# 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：** 朱岩 **学号：** 2016522039

**指导教师：** 管皓

**日　　期：** 2018 年 12 月 28 日

# 目录

# 一、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。

# 二、实验内容

**概述：**

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：

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathcal{X}=\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$，$i=1,2,\cdots,N$；学习率 $\eta\ (0<\eta\leqslant 1)$；

输出：$w,b$；感知机模型 $f(x)=\mathrm{sign}(w\cdot x+b)$.

（1）选取初值 $w_0,b_0$

（2）在训练集中选取数据 $(x_i,y_i)$

（3）如果 $y_i(w\cdot x_i+b)\leqslant 0$

$$w\leftarrow w+\eta y_i x_i$$

$$b\leftarrow b+\eta y_i$$

（4）转至（2），直至训练集中没有误分类点.

**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。**

# 三、实验环境

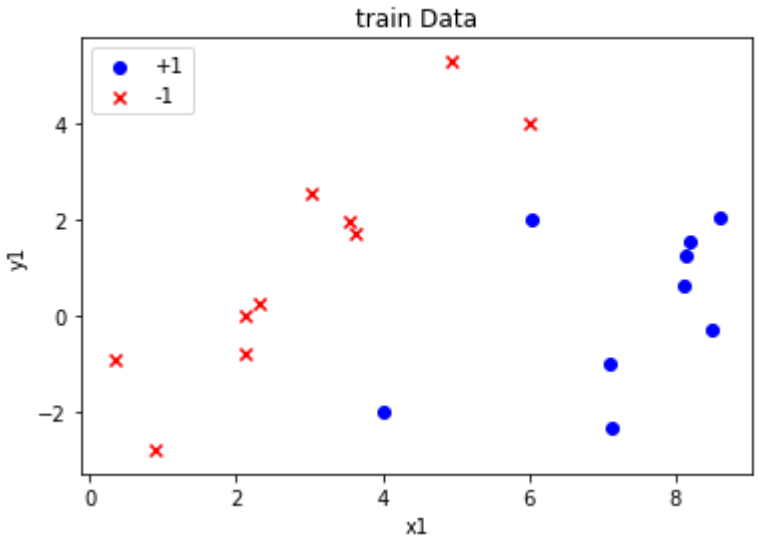
Windows 操作系统

编程语言：python

编程工具：Spyder 3.7

# 四、实验结果

## 4.1 请可视化训练样本点。




图表 1-可视化训练样本点

## 4.2 编程实现感知机算法。

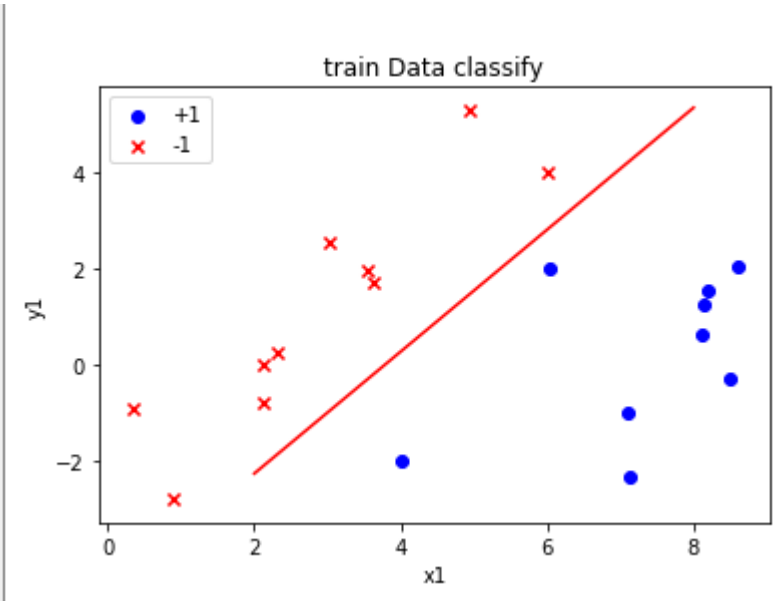
```
#初始化参数
weight = [0, 0]  # 将权重初始化为0
bias = 0  # 偏置量初始化为0
learning_rate = 0.5  # 学习速率

#进行训练
for i in range(100):
    worse=0
    for train in train_datas:
        x1, x2, y =train
        predict = sign(weight[0] * x1 + weight[1] * x2 + bias)  # 预测结果
        print("train data: x: (%d, %d) y: %d  ==> predict: %d" % (x1, x2, y, predict))
        #如果有错误点，则更新参数
        if y !=predict:
            worse+=1
            weight[0] = weight[0] + learning_rate * y * x1  # 更新权重
            weight[1] = weight[1] + learning_rate * y * x2
            bias = bias + learning_rate * y  # 更新偏置量
            print("update weight and bias: "),
            print(weight[0], weight[1], bias)
    if worse==0:break#没有错误点则退出训练

print("stop training: ")
print(weight[0], weight[1], bias)#将训练出参数打印在屏幕上
```

图表 2-感知机算法重要代码

```
train data: x: (6, 4) y: -1  ==> predict: -1
train data: x: (8, 1) y: 1  ==> predict: 1
train data: x: (7, 0) y: 1  ==> predict: 1
train data: x: (8, 2) y: 1  ==> predict: 1
train data: x: (7, -2) y: 1  ==> predict: 1
train data: x: (8, 0) y: 1  ==> predict: 1
train data: x: (8, 0) y: 1  ==> predict: 1
train data: x: (8, 1) y: 1  ==> predict: 1
train data: x: (4, -2) y: 1  ==> predict: 1
train data: x: (6, 2) y: 1  ==> predict: 1
train data: x: (4, 5) y: -1  ==> predict: -1
train data: x: (3, 1) y: -1  ==> predict: -1
train data: x: (3, 2) y: -1  ==> predict: -1
train data: x: (2, 0) y: -1  ==> predict: -1
train data: x: (2, 0) y: -1  ==> predict: -1
train data: x: (3, 1) y: -1  ==> predict: -1
train data: x: (0, 0) y: -1  ==> predict: -1
train data: x: (2, 0) y: -1  ==> predict: -1
train data: x: (0, -2) y: -1  ==> predict: -1
train data: x: (6, 4) y: -1  ==> predict: -1
stop training:
3.4389890000000025 -2.7107820000000045 -13.0
```

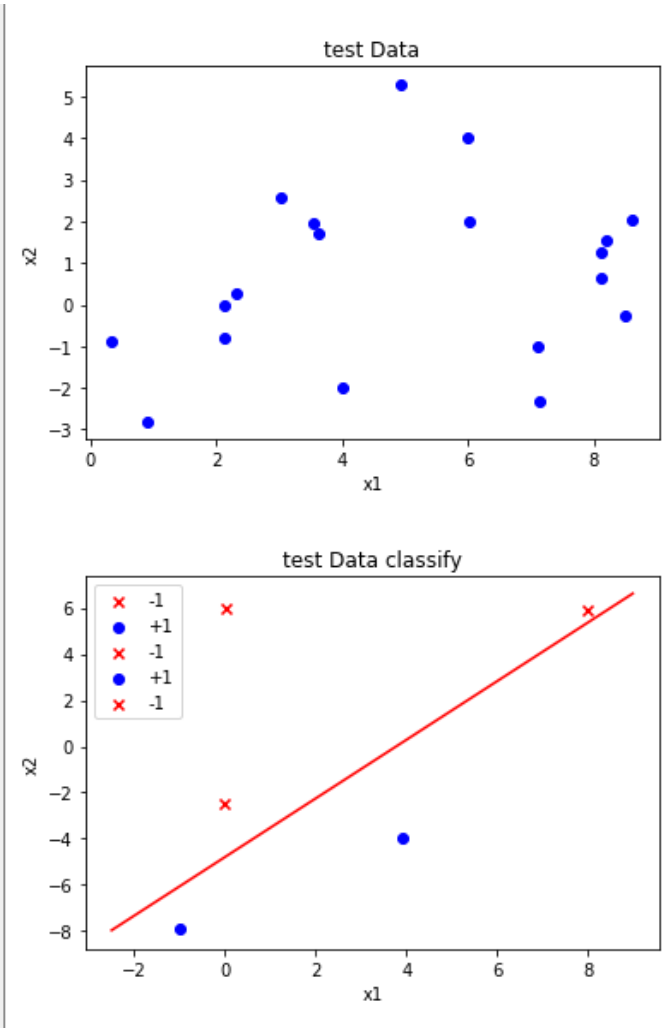
图表 3-训练结果

## 4.3 可视化训练完成的感知机。




图表 4-可视化训练完成的感知机

## 4.4 对 5 个未知样本点进行分类。




图表 5-测试数据可视化分类

# 五、遇到问题及解决方案

## 5.1 遇到问题 1：

使用 pandas 读取 txt 文件，后续对文件按列进行拆分时出现错误。

解决方案：经输出中间结果发现 pandas 将 txt 文件中的制表符也当作数据存储下来，并不是根据制表符分隔，而是根据逗号分隔的，故将原数据转化为 csv 文件。

问题成功解决

```
      7.55161\t-1.580025\t1
0     8.127123\t1.274372\t1
1    7.108772\t-0.986906\t1
2     8.610639\t2.046708\t1
3    7.139979\t-2.329896\t1
4     8.117032\t0.623493\t1
5    8.497162\t-0.266649\t1
6     8.197181\t1.545132\t1
7     4.001389\t-2.00992\t1
8     6.021212\t2.004598\t1
9    4.929821\t5.307212\t-1
10   3.542485\t1.977398\t-1
11    3.018896\t2.55416\t-1
12  2.114999\t-0.004466\t-1
13   2.326297\t0.265213\t-1
14   3.634009\t1.730537\t-1
15  0.341367\t-0.894998\t-1
16  2.123252\t-0.783563\t-1
17  0.887835\t-2.797792\t-1
18    6.001298\t4.00012\t-1
```

图表 6-txt 文件读入

```
**********初始训练数据文件**********
     7.55161  -1.580025  1
0   8.127123   1.274372  1
1   7.108772  -0.986906  1
2   8.610639   2.046708  1
3   7.139979  -2.329896  1
4   8.117032   0.623493  1
5   8.497162  -0.266649  1
6   8.197181   1.545132  1
7   4.001389  -2.009920  1
8   6.021212   2.004598  1
9   4.929821   5.307212 -1
10  3.542485   1.977398 -1
11  3.018896   2.554160 -1
12  2.114999  -0.004466 -1
13  2.326297   0.265213 -1
14  3.634009   1.730537 -1
15  0.341367  -0.894998 -1
16  2.123252  -0.783563 -1
17  0.887835  -2.797792 -1
18  6.001298   4.000120 -1
```

图表 7-csv 文件读入

## 5.2 遇到问题 2：

Pandas 读数据时原本是五组数据但只显示了 4 行

```
[ 3.902320e+00 -4.001120e+00]
[ 1.200000e-04 -2.500010e+00]
[-9.999100e-01 -7.888120e+00]
[ 8.001920e+00  5.901201e+00]
```

解决方案：Pandas 在读取文件时是认为第一行为表头，正式数据从第二行开始，故将自己将数据加上表头即可

# 六、实验总结：

通过本次实验，使我更加深入了解了感知机算法的原理及其实现，刚开始看到题目时真的无从下手，通过查找相关资料，和自己不断地尝试。最终理解并实现了感知机二分类算法。我感觉这个就像中学时期的数学题给定一个已知直线 y=ax+b，问你某点是在直线上，还是在直线下。而感知机算法就是通过不断的猜测 a 和 b 这些参数将

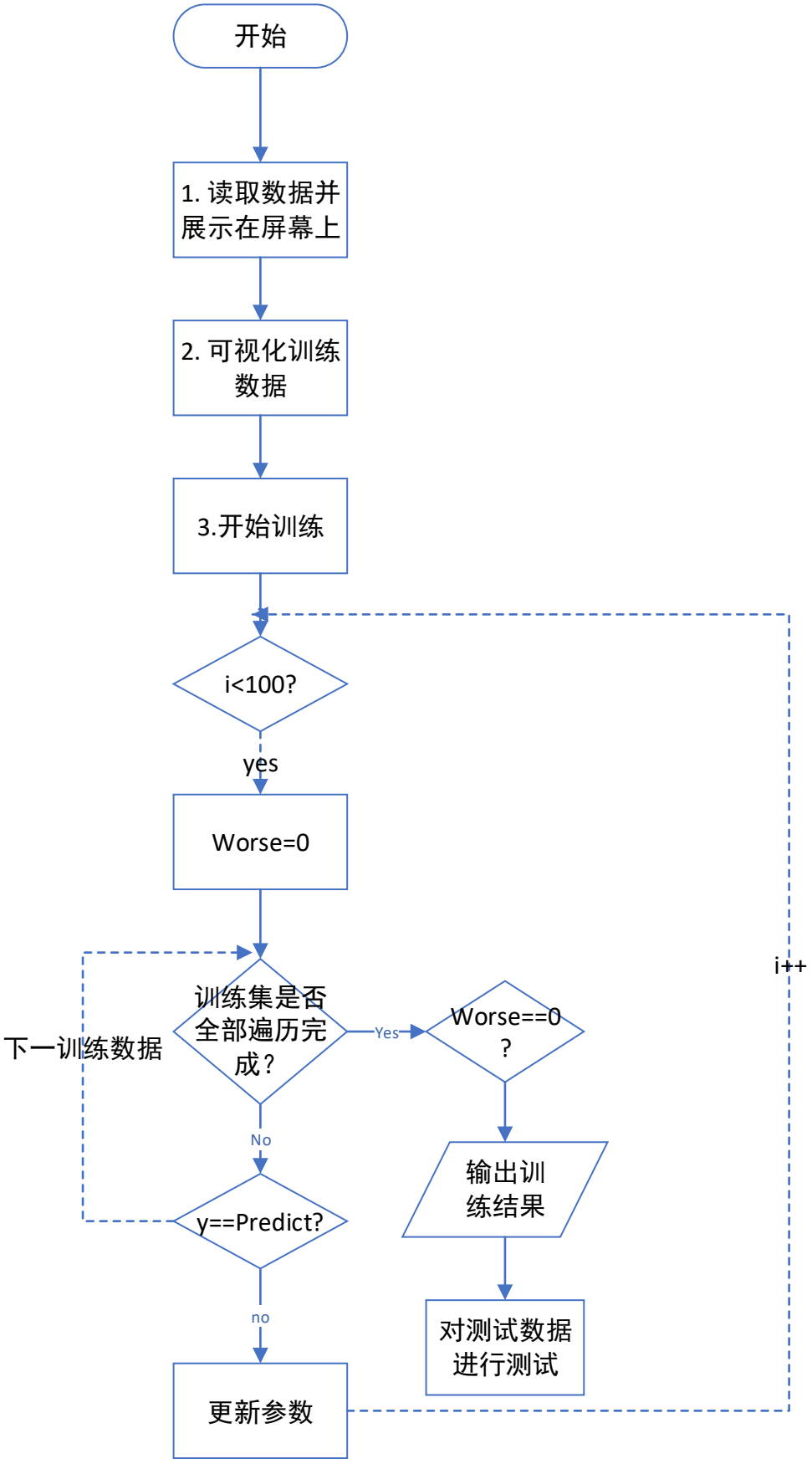
未知的分界线找出，然后根据点与线的位置关系判断该点属于哪类。如果数据集是二维的则分界线可能为直线或各种曲线，推广到三维，分界线可能为各种曲面，再向高维扩展就超出我的认知水平了，此时应该可以在预处理时应用降维技术来减少后续计算量。

# 七、附录

## 7.1 选用 Python 原因：

由于本实验主要是关于机器学习方面问题的，且目前主流的用于机器学习方面开发最主流的语言就是 python，python 对这类应用的支持度也相比其他语言支持度更高，相关的库更全，使得其更适用于本实验。

## 7.2 流程图


开始
1. 读取数据并展示在屏幕上
2. 可视化训练数据
3.开始训练
i<100?
yes
Worse=0
训练集是否全部遍历完成？
Yes
Worse==0？
No
输出训练结果
y==Predict?
no
对测试数据进行测试
更新参数
下一训练数据
i++

图表 8-流程图

## 7.3 思路及实现:

### 7.3.1. 将 excel 文件转化为 csv 文件

### 7.3.2. 读取 txt 文件

首先引用 pandas 库，方便后续对数据进行切片处理

import pandas as pd

data = pd.read_csv('./train_data. CSV')

X = data.iloc[:,:2].values，将前两列数据存入 X 中

y = data.iloc[:,2].values　将第三列标签存入 y 中

```
**********训练数据**********
[[ 8.127123e+00  1.274372e+00]
 [ 7.108772e+00 -9.869060e-01]
 [ 8.610639e+00  2.046708e+00]
 [ 7.139979e+00 -2.329896e+00]
 [ 8.117032e+00  6.234930e-01]
 [ 8.497162e+00 -2.666490e-01]
 [ 8.197181e+00  1.545132e+00]
 [ 4.001389e+00 -2.009920e+00]
 [ 6.021212e+00  2.004598e+00]
 [ 4.929821e+00  5.307212e+00]
 [ 3.542485e+00  1.977398e+00]
 [ 3.018896e+00  2.554160e+00]
 [ 2.114999e+00 -4.466000e-03]
 [ 2.326297e+00  2.652130e-01]
 [ 3.634009e+00  1.730537e+00]
 [ 3.413670e-01 -8.949980e-01]
 [ 2.123252e+00 -7.835630e-01]
 [ 8.878350e-01 -2.797792e+00]
 [ 6.001298e+00  4.000120e+00]]
**********标签**********
[ 1  1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
```

图表 9-训练数据

### 7.3.3. 将原数据可视化

**plt.scatter(X[:9, 0], X[:9, 1], color='blue', marker='o', label='+1')**

**将标签为+1 的数据用蓝色圈圈表示出来**

**plt.scatter(X[9:, 0], X[9:, 1], color='red', marker='x', label='-1')**

**将标签为-1 的数据用蓝色圈圈表示出来**

**plt.xlabel('x1')//横轴**

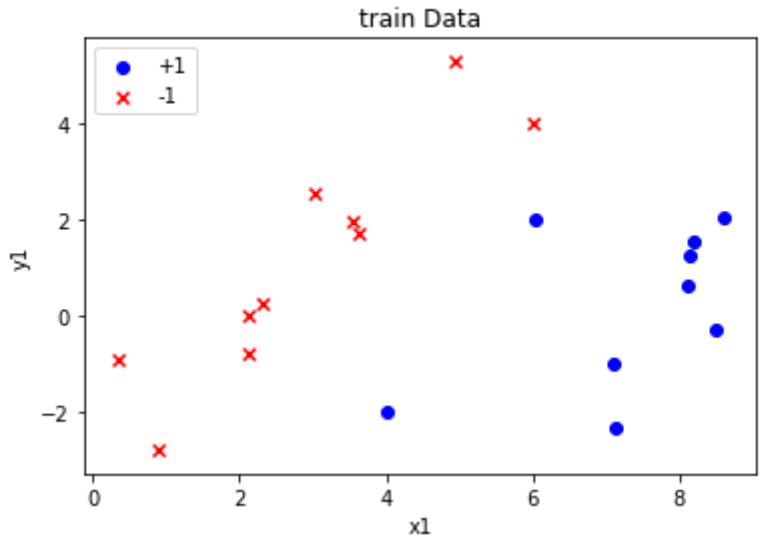
**plt.ylabel('y1')//纵轴**

**plt.title('Original Data')#图表的标题**

**plt.show()#将上述图表显示出来**

```
#可视化训练数据
plt.scatter(X[:9, 0], X[:9, 1], color='blue', marker='o', label='+1')
plt.scatter(X[9:, 0], X[9:, 1], color='red', marker='x', label='-1')
plt.xlabel('x1')
plt.ylabel('y1')
plt.legend(loc = 'upper left')
plt.title('train Data')
plt.show()
```

图表 10-数据可视化部分代码




图表 11-可视化原数据

### 7.3.4. 对感知机的认识

公式：

$$f(x)=\text{sign}(w\cdot x+b)$$

其中 x 是实例的特征向量

W 是权重，维度与 x 维度相等

B 是偏置

F(x)是预测值

Y 为真实值

### 7.3.5. 编写感知机训练部分代码

7.3.5.1. 设置初始参数值

使用数组 weight 来存放权重

Weight[0]是 x1 的权重

Weight[1]是 x2 的权重

```
#初始化参数
weight = [0, 0]  # 将权重初始化为0
bias = 0  # 偏置量初始化为0
learning_rate = 0.5  # 学习速率
```

图表 12-设置初始参数值部分代码

7.3.5.2. 进行训练

1. 设置最大迭代次数为 100
2. 开始训练
3. 设置一个局部变量 worse 初始化为 0,作用：标记是否有判断错误的点
4. If 所有数据训练完成 goto 6
   else 对每组数据进行预测 sign(weight[0] * x1 + weight[1] * x2 + bias)//sign 函数将括号中内容变为-1/1，方便比较
5. if 预测值等于标签，则进行下一组数据，goto 4
   else 对参数进行更新 $w \leftarrow w+\eta y_i x_i$ $b \leftarrow b+\eta y_i$ goto 2

```
if y !=predict:
    worse+=1
    weight[0] = weight[0] + learning_rate * y * x1  # 更新权重
    weight[1] = weight[1] + learning_rate * y * x2
    bias = bias + learning_rate * y  # 更新偏置量
```

图表 13-参数更新代码

6. if worse>0 说明训练未完成，goto 2
   else 说明我们已经拿到了我们想要的模型 break

```
#进行训练
for i in range(100):
    worse=0
    for train in train_datas:
        x1, x2, y =train
        predict = sign(weight[0] * x1 + weight[1] * x2 + bias)  # 预测结果
        print("train data: x: (%d, %d) y: %d  ==> predict: %d" % (x1, x2, y, predict))
        #如果有错误点，则更新参数
        if y !=predict:
            worse+=1
            weight[0] = weight[0] + learning_rate * y * x1  # 更新权重
            weight[1] = weight[1] + learning_rate * y * x2
            bias = bias + learning_rate * y  # 更新偏置量
            print("update weight and bias: "),
            print(weight[0], weight[1], bias)
    if worse==0:break#没有错误点则退出训练

print("stop training: ")
print(weight[0], weight[1], bias)#将训练出参数打印在屏幕上
```

图表 14-训练部分代码

### 7.3.6. 将训练结果可视化

1. 将训练数据显示出来，与 7.3.3 相同
2. 将训练出的模型可视化（即将那条线显示出来）
   0=weight[0]*x1+weight[1]*x2+bias
   两点确定一条直线
   给定 x1
   则 x2= - ( weight[0]*x1+ bias)/ weight[1]

```
#根据上述训练出的参数，求出两个点，方便后续画线
x11 = 2
x21 = -(weight[0] * x11 + bias)/weight[1]

x12 = 8
x22 =-(weight[0] * x12 + bias)/weight[1]
```

图表 15-两点确定一条直线

3. 可视化

```
plt.scatter(X[:9, 0], X[:9, 1], color='blue', marker='o', label='+1')
plt.scatter(X[9:, 0], X[9:, 1], color='red', marker='x', label='-1')
plt.plot([x11,x12], [x21,x22],'r')#第一个[]中是两个点的横坐标，第二个[]是纵坐标
plt.xlabel('x1')
plt.ylabel('y1')
plt.legend(loc = 'upper left')
plt.title('train Data classify')
plt.show()
```

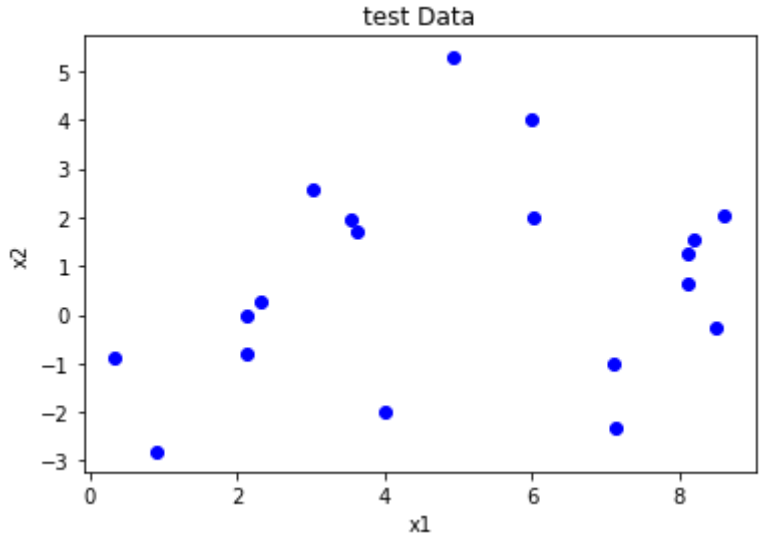
图表 16-训练结果可视化

### 7.3.7. 对测试数据进行测试

1. 读取测试数据（与 7.3.2 中读取训练数据基本相同）不同点就是测试数据只需读取两列即可，并将其展示在屏幕上

```
# 测试样本输入
test_data = pd.read_csv('test_data .CSV')
print("**********测试数据文件**********")
print(test_data.iloc[:,:2].values)
plt.scatter(X[:, 0], X[:, 1], color='blue', marker='o')
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('test Data')
plt.show()
```

图表 17-读取测试数据并可视化代码




图表 18-测试数据可视化

2. 对测试数据进行分类

   使用公式 sign(weight[0] * x1 + weight[1] * x2 + bias)

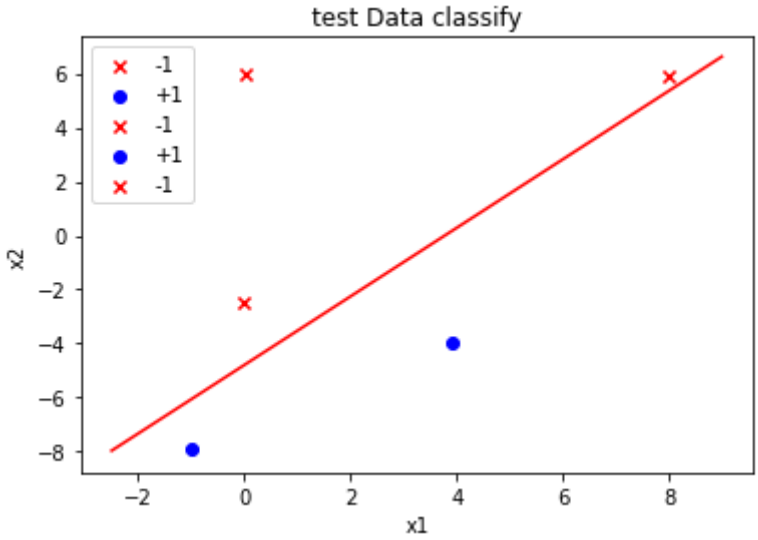
   大于 0 即为 1

   小于 0 即为-1

3. 测结果可视化（与训练结果可视化基本相同）

```
#对测试数据进行分类
for testData in test_data.iloc[:,:2].values:
    x1,x2=testData
    predict = sign(weight[0] * x1 + weight[1] * x2 + bias)
    if predict>0:
        plt.scatter(x1,x2, color='blue', marker='o', label='+1')
    else:
        plt.scatter(x1,x2, color='red', marker='x', label='-1')

x11 = -2.5
x21 = -(weight[0] * x11 + bias)/weight[1]
# 直线第二个坐标（x2，y2）
x12 = 9
x22 =-(weight[0] * x12 + bias)/weight[1]
plt.plot([x11,x12], [x21,x22],'r')

plt.xlabel('x1')
plt.ylabel('x2')
plt.legend(loc = 'upper left')
plt.title('test Data classify')
plt.show()
```

图表 19-测试数据代码




图表 20-测试结果可视化