Predicting NSF Award Money from Abstracts

Kyle Brogle Sean Ma Laura Stelzner
Stanford University

December 14, 2012

Abstract

The NSF receives tens of thousands of proposals requesting funding each year. This leads to a significant amount of time being spent by humans reviewing these proposals and deciding what amount to award. In order to help reduce this time frame, we explored different supervised machine learning techniques to see if any could accurately classify the amount of money that should be awarded based on past abstracts and amount award. With an accurate classifier, the process for human reviewers could be streamlined to find the promising projects first and prioritize giving them funding. Our results showed that multinomial Naive Bayes was the best model to use out of those tested, with an accuracy of 33 percent.

1 Introduction

The National Science Foundation is the gatekeeper of billions of dollars used for scientific research in the United States. This funding is critical to making advancements in science, mathematics, and other related fields such as medicine. In 2012 the NSF received 7.033 billion dollars of funding [3], most of which will be allocated to distribution of research awards to researchers. As mentioned from the NSF funding description page, "NSF receives approximately 40,000 proposals each year for research, education and training projects, of which approximately 11,000 are funded [2]". We used machine learning techniques to classify the amount of funding a NSF proposal would receive based on past funding allocations. Reviewing 40,000 proposals by hand takes a lot of time. Initial data from machine learning could help speed the process by helping determine which proposals are most promising given what theyve awarded in the past, and the classifier can continually be adjusted over the years as more proposals are available as training data.

2 Data

For our data we have 129,079 proposals submitted to the NSF between 1990 and 2003 courtesy of University of California, Irvine's Machine Learning Repository[1]. We began by giving each proposal a unique identifier. The proposals were all parsed into a bag-of-words format in order to transform them into feature vectors to be classified.

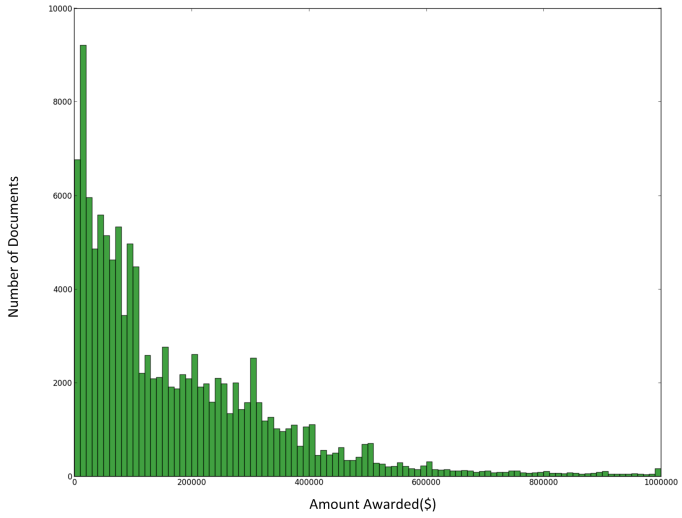

Figure 1: A histogram detailing an approximate distribution of the data

We removed stop words from our vocabulary, as they are too common in occurrence to aid in classification. Our dictionary of words produced was of length 30,799. We also extracted the awarded amounts from the proposals themselves. An initial histogram of the training data, as shown in Figure 1, suggests that 21 class labels would nicely divide the proposals: twenty buckets in increments of $50,000 up to $1 million, and one bucket for awards greater than $1 million.

3 Method and Results

After our data was processed into a usable format, we decided to start by training a multi-class SVM with linear kernel to classify our data. Although we have approximately 130,000 training examples, we trained on smaller subsets. In order to determine whether the size of the training set impacted our classification error, we decided to train the SVM with training set sizes 1000, 2000, 5000, and 10000. These sets were built by randomly choosing documents from our dataset, to try to avoid training sets that are similar in topic or year requested. In each training scenario, we used 5-fold cross-validation in order to test the accuracy of our model.

When analyzing the results, we found that the linear SVM suffers from extremely high variance, as our training error is near zero (refer to the first subplot in Figure 2). In order to try and remedy this, we decided to simplify our model by reducing the number of features. In order to do this, we reduced the vocabulary size by using the Porter2 stemming algorithm to stem all the words in our vocabulary. This reduced vocabulary size from 30,799 to 19,910. This resulted in less overfitting of our model, but the high variance persisted (refer to the second subplot of Figure 2). Before spending time training on larger training sets, we decided to investigate the performance of some other classifiers.

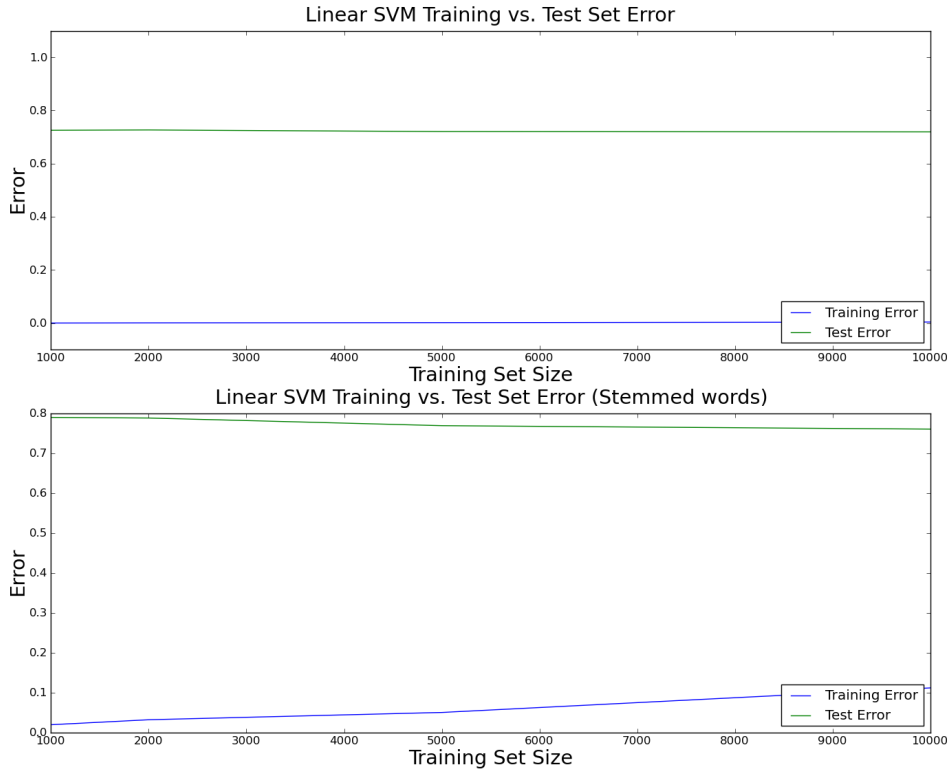

Figure 2: Training vs. Test Error for Linear SVM

The additional classifiers we chose to test were multinomial Naive Bayes and Stochastic Gradient Descent. We then ran these classifiers on both the stemmed and non-stemmed training sets of sizes 1000, 2000, 5000, and 10000. The Naive Bayes classifier seemed to perform the best of all our classifiers, exhibiting classification accuracy of 33 percent. With 21 classes, the trivial classifier that uniformly selects a class for each document would achieve 4.8 percent accuracy if the distribution of classes over documents was uniform. It would actually achieve less in this application, since the documents are concentrated in categories with lower award amounts. We found that the training error was also high (around 40 percent), indicating high bias. Because of the high bias, the reduction of the feature space provided by the stemmed dataset caused Naive Bayes to perform extremely poorly (refer to Figure 3 for the results).

On the other hand, when using a stochastic gradient descent classifier, we found classification accuracy to be around 28 percent (refer to Figure 3). Like the linear SVM, this classifier had very small training error, due to overfitting the model to the training set. When training on the stemmed dataset, the reduced feature set had a more significant impact here than with the linear SVM. With the reduced feature set, the classifier no longer overfit the model to the training data, and started to exhibit

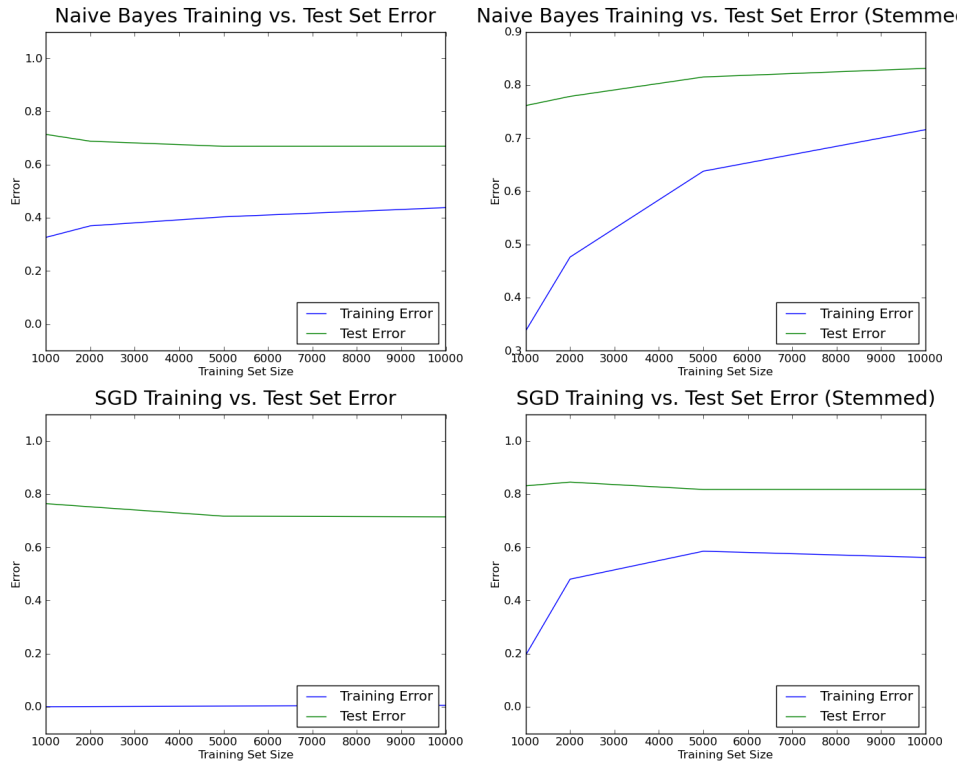

Figure 3: Training vs. Test Error for Naive Bayes and Stochastic Gradient Descent

signs of high bias. The accuracy of our classifications decreased when running on the stemmed data in this scenario.

4 Conclusions

From our analysis, we see that the best performance was realized with the Naive Bayes classifier on the un-stemmed dataset. Unlike the linear SVM and stochastic gradient descent classifiers, Naive Bayes did not overfit our model to the training set. The high training error in Naive Bayes indicates high bias, which suggests that classification error can be lowered with the addition of more features to the model.

5 Future Work

In order to further improve the accuracy of the Naive Bayes classifier, we plan to address the high bias by adding more features. One specific feature that we feel will improve the classification accuracy greatly is the year of submission. This should really be considered in our model, as popular research areas, especially in the field of computer science, tend to change over time, with new topics frequently emerging. This feature will be weighed much higher than the bag of words features, as we suspect it

plays a significant role in funding decisions. Other features that were suggested to us during poster presentations were the sponsoring university and the department, which we could extract from either the principal investigators mailing address or the NSF program ID.

It is also possible to improve upon the model itself by changing the buckets to better fit the distribution outlined in the histogram from Figure 1. If we were to choose buckets so that there are approximately equal number of proposals in each bucket, it could make our results more accurate, since there will be an even distribution of samples per class. One unfortunate problem with this change is that the actual range of how much a paper could earn is no longer a fixed range, so the information may not be as useful. It is also possible change our model from a multi-class classification problem to a regression problem, so that instead of discretizing the amount of money into buckets, we attempt to give a best guess as to exactly what the paper will earn. However, this may not be as feasible with such a large feature set, as our vocabulary exceeds 30,000 words.

We also wish to extend the dataset by contributing proposals up to the current year. We are currently writing a script to parse the award format provided by the NSF awards website in order to further look into how well this problem can actually be solved with machine learning.

6 Citations

[1] Frank, A. & Asuncion, A. (2010). *UCI Machine Learning Repository.* Irvine, CA: University of California, School of Information and Computer Science. Retrieved Oct 15, 2012, http://archive.ics.uci.edu/ml.

[2] National Science Foundation. About Funding. In *National Science Foundation Where Discoveries Begin.* Retrieved Nov 15, 2012, from http://www.nsf.gov/funding/aboutfunding.jsp

[3] National Science Foundation. FY 2012 Appropriations Signed Into Law–NSF to Receive $7.033 Billion. In *National Science Foundation Where Discoveries Begin.* Retrieved Nov 15, 2012, http://www.nsf.gov/about/congress/112/highlights/cu11_1118.js