Robert Donohue - Data Mining Spring 2025 - HW1: Linear Regression

1. Implement L2 regularized linear regression algorithm with λ ranging from 0 to 150 (integers only). For each of the 6 dataset, plot both the training set MSE and the test set MSE as a function of λ (x-axis) in one graph.

SCREENSHOTS OF GRAPHS AT END OF DOCUMENT

(a) For each dataset, which λ value gives the least test set MSE?

Best λ for train-50(1000)-100: **8** with Test MSE: **6.1231**
Best λ for train-100(1000)-100: **11** with Test MSE: **5.8874**
Best λ for train-150(1000)-100: **11** with Test MSE: **5.4375**
Best λ for train-100-10: **7** with Test MSE: **4.1797**
Best λ for train-100-100: **19** with Test MSE: **6.0816**
Best λ for train-1000-100: **8** with Test MSE: **4.3695**

(b) For each of datasets 100-100, 50(1000)-100, 100(1000)-100, provide an additional graph with λ ranging from 1 to 150.

GRAPHS PROVIDED AT END OF DOCUMENT

(c) Explain why λ = 0 (i.e., no regularization) gives abnormally large MSEs for those three datasets in (b).

All three of these datasets have a very high ratio of features to samples. Specifically, for train-50(1000)-100, the matrix has fewer rows than it does features (or x-values). In this case, the matrix is singular. For train-100(1000)-100, the number of features and rows are the same, but the matrix could still be singular if rows are linearly dependent. This could also be true of train-150(1000)-100.

If the matrices truly are singular then their inverses cannot be computed and the program should throw an error. However, instead of an error, we simply get very high MSE values for λ = 0. This is likely due to floating point imprecision in Python. The determinants are not precisely 0, but are very close to 0 which leads to very high values for the inverse. This is why we see such high MSE values, especially for train-50(1000)-100. And, even in the case of train-150(1000)-100, while the matrix may not be singular, it is still 'ill-determined' which means that its number of rows is relatively close to its number of features. Again, this leads to a small determinant and a large inverse.

This is all mitigated when λ > 0 because in this case, adding λ*I makes XT*X easily invertible.

2. From the plots in question 1, we can tell which value of λ is best for each dataset once we know the test data and its labels. This is not realistic in real world applications. In this part, we use cross validation (CV) to set the value for λ. Implement the 10-fold CV technique discussed in class (pseudo code given in Appendix A) to select the best λ value from the training set.

(a) Using CV technique, what is the best choice of λ value and the corresponding test set MSE for each of the six datasets?

Best λ for train-50(1000)-100: **150**, Test MSE: **6.5907**
Best λ for train-100(1000)-100: **40**, Test MSE: **6.1412**
Best λ for train-150(1000)-100: **56**, Test MSE: **5.9588**
Best λ for train-100-10: **9**, Test MSE: **4.1819**
Best λ for train-100-100: **7**, Test MSE: **6.4004**
Best λ for train-1000-100: **13**, Test MSE: **4.3782**

(b) How do the values for λ and MSE obtained from CV compare to the choice of λ and MSE in question 1(a)?

The MSEs obtained from CV are higher in every case. This makes sense because in part 1-a we are choosing the optimal lambda. However, in part 2 we have to guess at the optimal lambda value based on the training data. The lambda values also differ in every case. Note that for train-50(1000)-100, the lambda is 150. If lambda were allowed to be even higher, a higher value may have been chosen. This is because the training data is so small compared to the

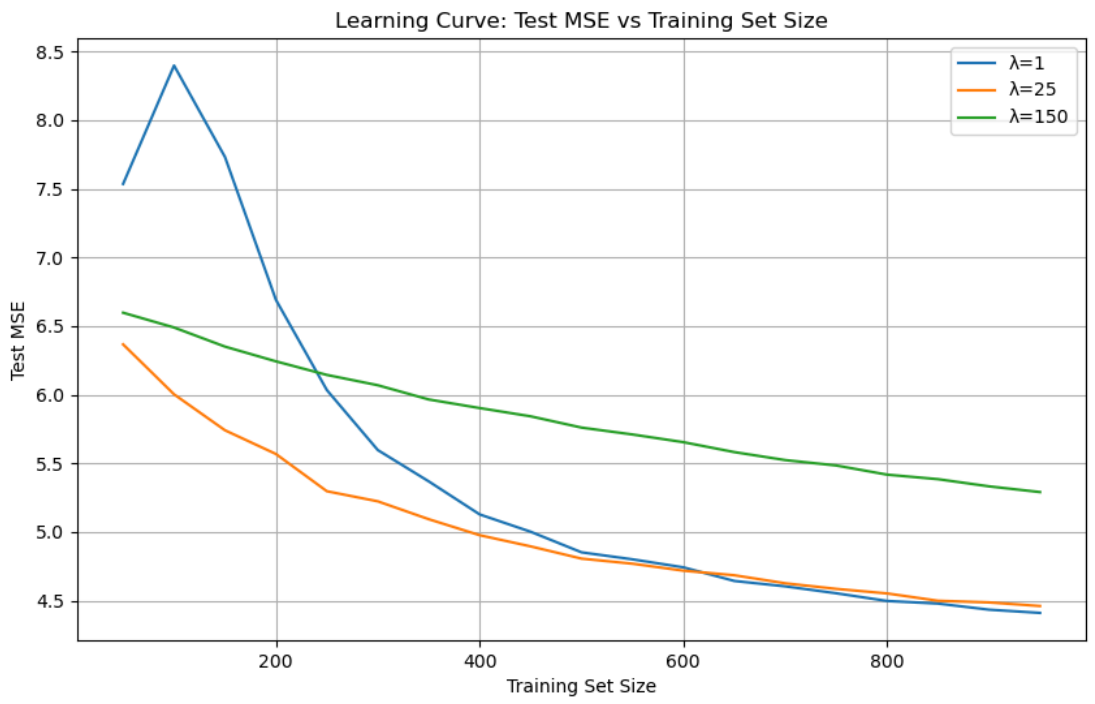
(c) What are the drawbacks of CV?

We can tell simply from the noticeable difference in runtime when using CV that it is more computationally expensive. It also performs poorly in smaller datasets because there is an increased chance of significant variance between the folds. Additionally, careful selection of the k-value (number of folds) is important to find an optimal balance between accuracy and computational efficiency.

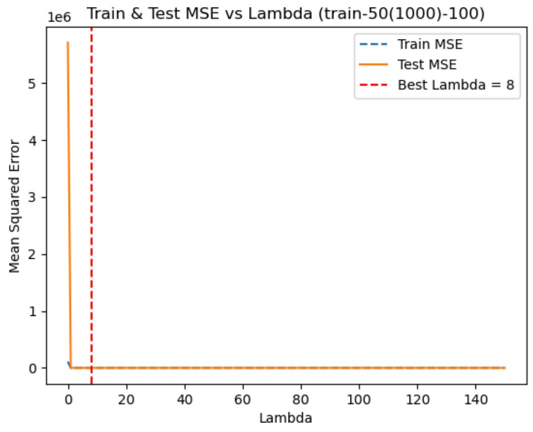
(d) What are the factors affecting the performance of CV?

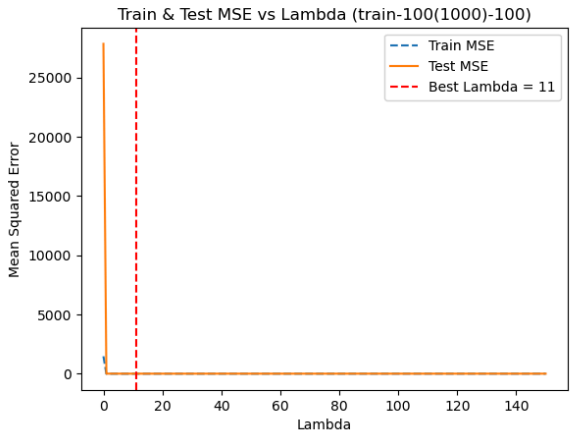
The size of the training data set, along with its variance, ratio of features to rows, the quality of data, and the number of folds all affect the performance of cross validation. Large data sets increase computational cost, as do higher values of k. Higher numbers of features with lower numbers of samples will lead to higher MSEs for CV, and will also likely lead to higher values of lambda. Well balanced data also can affect performance. If one subset of samples is relatively unlike other samples, it may impact the model heavily. This is because if this set takes up a large portion of a fold, this fold may misrepresent the data as a whole.

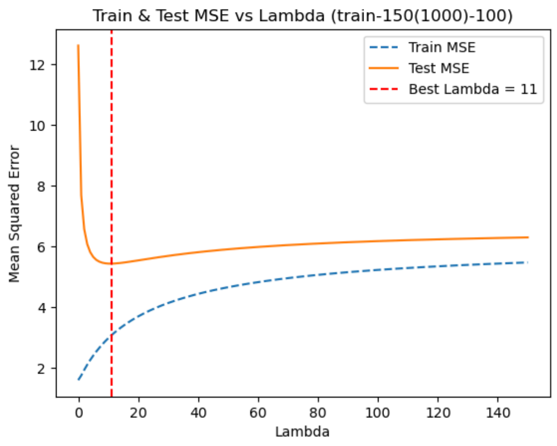
3. Fix λ = 1, 25, 150. For each of these values, plot a learning curve for the algorithm using the dataset 1000-100.csv.

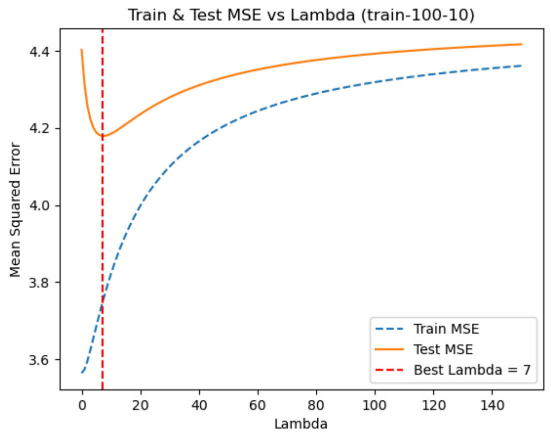


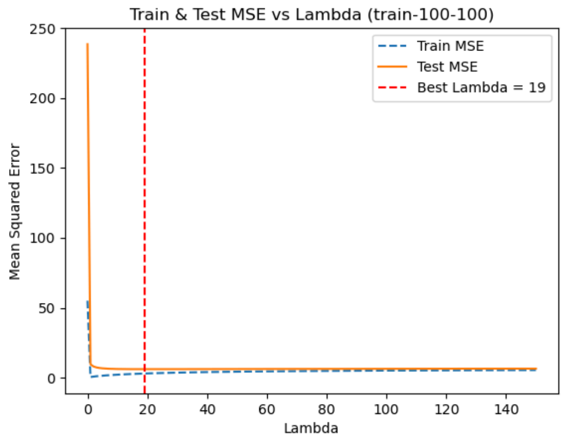
Graphs for part 1 - λ = 0 - 150

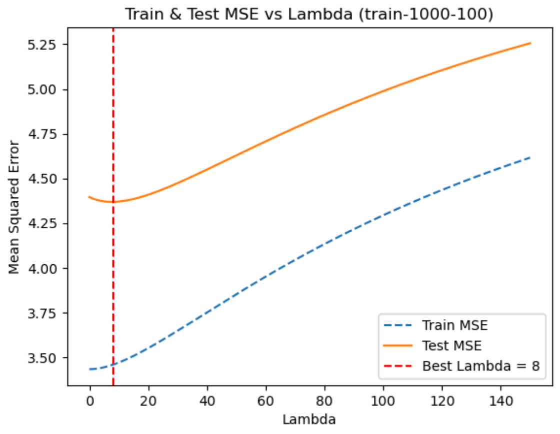




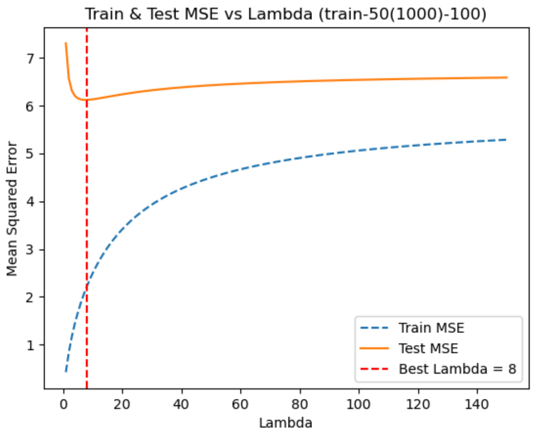


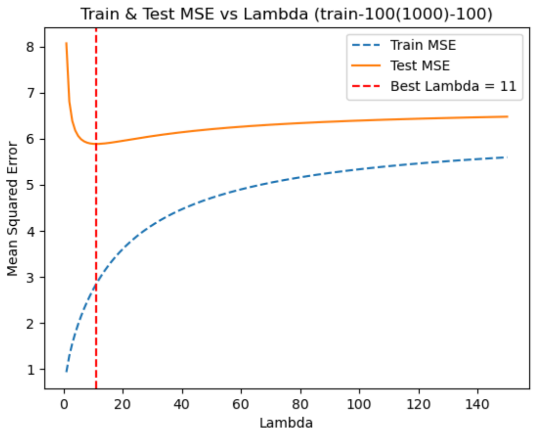


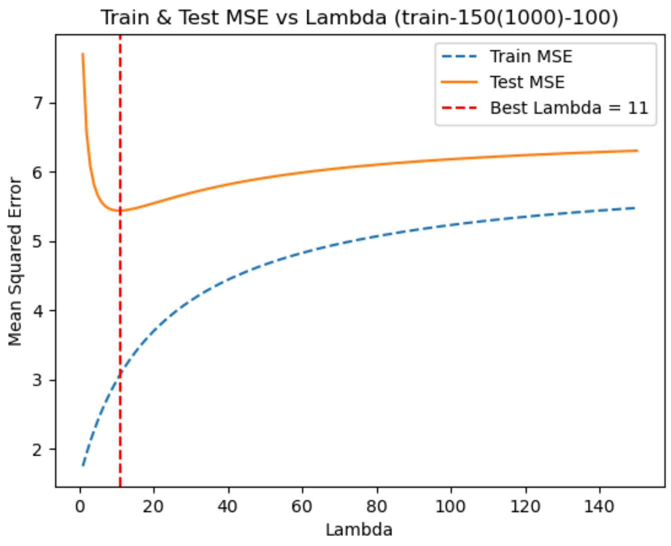




Graphs for part 1 - λ = 1 - 150


Train & Test MSE vs Lambda (train-50(1000)-100)
Mean Squared Error
Lambda
Train MSE
Test MSE
Best Lambda = 8


Train & Test MSE vs Lambda (train-100(1000)-100)
Mean Squared Error
Lambda
Train MSE
Test MSE
Best Lambda = 11


Train & Test MSE vs Lambda (train-150(1000)-100)
Mean Squared Error
Lambda
Train MSE
Test MSE
Best Lambda = 11