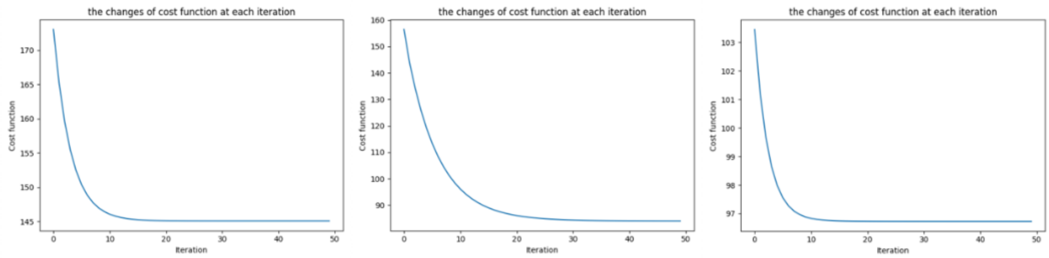
Rongtao Shen
z5178114

Homework 1

1. The θ parameters (θ_0, θ_1) from step 3 when you are using house age feature.
   $\theta_0 = 42.54098352$
   $\theta_1 = -10.32158102$

2. A plot, which visualizes the change in cost function $J(\theta)$ at each iteration.



3. RMSE for your training set when you use house age feature.
   RMSE for my training set when I use house age feature is 12.045471635151399

4. RMSE for test set, when you use house age feature.
   RMSE for test set when I use house age feature is 16.587314577458564

5. RMSE for test set, when you use distance to the station feature.
   RMSE for test set when I use distance to the station feature is 12.65187816696171

6. RMSE for test set, when you use number of stores feature.
   RMSE for test set when I use number of stores feature is 14.732079954030375

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

	RMSE for training set	RMSE for test set
house age	12.045471635151399	16.587314577458564
distance to the nearest MRT station	9.165812661768193	12.65187816696171
number of convenience stores	9.834850879113743	14.732079954030375

To compare the value of the RMSE in the form above, we can easily get the ranking of three models:
Rank1: Model by using the feature of distance to the nearest MRT station
Rank2: Model by using the feature of number of convenience stores
Rank3: Model by using the feature of house age

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import math


def pre_processing(x):
    max_x = x.max()
    min_x = x.min()
    for i in range(x.size):
        x_new = (x[i] - min_x) / (max_x - min_x)
        x[i] = x_new
    return x


def sgd(x, y):
    theta = np.array([-1, -0.5])
    learning_rate = 0.01
    max_iteration = 50
    cost = np.zeros(max_iteration)
    for i in range(max_iteration):
        for j in range(x.size):
            theta[0] = theta[0] + learning_rate * (y[j] - (theta[0] + theta[1] * x[j]))
            theta[1] = theta[1] + learning_rate * (y[j] - (theta[0] + theta[1] * x[j])) * x[j]
        cost[i] = cost_function(x, y, theta)
    rmse = RMSE(x, y, theta)
    return theta, cost, rmse


def cost_function(x, y, theta):
    m = x.size
    total = 0
    for i in range(m):
        total += (y[i] - (theta[0] + theta[1] * x[i])) ** 2
    return total / m


def RMSE(x, y, theta):
    m = x.size
    total = 0
    for i in range(m):
        total += (y[i] - (theta[0] + theta[1] * x[i])) ** 2
    return math.sqrt(total / m)


def test_RMSE(x_test, y_test, theta):
    m = x_test.size
    total = 0
    for i in range(m):
        total += (y_test[i] - (theta[0] + theta[1] * x_test[i])) ** 2
    return math.sqrt(total / m)
```

```python
if __name__ == '__main__':
    x1 = np.loadtxt('house_prices.csv', delimiter=',', usecols=1, skiprows=1)
    x2 = np.loadtxt('house_prices.csv', delimiter=',', usecols=2, skiprows=1)
    x3 = np.loadtxt('house_prices.csv', delimiter=',', usecols=3, skiprows=1)
    y = np.loadtxt('house_prices.csv', delimiter=',', usecols=4, skiprows=1)
    # pre-processing & normalisation
    x1_new = pre_processing(x1)
    x2_new = pre_processing(x2)
    x3_new = pre_processing(x3)
    # creating test and training set
    x1_training = x1_new[:300]
    x1_test = x1_new[300:]
    x2_training = x2_new[:300]
    x2_test = x2_new[300:]
    x3_training = x3_new[:300]
    x3_test = x3_new[300:]
    y_training = y[:300]
    y_test = y[300:]
    # stochastic gradient descent
    theta_new_1, cost_1, rmse_1 = sgd(x1_training, y_training)
    rmse_1_test = test_RMSE(x1_test, y_test, theta_new_1)
    theta_new_2, cost_2, rmse_2 = sgd(x2_training, y_training)
    rmse_2_test = test_RMSE(x2_test, y_test, theta_new_2)
    theta_new_3, cost_3, rmse_3 = sgd(x3_training, y_training)
    rmse_3_test = test_RMSE(x3_test, y_test, theta_new_3)
    print('theta_new_1: ', theta_new_1)
    print('rmse_1: ', rmse_1)
    print('rmse_1_test: ', rmse_1_test)
    print('theta_new_2: ', theta_new_2)
    print('rmse_2: ', rmse_2)
    print('rmse_2_test: ', rmse_2_test)
    print('theta_new_3: ', theta_new_3)
    print('rmse_3: ', rmse_3)
    print('rmse_3_test: ', rmse_3_test)
    # visualization
    plt.title('the changes of cost function at each iteration')
    plt.xlabel('Iteration')
    plt.ylabel('Cost function')
    plt.plot(cost_1)
    #plt.plot(cost_2)
    #plt.plot(cost_3)
    plt.show()
```