Jinu Jacob
CS 383
HW 2

1. Closed-form Linear Regression

Final Model: $g(x)$ = 1.0e+03 * [3.4256 + 0.8469x_1 + -0.3692x_2]

RMSE: 853.3806

To perform closed-form linear regression the data was first randomized by shuffling the order of the rows. The first two-thirds of the data was chosen for training and the remaining one-third for testing. Both training and testing sets were standardized using the mean and standard deviation of the training set. An additional feature was also added. The solution was computed using the following formula:

$$w = (x^T x)^{-1} x^T r$$

The solution was then applied to the testing samples and resulted in a root mean squared error of 853.3806.

2. S-Folds Cross-Validation

RMSE: 753.2995

To perform s-folds cross-validation the data is randomized and standardized. The data is then split into s different folds. Each fold is assigned to a separate cell in a cell array. Each fold is rotates being the testing set while the rest are used as the training set for closed-form linear regression. The square errors of each iteration are averaged and then square-rooted to get the root mean squared error.

Jinu Jacob
CS 383
HW 2

3. Gradient Descent

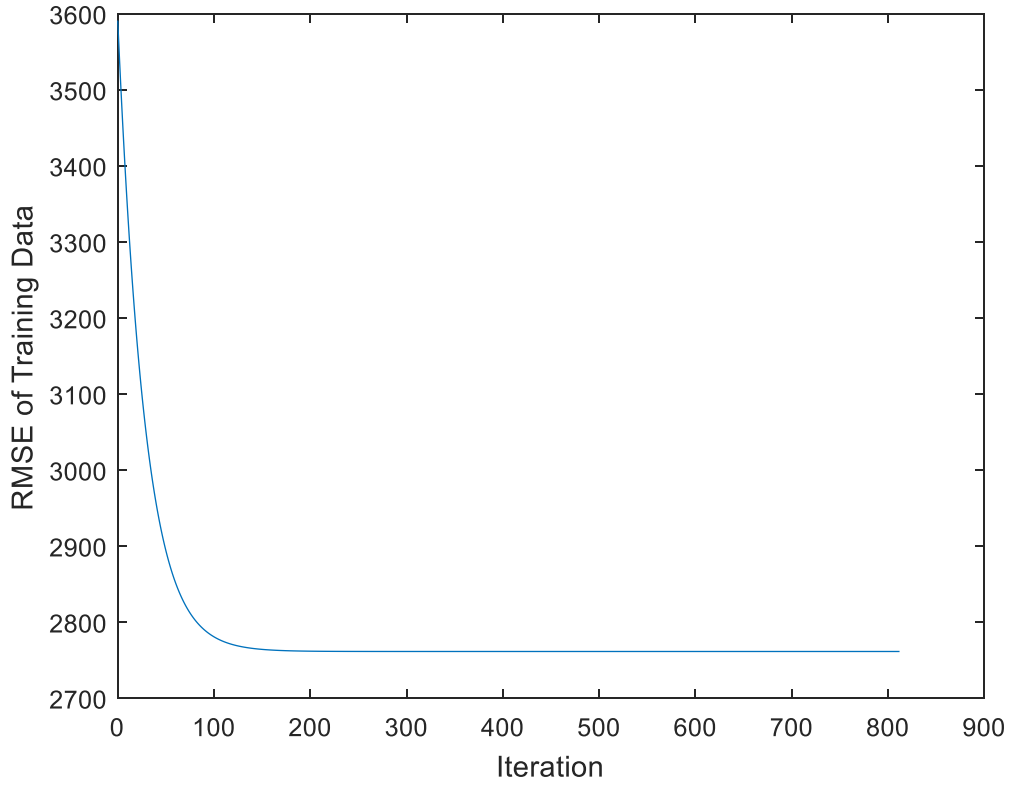
While writing this script I attempted two methods of updating the weights. I was not sure if the update step could be in one step or should be done in a loop.

The first method used the following code to update the weight for each parameter *j*:

```
w = w - ((lrnRate/n)*sum((predicted' - r_train)'*training(:,j)));
```

Final Model: $g(x)$ = 1.0e+03 * [1.3634 + 1.3641x_1 + - 1.3640x_2]

RSME: 3.2157e+03



The RMSE was very high but the plot looked similar to the example.

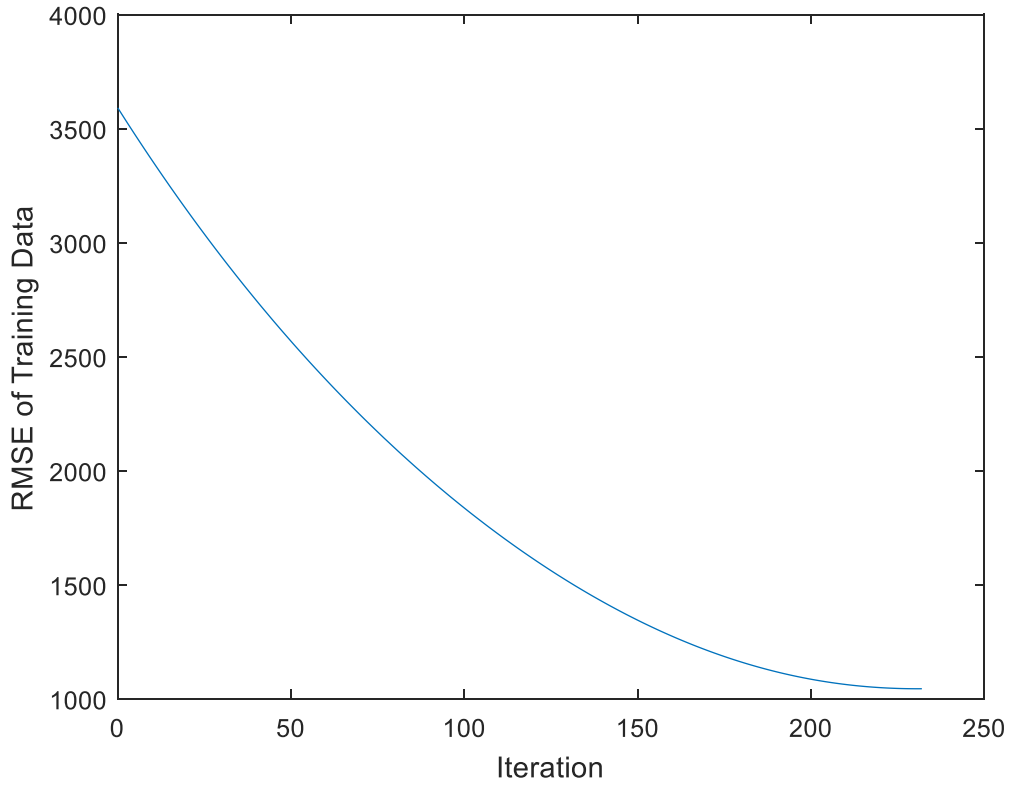
Jinu Jacob
CS 383
HW 2

In the second method for each parameter I iterated through each observation and individually calculated the gradient. I then summed the gradients multiplied the sum by the learning rate divided by the number of observations, and subtracted it the old weight to get the new weight.

```
% For each parameter
    for j=1:cols
        sum1 = 0;
        for i=1:length(training)
            p1 = (((w*training(i,:))' - r_train(i,:))' * training(i,j))
            sum1 = sum1 + p1(1);
        end
        w(j,:) = w(j,:) - ((lrnRate/n) * sum1);
    end
```

Final Model: $g(x)$ = 1.0e+03 * [2.7249 + 1.3244x_1 + -0.6320x_2]

RMSE: 791.0219



The root mean squared error was much lower but the plot does not resemble the example plot. The weights are also closer to the weights computed in closed-form linear regression.