Pattern Recognition - Homework 1

520021910494 张一帆

一、数据集生成

由于生成的数据集要用于十倍交叉验证，故生成数据的代码应单独放在一个文件里。生成数据的程序如下。

```
import numpy as np
#生成正态分布的 X w  并计算 y
m = 600
n = 100
w = np.random.randn(n, 1)
X = np.random.randn(m, n)
y = np.dot(X, w)
#给 y 添加噪声
mean = np.mean(y)
var = np.var(y)
svar = np.sqrt(var)
noise = (np.random.normal(0, svar, (m, 1)))/100
y = y+noise
#输出为 csv 表格
np.savetxt("X.csv", X, delimiter=",")
np.savetxt("w.csv", w, delimiter=",")
np.savetxt("y.csv", y, delimiter=",")
```

如上程序所示，首先根据需要定义 m 和 n 的数值，然后生成符合 N~(0,1)的正态分布的矩阵 w、X，并根据 w 和 X 计算 y，y 加上噪声边成新的 y；最后保存为 csv 表格。

二、梯度下降法求解 w 的代码及思路

梯度下降法求解 w 的代码如下所示。

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


X_csv = pd.read_csv(r'D:\python\data\X.csv', sep=',', header=None)
X_data = np.array(X_csv)
Y_csv = pd.read_csv(r'D:\python\data\y.csv', sep=',', header=None)
Y_data = np.array(Y_csv)
w_csv = pd.read_csv(r'D:\python\data\w.csv', sep=',', header=None)
w = np.array(w_csv)
X = X_data[:200, :]   #设定 m
```

```
Y = Y_data[:200, :]

learn_rate = 0.0001
iteration = 10000    # 迭代次数

W = np.random.rand(X.shape[1], 1)
mse = []    # 这是个 Python 列表，用来保存每次迭代后的损失值

# 下面使用 for 循环来实现迭代
# 循环变量从 0 开始，到 101 结束,循环 101 次，为了描述方便，以后就说迭代 100 次
# 同样，当 i 等于 10 时，我们就说第十次迭代
for i in range(0, iteration+1):
    # 首先计算损失函数对 W 的偏导数
    dL_dW = np.dot(np.transpose(X), np.dot(X, W)-Y)
    # 然后使用迭代公式更新 W
    W = W - learn_rate*(dL_dW+0.01*W)    # γ 可改变

    # 我们希望能够观察到每次迭代的结果，判断是否收敛或者什么时候开始收敛
    # 因此需要使用每次迭代后的 W 来计算损失，并且把它显示出来

    Y_PRED = np.dot(X, W)    # 使用当前这次循环得到的 W，计算所有样本的估计值
    Loss = np.mean(np.square(Y - Y_PRED)) / 2    # 使用样本的估计值和实际值计算均方误差
    mse.append(Loss)    # 把得到的均方误差加入列表 mse

# 创建 Figure 对象
plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
plt.plot(range(0, 4000), mse[0:4000])
plt.xlabel('Iteration', color='r', fontsize=14)
plt.ylabel('Loss', color='r', fontsize=14)
plt.title("Curve of loss value change for the first 5000 iterations", fontsize=10)

plt.subplot(1, 2, 2)
Y_PRED = Y_PRED.reshape(-1)
plt.plot(Y, color="red", marker='o', label="real")
plt.plot(Y_PRED, color="blue", marker='.', label="pred")
plt.xlabel('Sample', color='r', fontsize=14)
plt.ylabel('Price', color='r', fontsize=14)
plt.title("pred & real", fontsize=14)
plt.legend(loc="upper right")
plt.suptitle("Gradient descent method for multiple linear regression", fontsize=14)
# 将创建好的图像显示出来
plt.show()
```

代码的大致思路为，首先导入生成数据的 csv 表格，根据需要选取其中的数据；接着设置有关的参数，比如迭代次数、学习率；然后随机生成一个 w；最重要的是训练部分，使用梯度下降法进行训练，并记录每次梯度下降后的 mes；最后画图使数据可视化。

三、十倍交叉验证选择 γ

十倍交叉验证的基本步骤为，选取 m=200，n=100，将数据分为十份，运行十次程序，用训练集训练出 w，然后用测试集的 X 与 w 相乘算出预测的 y，接着用真实值与之比较，计算 mse。最终比较 10 组数据的 mse 来决定 γ 的值。具体验证时的数据表格见下方。

根据表格的十倍交叉验证，我们在后续程序中选择 γ 为 0.01。

表 1 十倍交叉验证选择合适的 γ 时的 mse*10^{-3} 大小

份数\γ	0.01	0.05	0.1	0.5	1	最佳 γ
1	9.667	9.625	9.593	10.152	12.817	0.1
2	5.506	5.481	5.472	6.305	9.447	0.1
3	12.100	12.066	12.049	12.890	16.259	0.1
4	4.932	5.055	5.242	8.053	14.604	0.01
5	10.136	10.150	10.324	17.779	41.259	0.01
6	6.712	6.920	7.217	10.961	18.907	0.01
7	5.139	5.251	5.417	7.725	12.944	0.01
8	7.471	7.334	7.184	6.829	8.389	0.5
9	5.010	4.961	4.909	4.875	5.732	0.5
10	7.032	7.066	7.147	9.328	15.674	0.01
平均	7.3705	7.3909	7.4554	9.4897	15.6032	0.01

四、绘制收敛性图

由程序的最后的可视化部分得知，每次运行程序后画出两个图，第一个图为：x 轴为迭代次数，y 轴为每次计算得到的 mes；第二个图为，最终计算得到的预测值（计算得到的 w 与训练集 X 相乘）与实际的 Y 相比较的图。当 n=100，m=200 时，画图如图 1 所示。

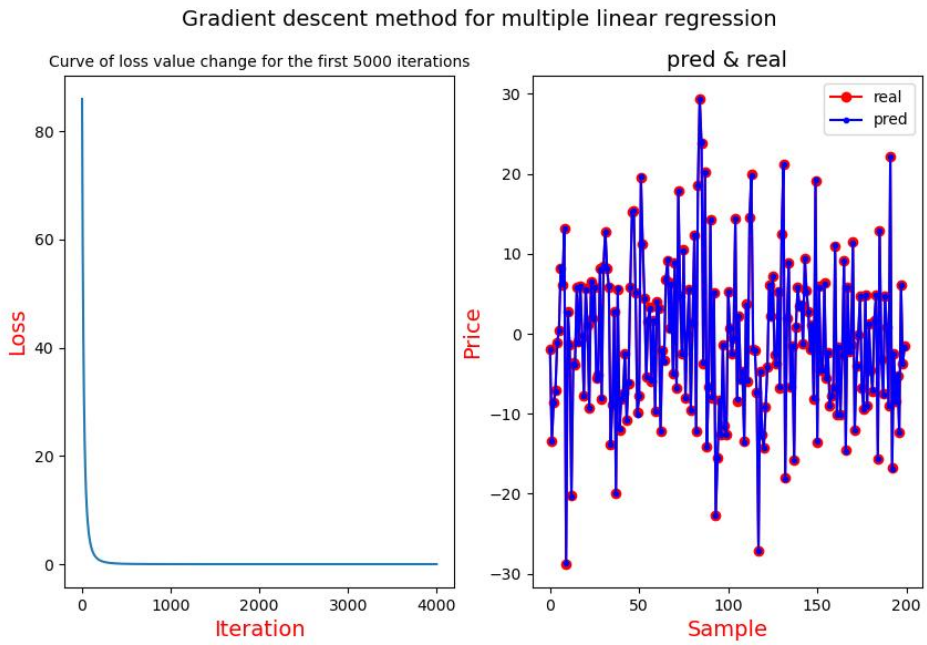

图 1　n=100,m=200 时的收敛图 和 预测与真实结果的对比图

与上图结合对比可知，当迭代次数较小时，mes 较大，即误差较大。当迭代次数较小时，mse 迅速下降。在之后的一段时间内，mse 持续下降，但是由于 mse 本身已较小，所以变化不明显。大约当迭代次数为 3000-4000 左右时，mse 不变化，收敛。

四、计算得到的 w 与真实值比较

当 n=100 时，改变 m 的大小，进行预测，得到的 w 与真实的进行比较，得出表 2。

表 2 真实 w 与预测 w 相比较

w	Real	m=120	m=200	m=300	m=500
$Mse*10^{-5}$	0	20.13	4.978	2.514	0.987

可见，随着 m 的增大，即训练的数据越来越多，w 的预测值和真实值的差距越来越小，两个 w 向量之间的均方误差也越来越小。

五、总结

本次作业中，首先随机生成正态分布的 X 和 w 的数据，计算出 y 后再加上噪声，然后用得到的数据进行分析。使用梯度下降法得到正确的 w，我们发现，不同的数据会有不同适合的 γ，最终我们选择了 0.01。通过迭代次数的增大，mse 逐渐下降最终收敛。同时我们也得知，m 越大，即数据量越大，我们预测的准确性就越大。总之，本次作业完成的较为成功。