COMP 551 - Assignment 1

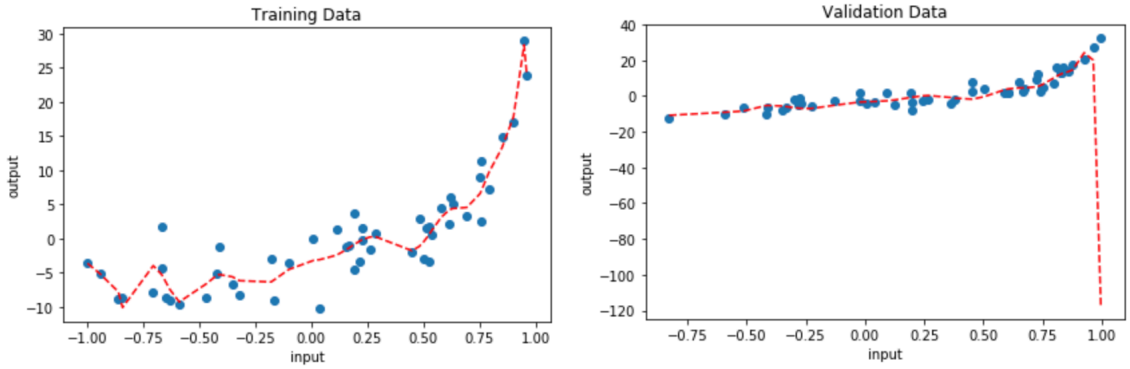
John Wu - 260612056

January 29, 2018

1 Model Selection

1.1 20 Degree Polynomial Without Regularization

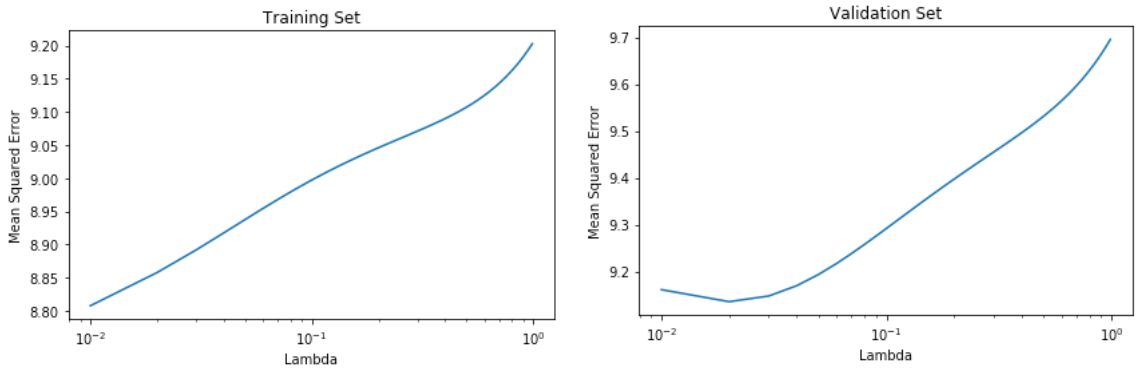
Fitting a 20 degree polynomial to the provided data set without regularization, we manage to achieve mean squared errors of 7.1524955066238824 and 459.03471278259 on our training and validation sets respectively. Visualized below are the fitted data for the corresponding polynomial used to predict our outputs.



Based on the MSE calculations, we can immediately notice that our model performs significantly worse when tested on the validation set as opposed to the training data. This is the result of over fitting and a prediction that is too biased towards the training data.

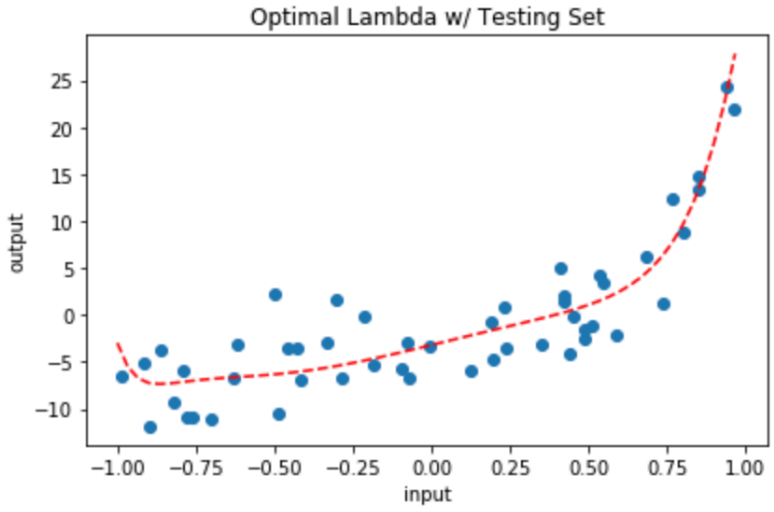
1.2 With L2 Regularization

Now we try fitting a 20 degree polynomial once again, but this time we include a regularization value, λ. In the graphs shown below, we vary λ from (0.0, 1.0) in constant intervals. The best MSE value when tested on the training data is 8.807610880442788 which occurs at $\lambda = 0.01$. When tested on the validation data, the best MSE is 9.135098784694684 which occurs at $\lambda = 0.02$.



As shown in the diagram, the MSE increases as we increase our value of λ in the training set. This occurs when we calculate our error on the same set that we trained our data with, because we technically know the exact function of the output, so any regularization will deviate our prediction further from the true function. However, when we calculate our error on the validation set, we can see that the MSE does improve as we increase our λ towards 0.02 at which point the error starts getting worse. This makes sense, because in this case we don't actually know the true function of the data which means that regularization will make our prediction less biased towards the data it had previously seen in the training set. Too much regularization will result in not enough bias towards the examples seen in our training set.

Using our optimal value as show above of λ, we now try out the new model on the testing data set. Shown below is the visualization of the fitted data into the new polynomial function.



The MSE above was calculated to be 10.730218400927187 which is about what we expect and fairly close to the optimal MSE values found when trying different lambdas earlier.

1.3 True Polynomial Of Data

When we tested our 20 degree polynomial without regularization, we found there was error even when compared to the same values it was trained on. This shows that the true degree of the source polynomial is probably more than 20, because otherwise one coefficient could have just

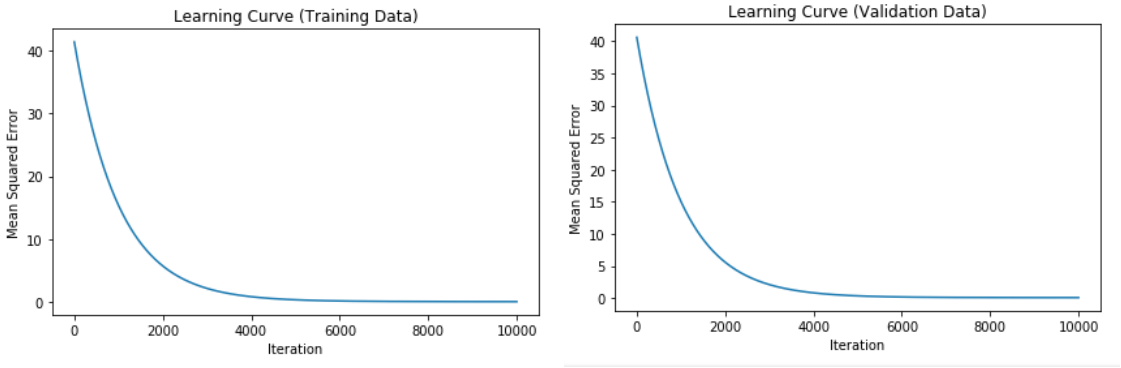
been mapped to one input to create zero error. In order to compute the actual degree of the source polynomial, we would have to keep increasing the polynomial degree until we reach the point where there is zero error on the training set.

2 Gradient Descent For Regression

For the following section, the initial parameters in our gradient descent were generated randomly on a scale from 1 to 10. In this specific test, the generated parameters were w0 = 7 and w1 = 8.

2.1 Stochastic Gradient Descent

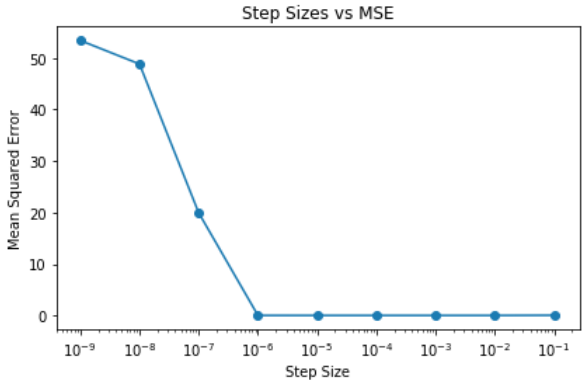
Using a step size of 1e-6 and iterating through 10,000 epoch cycles using a stochastic gradient descent, we obtain the learning curves shown below. The final MSE's in the test below are 0.11571391049627451 and 0.08595402153637659 for the training and validation sets respectively.



Based on the graph, we can see that early iterations of the gradient descent improve our MSE significantly and this rate decreases as we slowly approach the minimum.

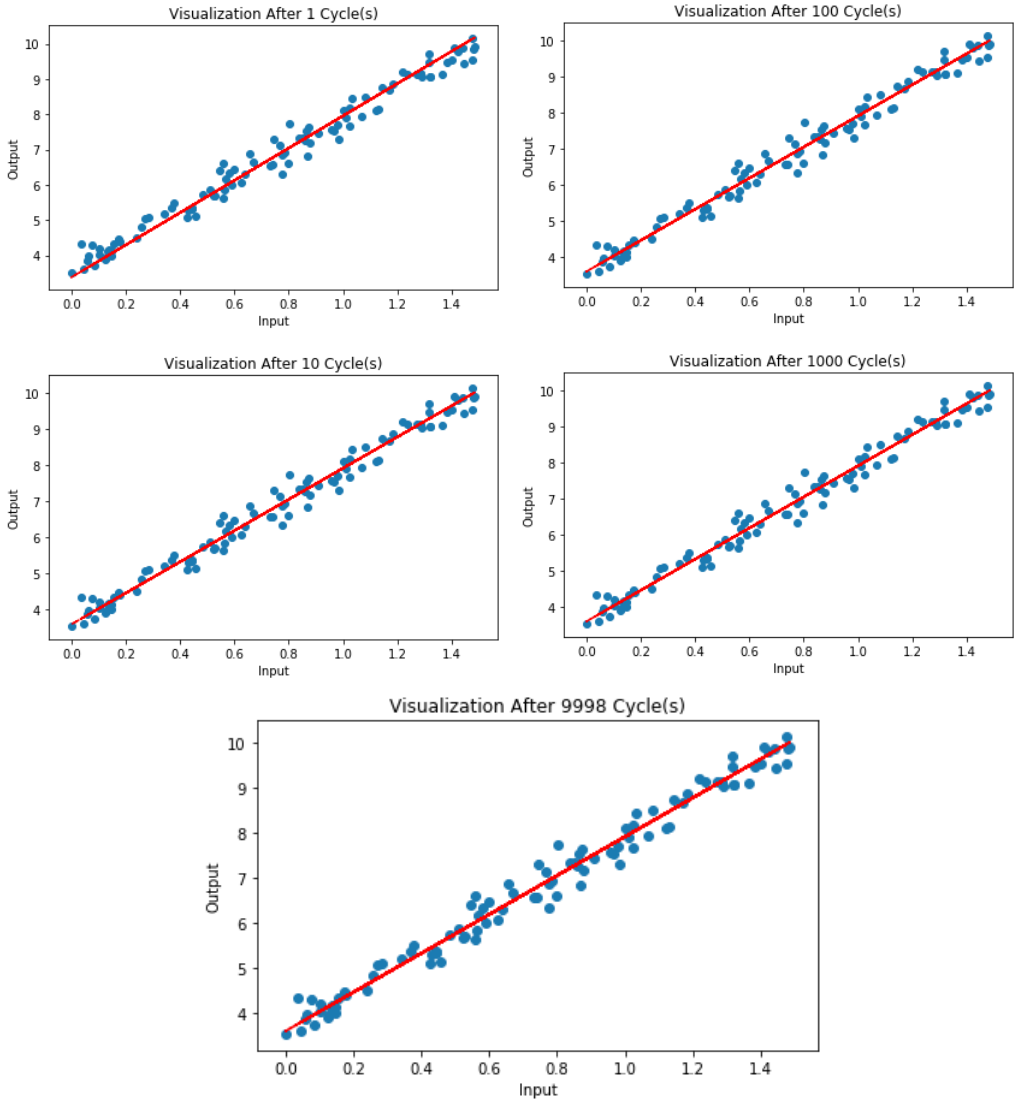
2.2 Varying Step Sizes

Using the same approach as above and varying our step size, we can see the varying accuracies shown in the following diagram. The optimal step size is calculated to be 0.01. Running the regression on the testing data using the new step size, we achieve an MSE of 0.06931249757133523.



2.3 Epoch Visualizations

Using the optimal step size obtained, we now repeat our gradient descent once again. This time, we visualize the progression by graphing the data fit during 5 different points during our training.



Although it is very subtle, we can see in the visualizations that our data fit improves on each iteration of an epoch. The reason for the very subtle improvements is because we chose the optimal step size which allowed us to reach a value close to the minimum very quickly at which point it starts making very small changes to our parameters as we approach the true minimum.

3 Real Life Dataset

3.1 Data Cleanup

In the given dataset, we fill in missing values by replacing it with the mean values of that specific column. In addition, the dataset description indicates that the first 5 columns are not predictive so

the data set was further cleaned out by eliminating these column. Other methods which were not implemented would be to also remove columns where there is too much missing data. After we clean our data set, we generate 5 random 80-20 splits of our data which can be found in the jupyter notebook.

3.2 Linear Regression Fit

When applying a linear regression model to each of our data sets, we achieve a 5-fold-cross-validation error of 0.633717360182823. In the output of our juypter notebook, we can see the final parameters for each of the data sets.

3.3 Ridge Regression

Using the ridge regression method, we repeat the experiment for different values of λ and we achieve the following results when calculating our error on the testing set:

- $\lambda = 0.0$, Average MSE = 0.41616949165723593
- $\lambda = 0.1$, Average MSE = 0.019564559023821256
- $\lambda = 1.0$, Average MSE = Average MSE: 0.01916695741384915
- $\lambda = 10.0$, Average MSE = 0.019114903079419255
- $\lambda = 100.0$, Average MSE = 0.019802911243695882

We can see here that the best λ value is calculated to be 10.0 and the parameters for each dataset can be shown in our jupyter notebook. We can use the parameters learned in the training to see which columns have the lowest weighting and drop those columns out of our training data. This is due to the fact that if the weighting is small, it likely has little or no correlation to predicting our output and is only deviating us slightly away from the actual result.