9417 Assignment1

Zheng Qiwen z5240149

Q1

theta0 for house age: 42.54078538346594

theta1 for house age: -10.319399022339129

Q2

The red curve: house age

The blue curve: distance to the nearest MRT station

The yellow curve: number of convenience stores

图片包含 屏幕截图

描述已自动生成

Q3

RMSE for training set of house age: 12.045510305912353

Q4

RMSE for test set of house age: 16.58731450340051

Q5

RMSE for test set of distance to the nearest MRT station: 12.652088009723935

Q6

RMSE for test set of number of convenience stores: 14.731993508206784

Q7

From the graph of Q2, we can see that as iteration grows to 50, the value of cost function for distance to the nearest MRT station is smallest, and number of convenience stores is the second smallest, house age is the largest. We can also observe the same conclusion when it comes to the RMSE for test sets from Q4, Q5 and Q6. Therefore, the rank for these three features would be distance to the nearest MRT station, followed by number of convenience stores, then house age.

The whole code is shown as follow:

**import** matplotlib.pyplot **as** plt  
**import** numpy **as** numpy  
**import** pandas **as** pd   
**from** math **import** sqrt  
  
**def** square(x):  
 **return** x\*x  
*#read house\_prices.csv and read all the values*df = pd.read\_csv(**'house\_prices.csv'**)  
target\_name = **"house price of unit area"**target = df[target\_name].values  
house\_age\_all = df[**"house age"**].values  
dis\_to\_n\_MRT\_station\_all = df[**"distance to the nearest MRT station"**].values  
num\_of\_con\_stores\_all = df[**"number of convenience stores"**].values  
  
*#normalization*house\_age\_norm = [(i-min(house\_age\_all))/(max(house\_age\_all)-min(house\_age\_all))**for** i **in** house\_age\_all]  
dis\_to\_n\_MRT\_station\_norm = [(i-min(dis\_to\_n\_MRT\_station\_all))/(max(dis\_to\_n\_MRT\_station\_all)-min(dis\_to\_n\_MRT\_station\_all))**for** i **in** dis\_to\_n\_MRT\_station\_all]  
num\_of\_con\_stores\_norm = [(i-min(num\_of\_con\_stores\_all))/(max(num\_of\_con\_stores\_all)-min(num\_of\_con\_stores\_all))**for** i **in** num\_of\_con\_stores\_all]  
*#stochatic gradient descent*learning\_rate = 0.01  
theta\_0 = -1  
theta\_1 = -0.5  
iterate = 0  
  
**while** iterate < 50:  
 **for** j **in** range(300):  
 h\_fun = theta\_0 + theta\_1 \* house\_age\_norm[j]  
 theta\_0 = theta\_0 + learning\_rate \* (target[j] - h\_fun)  
 theta\_1 = theta\_1 + learning\_rate \* (target[j] - h\_fun) \* house\_age\_norm[j]  
 iterate = iterate + 1;  
 Loss = sum([square(target[i]-theta\_0-theta\_1\*house\_age\_norm[i]) **for** i **in** range(300)])/300.0  
 plt.scatter(iterate,Loss,color = **"red"**)  
*#plt.show()***print** (**"theta0 for house age: "**,theta\_0)  
**print** (**"theta1 for house age: "**,theta\_1)  
  
*#RMSE for houseage,trainging set and test set respectively*RMSE\_train\_houseage = sqrt(sum([square(target[i]-theta\_0-theta\_1\*house\_age\_norm[i]) **for** i **in** range(300)])/300.0)  
**print** (**"RMSE for training set of house age: "**,RMSE\_train\_houseage)  
RMSE\_test\_houseage = sqrt(sum([square(target[i]-theta\_0-theta\_1\*house\_age\_norm[i]) **for** i **in** range(300,400)])/100.0)  
**print** (**"RMSE for test set of house age: "**,RMSE\_test\_houseage)  
  
*#reset the thetas and iterate and do the same procedure*iterate = 0  
theta\_0 = -1  
theta\_1 = -0.5  
**while** iterate < 50:  
 **for** j **in** range(300):  
 h\_fun = theta\_0 + theta\_1 \*dis\_to\_n\_MRT\_station\_norm[j]  
 theta\_0 = theta\_0 + learning\_rate \* (target[j] - h\_fun)  
 theta\_1 = theta\_1 + learning\_rate \* (target[j] - h\_fun) \* dis\_to\_n\_MRT\_station\_norm[j]  
 iterate = iterate + 1;  
 Loss = sum([square(target[i]-theta\_0-theta\_1\*dis\_to\_n\_MRT\_station\_norm[i]) **for** i **in** range(300)])/300.0  
 plt.scatter(iterate,Loss,color = **"blue"**)  
  
**print** (**"theta0 for distance to the nearest MRT station: "**,theta\_0)  
**print** (**"theta1 for distance to the nearest MRT station: "**,theta\_1)  
  
RMSE\_train\_dis\_to\_n\_MRT\_station\_norm = sqrt(sum([square(target[i]-theta\_0-theta\_1\*dis\_to\_n\_MRT\_station\_norm[i]) **for** i **in** range(300)])/300.0)  
RMSE\_test\_dis\_to\_n\_MRT\_station\_norm = sqrt(sum([square(target[i]-theta\_0-theta\_1\*dis\_to\_n\_MRT\_station\_norm[i]) **for** i **in** range(300,400)])/100.0)  
**print** (**"RMSE for training set of distance to the nearest MRT station: "**,RMSE\_train\_dis\_to\_n\_MRT\_station\_norm)  
**print** (**"RMSE for test set of distance to the nearest MRT station: "**,RMSE\_test\_dis\_to\_n\_MRT\_station\_norm)  
  
iterate = 0  
theta\_0 = -1  
theta\_1 = -0.5  
**while** iterate < 50:  
 **for** j **in** range(300):  
 h\_fun = theta\_0 + theta\_1 \* num\_of\_con\_stores\_norm[j]  
 theta\_0 = theta\_0 + learning\_rate \* (target[j] - h\_fun)  
 theta\_1 = theta\_1 + learning\_rate \* (target[j] - h\_fun) \* num\_of\_con\_stores\_norm[j]  
 iterate = iterate + 1;  
 Loss = sum([square(target[i]-theta\_0-theta\_1\*num\_of\_con\_stores\_norm[i]) **for** i **in** range(300)])/300.0  
 plt.scatter(iterate,Loss,color = **"yellow"**)  
  
**print** (**"theta0 for number of convenience stores: "**,theta\_0)  
**print** (**"theta1 for number of convenience stores: "**,theta\_1)  
  
RMSE\_train\_num\_of\_con\_stores\_norm = sqrt(sum([square(target[i]-theta\_0-theta\_1\*num\_of\_con\_stores\_norm[i]) **for** i **in** range(300)])/300.0)  
RMSE\_test\_num\_of\_con\_stores\_norm = sqrt(sum([square(target[i]-theta\_0-theta\_1\*num\_of\_con\_stores\_norm[i]) **for** i **in** range(300,400)])/100.0)  
**print** (**"RMSE for training set of number of convenience stores: "**,RMSE\_train\_num\_of\_con\_stores\_norm)  
**print** (**"RMSE for test set of number of convenience stores: "**,RMSE\_test\_num\_of\_con\_stores\_norm)  
  
*#show the graph*plt.xlabel(**"iteration"**)  
plt.ylabel(**"cost function"**)  
plt.show()