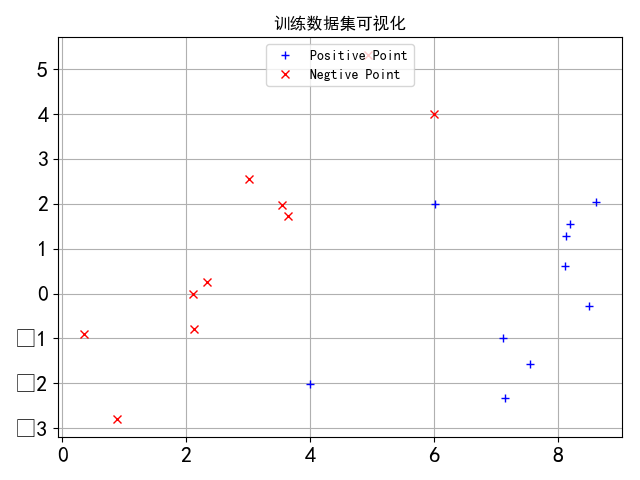
题目：

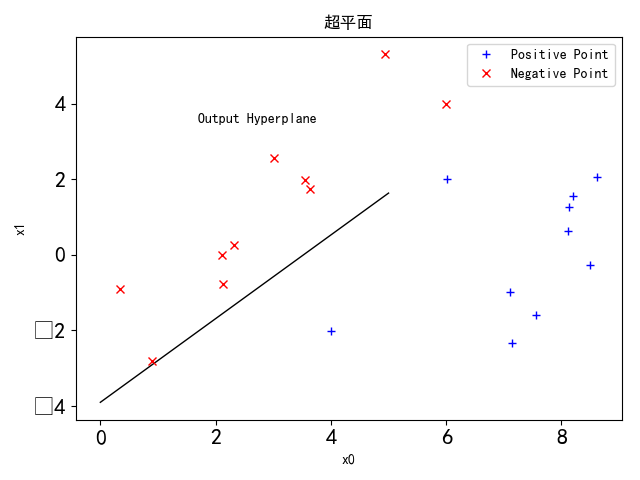
现有二维空间的样本点 (x,y),样本分为两类(类别1，类别-1)。现提供了20个数据样本及其标签供使用（train\_data）。有5个未知样本待分类(test\_data)。

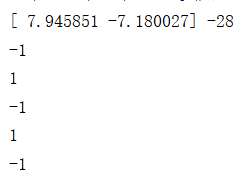
1. 请可视化训练样本点。
2. 编程实现感知机算法。
3. 可视化训练完成的感知机。
4. 对5个未知样本点进行分类。
5. **实验环境**

Windows10、pycharm下的python3

1. **实验结果**





w、b的值及测试集结果

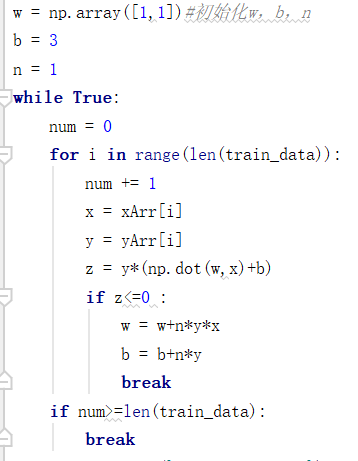
1. **附录**

**实验思路**

如果存在某个超平面S，w \* x + b=0，能将正负样例分到S两侧，则说明数据集T可分。这个超平面的关键在于二维向量w和整数值b，其实从我们的场景来说，这个模型函数中的x实际上同样是一个向量，即它包含两个维度的值。这样我们就可以将能够影响一个点位置信息的维度上的值x和y维度上的值进行评估来得到一个综合的评估值。其结果实际上是求到w1x1+w2x2+b=0平面的距离。

**设计思路说明**

本次实验我使用numpy包（数组、矩阵处理功能）和matplotlib包（生成统计图表功能），为减轻工作量我事先手动导入数据内容至列表。训练数据使用离散点图来表示，超平面使用二维直线标识。核心代码如下：



**代码：**

**import** numpy **as** np  
**import** matplotlib **as** mpl  
**import** matplotlib.pyplot **as** plt  
  
*#使用matplotlib对于训练数据集进行可视化*train\_data = [[(7.55161, -1.580025), 1], [(8.127123, 1.274372), 1], [(7.108772, -0.986906), 1], [(8.610639, 2.046708), 1], [(7.139979, -2.329896), 1], \  
 [(8.117032,0.623493),1], [(8.497162, -0.266649), 1], [(8.197181, 1.545132), 1], [(4.001389, -2.00992), 1], [(6.021212, 2.004598), 1], \  
 [(4.929821, 5.307212), -1], [(3.542485, 1.977398), -1], [(3.018896, 2.55416), -1], [(2.114999, -0.004466), -1], [(2.326297, 0.265213), -1], \  
 [(3.634009, 1.730537), -1], [(0.341367, -0.894998),-1], [(2.123252, -0.783563),-1], [(0.887835, -2.797792),-1], [(6.001298, 4.00012), -1]]  
fig = plt.figure(**'Input Figure'**)  
mpl.rcParams[**'xtick.labelsize'**] = 16  
mpl.rcParams[**'ytick.labelsize'**] = 16  
plt.rcParams[**'font.sans-serif'**]=[**'SimHei'**] *#用来正常显示中文标签  
#plt.rcParams['axes.unicode\_minus']=False #用来正常显示负号*xArr = np.array([x[0] **for** x **in** train\_data])*#取得数据集中的二维坐标向量*yArr = np.array([x[1] **for** x **in** train\_data])  
xPlotx,xPlotx\_,xPloty,xPloty\_ = [],[],[],[]*#列表创建***for** i **in** range(len(train\_data)):  
 y = yArr[i]  
 **if** y>0:  
 xPlotx.append(xArr[i][0])  
 xPloty.append(xArr[i][1])  
 **else**:  
 xPlotx\_.append(xArr[i][0])  
 xPloty\_.append(xArr[i][1])  
plt.title(**'训练数据集可视化'**)  
plt.grid(**True**)  
pPlot1,pPlot2 = plt.plot(xPlotx,xPloty,**'b+'**,xPlotx\_,xPloty\_,**'rx'**)  
plt.legend(handles = [pPlot1,pPlot2],labels=[**'Positive Point'**,**'Negtive Point'**],loc=**'upper center'**)  
plt.show()  
  
*#使用matplotlib对超平面数据集进行可视化*w = np.array([1,1])*#初始化w，b，n*b = 3  
n = 1  
**while True**:  
 num = 0  
 **for** i **in** range(len(train\_data)):  
 num += 1  
 x = xArr[i]  
 y = yArr[i]  
 z = y\*(np.dot(w,x)+b)  
 **if** z<=0 :  
 w = w+n\*y\*x  
 b = b+n\*y  
 **break  
 if** num>=len(train\_data):  
 **break**fig = plt.figure(**'Output Figure'**)  
x0 =np.linspace(0,5,100)  
w0 = w[0]  
w1 = w[1]  
x1 = -(w0/w1)\*x0-b/w1  
plt.title(**"超平面"**)  
plt.xlabel(**'x0'**)  
plt.ylabel(**'x1'**)  
plt.annotate(**'Output Hyperplane'**,xy=(0.5,4.5),xytext=(1.7,3.5))  
plt.plot(x0,x1,**'k'**, lw=1)  
pPlot3, pPlot4= plt.plot(xPlotx,xPloty,**'b+'**,xPlotx\_,xPloty\_,**'rx'**)  
plt.legend(handles = [pPlot3,pPlot4],labels=[**'Positive Point'**,**'Negative Point'**],loc=**'upper right'**)  
plt.show()  
print(w,b)  
  
*#使用求得的w,b补全测试集  
#test\_data = [[(0.00789, 5.99876)], [(3.90232, -4.00112)], [(7.108772, -0.986906)], [(8.610639, 2.046708)], [(7.139979, -2.329896)]]*test\_data = [[0.00789, 5.99876],[3.90232, -4.00112],[0.00012,-2.50001],[-0.99991,-7.88812],[8.00192,5.901201]]  
**for** q **in** range(len(test\_data)):  
 **if** (np.dot(w, test\_data[q]) + b)>0:  
 print(1)  
 **else**:  
 print(-1)