Boinapalli, Pavan Kumar

PXB180007@utdallas.edu

Abstract

This report is written by Pavan Kumar Boinapalli as part of coursework for Applied Machine learning with course number BUAN 6341.501 at The university of Texas at Dallas

Assignment 1

BUAN 6341.501

## **Introduction:**

In this assignment, our goal is to predict the number of comments a facebook post would get in H number of hours. We will implement linear regression to estimate the coefficients of features. We will use batch gradient descent to find the coefficient parameters at which gradient function converges with minimum error. We shall use a modified Mean Squared Error as our loss function i.e; mean squared error divided by 2(for convenient gradient descent update). We shall perform experimentation with various alpha(learning parameter) values, different feature selection criteria, different thresholds(minimum change in loss function which defines convergence) and various initial parameter values.

## **Dataset:**

The data set is collected from UCI repository. It is available at the following link:

<https://archive.ics.uci.edu/ml/datasets/Facebook+Comment+Volume+Dataset>

**Features**:

"PageLikes","PageCheckins","PageTalkingAbout","PageCategory", "CC1","CC2","CC3","CC4","CC5","BaseTime","PostLength","PostShareCount","PostPromotionStatus","Hhours", "PostPublishedWeekday", "BaseTimeWeekDay" and "Target". Apart from these there are 25 derived features, derived from “Page” features. They are named “Derived\_1”,”Derived\_2”,…..,”Derived\_25”. Only “PostPublishedWeekDay” and “BaseTimeWeekDay” are categorical variables with seven levels and are already one-hot encoded in this dataset. Rest of the variables are continuous.

## **Linear Hypothesis Equation:**

B= coefficient vector containing values of coefficients

X = training example matrix containing feature values of each training example

= intercept

## **Loss Function:**

This is modified mean squared error because we divided mean squared error by 2.

B= coefficient vector containing values of coefficients

X = training example matrix containing feature values of each training example

Y = target values or vector containing true outcome values of each training example

n= total number of training examples.

## **Gradient Step:**

=coefficient of feature j

= configurable learning rate

=Value of feature j for training example i

# **TASKS:**

## **Feature Selection:**

I have selected 11 features for our regression model. The feature selection technique used, consists of two steps. First is a “f\_regression” method from sklearn. This method computes correlation between each feature to outcome variable and estimates the significance of feature using F-test and p-value. Second method is to find correlation between features and eliminate features that have correlation between them as more than 0.5. I have selected 42 features from first method and again took subset of these features to end up with 11 features.

The 11 features are:

['PageCheckins', 'PageCategory', 'derived\_11', 'derived\_21',

'BaseTime', 'PostShareCount', 'Hhours', 'PostPublishedWeekday1',

'PostPublishedWeekday4', 'PostPublishedWeekday7', 'BaseTimeWeekDay4']

## **Regression Equation:**

Target = b0 + bPageCheckins \* PageCheckins + bPageCategory \* PageCategory + bderived\_11 \* derived\_11 + bderived\_21 \* derived\_21 + bBaseTime \* BaseTime + bPostShareCount \* PostShareCount + bHhours \* Hhours + bPostPublishedWeekday1 \* PostPublishedWeekday1 + bPostPublishedWeekday4 \* PostPublishedWeekday4 + bPostPublishedWeekday7 \* PostPublishedWeekday7 + bBaseTimeWeekDay4 \* BaseTimeWeekDay4

Where Target = outcome variable = number of comments in Hhours.

b0 = number of comments post will receive if all the features are zero.

## **Initial Parameter values:**

('b0' : 0.006964691855978616), ('bPageCheckins': 0.0028613933495037947),

('bPageCategory' : 0.0022685145356420313), ('bderived\_11' : 0.005513147690828913), ('bderived\_21' : 0.007194689697855631), ('bBaseTime' : 0.004231064601244609),

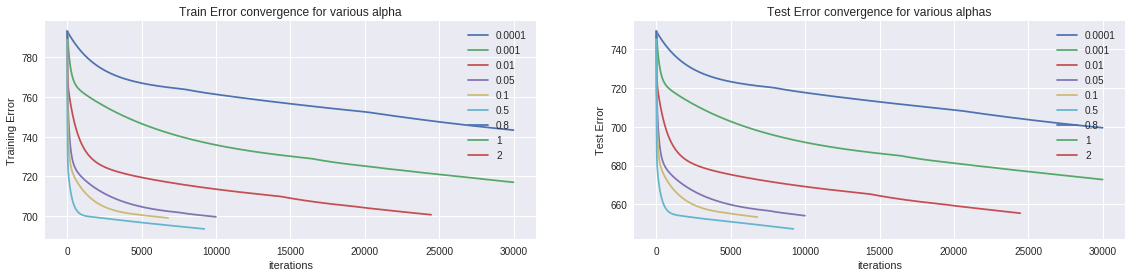
('bPostShareCount' : 0.009807641983846154), ('bHhours' : 0.006848297385848633), ('bPostPublishedWeekday1' : 0.004809319014843609), ('bPostPublishedWeekday4' : 0.003921175181941505), ('bPostPublishedWeekday7' : 0.003431780161508694),

('bBaseTimeWeekDay4' : 0.007290497073840416)

# **PART 1**

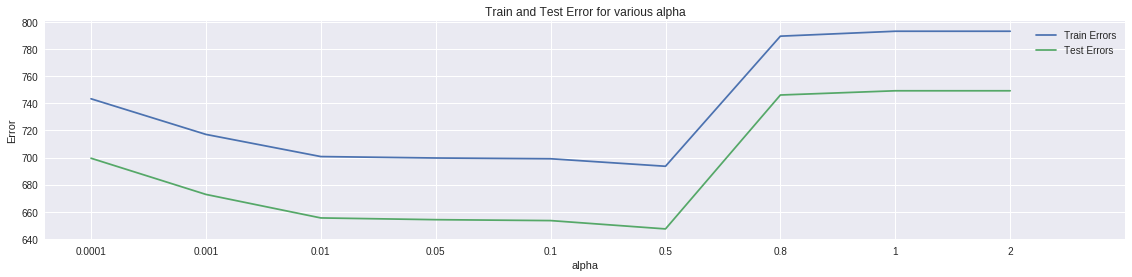
Let’s call the learning rate be alpha. Let convergence be defined as when the error value does not change by threshold% for the next iteration.

When the alpha is too low, the gradient descent algorithm takes high number of iterations to reach minimum value. When the alpha is high, the gradient descent algorithm takes less number of iterations to reach minimum but the training loss wobbles around the lowest value, but never actually reaches the lowest value. And for very high alpha values, the algorithm may not even converge but diverges. Hence, it is important to choose best alpha such that we reach close to lowest training error in not many iterations. **I experimented with alpha values of 0.0001,0.001,0.01,0.05,0.1,0.5,0.8,1,2** to check how the gradient descent algorithm converges. The plot of training error vs iterations and test error vs iterations can be seen as below. In this experiment threshold is fixed at 0.000001% change in error.



For alpha of 0.0001, the algorithm converges earlier and does not reach minimum value because the value is changing very slowly that threshold criteria is met. Even if threshold is set to lower than the current value, the algorithm takes too many iterations to reach the minimum as we can see from the graph. Hence this is not a good learning rate.

For alpha of 0.5, the algorithm converges faster in less number of iterations. For this threshold value alpha value, of 0.5 seems to be best learning rate since the algorithm meets both criteria that it reaches minimum error value in less number of iterations and also doesn’t diverge before threshold criteria is met.



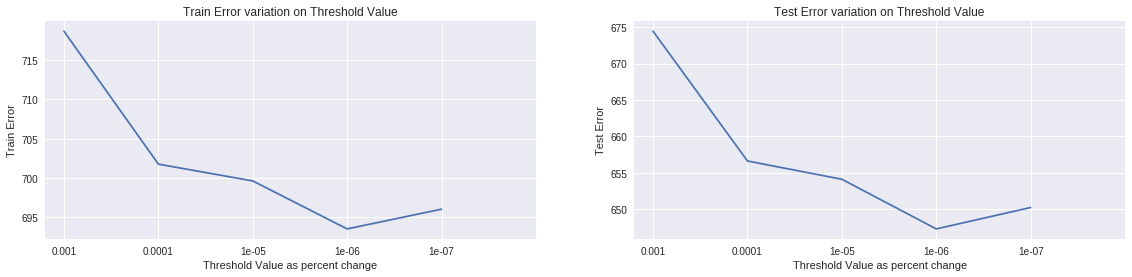
We can also see from the above plot that the error rate is lowest at alpha value of 0.5. For higher alpha values the algorithm diverges.

**We will pick alpha of 0.5 as our best learning rate for the threshold of 0.00001%**

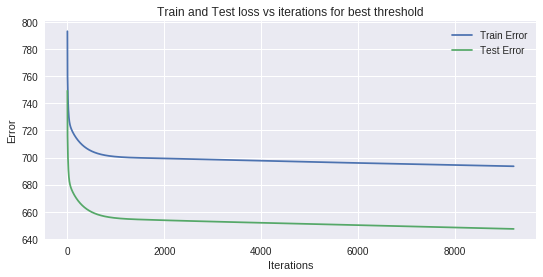
# **PART 2**

We Should also note that choice of threshold value matters because for low alpha values, since the algorithm converges slowly, the threshold must be low, for best alpha values, if the threshold is too low, algorithm does not stop at minimum error value and jumps around the minimum value. In this experiment, we will experiment with different threshold values for the best alpha of 1, that we picked from previous experiments.

I have picked threshold values of 0.001,0.0001,0.00001,0.000001,0.0000001 for experimentation. The plot of training error vs threshold values is as follows.



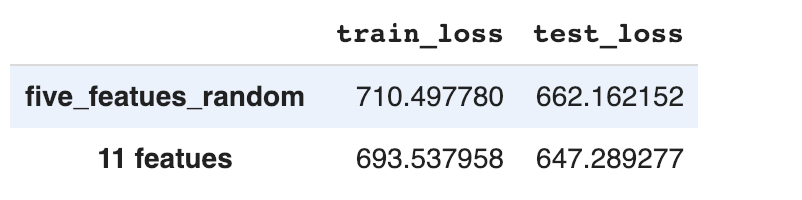
As we have not picked the lowest learning rate to make the algorithm converge faster, when threshold is too low, the error will move around true low error value, as we can see from the plot. **Hence, we pick the best threshold as 0.000001% for alpha value of 0.5** because at this point the training error will definitely converge closer to the minimum value.



# **PART 3**

In this experiment, we train the model with five randomly chosen features. I have used NumPy “Random” library to randomly pick features. My random features are:

The training error, test error comparison for this 5 feature model and original 11 feature model is follows: 'BaseTimeWeekDay7' 'derived\_14' 'Hhours' 'CC3' 'derived\_22'

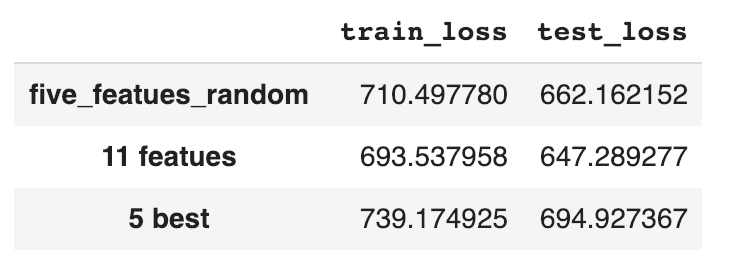


We get better error in our original 11 feature model than 5 random feature model because, in the original model, we used statistics like correlation between features and correlation of feature with target variable to pick features. In randomly selected features these statistics are ignored and hence error will be higher. This may be either due to multicollinearity or weak features.

# **PART 4**

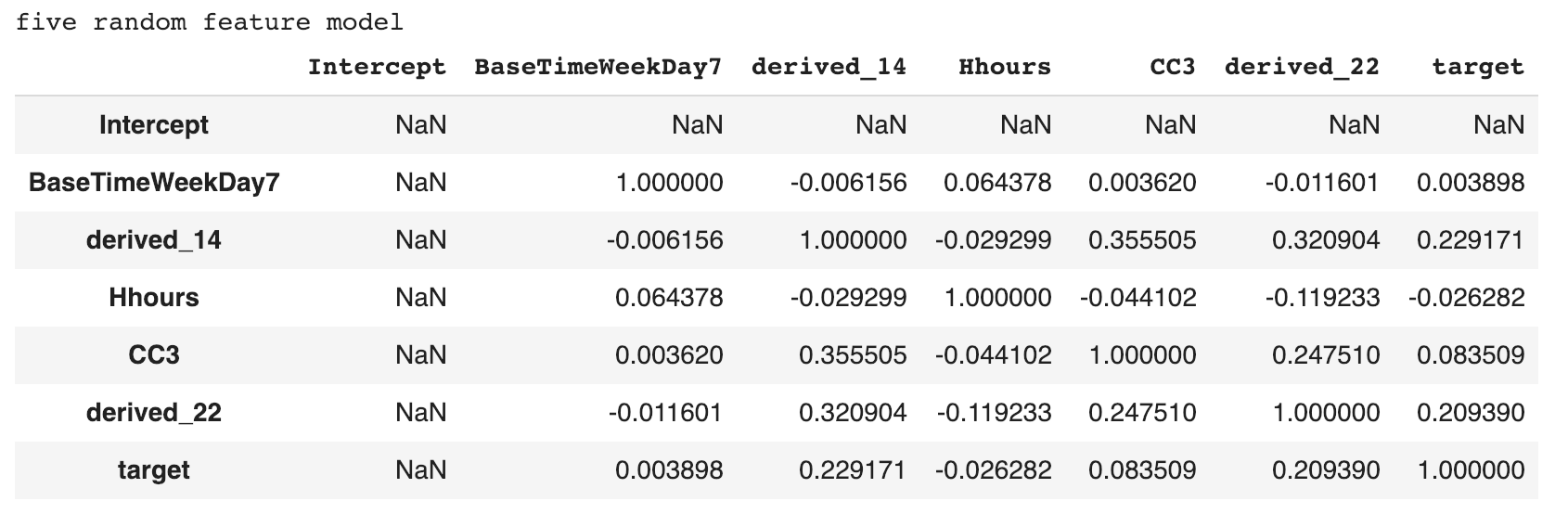
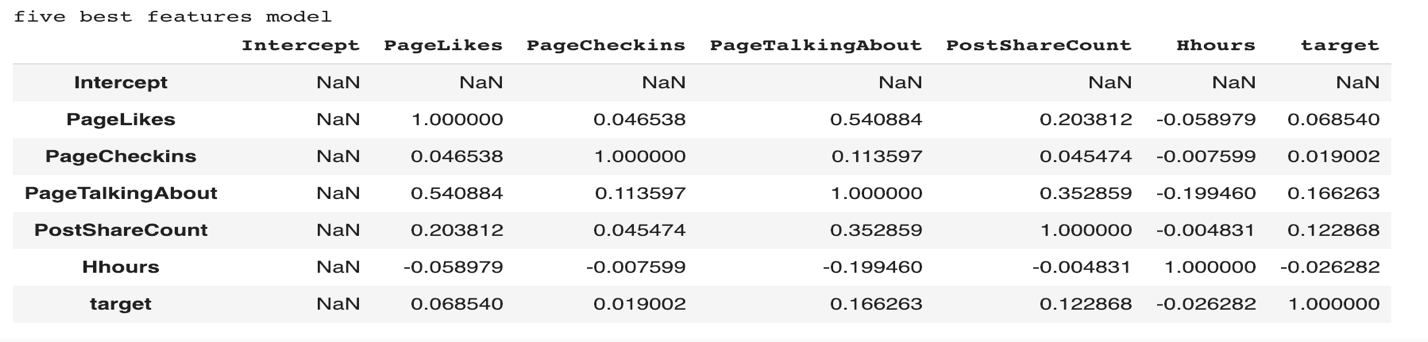
In this experiment, we train our model with 5 features that we think are good predictors intuitively. We will compare the train and test errors of this model with five randomly selected feature model and our original model with 11 features. The five features in this model are:

"PageLikes","PageCheckins","PageTalkingAbout","PostShareCount","Hhours"



We can see the comparison of train and test errors in 5 random feature model, original 11 feature model and 5 best feature model.

The 5 best feature model is not better than 5 random feature model because, the five features I selected are not correlated with target variable two of the features “PageTakingAbout” and “PageLikes” are correlated with each other. Multicollinearity and poor dependence of target variable on features may have resulted in this performance.



The 5 best feature model is not better than original 11 feature model because even though if 5 are good predictors, much of the information is lost when we decrease the number of features.

Counter-intuitively using all of the features, only results in fitting the noise(overfitting) and hence we should seek balance in choosing number of features.

## **Final Equation and Train and Test Error:**

**Final Equation:** 72.64771615894486 \* Intercept -32.46862789882107 \* PageCheckins -3.599246689043816 \* PageCategory + 73.77937688612116 \* derived\_11 -100.19957934375893 \* derived\_21 -28.45071471329087 \* BaseTime + 326.55193326328333 \* PostShareCount -3.4441248182697026 \* Hhours + 0.3334108368292466 \* PostPublishedWeekday1 + 0.3803835063246639 \* PostPublishedWeekday4 -0.2438554460733463 \* PostPublishedWeekday7 -0.04859933197221717 \* BaseTimeWeekDay4

Train Error: 693.537958

Test Error: 647.289277

## **Discussion:**

So according to our regression model, the coefficient values of derived\_21 , derived\_11 and post share count are high, and hence these are the important features. If the post share count is high, the number of comments will be high. Similarly derived\_11 and number of comments are similarly related. Derived\_21 and number of comments are inversely related. We can conlude that “**PostShareCount**” is an important factor in predicting number of comments.

I have used correlation and F-test feature selection techniques to reduce the error, apart from building the parameters of the model. I have also used MinMaxScale so that model will not show bias towards high scale features. However the error might be improved by removing the outliers.

## **Conclusion:**

From all these experiments we can get to conclusion that:

1)The learning rate must not be too high or too low, and best learning rate can be decided by plotting learning rate vs error plots. The one that reaches the minimum iterations is the best.

2)The threshold value is dependent on learning rate; low learning rates require low threshold to reach minimum error value. We choose the best threshold for the selected alpha by plotting threshold vs error plot.

3)It is better to use features that are selecting with initial data exploration than randomly chosen features.

4)Choosing a smaller number of features results in information loss and choosing too high features may result in overfitting. Also using less features takes less number of iterations to converge, although it doesn’t guarantee a good model.

## **References:**

<http://www.numpy.org/>

<https://scikit-learn.org>

https://en.wikipedia.org/wiki/Gradient\_descent