**《机器学习与数据挖掘》实验一**

实验题目： 使用梯度下降法训练线性回归模型

实验目的： 掌握线性回归的基本原理，以及梯度下降法和最小二乘法；

实验环境（硬件和软件） Anaconda/Jupyter notebook/Pycharm

实验内容：

（1）编码实现基于梯度下降的单变量线性回归算法，包括梯度的计算与验证；

（2）画数据散点图，以及得到的直线；

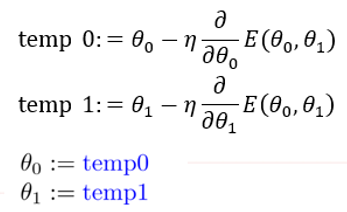
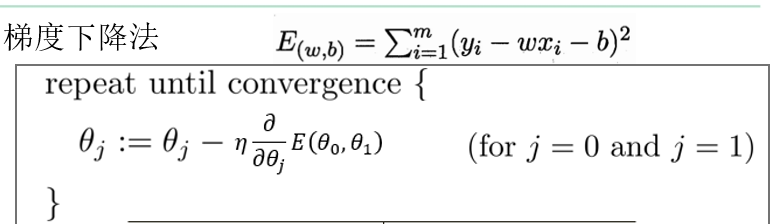
（3）画梯度下降过程中损失的变化图；

（4）基于训练得到的参数，输入新的样本数据，输出预测值；

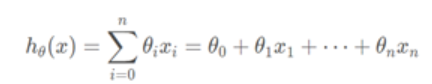
原理：

梯度下降算法原理：

每次选定一段距离，从当前位置高度下降最快的方向走。每走一段距离，就重新确定当前高度下降最快的方向走。这个思想引入到线性回归，就是找到参数矩阵θ值使得损失函数J(θ)最小。山底就是损失函数最小的地方，求解参数矩阵θ的过程，就是人走道山底的过程。



拟合函数：

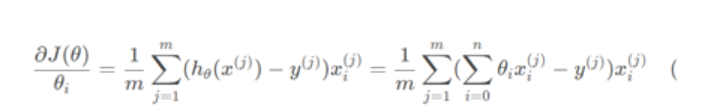


损失函数:

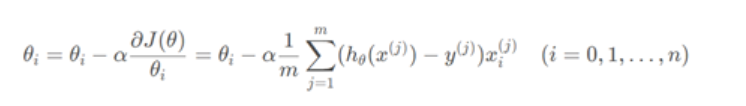
（多变量）

（单变量）

损失函数的偏导数：



θ每次更新参数的操作：



每一次θ的更新，就相当于下山走的每一段，学习率就类似下山走的步长，而损失函数的偏导数就类似于每段下山时当前位置高度下降最快的方向。

实验源码：

import numpy as np

import matplotlib.pyplot as plt

#读入训练数据

train = np.loadtxt('data1.txt', delimiter=',', dtype='double', skiprows=1)

train\_x = train[:,0]

train\_y = train[:,1]

train\_x.shape

# for i in train\_x:

#     print(i)

plt.plot(train\_x,train\_y,'o')

plt.show()

theata0 = np.random.rand()

theata1 = np.random.rand()

#预测函数

def f(x):

    return theata0 + theata1 \* x

#目标函数

def E(x, y):

    return 0.5 \*np.sum((y - f(x)) \*\* 2)

mu = train\_x.mean()

sigma = train\_x.std()

def standardize(x):

    return (x - mu) / sigma

train\_z = standardize(train\_x)

plt.plot(train\_z, train\_y,'o')

plt.show()

ETA = 1e-4 #步长

diff = 1    #两次误差值的差

count = 0   #循环次数

ll = []

xx = []

i = 1

error = E(train\_z, train\_y)

while diff > 1e-3:

    #更新结果保存到临时变量

    tmp\_theata0 = theata0 - ETA \* np.sum((f(train\_z) - train\_y))

    tmp\_theata1 = theata1 - ETA \* np.sum((f(train\_z) - train\_y) \* train\_z)

    theata0 = tmp\_theata0

    theata1 = tmp\_theata1

    current\_error = E(train\_z, train\_y)

    ll.append(current\_error)

    xx.append(i)

    i += 1

    diff = error - current\_error

    error = current\_error

    count += 1

    log = '第 {} 次 : theta0 = {:.3f}, theta1 = {:.3f}, 差值 = {:.4f}'

    print(log.format(count, theata0, theata1, diff))

print(theata0)

print(theata1)

x = np.linspace(-3, 3, 100)

plt.plot(train\_z, train\_y, 'o')

plt.plot(x, f(x))

plt.show()

plt.plot(xx, ll)

plt.show()

# 做出预测

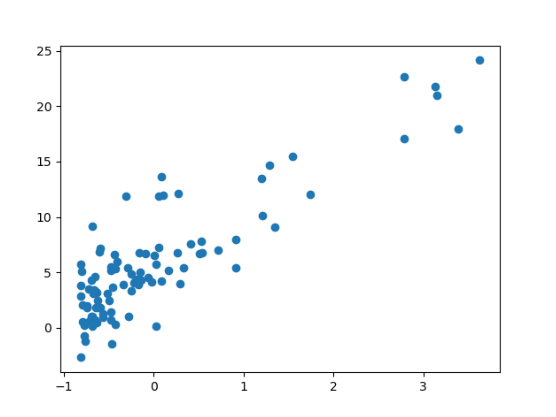
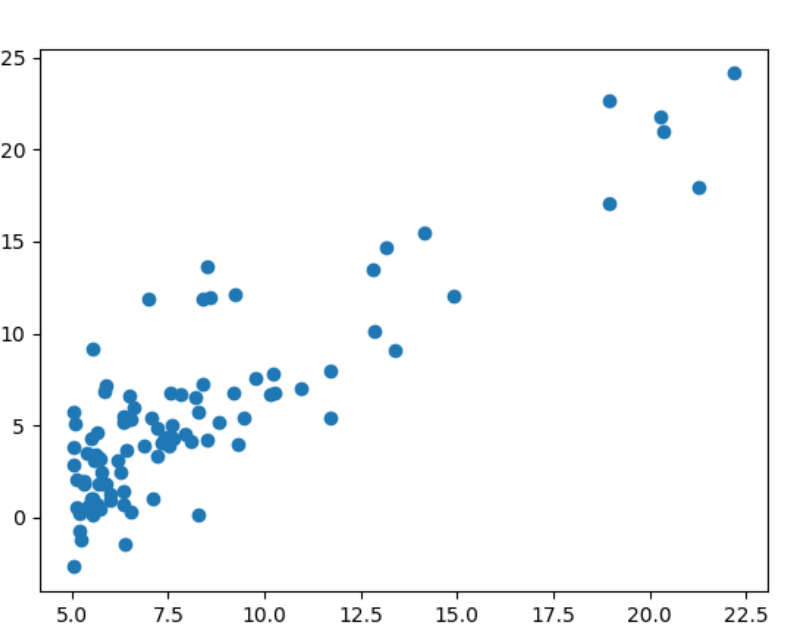
print(f(standardize(100)))

print(f(standardize(200)))

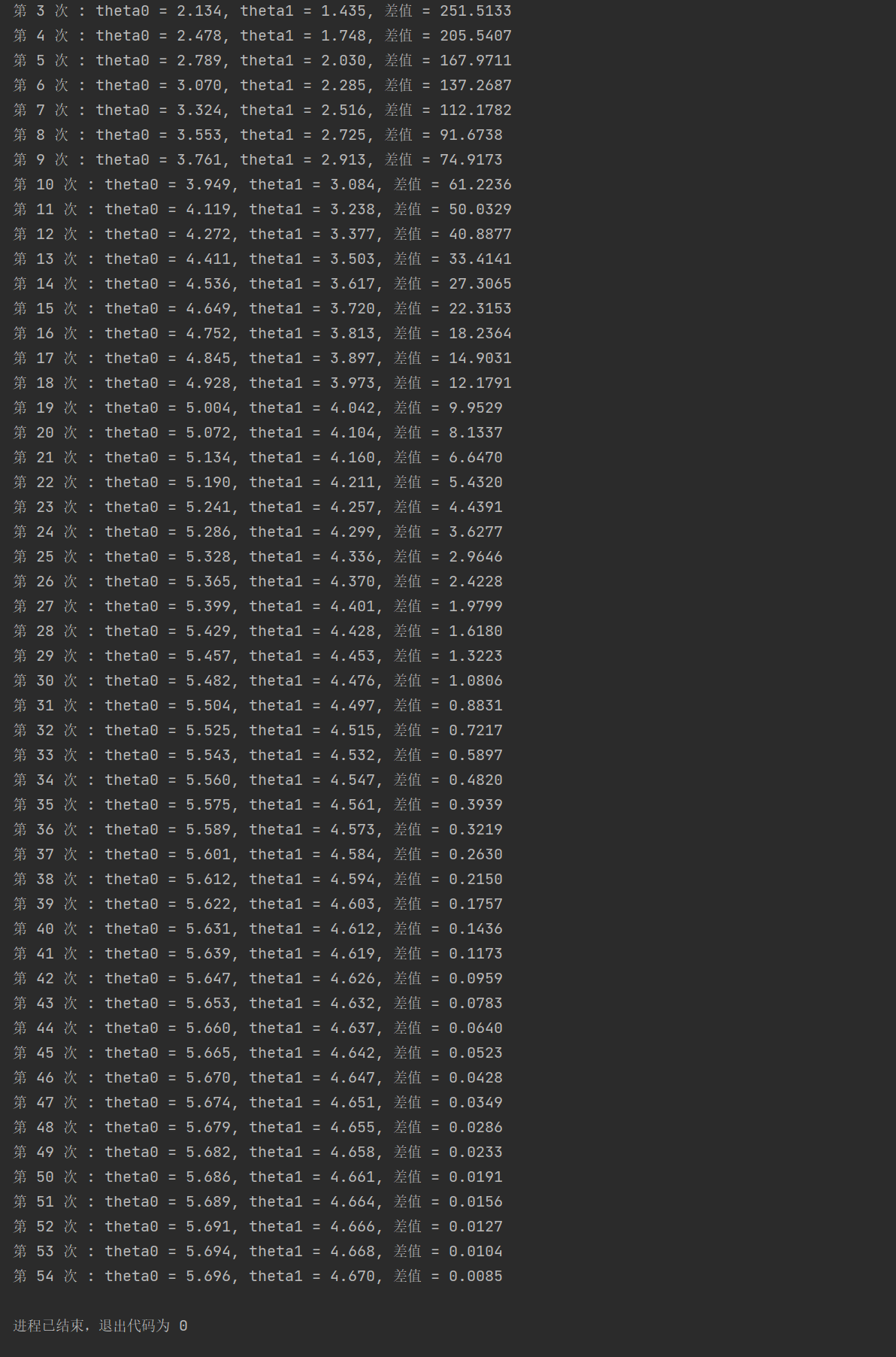
print(f(standardize(300)))

实验结果：

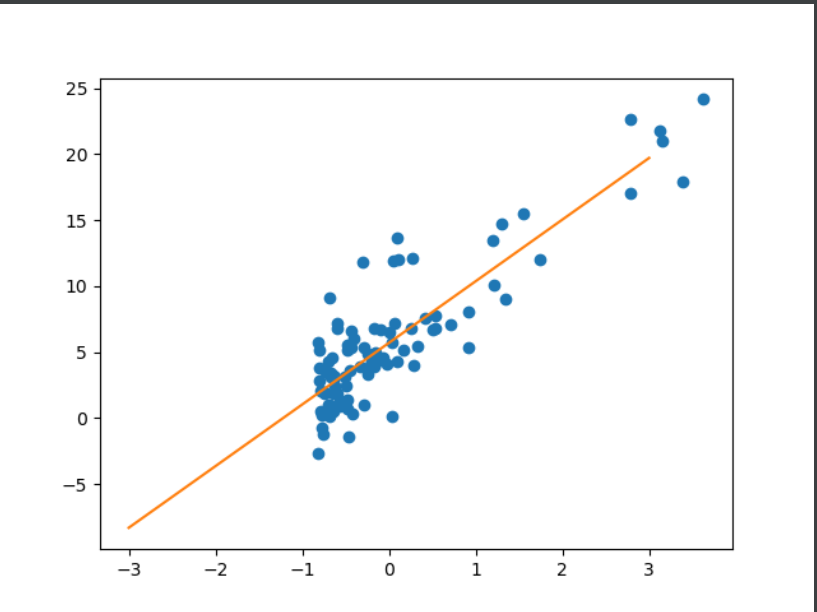
一、读入训练数据之后，进行训练并画出数据散点图

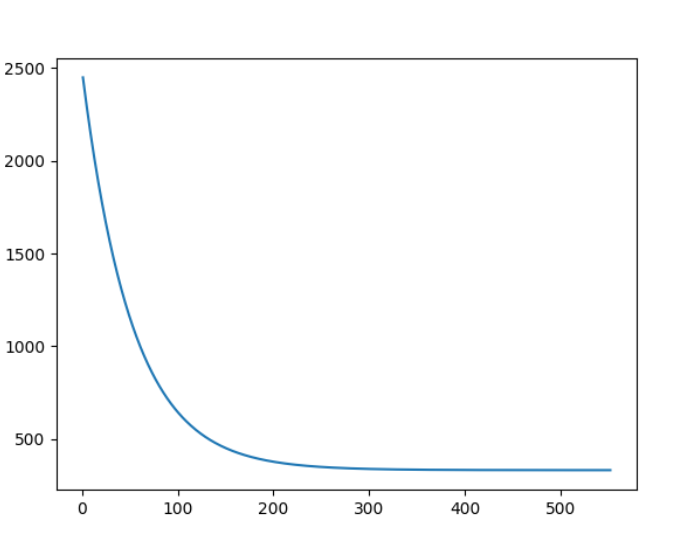
二、经过训练后计算机的预测结果与实际结果之间的误差越来越小



根据画出的散点图得到对应的直线：



梯度下降过程中损失的变化图



三、基于训练得到的参数输入新的样本数据并作出预测：

输入：standardize(100)

输入：standardize(200)

输入：standardize(300)

输入： print(theata0)

输入： print(theata1)

