9417 Assignment1

Zheng Qiwen   z5240149

Q1

theta0 for house age: 42.54078538346594

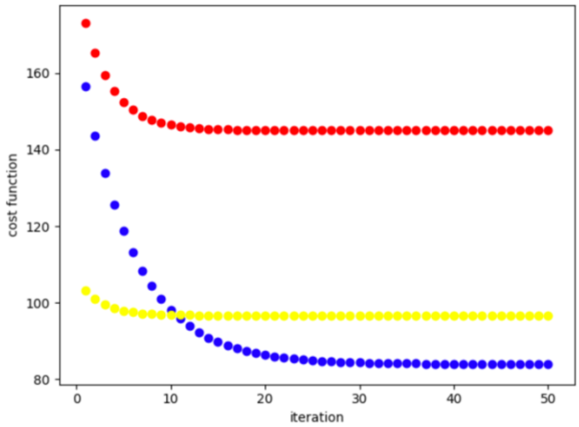
theta1 for house age: -10.319399022339129

Q2

The red curve: house age

The blue curve: distance to the nearest MRT station

The yellow curve: number of convenience stores



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)
```

```python
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)
```

```python
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()
```