Predicting Bill Passage

Kyle Gulshen, Noah Makow, Pablo Hernandez
Stanford University - CS 229

***Abstract*—In the legislative process, vast amounts of time and effort are spent on working to understand how various congress-people will vote on a bill. Our research has sought to build a system that will model how each congressperson will react to a newly introduced bill. Such a model might help congress-people to get a sense of the reaction a bill they are seeking to propose would provoke, and provide the general public with a sense of how likely a given bill in Congress is to pass and become a law. Our algorithm uses a training set where each previous vote from a congressperson becomes a training example, properties of the bill and voting circumstances are features, and the outcome is how the congressperson actually voted on the bill. Our research has explored the effectiveness of different machine learning algorithms (GLMs and SVMs) on modelling this data. Next steps will include building models for each congressperson, and aggregating our predictions over the entire Congress to determine whether or not a bill will ultimately pass.**

***Keywords*—Bills, Congress, Machine Learning, Logistic Regression, SVM, Naive Bayes, DW1, Gaussian, Kernel**

I. Introduction

In the legislative process, vast amounts of time and effort are spent on working to understand how various congress-people will vote on a bill. Our group sought to explore what ultimately goes into having an individual congress-person vote in favor of or against a certain bill. The thought process behind taking this approach revolved around the idea of lobbying individual representatives. Equipped with a model of how representatives might vote, lobbyists would know whom to target and how to allocate resources in order to gain support for a particular bill. In addition to lobbying, legislators can get a sense of which representatives will be in favor of a bill they are crafting, and tailor their efforts to reflect these results.

Our approach to this problem involved training a model for each sitting representative. Merging vote history data from GovTrack and bill-specific features from Congressional Bills, we were able to train our models based on a representative's voting history. Given a new bill as input, the model would output whether or not the given representative would vote in favor of or against the bill. For the purposes of this paper, we have trained models for three different representatives. In practice, a model would be trained for every single sitting representative, and it would be possible to aggregate results across the entire congress to predict whether or not a bill would pass and with what margin.

II. Related Work

Our approach to this problem has not been extensively researched in the past. Most research in this space has been focused on leveraging bill text and making a single prediction as to whether or not a bill will pass or fail. Yano, Smith, and Wilkerson (2012) demonstrate this approach, augmenting bill features related to bill sponsorship, committee membership, state, and time of introduction with the text of the bill. Having trained on bills in the 103-110th Congresses, their model was able to attain a 10% error rate on bills in the 111th Congress. Smith (2010) generalizes the challenge addressed in his 2012 paper: Given a body of text T pertinent to a social phenomenon, make a concrete prediction about a measurement M of that phenomenon, obtainable only in the future, that rival the best-known methods for forecasting M. This approach is starkly different from the approach we took, which focuses on making predictions for individual representatives. We believe that our approach to the problem provides finer granularity and, as mentioned above, gives valuable insight for lobbyists and others interested in swaying or learning about the reactions of individual representatives.

Smith, Baek, et al. (2012) take another novel approach to the problem: utilizing campaign finance data as input features for their learning. Data was collected from publicly available sources on donations from corporations and individuals to politicians, the stated opinions of corporations and other organizations on legislative actions, and the records of how members of Congress voted on these measures. Smith, Baek, et al. (2012) take a similar approach of predicting votes for each congressperson. The accuracy of their models varies significantly based on the approach. In the end, they conclude that there is no strong evidence that politicians vote based solely on the financial contributions they receive from certain industries. However, there does exist a strong correlation between money flow and political party that is reflected in the voting process where an individaul politican is very likely to vote along his or her party line.

Poole, Rosenthal (1989, 2001) provide a description of their D-NOMINATE score, which is a critical feature in our training set. The D-NOMINATE score is a multidimensional classification that attempts to ideologically cluster representatives based on their voting history and other features. The first dimension provides a rough approximation of how liberal or conservative a representative is. In this paper, this score is referred to as DW1, where DW1 $\in [-1, 1]$ and -1 corresponds to liberal, whereas 1 corresponds to conservative.

III. Dataset and Features

Our dataset comes from two different sources: Congressional Bills had the data that we used for the features on our bills starting at the 93rd Congress. GovTrack

helped us to collect information on how individual congress-people voted while they were active in legislation. The three congresspeople we collected data for were: Roy Blunt - a Republican Senator from Missouri, Nancy Pelosi - a Democratic Representative from California (and former Speaker of the House), and Paul Ryan - current Speaker of the House for the Republican Party. For each individual, we have around 2,000 examples that served as our dataset.

Significant preprocessing had to be done. Using SQL, the two data sets had to be merged, according to the respective Bill ID, Congress Number, and house it was proposed in. Data examples missing key features were discarded. The information needed for the merge did not match correctly, so the congress-people information had to be processed in a manner such that it could be compared for a join. The final set of features we examined were: Congress, Bill Number, Chamber, Commemoratory Bill Indicator, Major, Minor, Private Bill Indicator, Age, Delegate, District, DW1, DW2, FrstConH, FrstConS, Gender, LeadCham, Majority, Mref, Party, and State. Ultimately, our feature set was narrowed down to the bill author's DW1-NOMINATE score, along with each bill's Major Topic and Minor Topic. The DW1-NOMINATE score is a measure of how liberal or conservative a particular author is - on a score ranging from (-1, 1). Our dataset was all included in tables, for example:

Congress	Bill ID	HR/S	DW1	Major	Minor
112	1	0	0.531	16	1600
108	1018	0	-0.322	20	2008
93	135	0	0.195	3	321
94	1834	1	0.144	2	200
94	2486	1	-0.194	21	2101
106	1278	1	0.356	8	802

IV. Methods

We investigated fitting our data using 3 different approaches: (A) logistic regression, (B) support vector machines with a Gaussian kernel, and (C) naive bayes. All the aforementioned algorithms are suited to classification problems, but there are formulations and assumptions specific to each. Methods are described below.

A. Logistic Regression

We first explored using logistic regression to model our data. Here, we aim to minimize the following cost function:

$$J_\theta = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{1}$$

Where $h_\theta = \theta^T x$, and the θ_i's are the weights parametrizing the space of linear function from X to Y. In order to learn the parameters θ, we minimize the cost function $J(\theta)$ using stochastic gradient descent. In this algorithm, we repeatedly run through our training set, and update our parameters θ according to the gradient of the error with respect to the current training point under observation, i.e:

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} \tag{2}$$

B. SVM with a Gaussian Kernel

Our data was also learned on using a support vector machine with a Gaussian Kernal as our model. Here the objective is to minimize:

$$min_{\gamma,w,b}\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \tag{3}$$

Such that:

$$y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, ..., m \tag{4}$$

$$\xi_i \geq 0, i = 1, ..., m \tag{5}$$

Our Kernel selected in this algorithm is in the form of:

$$K(x, z) = exp(-\frac{\|x - z\|^2}{2\sigma^2}) \tag{6}$$

Providing us with a value close to 1 when x and z approximately equal, and 0 as their difference grows. The goal of the SVM algorithm is to find the maximum margin separating hyperplane between the data, and by examining/solving the dual of this optimization problem, we are then able to learn efficiently in high dimensional spaces.

C. Naive Bayes

For our naive bayes approach, we seek to now use a generative model in order to be able to accurately fit our data. Here, the goal is to model the class prior and the conditional feature prior - $p(y)$ and $p(x_x|y)$, respectively - using a multinomial, multivariate distribution. This is done since it is for predicting a model who's observations are categorical. Thus the probability of a congressperson voting yes on a bill becomes:

$$p(y)\prod_{i=1}^{m} p(x_i \mid y) \tag{7}$$

where $x_i, ..., x_m$ are the features of the dataset corresponding to the voting decision tethered to class y, and where our model is parametrized by:

$$\phi_{i|y} = p(x_i \mid y) \tag{8}$$

$$\phi_y = p(y) \tag{9}$$

Here we also institute Laplace smoothing to shift some of the probabilities towards potential test sets that were not seen in the training set. To allow for this, the estimates for our parameter (for predicting "1" as the output) now becomes:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m} 1\{x_j^{(i)} = 1^{y(i)} = 1\} + 1}{\sum_{i=1}^{m}\{y^{(i)} + 2\}} \tag{10}$$

V. Experiments/Results/Discussion

We ran into interesting challenges along the way when trying to model this problem. The following sections address these issues.

A. Establishing a Baseline

To properly measure our performance it was necessary to establish an appropriate baseline to measure against. One approach, given our knowledge of the partisan nature of congress, is to measure against a "party line" baseline in which the predicted vote is 'yes' if and only if the congressperson proposing the bill is in the same party as the congressperson who is voting. This approach, however, misclassified around 35% of votes. In fact, a more accurate baseline model is simply to predict 'yes' for all votes, resulting in around 15-25% misclassification, depending on the congressperson. This model is superior likely due to the structure of the voting process in congress–most bills that make it onto the floor are favored by the majority, and there are a number of bills that are likely to receive a near-unanimous vote.

B. Unbalanced Classes

Our data suffered from unbalanced classes. The following table demonstrates the disparity between the number of yeas and nays:

	Blunt	Pelosi	Ryan
Y	1539	1509	1635
N	250	451	276
%N	13.9	23.0	14.4

Undersampling our data–that is, excluding some 'yes' votes from the data until there was a roughly equal number of yes and no votes–failed to improve anything; the model still almost always predicted yes for the test set (meanwhile maintaining near 0% training error for the SVM model). We then tried to change the cost associated with false positives. By weighting false positive cost at 1.7 times or more of the cost of false negatives, the model would switch from voting all yes to all no on the test set. In fact, manipulation of this parameter only resulted in an oscillation between these two extremes. [Note: this approach was also tried on the original unbalanced data, but no weighting appeared able to change the voting behavior from all yes to all no.]

We also went the route of oversampling, since our research resulted in information that oversampling could increase the minority class recognition with sacrificing less of the majority class recognition rate. Adding new examples would increase the time to learn over the training set, but this proved to be negligible in practice. Attempts to oversample also failed to improve the results of our models.

C. Results

The following are the confusion matrices reported when running our three methods on Nancy Pelosi's dataset (training on 70%, testing on remaining 30%):

Logistic Regression

	p'	n'
p	434	23
n	111	21

SVM

	p'	n'
p	457	0
n	131	1

Naive Bayes

	p'	n'
p	381	76
n	71	61

Below are precision & accuracy results across all algorithms, when testing on 30% of each dataset:

Logistic Regression

Person	Precision	Accuracy	F1-score
Blunt	1.0	.8701	0.9305
Pelosi	0.9538	0.7963	0.8680
Ryan	0.9979	0.8394	0.9118

SVM

Person	Precision	Accuracy	F1-score
Blunt	1.0	0.8701	0.9305
Pelosi	0.9978	0.7772	0.8738
Ryan	1.0	0.8380	0.9118

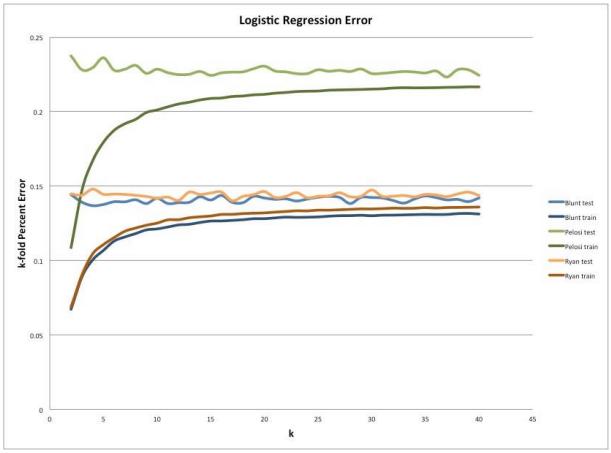
Naive Bayes

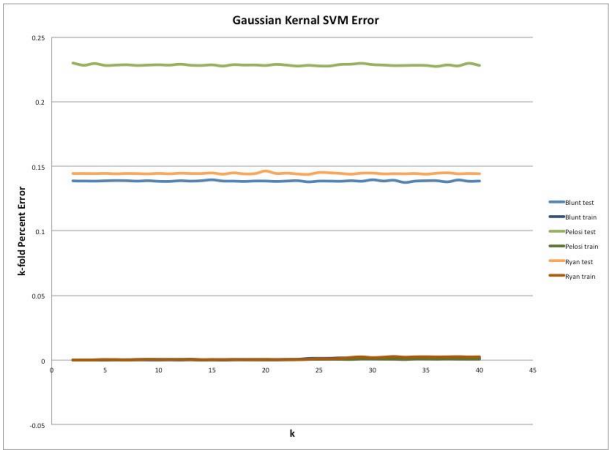
Person	Precision	Accuracy	F1-score
Blunt	0.9912	0.8735	0.9286
Pelosi	0.8620	0.8429	0.8523
Ryan	0.9812	0.8486	0.9101

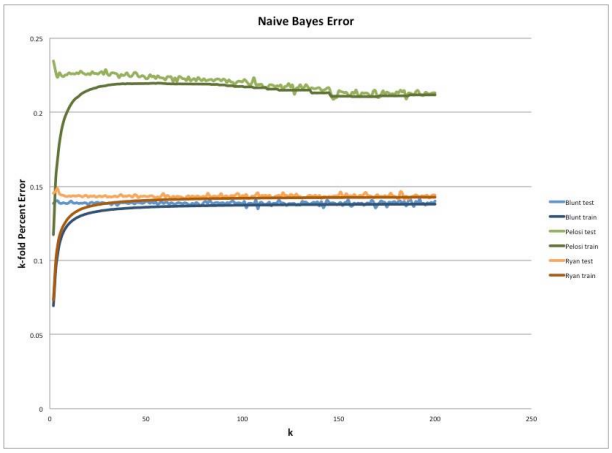
For logistic regression, the average errors were produced when running k -fold cross validation with $2 \leq k \leq 40$, when done across all congresspeople:

Person	Train	Test
Blunt	12.36%	14.08%
Pelosi	20.42%	22.75%
Ryan	12.75%	14.40%

Below are the plots comparing test vs. train error across the congresspeople, when running k-fold cross validation with $2 \leq k \leq 40$.







D. Discussion

Our first approach was to quickly run logistic regression on each individual congressperson's dataset. We then measured the training and test error on a 70/30 train/test breakdown of the data. We determined that the training error was unacceptably high and so sought out a better model to reduce the high bias we observed.

Having implemented a general GLM, we tried a variety of different link functions, but none outperformed logistic regression. We then switched to a SVM model. A linear kernel also failed to outperform logistic regression. An SVM with Gaussian kernel was finally able to produce less training error then logistic regression and in fact produced training errors close to 0%. With our test error still high, we concluded that the SVM was subject to high variance.

While this was encouraging, the model nearly always predicted a yes vote on the test data. Realizing this was likely due to unbalanced classes, we tried a variety of methods to improve our results. As a result, we experimented with both oversampling and undersampling. Duplicating negative examples to balance (oversampling) the classes did not improve our models. Similarly, undersampling increased training and test error. We also experimented with different threshold values. In other words, our decision boundary hyperplane became $\theta^T x = \tau$ where we were able to vary $\tau \in [0, 1]$.

While altering the threshold gave us a better balance of misclassified examples, it increased the overall error and we decided to not include this in our final models. We believe that altering the threshold is helpful, but in doing so exposed other issues with our model, such as that our feature set may not actually be very good. Lastly, we also tried different cost functions with the SVM to weight the cost of false positives versus false negatives. As discussed above, for very small changes in these costs our model oscillated between always outputting 1 or always outputting 0.

VI. Conclusion/Future Work

Similar approaches to tackling a problem like this revolved around analyzing bill text. Our approach focused on features pulled outside of the bill itself – and produced comparable results. Looking at the graphs across all our models, it seems that both the logistic and the naive bayes model are suffering from high bias, a problem ultimately tracing back to feature selection. The SVM algorithm suffers from high test variance. Naive Bayes Classification was our best performing algorithm.

Future work would focus mainly on manipulating the dataset. Most of the issues that our team ran into revolved around not having sufficient data (on the "no" votes, to be able to learn on). With more time, we would analyze the feature selection process in finer detail, and potentially explore different sources of data to tackle the same problem. We would also seek to integrate existing research into our approach, such as including bill text analysis and financial data.

REFERENCES

[1] Yano, T., Smith, N. A., Wilkerson, J. D. (2012, June). Textual predictors of bill survival in congressional committees. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 793-802). Association for Computational Linguistics.

[2] Smith, S., Baek, J. Y., Kang, Z., Song, D., El Ghaoui, L., Frank, M. (2012, December). Predicting Congressional Votes Based on Campaign Finance Data. In Machine Learning and Applications (ICMLA), 2012 11th International Conference on (Vol. 1, pp. 640-645). IEEE.

[3] Smith, N. A. (2010). Text-driven forecasting.

[4] Poole, K. T., Rosenthal, H. (2000). Congress: A political-economic history of roll call voting. Oxford University Press. Chicago

[5] Poole, K. T., Rosenthal, H. (2001). D-nominate after 10 years: A comparative update to congress: A political-economic history of roll-call voting. Legislative Studies Quarterly, 5-29.

[6] "GovTrack API Documentation." GovTrack.us. Web. 1 Dec. 2015. https://www.govtrack.us/developers/api .

[7] "Download Congressional Bills Project Data." Congressional Bills Project: Download. Web. 1 Dec. 2015. http://www.congressionalbills.org/download.html .

[8] "Efficient Resampling Methods for Training Support Vector Machines with Imbalanced Datasets." Web. 1 Dec. 2015. http://www.cs.ox.ac.uk/people/vasile.palade/papers/Resampling_methods_SVM.pdf.