David Sahni, Burton Jaursch, Chase McWhirt

April 12th, 2019

CS 434: Machine Learning and Data Mining

Assignment 1: Linear & Logistic Regression

1 Linear Regression

1. *Learned Weight Vector:*

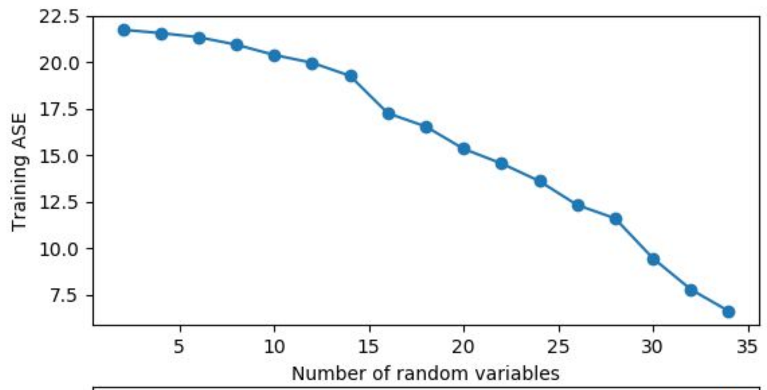
```
[ 3.95843212e+01]
[-1.01137046e-01]
[ 4.58935299e-02]
[-2.73038670e-03]
[ 3.07201340e+00]
[-1.72254072e+01]
[ 3.71125235e+00]
[ 7.15862492e-03]
[-1.59900210e+00]
[ 3.73623375e-01]
[-1.57564197e-02]
[-1.02417703e+00]
[ 9.69321451e-03]
[-5.85969273e-01]
```

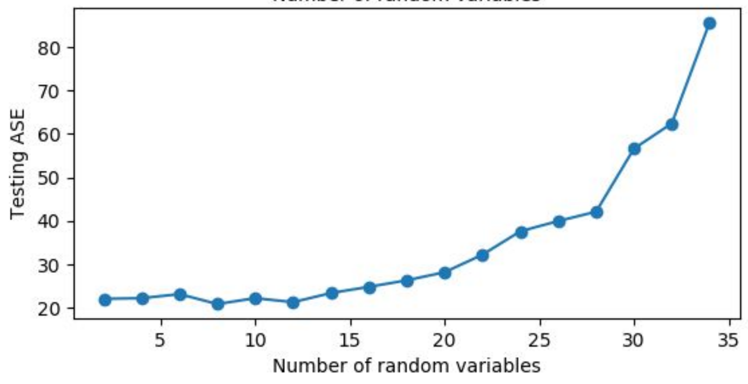
2. Training ASE: 22.08127, Testing ASE: 22.63825

The testing ASE is larger, which is consistent with our expectations as the learned weights we calculated were created using the training data. A lower average squared error value in the training data indicates that the error between the calculated and given results is less than the difference in the testing data.

3. Removing the dummy variable results in higher ASE values for both the training and testing data. The dummy variable is considered the baseline input feature, and generally helps to include this feature for more accurate learning data.

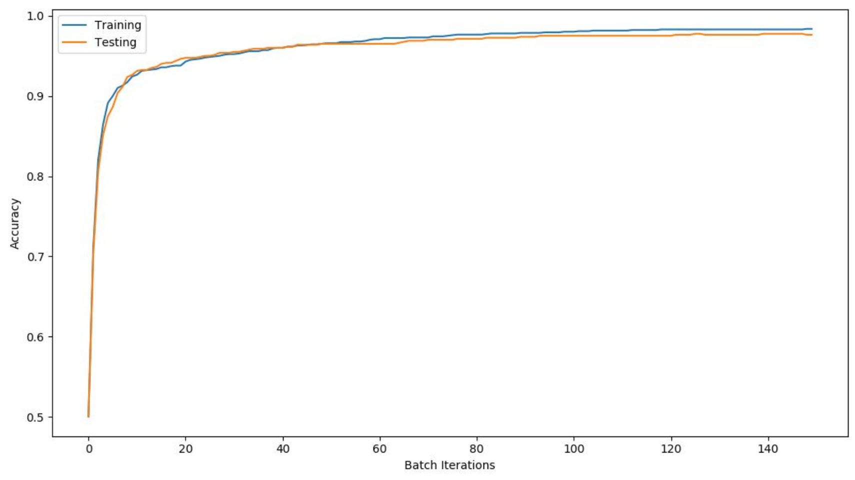
4. From the data we've collected, we see diverging results as we added random features to our data. In regards to our training data, we found that the increased data that we added to our model, the lesser the ASE. This was largely because our weight could better fit the specific situation being simulated within the training data. However, within the testing data the result was the opposite, and we found an increased ASE as more random features were implemented. This is most likely due to the overfitting of our model to the training data, and did not help with the data we used for testing.





2 Logistic Regression with Regularization

1. From our data, we notice that as we provide more batch iterations, our algorithm becomes increasingly accurate, and does so in a logarithmic manner. Further, it should be noted that it appears as the batch gets into larger numbers, the training data seems to become slightly more accurate relative to the testing data, and slowly pulls away. This would make sense considering the data is still fitting the testing data, and the training data is only reacting to it.

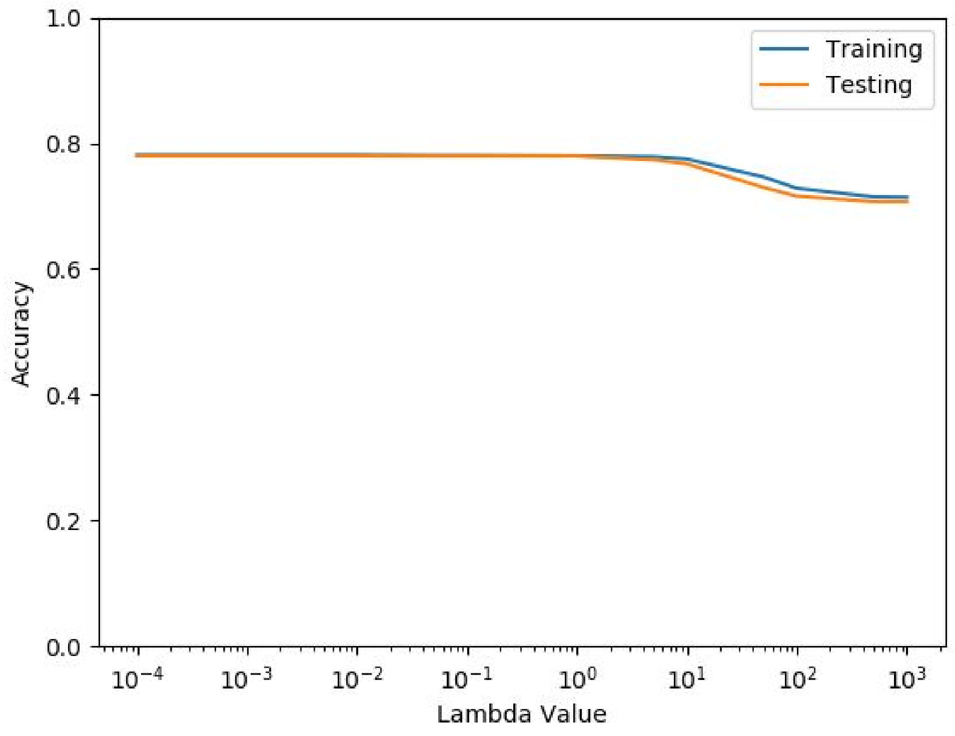


2. Pseudo Code for Adding a Regularization Term

```
Given: training examples ($x_1$, $y_i$), i=1, …, n
Let w ← (0, …, 0) //initializing weight values
Let ɳ = .000001 //learning curve
For(int i=0; i<150; i++){ //decided to run 150 times
    Δ = (0, … , 0) //Temporary holding of estimated weight values
    For i in range(1, n + 1) { // Note range(inclusive integer, exclusive integer)
        $\hat{y}_i \leftarrow \frac{1}{1 + e^{-w^T x_i}}$ //find prediction
        Δ ← Δ + ($\hat{y}_i$ - $y_i$ )$x_i$//add prediction difference to weight values
    }
    w ← w - (ɳ * Δ) //add difference of calc. weights back to original weight vector
}
```

3. Implementing the Derived Algorithm

After running the logistic regression formula with the additional regularization term, we didn't see much of a difference in accuracy for lambda (λ) values equal to one or less. This was likely because there were too few features for the impact of an additional regularization term to be noticable, even if applied multiple times. However, once lambda becomes greater than one, the effects on accuracy become more noticable. Once lambda is one thousand, we can see an approximate 10% drop in accuracy.


Training
Testing
Accuracy
Lambda Value
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}