Predicting Useful Votes Received by a Yelp Review

Bharath Kumar Suresh
1211182086
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
bsuresh@asu.edu

***Abstract*— Regression techniques are widely used in the field of machine learning for prediction of a dependent variable given a set of independent variables [1]. In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known [2]. In this project, we predict the number of useful votes a yelp review will receive by using linear regression (regularized and non-regularized) and Iterative Boosting, a technique in which we model our regression problem as a classification problem and solve it recursively. We developed both the models and compared the results.**

Keywords— Yelp, Regression, Classification, Boosting

I. INTRODUCTION

Yelp is an American multinational corporation headquartered in San Francisco, California. It develops, hosts and markets Yelp.com and the Yelp mobile app, which publish crowd-sourced reviews about local businesses, as well as the online reservation service Yelp Reservations and online food delivery service Eat24. The company also trains small businesses in how to respond to reviews, hosts social events for reviewers, and provides data about businesses. Yelp tracks 3 community-powered metrics of review quality: Useful, Funny and Cool. The goal our project is to estimate the number of Useful votes a review will receive. The dataset for the project was obtained from Kaggle [3].

II. METHODOLOGY

A. Data Pre-Processing

The dataset obtained from Kaggle [3] contained Json files corresponding to the reviews, users who posted the review and the business for which the review was posted. The first step in data processing was to de-normalize the data. This involved merging the given data set Json files into a single CSV file by using the unique entries present in each of those Json files such as Review, User and Business IDs. For the review texts, we applied some filtering measures to remove special characters and converted upper case characters to lower case. N-grams of size 1,2 and 3 were generated for the reviews. Our approach was to use a bag-of-words model and this required reduction in the number of dimensions as we had a very high number to manage. So, we removed stop words from the dataset by setting a threshold on the frequency of all the words/N-grams in the dataset. The next step was to apply PCA on it. Based on our analysis, we chose a value 'k', the reduced number of dimensions that preserved about 95% of the variance in the original dataset for 1,2 and 3 grams. The datasets thus generated were used for our subsequent training and testing.

B. Boosting

We implemented the Ada Boost algorithm for Boosting. We restricted the maximum number of iterations to 8 since the time taken for training was high. We use Logistic Regression as a weak learner for each of the iterations. We resample [4] the training original data for each iteration of Boosting based on the weights associated with the examples. The test data for each of the iterations is the original data set. The algorithm returns the values of Alphas and coefficients of Logistic Regression.

C. Iterative Boosting

As mentioned, we remodeled our Regression problem as a classification problem. Since our objective was to predict the number of useful votes a review will receive, our classes were the number of useful votes themselves. Since the number of useful votes could take any Integral value, we had to build a model that predicts multiple classes. In order achieve this, we built a binary tree data structure represented in the form of an XML Tree. Each node of the tree would correspond to a particular portion of the training data set. Since Ada Boost algorithm works on binary class labels, we assign each of the examples in the set a class label of +1 or -1 based on the number of votes they have. We use the mean of the number of votes to split the values into two classes i.e. examples with votes less than the mean will have a class of -1 and rest of them +1. This is passed to the Boosting algorithm which returns the values of Alphas and coefficients of each learner. These values are stored in that node of the XML tree. Examples with class values -1 are then passed on to the Right Child and the ones with +1 to the Left Child of the node and the same process is repeated. Initially, the root node of the tree has the whole dataset associated with it. This recursive process is carried on until the dataset associated with the node becomes homogeneous in terms of the number of useful votes. i.e. when all the examples have the same number of useful votes due to which we can no longer split it based on the mean into two classes. Such nodes become the leaves of the tree.

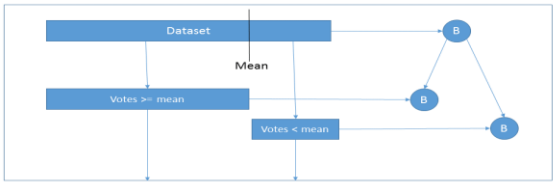


Fig. 1. Iterative Boosting Tree

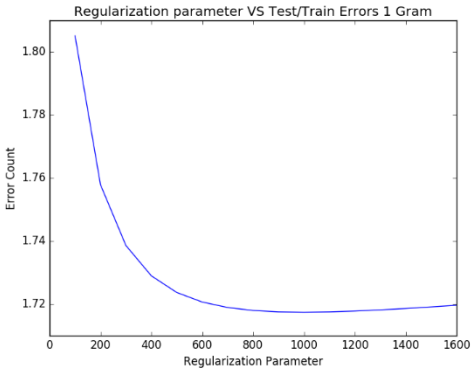
The Alphas and coefficients present in the nodes of the tree (except the leaves that do not have) enables us traverse the tree during prediction. For an unknown example for which we must predict the number of useful votes, we start with the root of the tree. By using the Alphas and Coefficients, we classify the example into +1 or -1 class label. We traverse to the Left Child if the class is +1 else we go to the Right Child of the root and the same process is repeated until we reach one of the leaves of the tree. When we encounter a leaf, we return the value of the votes the dataset corresponding to that leaf has (all the examples in the leaf have the same number of useful votes). That would be the number of useful votes predicted by the model.

D. Linear Regression

We developed a linear regression model to evaluate accuracy of the boosting model we developed. We ran the regression model for different values of λ, regularization parameter and found the value of λ that minimized the test error.

III. Results

We ran the Linear regression model for different values of λ in range [100, 200, ... , 1600] on different datasets like 1-Gram, 2-Gram and 3-Gram and plotted the RMSE values of the test errors vs. λ. From the graphs below we infer the λ that minimizes the test error for 1-Gram, 2-Gram and 3-Gram datasets are around 1000,500 and 700 respectively.



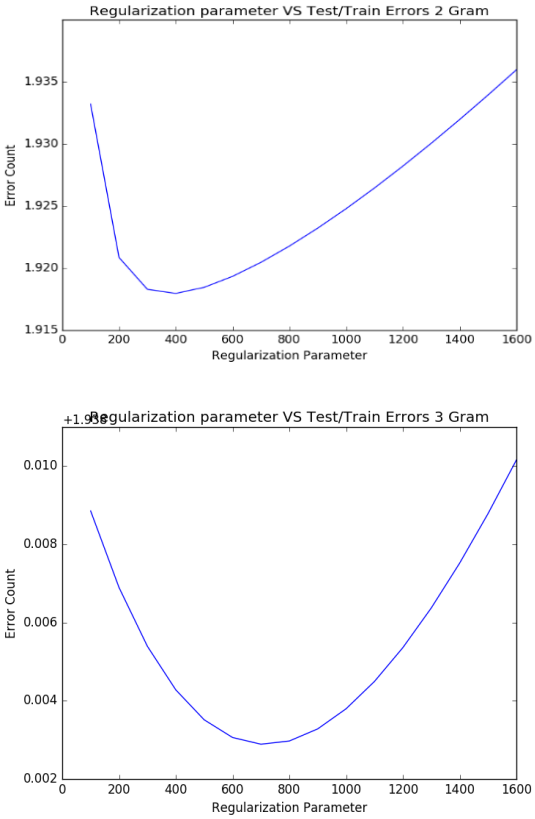
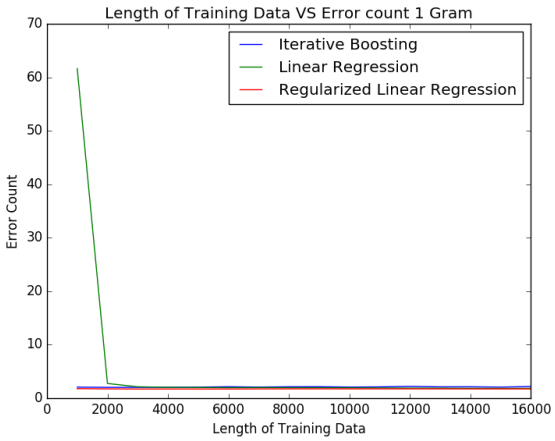


Fig. 2. *Regularization parameter vs Error*

We also compared the results of Regularized Linear regression, Iterative Boosting and Non – Regularized Linear Regression for different Lengths of training data on 1-Gram, 2-Gram and 3-Gram datasets. The plots are shown below.



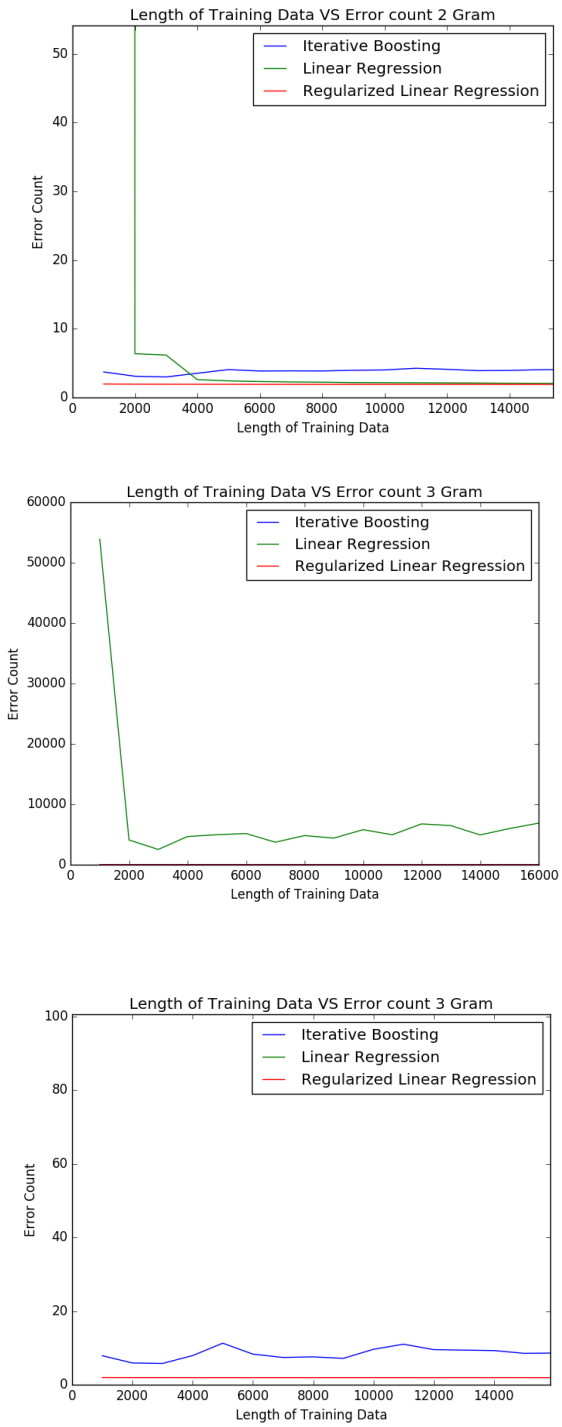


Fig. 3. *Length of training data vs error*

IV. Personal Contribution

I primarily worked on developing the Iterative Boosting Tree Data Structure. I also worked on developing the Regularized Linear regression model and deducing the value of the regularization parameter (λ) that minimized the test error. I also contributed for the data pre-processing phase by building the De-normalized dataset.

V. Conclusions

This project helped me learn various aspects of Machine Learning. I learned how various Machine Learning algorithms are implemented and the different challenges involved while selecting a model. I understood how regularization helps us avoid overfitting. I realized how regularization parameter could affect the test error and learned how to find the regularization parameter that would minimize test error. This project also gave me an opportunity to learn about how regression can be emulated through recursive classification. I also learnt about bag-of-words model. I understood how removing stop-words and reducing dimensionality can make the model efficient. I also learned how the different values of 'n' in n-grams can affect the error rate.

VI. Team Members

1.) Ankur Bhardwaj
2.) Bharath Kumar Suresh
3.) Deepak
4.) N V K Chaitanya Jetti
5.) Sahil Rajesh Dhayalkar
6.) Tejinder Singh Kang

References

[1] https://en.wikipedia.org/wiki/Regression_analysis

[2] https://en.wikipedia.org/wiki/Statistical_classification

[3] https://www.kaggle.com/c/yelp-recruiting

[4] Seiffert, Chris, et al. "Resampling or reweighting: a comparison of boosting implementations." 2008 20th IEEE International Conference on Tools with Artificial Intelligence. Vol. 1. IEEE, 2008