**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：\_\_\_\_孙泽文\_\_\_\_学号：\_\_2016522047\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

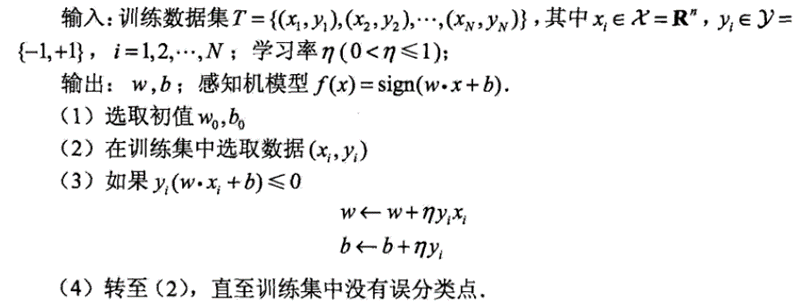
**指导教师： \_\_管皓\_\_\_\_\_\_\_\_\_\_\_**

**日 期： 2019 年 1 月 10 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**

VS Code,anaconda

1. **实验结果**

完成了实验指导书上的实验要求，理解并实现了感知机对数据的分类。

以下为代码，截图等实验结果

（附上实验文档，如：问题分析、设计方案、算法、设计图、程序、仿真结果、运行结果、调试心得等，具体内容根据实验要求来

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv('./Desktop/train\_data.csv', header=None)

# 样本输入，维度（20，2）

X = data.iloc[:,:2].values

# 样本输出，维度（20，）

y = data.iloc[:,2].values

plt.scatter(X[:10, 0], X[:10, 1], color='blue', marker='o', label='Positive')

plt.scatter(X[10:, 0], X[10:, 1], color='red', marker='x', label='Negative')

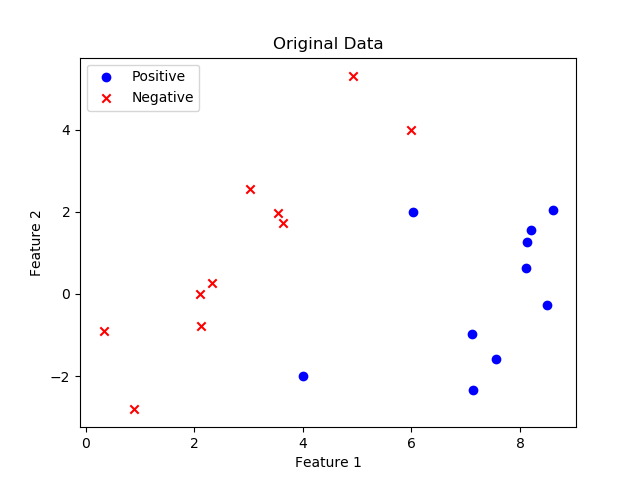
plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend(loc = 'upper left')

plt.title('Original Data')

plt.show()



**以上为对训练样本点的可视化**

# 均值

u = np.mean(X, axis=0)

# 方差

v = np.std(X, axis=0)

X = (X - u) / v

# 作图

plt.scatter(X[:10, 0], X[:10, 1], color='blue', marker='o', label='Positive')

plt.scatter(X[10:, 0], X[10:, 1], color='red', marker='x', label='Negative')

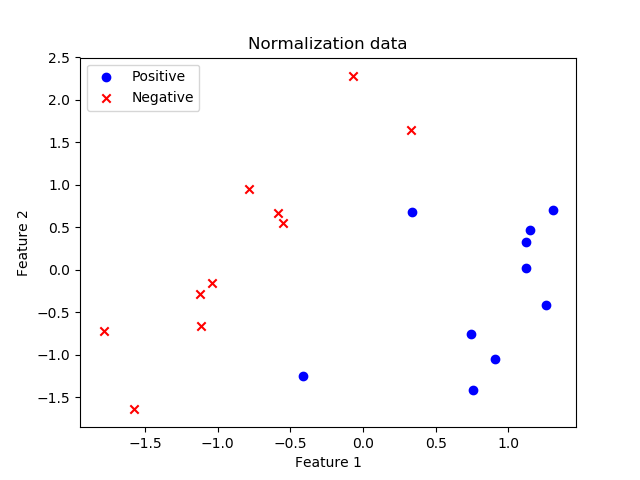
plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend(loc = 'upper left')

plt.title('Normalization data')

plt.show()



**对数据进行归一化，样本点可视化如上图**

#X加上偏置项

X = np.hstack((np.ones((X.shape[0],1)), X))

# 权重初始化

w = np.random.randn(3,1)

# 直线第一个坐标（x1，y1）

x1 = -2

y1 = -1 / w[2] \* (w[0] \* 1 + w[1] \* x1)

# 直线第二个坐标（x2，y2）

x2 = 2

y2 = -1 / w[2] \* (w[0] \* 1 + w[1] \* x2)

# 作图

plt.scatter(X[:10, 1], X[:10, 2], color='blue', marker='o', label='Positive')

plt.scatter(X[10:, 1], X[10:, 2], color='red', marker='x', label='Negative')

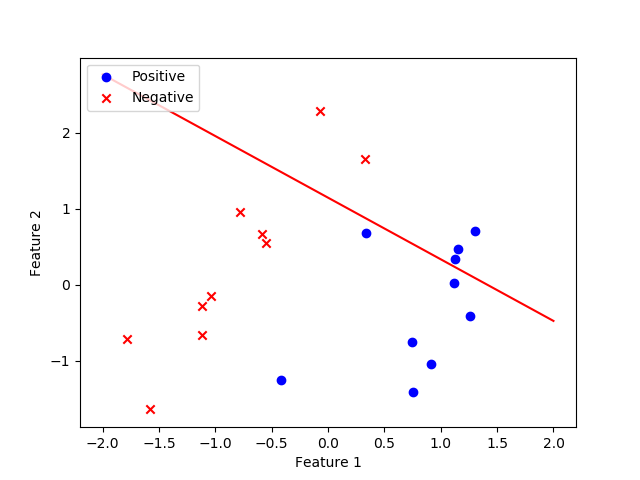
plt.plot([x1,x2], [y1,y2],'r')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend(loc = 'upper left')

plt.show()



**初始化一个直线用于分类，可视化如上图**

s = np.dot(X, w)

y\_pred = np.ones\_like(y) # 预测输出初始化

loc\_n = np.where(s < 0)[0] # 大于零索引下标

y\_pred[loc\_n] = -1

# 第一个分类错误的点

t = np.where(y != y\_pred)[0][0]

# 更新权重w

w += y[t] \* X[t, :].reshape((3,1))

for i in range(20):

s = np.dot(X, w)

y\_pred = np.ones\_like(y)

loc\_n = np.where(s < 0)[0]

y\_pred[loc\_n] = -1

num\_fault = len(np.where(y != y\_pred)[0])

print('第%2d次更新，分类错误的点个数：%2d' % (i, num\_fault))

if num\_fault == 0:

break

else:

t = np.where(y != y\_pred)[0][0]

w += y[t] \* X[t, :].reshape((3,1))

# 直线第一个坐标（x1，y1）

x1 = -2

y1 = -1 / w[2] \* (w[0] \* 1 + w[1] \* x1)

# 直线第二个坐标（x2，y2）

x2 = 2

y2 = -1 / w[2] \* (w[0] \* 1 + w[1] \* x2)

# 作图

plt.scatter(X[:10, 1], X[:10, 2], color='blue', marker='o', label='Positive')

plt.scatter(X[10:, 1], X[10:, 2], color='red', marker='x', label='Negative')

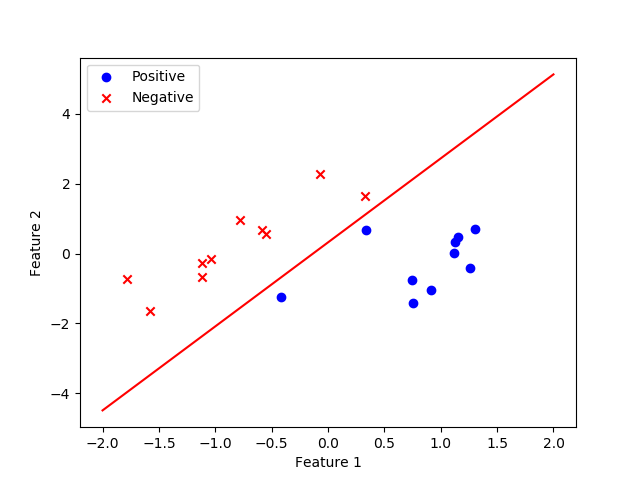
plt.plot([x1,x2], [y1,y2],'r')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend(loc = 'upper left')

plt.show()

****

**训练感知机，并将其可视化如上图**

#更正记录

s = np.dot(X, w)

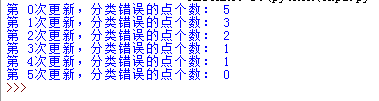
y\_pred = np.ones\_like(y)

loc\_n = np.where(s < 0)[0]

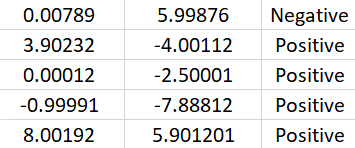
y\_pred[loc\_n] = -1

accuracy = len(np.where(y == y\_pred)[0]) / len(y)

print('accuracy: %.2f' % accuracy



**更正记录如上图**



**判断测试机的特征如上**