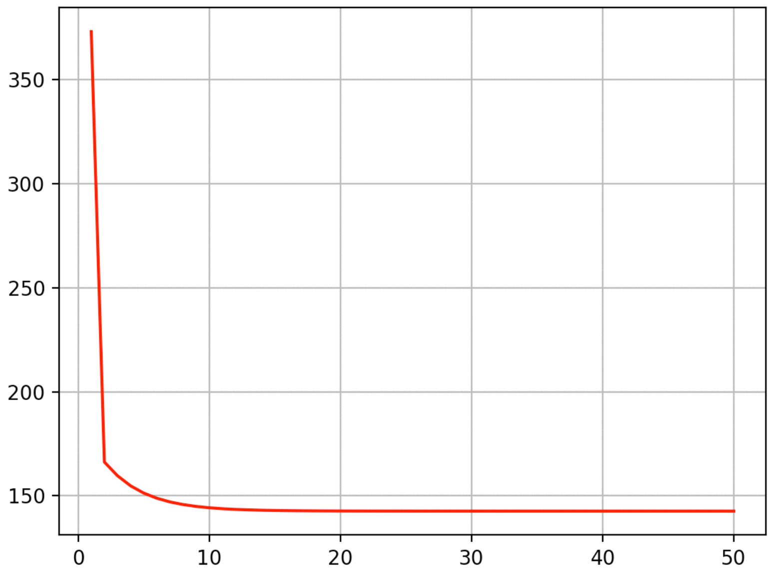
Comp9417 homework 1 report

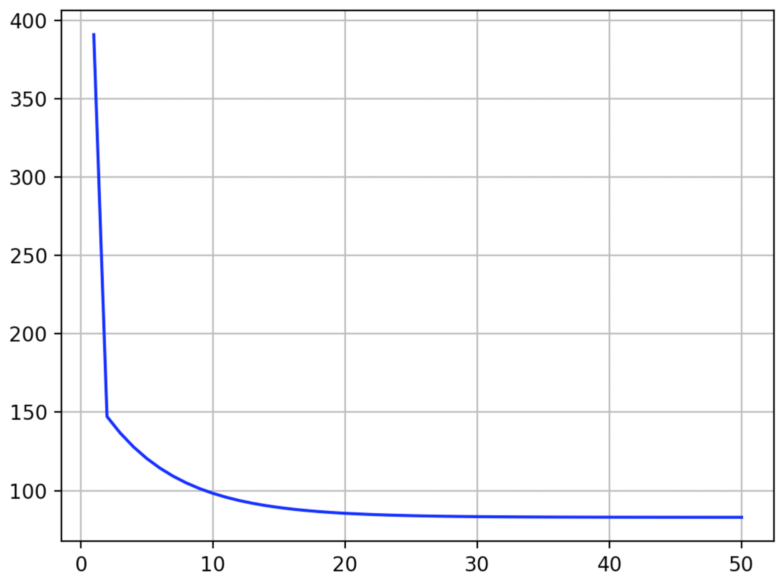
Z5196480
HUIYAO ZUO

1. The θ parameters ($\theta_<$, $\theta_;$) from step 3 when you are using house age feature.
   house age : θ 0 = 42.54078538346594 θ 1 = -10.319399022339129
   distance to the nearest MRT station : θ 0 = 44.766087037899375
   θ 1 = -46.500633970906314
   number of convenience stores : θ 0 = 27.486676129636784
   θ 1 = 25.642117651334722

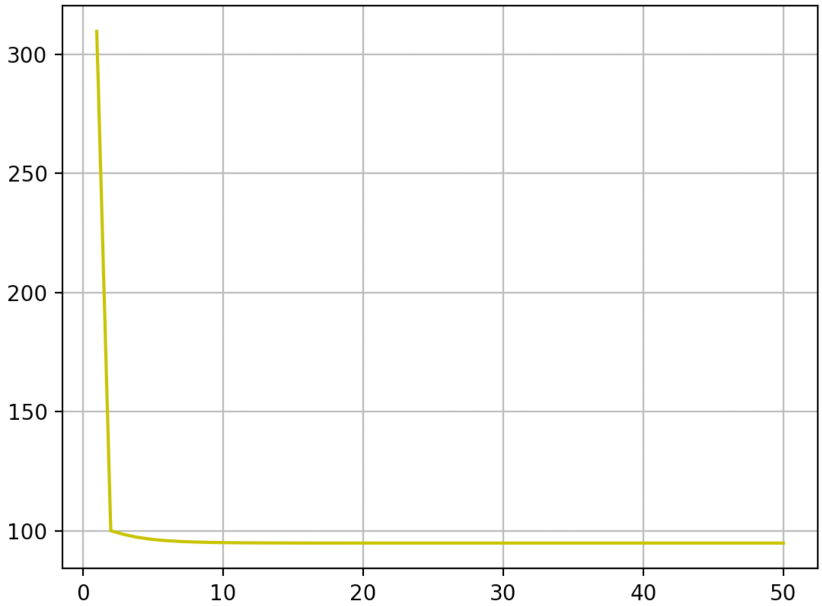
2. house age :



distance to the nearest MRT station :



number of convenience stores :



3.RMSE for your training set when you use house age feature.

12.045510305912353

4.RMSE for test set, when you use house age feature.

16.58731450340051

5.RMSE for test set, when you use distance to the station feature.

12.652088009723935

6.RMSE for test set, when you use number of stores feature.

14.731993508206784

7.Compare the performance of your three models and rank them accordingly.

The best performance one is the second one , it has the smallest RMSE and the gap between the train set and test set is also the smallest. And from the plot of cost , can find the cost decrease much after one round of calculate . By compare the RMSE, the second performance one is the number of convenience stores , with lower train RMSE and lower test RMSE than the house age. So the performance rank is distance to the nearest MRT station > number of convenience stores > house age.

Code:

```
#9417 homework1
#z5196480

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


###prepocess the data
df_house = pd.read_csv('house_prices.csv',header=None)
df_house = df_house.drop(df_house.columns[0], axis=0)
df_house = df_house.drop(df_house.columns[0],axis = 1)
df_x1 = pd.to_numeric(df_house[1])
df_x2 = pd.to_numeric(df_house[2])
df_x3 = pd.to_numeric(df_house[3])
df_y = pd.to_numeric(df_house[4])
a = df_x1.min()
b = df_x1.max()
df_x1 = df_x1.apply(lambda x : (x-a)/(b-a))
a = df_x2.min()
b = df_x2.max()
df_x2 = df_x2.apply(lambda x : (x-a)/(b-a))
a = df_x3.min()
b = df_x3.max()
df_x3 = df_x3.apply(lambda x : (x-a)/(b-a))

### creating test and training set
slice_num = 300
df_x1_train = df_x1[0:slice_num]
df_x1_test = df_x1[slice_num:]
df_x2_train = df_x2[0:slice_num]
df_x2_test = df_x2[slice_num:]
df_x3_train = df_x3[0:slice_num]
df_x3_test = df_x3[slice_num:]

df_y_train = df_y[0:slice_num]
df_y_test = df_y[slice_num:]

####Stochastic gradient descent
def stochastic(x,y,num_0,num_1) :
```

```
    list_cost= []
    a = 0.01
    for i in range(50):
        cost = 0
        for j in range(1,len(x)+1):
            x_y = num_0 + num_1 * x[j]
            num_0 = num_0 + a * (y[j] - x_y) *1
            num_1 = num_1 + a * (y[j] - x_y) *x[j]
            cost = cost + pow((y[j] - (num_0 + num_1*x[j])),2)
        list_cost.append(cost/len(x))
    return num_0 , num_1 ,list_cost

def evaluation(x,y,num_0,num_1) :
    sum = 0
    for i in range(len(x)):
        x_y = num_0 + num_1 * x[i]
        sum = sum + pow(y[i] - x_y,2)
    rmse = pow(sum/len(x),0.5)
    return rmse

#####start training
####for x1
x_mark = [i for i in range(1,51)]

num_x1_0, num_x1_1 ,list_cost_x1 = stochastic(df_x1_train,df_y_train,-1,-0.5)
print(num_x1_0, num_x1_1 ,list_cost_x1)

num_x2_0, num_x2_1 ,list_cost_x2 = stochastic(df_x2_train,df_y_train,-1,-0.5)
print(num_x2_0, num_x2_1 ,list_cost_x2)

num_x3_0, num_x3_1 ,list_cost_x3 = stochastic(df_x3_train,df_y_train,-1,-0.5)
print(num_x3_0, num_x3_1 ,list_cost_x3)

#plt.plot(x_mark, list_cost_x1,'r' )

#plt.plot(x_mark, list_cost_x2,'b' )

plt.plot(x_mark, list_cost_x3,'y' )

plt.grid()
plt.show()

####start evaluation
df_x1_train_rems = np.array(df_x1_train).tolist()
df_x1_test_rems = np.array(df_x1_test).tolist()

df_x2_train_rems = np.array(df_x2_train).tolist()
df_x2_test_rems = np.array(df_x2_test).tolist()

df_x3_train_rems = np.array(df_x3_train).tolist()
df_x3_test_rems = np.array(df_x3_test).tolist()

df_y_train_rems = np.array(df_y_train).tolist()
df_y_test_rems = np.array(df_y_test).tolist()

rems_x1_train = evaluation(df_x1_train_rems,df_y_train_rems,num_x1_0,num_x1_1)
rems_x1_test = evaluation(df_x1_test_rems,df_y_test_rems,num_x1_0,num_x1_1)
rems_x2_train = evaluation(df_x2_train_rems,df_y_train_rems,num_x2_0,num_x2_1)
rems_x2_test = evaluation(df_x2_test_rems,df_y_test_rems,num_x2_0,num_x2_1)
rems_x3_train = evaluation(df_x3_train_rems,df_y_train_rems,num_x3_0,num_x3_1)
```

```
rems_x3_test = evaluation(df_x3_test_rems,df_y_test_rems,num_x3_0,num_x3_1)

print(rems_x1_train ,rems_x1_test)
print(rems_x2_train ,rems_x2_test)
print(rems_x3_train ,rems_x3_test)
```