Homework 2: Multivariate Linear Regression

Student ID: 16341023 Student Name:周芙蓉

Lectured by 梁上松, Sun Yat-sen University

In this homework, you will investigate multivariate linear regression using Gradient Descent and Stochastic Gradient Descent. You will also examine the relationship between the cost function, the convergence of gradient descent, overfitting problem, and the learning rate.
在本次作业中，你将探讨使用梯度下降法（随机梯度下降法）的多变量线性回归模型。你将探讨损失函数、梯度下降法的收敛、过拟合问题和学习率等之间的关系。

Download the file “dataForTraining.txt” in the attached files called “Homework 2”. This is a training dataset of apartment prices in Haizhu District, Guangzhou, Guangdong, China, where there are 50 training instances, one line per one instance, formatted in three columns separated with each other by a whitespace. The data in the first and the second columns are sizes of the apartments in square meters and the distances to the Double-Duck-Mountain Vocational Technical College in kilo-meters, respectively, while the data in the third are the corresponding prices in billion RMB. Please build a multivariate linear regression model with the training instances by script in any programming languages to predict the prices of the apartments. For evaluation purpose, please also download the file “dataForTesting.txt” (the same format as that in the file of training data) in the same folder.
请在文件夹“作业 2”中下载文件名为“dataForTraining.txt”的文件。该文件包含广东省广州市海珠区的房价信息，里面包含 50 个训练样本数据。文件有三列，第一列对应房的面积（单位：平方米），第二列对应房子距离双鸭山职业技术学院的距离（单位：千米），第三列对应房子的销售价格（单位：万元）。每一行对应一个训练样本。请使用提供的 50 个训练样本来训练多变量回归模型以便进行房价预测，请用（随机）梯度下降法的多变量线性回归模型进行建模。为了评估训练效果，请文件夹中下载测试数据集“dataForTesting.txt”（该测试文件里的数据跟训练样本具有相同的格式，即第一列对应房子面积，第二列对应距离，第三列对应房子总价）。

Exercise 1: How many parameters do you use to tune this linear regression model? Please use Gradient Descent to obtain the optimal parameters. Before you train the model, please set the number of iterations to be 1500000, the learning rate to 0.00015, the initial values of all the parameters to 0.0. During training, at every 100000 iterations, i.e., 100000 , 200000,…, 1500000, report the current training error and the testing error in a figure (you can draw it by hands or by any software). What can you find in the plots? Please analyze the plots.
Exercise 1: 你需要用多少个参数来训练该线性回归模型？请使用梯度下降方法训练。训练时，请把迭代次数设成 1500000，学习率设成 0.00015，参数都设成 0.0。在训练的过程中，每迭代 100000 步，计算训练样本对应的误差，和使用当前的参数得到的测试样本对应的误差。请画图显示迭代到达 100000 步、200000 步、… … 1500000 时对应的训练样本的误差和测试样本对应的误差（图可以手画，或者用工具画图）。从画出的图中，你发现什么？请简单分析。

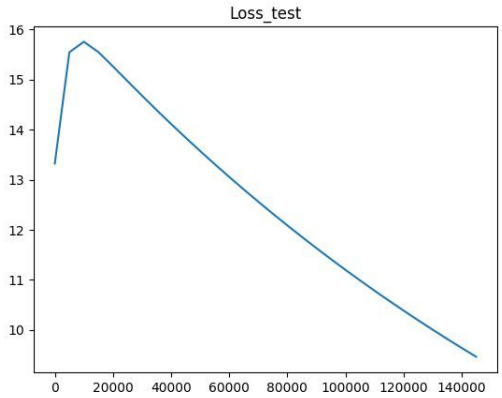
我需要三个参数来训练该线性回归模型。

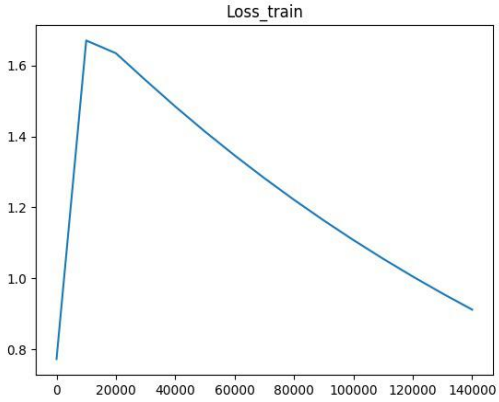
代码实现：

```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt

dataset = np.loadtxt("dataForTesting.txt")        # 将 txt 文本存为 numpy 数组
data_train = np.loadtxt("dataFortraining.txt")       # 将 txt 文本存为 numpy 数组

m, n = np.shape(dataset)
```

```
M, N = np.shape(data_train)

x, x_train = np.ones((m, n)), np.ones((M, N))

x[:, : -1], x_train[:, : -1] = dataset[:, : -1],data_train[:, : -1]  #获得（x0，x1，x2）
y, y_train = dataset[:, -1], data_train[:, -1]       #获得 Y
MAX_Inter = 150000
precision = 0.0001
step, s = 0.00015, 10000
err = [0, 0, 0]
error_train, error, num, Iter = 0,0,0,0
theta = [1, 1, 1]
loss, loss_train = 10, 10
Error, Error_train = [0]*15,  [0]*15
while (Iter < MAX_Inter and loss > precision):
    loss = 0
    loss_train = 0
    if (num > m - 1):  # 梯度下降方法
        num = 0
    prediction = theta[0] * x[num][0] + theta[1] * x[num][1] + theta[2] * x[num][2]
    err[0] = (prediction - y[num]) * x[num][0]
    err[1] = (prediction - y[num]) * x[num][1]
    err[2] = (prediction - y[num]) * x[num][2]
    num += 1
    for index in range(3):
        theta[index] = theta[index] - step * err[index]
    for index in range(m):     #计算测试误差
        prediction = theta[0] * x[index][0] + theta[1] * x[index][1] + theta[2] * x[index][2]
        error = (1 / 2/m) * (prediction - y[index]) ** 2    #loss
        loss += error
    for index in range(M):   #计算训练方差
        prediction_train = theta[0] * x_train[index][0] + theta[1] * x_train[index][1] + theta[2] *
x_train[index][2]
        error_train = (1 / 2/M) * (prediction_train - y_train[index]) ** 2    #loss
        loss_train +=error_train
    Iter = Iter + 1

    if(Iter == s):
        k = int(s/10000)
        Error[k-1] = error
        Error_train[k-1] = error_train
        print("theta", theta)
        s = s+10000  #每隔 10000 输出一次
```

```
print('Error',Error)
print('Error_train', Error_train)
X  = [i*10000for  i in range(15)]
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(X,Error)
ax.set_title('Loss_train')
plt.savefig(r"Loss_train.png")
fig = plt.figure()
bx = fig.add_subplot(1, 1, 1)
bx.plot(X,Error_train)
bx.set_title('Loss_test')
plt.savefig(r"Loss_tset.png")
```





训练集最佳参数：房价 = 7.24*面积 -63.96*距离 +2.79
测试集最佳参数：房价 = 7.07*面积 -67.84*距离 +25.14

从训练集误差图我们可以看出，在迭代次数很小的时候，误差很小，在迭代次数在 20000 附件时误差最大，后误差缓慢下降，下降趋势逐渐变慢。测试集与训练集图像上基本相同。测试集最优参数出现在迭代次数为 150000 时，训练集最优参数出现在迭代次数为 0 时。

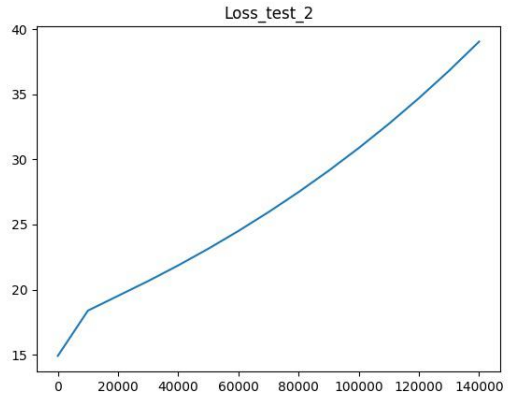
Exercise 2: Now, you change the learning rate to a number of different values, for instance, to 0.0002 (you may also change the number of iterations as well) and then train the model again. What can you find? Please conclude your findings.
Exercise 2: 现在，你改变学习率，比如把学习率改成 0.0002（此时，你可以保持相同的迭代次数也可以改变迭代次数），然后训练该回归模型。你有什么发现？请简单分析。

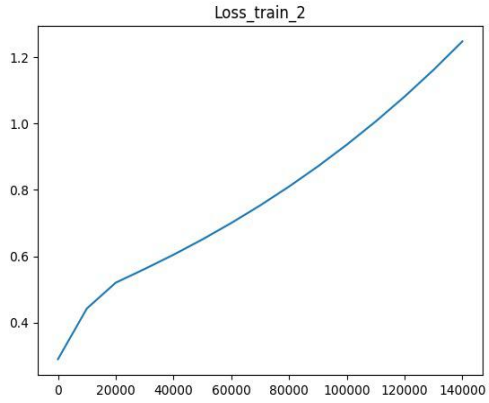
代码实现：

```
# -*- coding: utf-8 -*-
import numpy as np
import matplotlib.pyplot as plt

dataset = np.loadtxt("dataForTesting.txt")        # 将 txt 文本存为 numpy 数组
data_train = np.loadtxt("dataFortraining.txt")       # 将 txt 文本存为 numpy 数组
```

```
m, n = np.shape(dataset)
M, N = np.shape(data_train)

x, x_train = np.ones((m, n)), np.ones((M, N))

x[:, : -1], x_train[:, : -1] = dataset[:, : -1], data_train[:, : -1]  #获得（x0，x1，x2）
y, y_train = dataset[:, -1], data_train[:, -1]       #获得Y
MAX_Inter = 150000
precision = 0.0001
step, s = 0.0002, 10000
err = [0, 0, 0]
error_train, error, num, Iter = 0,0,0,0
theta = [1, 1, 1]
loss, loss_train = 10, 10
Error, Error_train = [0]*15,  [0]*15
while (Iter < MAX_Inter and loss > precision):
    loss = 0
    loss_train = 0
    if (num > m - 1):  # 梯度下降方法
        num = 0
    prediction = theta[0] * x[num][0] + theta[1] * x[num][1] + theta[2] * x[num][2]
    err[0] = (prediction - y[num]) * x[num][0]
    err[1] = (prediction - y[num]) * x[num][1]
    err[2] = (prediction - y[num]) * x[num][2]
    num += 1
    for index in range(3):
        theta[index] = theta[index] - step * err[index]
    for index in range(m):     #计算测试误差
        prediction = theta[0] * x[index][0] + theta[1] * x[index][1] + theta[2] * x[index][2]
        error = (1 / 2/m) * (prediction - y[index]) ** 2    #loss
        loss += error
    for index in range(M):   #计算训练方差
        prediction_train = theta[0] * x_train[index][0] + theta[1] * x_train[index][1] + theta[2] *
x_train[index][2]
        error_train = (1 / 2/M) * (prediction_train - y_train[index]) ** 2    #loss
        loss_train +=error_train
    Iter = Iter + 1

    if(Iter == s):
        k = int(s/10000)
        Error[k-1] = error
        Error_train[k-1] = error_train
        print("theta", theta)
```

```
        s = s+10000  # 每隔 10000 输出一次
print('Error',Error)
print('Error_train', Error_train)
X = [i*10000for  i in range(15)]
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(X, Error)
ax.set_title('Loss_train_2')
plt.savefig(r"Loss_train_2.png")
fig = plt.figure()
bx = fig.add_subplot(1, 1, 1)
bx.plot(X, Error_train)
bx.set_title('Loss_test_2')
plt.savefig(r"Loss_test_2.png")
```





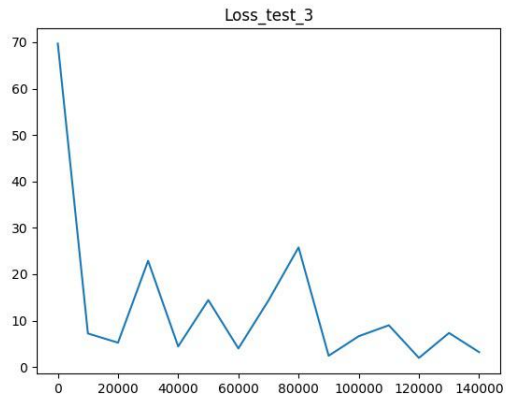
训练集最佳参数：房价 = 7.50*面积 -67.15*距离 -7.66
测试集最佳参数：房价 = 7.50*面积 -67.15*距离 -7.66
当学习率增大到 0.0002 时，无论是训练集还是测试集 loss 图像走势与题一 loss 图像完全不同。测试集和训练集在一开始就达到了最优参数，后随着迭代次数的增加，误差越来越大。通过对比误差的值，我们发现当学习率为 0.002 时，训练集的误差总体上低于学习率为 0.00015 训练集上的误差。不过学习率为 0.002 测试集的误差远大于学习率为 0.00015 测试集的误差。学习率为 0.0002 时，在训练集表现良好，在测试集表现较差，我猜测在模型拟合过程中，出现了过拟合现象。

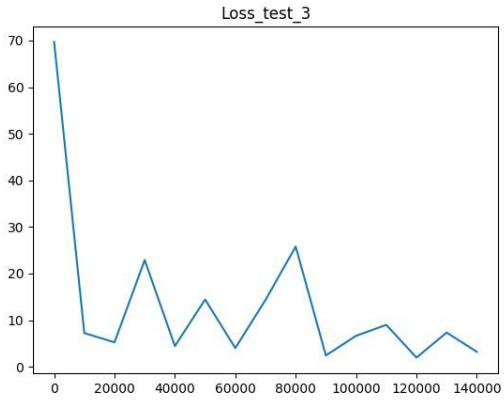
Exercise 3: Now, we turn to use other optimization methods to get the optimal parameters. Can you use Stochastic Gradient Descent to get the optimal parameters? Plots the training error and the testing error at each K-step iterations (the size of K is set by yourself). Can you analyze the plots and make comparisons to those findings in Exercise 1?
Exercise 3: 现在，我们使用其他方法来获得最优的参数。你是否可以用随机梯度下降法获得最优的参数？请使用随机梯度下降法画出迭代次数（每 K 次，这里的 K 你自己设定）与训练样本和测试样本对应的误差的图。比较 Exercise 1 中的实验图，请总结你的发现。

代码实现：

```
# -*- coding: utf-8 -*-
import numpy as np
import random
import matplotlib.pyplot as plt

dataset = np.loadtxt("dataForTesting.txt")       # 将 txt 文本存为 numpy 数组
data_train = np.loadtxt("dataFortraining.txt")       # 将 txt 文本存为 numpy 数组

m, n = np.shape(dataset)
M, N = np.shape(data_train)

x, x_train = np.ones((m, n)), np.ones((M, N))

x[:, : -1], x_train[:, : -1] = dataset[:, : -1], data_train[:, : -1]   #获得（x0，x1，x2）
y, y_train = dataset[:, -1], data_train[:, -1]        #获得 Y
MAX_Inter = 150000
precision = 0.0001
step, s = 0.00015, 10000
err = [0, 0, 0]
error_train, error, num, Iter = 0,0,0,0
theta = [1, 1, 1]
loss, loss_train = 10, 10
Error, Error_train = [0]*15,  [0]*15
while (Iter < MAX_Inter and loss > precision):
    loss = 0
    loss_train = 0
    num = random.randint(0, m-1)            #随机梯度下降方法
    prediction = theta[0] * x[num][0] + theta[1] * x[num][1] + theta[2] * x[num][2]
    err[0] = (prediction - y[num]) * x[num][0]
    err[1] = (prediction - y[num]) * x[num][1]
    err[2] = (prediction - y[num]) * x[num][2]
    num += 1
    for index in range(3):
        theta[index] = theta[index] - step * err[index]
    for index in range(m):     #计算测试误差
        prediction = theta[0] * x[index][0] + theta[1] * x[index][1] + theta[2] * x[index][2]
        error = (1 / 2/m) * (prediction - y[index]) ** 2     #loss
        loss += error
    for index in range(M):   #计算训练方差
        prediction_train = theta[0] * x_train[index][0] + theta[1] * x_train[index][1] + theta[2] *
x_train[index][2]
        error_train = (1 / 2/M) * (prediction_train - y_train[index]) ** 2     #loss
        loss_train +=error_train
```

```
    Iter = Iter + 1

    if(Iter == s):
        k = int(s/10000)
        Error[k-1] = error
        Error_train[k-1] = error_train
        print("theta", theta)
        s = s+10000  # 每隔 10000 输出一次
print('Error',Error)
print('Error_train', Error_train)
X = [i*10000for  i in range(15)]
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.plot(X, Error)
ax.set_title('Loss_train_3')
plt.savefig(r"Loss_train_3.png")
fig = plt.figure()
bx = fig.add_subplot(1, 1, 1)
bx.plot(X, Error_train)
bx.set_title('Loss_test_3')
plt.savefig(r"Loss_test_3.png")
```





训练集最佳参数：房价 = 7.245*面积 -71.08*距离 +16.37
测试集最佳参数：房价 = 7.944*面积 -71.05*距离 +12.47
这里学习率为 0.00015，通过对比梯度下降图和随机梯度下降图，发现随机梯度下降算法收敛速度大于梯度下降算法的收敛速度，最优参数出现在迭代过程中。训练集误差较大，但测试集误差与梯度下降测试集误差相差不是很多。