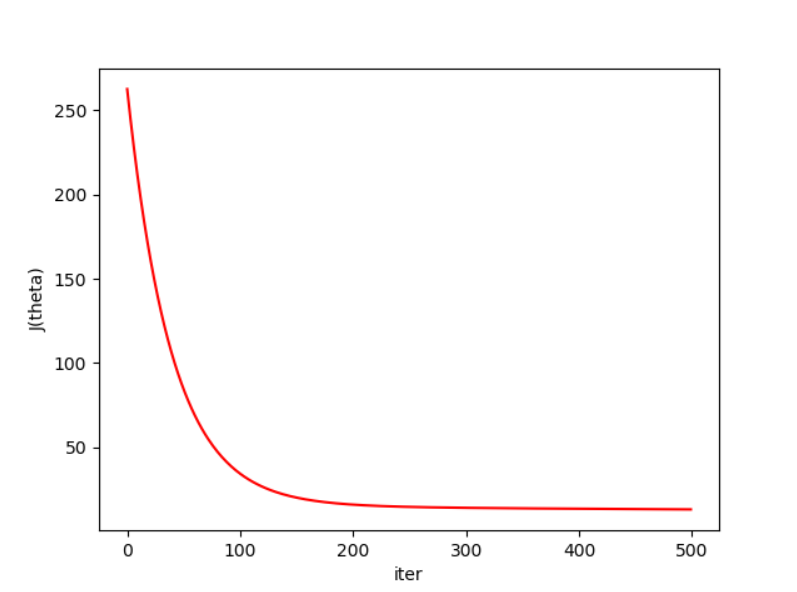
9417 homework1

1. You have to report the 𝜃 parameters in step 3 when you are using TV feature.

𝜃0 = 10.11283407, 𝜃1 = 8.27183129

1. A plot, which visualises the change in cost function 𝐽(𝜃) at each iteration.



1. RMSE for your training set when you use TV feature. RMSE = 3.64034549
2. RMSE for test set, when you use TV feature. RMSE = 3.90856034
3. RMSE for test set, when you use Radio feature. RMSE = 4.2004258
4. RMSE for test set, when you use newspaper feature. RMSE = 5.42790985
5. Compare the performance of your three models and rank them accordingly.

Here are the RMSE and 𝜃(fisrt one is 𝜃0 and second one is 𝜃1) for different features, so according to RMSE, it obvioursly TV feature perform better and then Radio, finally it is Newspaper.

TV

[ 10.11283407] [ 8.27183129]

RMSE for trainning [ 3.64034549]

RMSE for testing [ 3.90856034]

Radio

[ 10.82212248] [ 7.01320058]

RMSE for trainning [ 4.35732316]

RMSE for testing [ 4.2004258]

Newspaper

[ 12.9666593] [ 3.83741415]

RMSE for trainning [ 5.06890537]

RMSE for testing [ 5.42790985]

Code:

import matplotlib.pyplot as plt  
import pandas as pd  
  
J\_theta = []  
data = pd.read\_csv('Advertising.csv')  
  
  
def minmax(x, min\_x, max\_x):  
 return (x - min\_x) / (max\_x - min\_x)  
  
  
def RMSE(X, y, theta0, theta1):  
 res = 0  
 for i in range(len(X)):  
 res += (y[i] - (theta0 + X[i] \* theta1)) \*\* 2  
 res = res / len(X)  
 return pow(res, 0.5)  
  
  
def gradient\_descent(a, max\_iter, theta0, theta1, X, y):  
 cur\_iter = 0  
 newy = y.values.reshape(len(y), 1)  
 error0 = sum((newy - (theta0 + X \* theta1)) \*\* 2)  
 while cur\_iter < max\_iter:  
 # stochastic gradient descent  
 """  
 for i in range(len(X)):  
 diff = y[i] - (theta0 + X[i] \* theta1)  
 theta0 += a \* diff \* 1  
 theta1 += a \* diff \* X[i]  
  
 # batch gradient descent  
 """  
 J\_theta.append(error0/len(X))  
 res0, res1 = 0, 0  
 for i in range(len(X)):  
 diff = y[i] - (theta0 + X[i] \* theta1)  
 res0 += diff \* 1  
 res1 += diff \* X[i]  
  
 theta0 += a \* res0 / len(X)  
 theta1 += a \* res1 / len(X)  
  
 # print(theta0, theta1)  
 cur\_iter += 1  
 error1 = sum((newy - (theta0 + X \* theta1)) \*\* 2)  
 if abs(error1 - error0) < 0.0001:  
 break  
 error0 = error1  
  
 return theta0, theta1  
  
  
def show\_result(feature\_name):  
 features = [feature\_name]  
 X = data[features][:-10]  
 y = data.Sales[:-10]  
 max\_x = max(X[feature\_name])  
 min\_x = min(X[feature\_name])  
  
 X\_test = data[features][-10:]  
 y\_test = data.Sales[-10:]  
  
 Normalisation\_data\_X = minmax(X, min\_x, max\_x).values.reshape(len(X), 1)  
  
 Normalisation\_data\_test\_X = minmax(X\_test, min\_x, max\_x)  
 theta0, theta1 = gradient\_descent(0.01, 500, -1, -0.5, Normalisation\_data\_X, y)  
 print(theta0, theta1)  
 newy\_test = y\_test.values.reshape(10, 1)  
 newx\_test = Normalisation\_data\_test\_X[feature\_name].values.reshape(10, 1)  
  
 print("RMSE for trainning", RMSE(Normalisation\_data\_X, y, theta0, theta1))  
 print("RMSE for testing", RMSE(newx\_test, newy\_test, theta0, theta1))  
  
 plt.plot(J\_theta, color='red')  
 plt.ylabel("J(theta)")  
 plt.xlabel("iter")  
 plt.show()  
  
  
print("TV")  
show\_result('TV')  
print("Radio")  
show\_result('Radio')  
print("Newspaper")  
show\_result('Newspaper')