STAT 545: Final Project Report

Jingjing Guo
In Collaboration with Yue Zhou

Dec 15, 2017

Contents

1 Problem Overview

In this project, we conducted a sentiment study of an Amazon fine food review dataset from Kaggle. The dataset consists of 568,454 food reviews Amazon users left up to October 2012[1]. TheseCloud reviews consist of textual data written by amazon users to express either positive or negative feedback to various products. An word cloud is shown in Figure 1 to illustrate contents of the review.

Figure 1: Word Cloud of the Review Dataset

The objective of this project is two fold. First, implementing the random forest model in R. Secondly, use this method to develop a predictive classifier of the review data as positive or negative.

2 Overview of Random Forest

Random forests are an ensemble learning method for classification and other tasks. It uses a multitude of decision trees for training and learning and outputting the class that based on the statistical results from all decision trees. Random decision forests more effectively explores the model space than a single decision tree and therefore can overcome the over fitting issue of a decision tree, and reduces variance of the prediction.

2.1 Decision Trees

Decision tree is a simple and widely used method for classification. In the learning or training process, decision tree is built by recursively select the best feature as its node attribute. We use the CART algorithm which uses a Gini-index as its evaluation criterion.
Given the training set S, the target attribute (positive review or negative review), Gini index of S is defined as:

$$Gini(S) = 1 - \Sigma_x p_i^2$$

And the Gini Gain splitting S by the value of feature A is:

$$Gini(S, A) = Entropy(S) - \Sigma_{v \in values(A)} \frac{|S_A|}{|A|} Entropy(S_A)$$

where,

$$H(X) = -\sum_x p(x) log_2 p(x)$$

This given us how much uncertain we reduced by splitting the data according to feature A.
The model space of a decision tree includes both its structure and attributes at each node. In the learning process, decision trees can be developed by selecting the best attributes with minimal Gini gain and recursively grow until a predetermined constraint is met.

In the prediction with new data, each data point will route according to the node attributes of the tree and their corresponding values. The leaf value at the end of the route specifies the classification of the new data.

2.2 Bootstrap Aggregation

Random forests are a variant that aims to improve on bagged Decision Trees by reducing the correlation between the models. First, each tree is learned from a bootstrap sample, and second for each tree split, a random sample of k features is drawn first, and only those features are considered when selecting the best feature to split on (typically $k = \sqrt{p}$ or $k = logp$).

A single tree is highly sensitive to noise in its training. The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance [2].

3 Implementation

3.1 Data Pre-processing

The reviews are in a FastText Form with labels 1 or 2, with 1 being one-star or 2-star review and 2 being 4-star or 5-star review. An example of the data is shown in Figure 2.

To process these textual data, we first select m top most common words for all entries in dataset as features. Each entry is then converted into vectors of 0 or 1, with the 1 representing the entry containing the i-th feature word, and 0 otherwise. This way, we are able to generate a matrix of size $(m + 1) \times n$ and elements of 0 or 1.

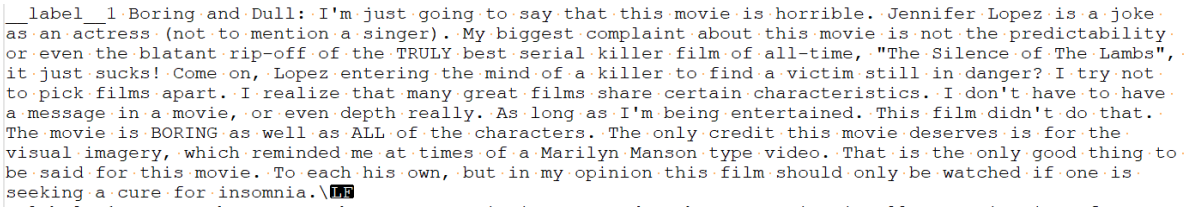


Figure 2: Review Example

3.2 Algorithms

Decision Tree Pseudo code for the algorithm we implement for decision tree is shown in Algorithm 1. Note in this algorithm, we specify depth of the tree and minimal size of data for leaves (bottom level nodes) as constraints. The more detailed code are appended to this report.

A nested list is used for storing the tree structure and attributes of each node. At each level, a node is represented with three attributes: index, left and right, where left or right themselves are may be lists of the same data structure as well.

Algorithm 1 Decision Tree

Root $\leftarrow$ *Divide data once with best feature*
procedure BUILD TREE
    Intialize variable to store tree attributes, e.g. a nested list, tree
    if *splitted table is pure* **then**
        $tree_{left} = toLeaf(data_{left} + data_{right})$
        $tree_{right} = toLeaf(data_{left} + data_{right})$ **return**
    if *Tree Level* $>= MaximumLevel$ **then**
        $tree_{left} = toLeaf(data_{left})$
        $tree_{right} = toLeaf(data_{right})$ **return**
    if *Left Data Size* $<= MininumSize$ **then**
        $tree_{left} = toLeaf(data_{left})$
    else
        Left Data Size $<= MininumSize$
        Recurse
    if *Right Data Size* $<= MininumSize$ **then**
        $tree_{right} = toLeaf(data_{right})$
    else
        right Data Size $<= MininumSize$
        Recurse
Root $\leftarrow BuildTree(Root, Maximum_Level, Min_Size, 1)$

Random Forest: The random forest algorithm use decision tree procedure, and randomly selected k features to create p trees.

Algorithm 2 Random Forest

$k \leftarrow \sqrt{m_{features}}$
procedure RANDOM FOREST
    Initialize variable to store trees
    for $i \in 1$ to number of trees **do**
        Sample k features from m features
        Recreate Training Data
        Use new Training data to build a new tree
        Attach tree to Trees

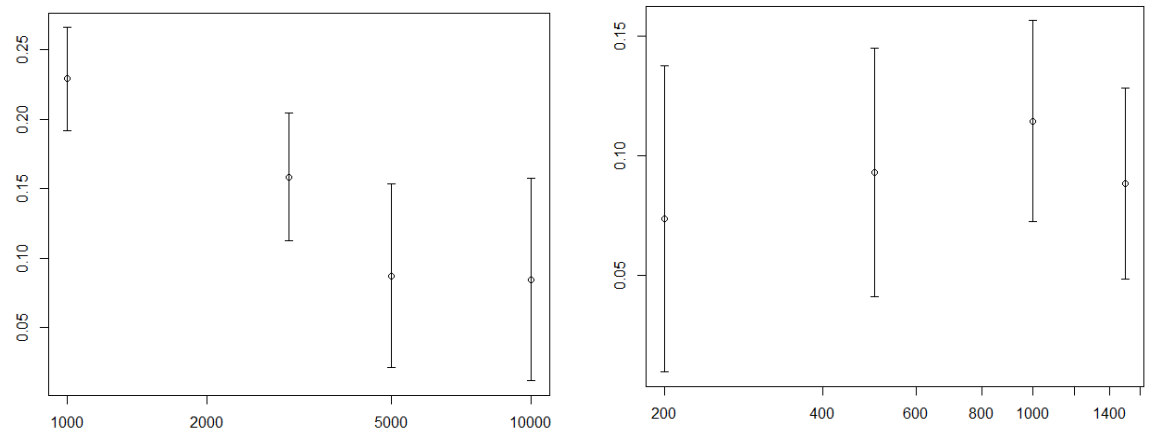
4 Results and Analysis

In studying the model, we varied four parameters: training data size, number of features, number of trees and maximal depth of tree. The results are shown in Figures 3 a)-d). The values of other variables are listed in Table 1. In all cases, we use 1000 examples for testing, and each case is run 10 times to estimate model variance. The test datasets are new data that do not overlap with the training data.

Table 1: Parameter Setting for Implementation Cases

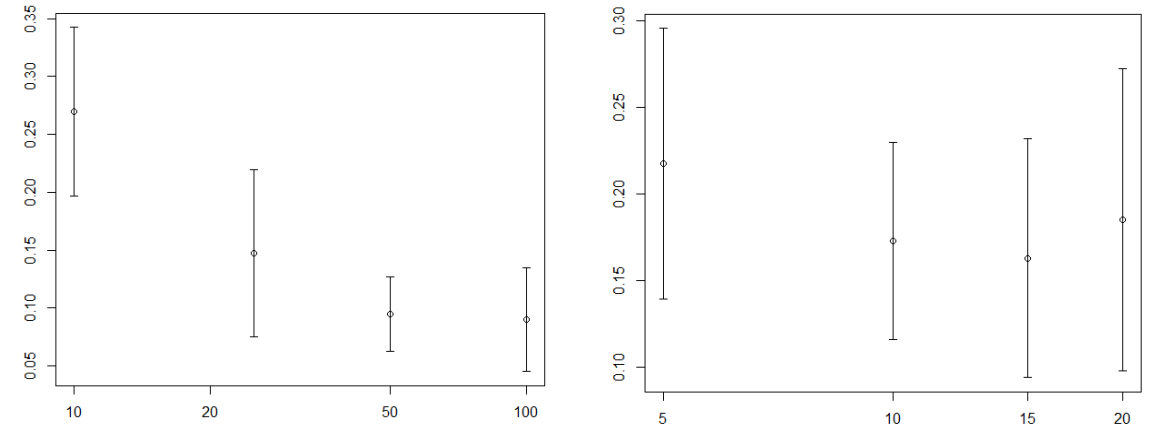
Figure Index	Training Data Size	Number of Features	Number of Trees	Maximum Depth
3-a)	Varying	1000	50	15
3-b)	5000	Varying	50	15
3-c)	5000	1000	Varying	15
3-d)	5000	1000	50	Varying

The performance metric we use to evaluate each model is prediction error rate, i.e. the percentage of erroneous prediction for the new dataset.



a) Training Size

b) Number of Features



c) Number of Trees

d) Maximal Depth

Figure 3: Plots of Prediction Error Rate for Varying Training Size, Number of Features, Number of Trees and Max Depth

From this result, we can see that with the increase of training size, the error rate decreases as expected. This is because larger data contains more information about the generative model and the trained model better approximate the true model and thus the prediction error rate for the new data set will decrease.

For the number of features, the performance did not seem to improve as the number of features increase, this is likely due to the fixed value of k which randomly selects k features from the m total of features and thus the feature space for the individual decision tree remain the same size. Normally, as the feature space increase, accuracy of the model should also increase. It is possible this is an implementation error in the random forest.

For the number of trees, as the number of trees increase, the trained model also seem to be more accurate. The decreasing rate however plateaus eventually. With the dataset size of 5000, this implies inadequacy of data with models that have larger model space. Comparing the cases when the number of trees is 50 and the number of trees is 100, the results show higher variance for 100

trees.

For the maximal tree depth of varying values, we also see an initial decrease of prediction error, but the value increases after 15 levels. The variance for 15 and 20 levels also seems excessive. This can also be due to over-fitting. It would be interesting to use a larger training dataset and examine the variance for these depth settings.

4.1 Reflection and Concluding Remarks

For this project, we intended to study and implement Random Forest models to an Amazon food review dataset for sentiment analysis. We were able to complete the implementation in R and develop random forest models. The original plan was to use most or all of the 500 MB data for training the model. Due to limited resources and extended implementation time, we were unable to accomplish the second objective.

We instead sampled about 10K entries of reviews and used that to explore the performance of the method. As was discussed in the preceding section, we looked at varying training sizes, feature sizes, number of trees, and levels of trees for the sampled data. Some of the trends are as expected, but we also observed trends I believe are inconsistent with characteristics of the random forest model, e.g. increased variance with lager model space, and decreased performance with increased number of features.

There are a few challenges I want to highlight. First, both team members are new to the R programming language and therefore, some unexpected features of the language, e.g. matrix assignment and manipulations introduced code errors and took us a long time to debug. Second, limited computing power: processing the original dataset exceeded both team member's laptop computing powers, and we were not able to find alternative solutions within the time frame of this project. Finally, there are some theories of random forest models that yet to be studied in depth, the results and analysis in Section 4 require further experiments and explanations.

Finally, through this process, I learned to implement the random forest model in R and develop predictive models for textual datasets such as online reviews. I also learned numerically how the tree depth, number of trees, etc. affect the performance of such models.

References

[1] Andy Liaw, and Matthew Wiener. *Classification and regression by random forest.* R News.

[2] J. McAuley and J. Leskovec. 2013. *From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews*

[3] Polikar, R., *Ensemble based systems in decision making.* IEEE Circuits and Systems Magazine, 2006. 6(3).

[4] Friedman, T.H.R.T.J, *he Elements of Statistical Learning.* 2008.