CS434 Assignment 1 Report
Jiongcheng Luo, Tao Chen, Jiayi Du

Problem 1~3

Data manipulation: load the training data into corresponding X and Y matrices, where X stores the features (first 13 columns) and Y stores the desired outputs (last column).
Introduce a dummy feature to X by adding an extra column of ones at the beginning of X. By adding this dummy feature of 1's, the weight vector will have a constant (b) that can move the line, plane, or the hyper-plane up and down to fit the data better.

Results: The optimal weight vector **w** is shown below.

Column	Weight
1	39.584321218
2	-0.101137046
3	0.045893530
4	-0.002730387
5	3.072013402
6	-17.225407182
7	3.711252355
8	0.007158625
9	-1.599002102
10	0.373623375
11	-0.015756420
12	-1.024177030
13	0.009693215
14	-0.585969273

Problem 4

Data manipulation: load the training data into corresponding X and Y matrices, where X stores the features (first 13 columns) and Y stores the desired outputs (last column). This time we do **NOT** introduce dummy variable to X.

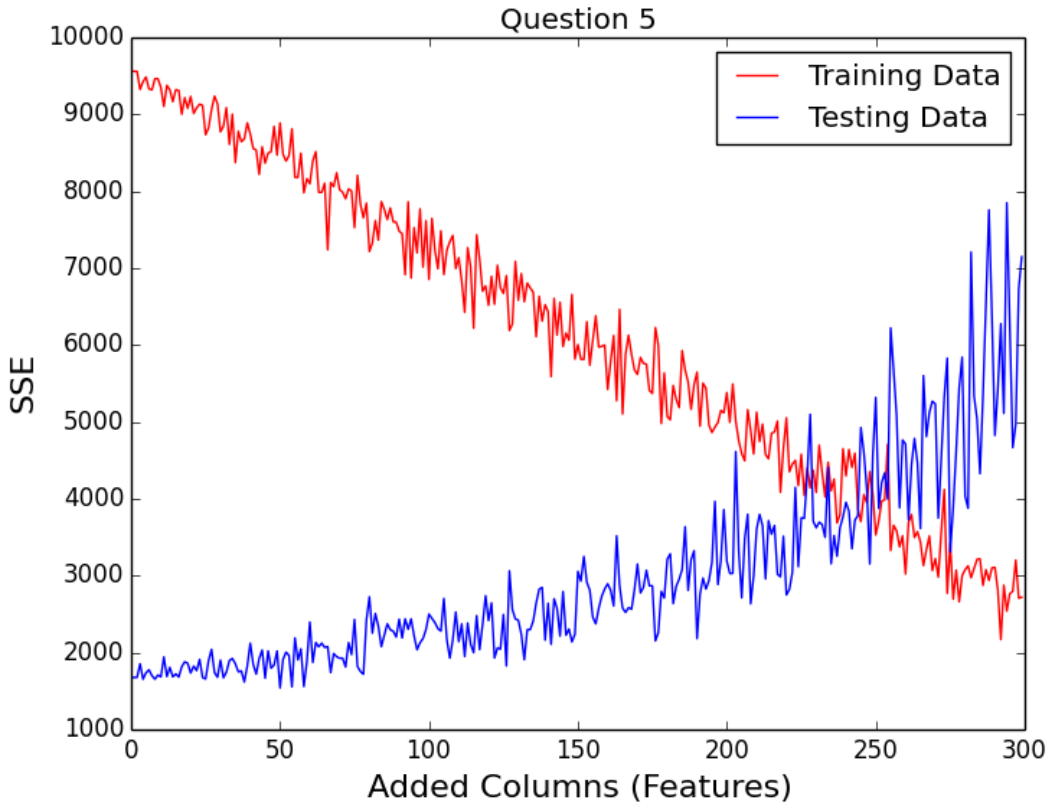
Results:

	Training Data	Testing Data
SSE with NO dummy var.	10598.057	1797.635
SSE with dummy var. (1)	9561.191	1675.231

Compare the results of SSEs, we found that the dummy variable help decrease the SSEs so that the existing of constant coefficient makes the linear model better in this case.

Problem 5

Data manipulation: load the training data into corresponding X and Y matrices, where X stores the features (first 13 columns) and Y stores the desired outputs (last column). Introduce a dummy feature to X by adding an extra column of ones at the beginning of X. Then generate additional features of random values, each from a specified uniform distribution [0,a] (a is different for each additional feature).



One of the Results:

# of different features	10	50	100	150	200	250	300
SSE_train_5	**9262.11**	**8472.96**	**7007.21**	**6225.62**	**4608.60**	**3792.19**	**2721.74**
SSE_test_5	**1746.72**	**1967.10**	**2497.94**	**2657.67**	**3459.25**	**3434.54**	**5478.19**

According to the graph, it is very easy to see that the SSEs for the training data set decrease as new features of random values are being added to the data set. And the SSEs for the testing data set do the opposite when applying the weight calculated from training. We ran the program a couple times. Because features were randomly generated, the numbers would be different. However, the general trend stayed the same, which led to a conclusion that more features might not be directly related to better performance.

The new features we generated were totally random. They had nothing to do with the given outputs, which was the Y column. So, the weight vector we got from the original data set was strongly tight to the real data, which was not random. When only a couple of new features were added to the set, the weight vector would get slightly tilted. As the number of new features got larger, and dominated the original set, the weight vector would get entirely offset. It would have a strong tight with the new features instead of the original data.

Since the new features added to the testing and training data sets were random numbers from a uniformly distribution, which means they were unrelated, to each other and to the output (Y), when applying the weight vector from the training set to the testing set, it wouldn't fit, which caused large SSEs.

Problem 6 ~ 8

Data manipulation: load the training data into corresponding X and Y matrices, where X stores the features (first 13 columns) and Y stores the desired outputs (last column). Introduce the dummy variable to X by adding an extra column of ones in the first column to X. To compute the optimal weight, we use the formula below with different values of λ (where we choose λ as 100 points between 0.0001 to 100).

$$w = (X^T X + \lambda I)\lambda^{-1} X^T Y$$

Results:

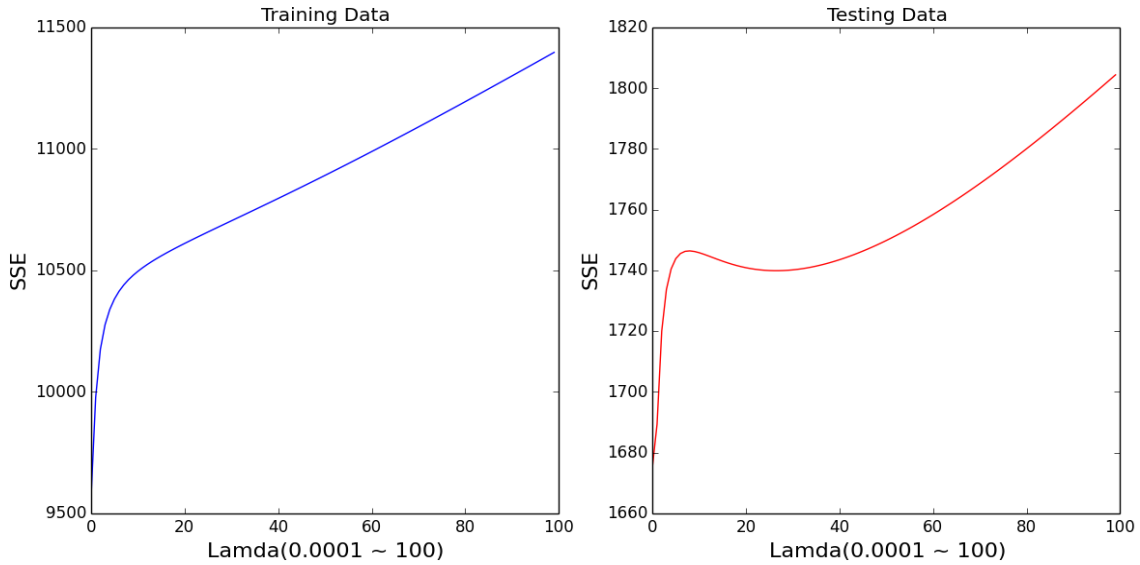


Figure 1. The sum of squared error (SSE) on training and testing data sets as a function of λ.

Q6: From the plots in Figure 1, we find that when λ is very small (<5) both SSE on training data and on testing data are increasing. We think that smaller λ encourages better fit of the data (driving SSE to zero). When λ is larger than 5, the larger the λ is, the larger SSE on training data is but SSE on testing data has a concave shape. We believe that the best λ is around 30 (between 20 and 40) where SSE on testing data is small.

Q7: Since "w" is a vector with 14 values, it's hard to see its changes with changing λ, so we take the first value for comparison, by comparing different λ (0.0001 ~ 100) to "w", as figure 2, we found that the as λ gets bigger, w gets approaching to 0.

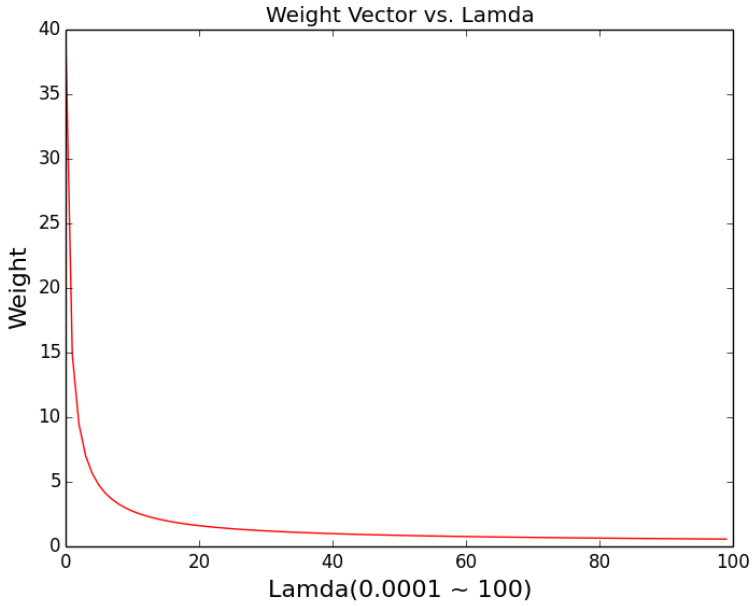


Figure 2

Q8:

After calculating the norm of the weight vector, we plot the norm of the weight vector as a function of λ, which is shown below. We can see that as λ gets bigger, the norm of the weight vector decreases. λ controls the trade-off between model complexity and the fit to the data. Large λ encourages simple model (driving more elements of w to 0) and Small λ encourages better fit of the data (driving SSE to zero).

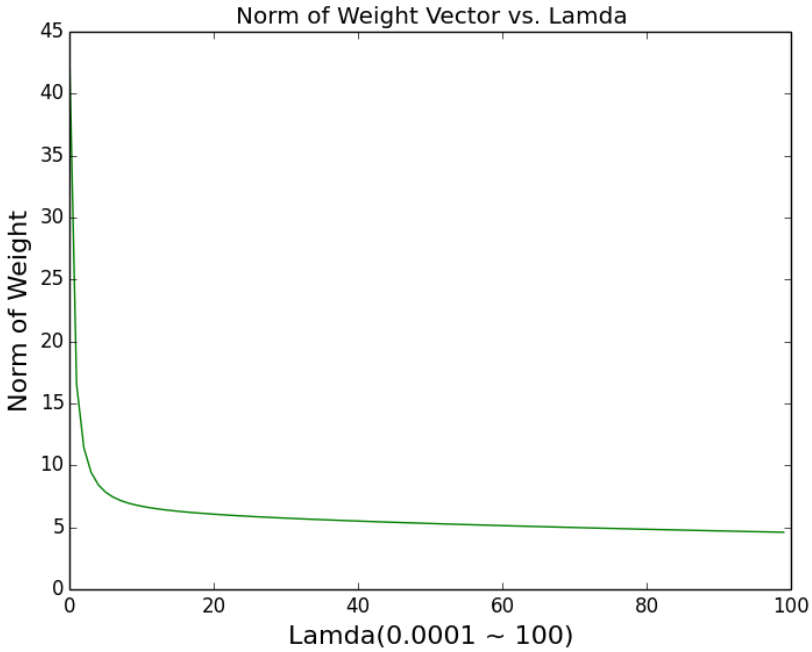


Figure 3