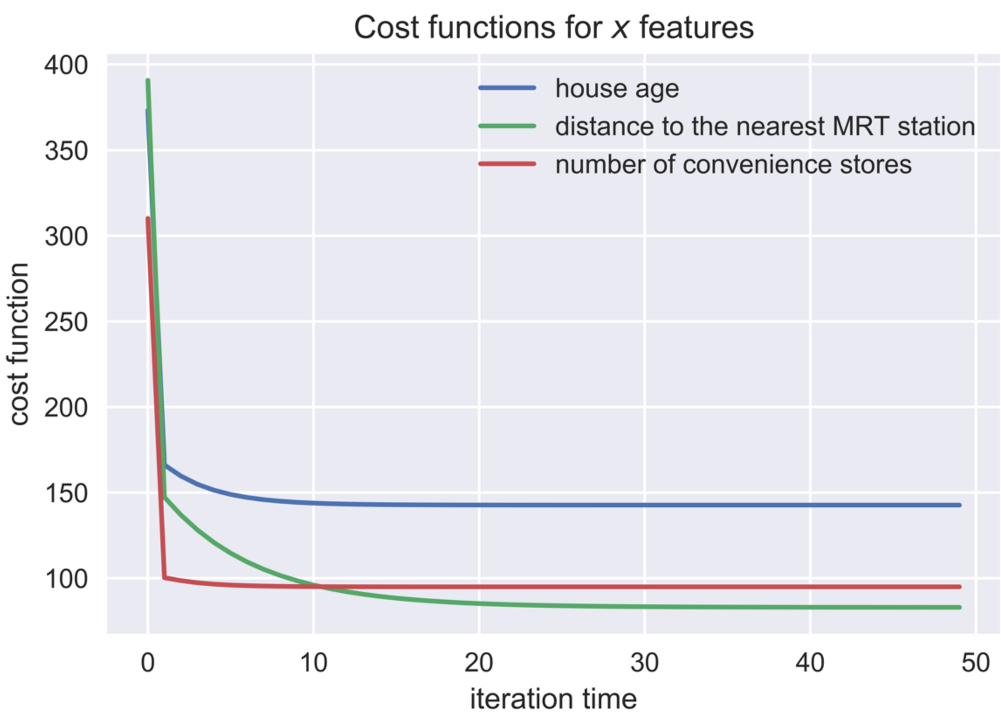
COMP9417 20T1 Assignment 1
Student Name: Minrui Lu
Student Number: 5277884

Question 1: The θ parameters (θ_0, θ_1) from step 3 when you are using house age feature. (2 marks)

In step 3, with the iteration times 50 and the training data set row number 300, we have updated the θ_0 and θ_1 $50 \times 300 = 15000$ times respectively. Each time, we update the θ value by $\theta_i = \theta_i + \alpha\left(y_j - h_\theta(x_j)\right)x_{ji}$.

From the program output, it can be seen that from step 3, using the house age feature, the value of θ_0 is 42.54098352098717, and the value of θ_1 is -10.321581018919572.

Question 2: A plot, which visualises the change in cost function $J(\theta)$ at each iteration. (1 mark)



Question 3: RMSE for your training set when you use house age feature. (0.5 mark)

By calculating the RMSE formula for the house age feature in the training set:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^i - \hat{y}^i)^2}$$
$$= \sqrt{\frac{1}{300}\sum_{i=1}^{300}\left(y^i - h(x_i, \theta_0, \theta_1)\right)^2}$$

We can get the RMSE for training set when I use house age feature is 12.045471635151399.

Question 4: RMSE for test set, when you use house age feature. (0.5 mark)

By calculating the RMSE formula for the house age feature in the test set:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^i - \hat{y}^i)^2}$$
$$= \sqrt{\frac{1}{100}\sum_{i=1}^{100}\left(y^i - h(x_i, \theta_0, \theta_1)\right)^2}$$

We can get the RMSE for test set when I use house age feature is 16.587314577458564.

Question 5: RMSE for test set, when you use distance to the station feature. (0.25 mark)

By calculating the RMSE formula for the distance to the station feature in the test set:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^i - \hat{y}^i)^2}$$
$$= \sqrt{\frac{1}{100}\sum_{i=1}^{100}\left(y^i - h(x_i, \theta_0, \theta_1)\right)^2}$$

We can get the RMSE for test set when I use distance to the station feature is 12.65187816696171.

Question 6: RMSE for test set, when you use number of stores feature. (0.25 mark)

By calculating the RMSE formula for the number of stores feature in the test set:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^i - \hat{y}^i)^2}$$
$$= \sqrt{\frac{1}{100}\sum_{i=1}^{100}\left(y^i - h(x_i, \theta_0, \theta_1)\right)^2}$$

We can get the RMSE for test set when I use number of stores feature is 14.732079954030375.

Question 7: Compare the performance of your three models and rank them accordingly. (0.5 mark)

In order to compare the performance of three models, we need to calculate the difference of RMSE between training model and test model among the three features respectively.

House Age Feature:

RMSE for house age training set is 12.045471635151399, RMSE for house age test set is 16.587314577458564, with $\theta_0 = 42.54098352098717, \theta_1 = -10.321581018919572$.

Distance to Station Feature:

RMSE for distance to station training set is 9.165812661768193, RMSE for distance to station test set is 12.65187816696171, with $\theta_0 = 44.76248196156705, \theta_1 = -46.474566532140706$.

Number of Stores Nearby Feature:

RMSE for number of stores nearby training set is 9.834850879113743, RMSE for number of stores nearby test set is 14.732079954030375, with $\theta_0 = 27.486960274404385, \theta_1 = 25.640465112647917$.

Considering that RMSE is the square root square root of the variance of the residuals, and it indicates how close are the observation data to the predicted values, lower values of RMSE shows better fit.

Therefore, by comparing the RMSE values of test set data, it could be seen that the model fitted by Distance to Station Feature has the best performance, followed by the model fitted by Number of Stores Nearby Feature, with the model fitted by House Age Feature ranking the last.

Namely, the performance ranking is:

Distance to Station > Number of Stores Nearby > House Age

The Python code is given below.

In [1]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

# prediction for instance x
def h(x,theta_0,theta_1):
    return (theta_0+theta_1*x)

# function of Stochastic Gradient Descent
def sgd(train_x,train_y,theta_0=-1,theta_1=-0.5,alpha=0.01):
    J_values = []
    for j in range(50):
        J_value = 0
        for i in range(train_x.shape[0]):
            theta_0 = theta_0 + alpha*(train_y[i]-h(train_x[i],theta_0,theta_1))
            theta_1 = theta_1 + alpha*\
            (train_y[i]-h(train_x[i],theta_0,theta_1))*train_x[i]
            J_value += (train_y[i]-h(train_x[i],theta_0,theta_1))**2
        J_values.append(J_value/train_x.shape[0])
    return theta_0,theta_1,J_values

# Normalize feature data and creat training sets and test sets
def pre_process(filename):
    # Read data file
    df = pd.read_csv(filename)
    min_list =[]
    max_list=[]
    for i in range(1,len(df.columns)-1):
        min_list.append(min(df[df.columns[i]]))
        max_list.append(max(df[df.columns[i]]))
        df[df.columns[i]] = (df[df.columns[i]]-min_list[i-1])\
        /(max_list[i-1]-min_list[i-1])
    # Data Split
    data_x = df.iloc[:,:-1]
    data_y = df.iloc[:,-1]
    # 300 rows for training and 100 rows for testing
    train_x,test_x,train_y,test_y = train_test_split\
    (data_x,data_y,test_size=0.25,shuffle=False)
    return (df,train_x,test_x,train_y,test_y)

# Build a linear regression model for given feature name
def feature_regression(train_x,test_x,train_y,test_y,feature_name):
    train_x = train_x[feature_name]
    test_x = test_x[feature_name]
    theta_0,theta_1,J_values = sgd(train_x,train_y)
    RMSE_train =0
    RMSE_test = 0
    for i in range(train_y.shape[0]):
        RMSE_train += (train_y[i]-h(train_x[i],theta_0,theta_1))**2
    for i in range(train_y.shape[0],train_y.shape[0]+test_y.shape[0]):
        RMSE_test += (test_y[i]-h(test_x[i],theta_0,theta_1))**2
    RMSE_train = np.sqrt(RMSE_train/train_y.shape[0])
    RMSE_test = np.sqrt(RMSE_test/test_y.shape[0])
    print('In the regression model for {0}, the theta 0 is {1},\
and the theta_1 is {2}'.format(feature_name,theta_0,theta_1))
    print('RMSE for {0} training set is {1:3},\
and RMSE for house age test set is {2:3}'.\
```

```
            format(feature_name,RMSE_train,RMSE_test))
    return (J_values)

# Plot all regression model's cost functions
def plot_all(data):
    if(len(data)==0):
        print('Input data for plotting is empty, exit now.')
        sys.exit()
    x_axis = np.arange(len(data[0][0]))
    plt.figure()
    plt.style.use('seaborn')
    for i in range(len(data)):
        plt.plot(x_axis,data[i][0],label=data[i][1])
    plt.legend()
    plt.xlabel('iteration time')
    plt.ylabel('cost function')
    plt.title('Cost functions for $x$ features')
    plt.savefig('plot.png',dpi=1000)
    plt.show()

def main():
    df,train_x,test_x,train_y,test_y = pre_process('house_prices.csv')
    plot_data = []
    for feature_name in df.columns[1:-1]:
        J_values = feature_regression\
          (train_x,test_x,train_y,test_y,feature_name)
        plot_data.append([J_values,feature_name])
    plot_all(plot_data)

if(__name__=="__main__"):
    main()
```

```
In the regression model for house age, the theta 0 is 42.54098352098
717,and the theta_1 is -10.321581018919572
RMSE for house age training set is 12.045471635151399,and RMSE for h
ouse age test set is 16.587314577458564
In the regression model for distance to the nearest MRT station, the
theta 0 is 44.76248196156705,and the theta_1 is -46.474566532140706
RMSE for distance to the nearest MRT station training set is 9.16581
2661768193,and RMSE for house age test set is 12.65187816696171
In the regression model for number of convenience stores, the theta
0 is 27.486960274404385,and the theta_1 is 25.640465112647917
RMSE for number of convenience stores training set is 9.834850879113
743,and RMSE for house age test set is 14.732079954030375
```