Classification and Regression Approaches to Predicting United States Senate Elections

Rohan Sampath, Yue Teng

Abstract

The United States Senate is arguably the finest democratic institution for debate and deliberation in the world. It also provides a fascinating opportunity to understand the complex dynamics that go into determining the outcome of Senate elections, using Machine Learning.

Motivation

Are elections decided even before they begin? Can political fundamentals predict elections regardless of candidates/campaigns?

Our goal is to get a bird's eye, forward-looking view of Senate elections using data available well in advance. We believe that if we reliably predict Senate elections well before they happen, that has several significant implications for various stakeholders, since individual senators wield a tremendous amount of legislative power.

Introduction

We use:

- A modified version of the LMS algorithm, called a discount-weighted least-mean squares algorithm to predict the margin of victory of Senate elections.
- An ordinary Support Vector Machine classifier to predict the outcome of Senate elections.
- Random Forest to also predict the outcome of Senate elections.

Data

Our data set consists of all biennial Senate elections that were held from 1998 to 2014; this data is publicly available.

Preprocessing: We preprocess the data to weed out elections where:

- There wasn't exactly one Republican and exactly one Democratic candidate.
- A third Party candidate either won the election or distorted the election by winning more than 20% of the vote (i.e. a third party candidate was a significant player).

After preprocessing, we are left with 273 data points. (There were 300 regularly scheduled Senate elections in the period 1998-2014, of which 27 were eliminated in preprocessing.)

The fundamental challenge we face is one of limited data. Senate elections, by their very nature, are limited – only around 33 happen every two years. Therefore, we had to keep in mind that an inescapable part of this project was having limited data.

Features

We use a feature vector of 71 features. These 71 features include original sourced features (such as margin of victory, unemployment rate etc.) and derived features (such as change in unemployment rate over a period of time).

The features are described below:

- Margin of victory in the Senate election held six and twelve years previously. (Note: Senators serve six-year terms.)
- Margin of victory in the state in the last three Presidential Elections
- Presidential Approval in the State.
- Annualized Changes in the Presidential Approval in the state.
- Percent African-American population (as extrapolated from the most recent Census)
- Percent Hispanic/Latino population ((as extrapolated from the most recent Census)
- Changes in the above demographic factors over time.
- Three-month average unemployment rates in the year before the election.

- 6-month, 12-month, 18-month 24-month changes in unemployment rate[1]
- Partisan Voting Index (PVI) over the past three Presidential elections[2]
- Change in the PVI from the second-last presidential election to the last one.
- Median income in the state.
- Variation in the median income in the state.
- Indicator variable: Whether Republican candidate is the incumbent senator.
- Indicator variable: Whether Democratic candidate is the incumbent senator.
- Number of years of incumbency for the President.
- Indicator variable: Whether the election was a midterm election or not.

Convention: In all cases, a positive result for the Republican is recorded as positive, and vice-versa. Example: A reduction of the unemployment rate during a Democratic President's term is means that the feature data point is (-), since it's good for the Democrats.

Cross-Validation

We use cross-validation frequently through the project. Our perusal of literature suggested that a direct-application of k-fold cross-validation was not appropriate for time-series data – it would not be appropriate to train on 2012 data, for example, and validate on a hold-out data point that happened before 2012!

Hence, we use a modified version called *forward chaining*. For example, say we have a training set consisting of data from years 2000, 2002, 2004 and 2006; we then design the folds as follows:

- Fold 1: train [2000], hold-out validation [2002]
- Fold 2: train [2000, 2002], hold-out validation [2004]
- Fold 3: train [2000, 2002, 2004], hold-out validation [2006]

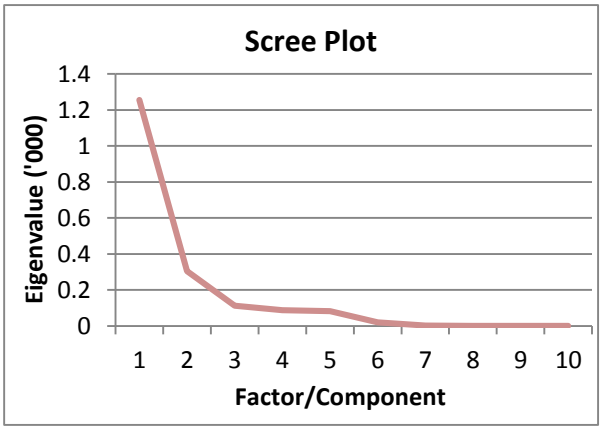
[1] A reduction is positive under incumbent Republican President, while negative under an incumbent Democratic President; vice-versa for an increase.

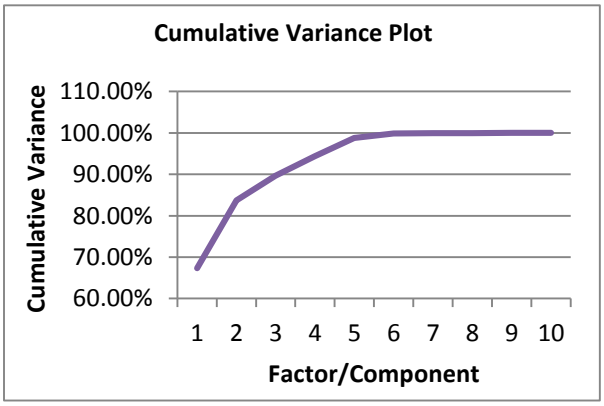
Principal Component Analysis

Motivation: *clear interdependencies* between certain variables: PVI and Previous US Presidential election result, for example.

In order to choose an appropriate k-dimension spanned by the first k principal components subspace (for $k \in (1, 10)$) and thus determine the k principal factors, we use the Scree Plot and the Cumulative Variance Plot.

(The plot below is for the 183 data points from 2002 to 2012. First 10 principal components are shown.)





[2] PVI of a state: On average, how much more Republican was the state in the last two presidential elections as compared to the nation as a whole)

Support Vector Machine (SVM) Classification

We solve 3 classification problems using standard SVM classification:

- classifying 2014 after learning on 2002-2012,
- classifying 2012 after learning on 2000-2010, and
- classifying 2010 after learning on 1998-2008.

The fundamental motivation behind SVM is carrying out binary classification in a high-dimension feature space efficiently, by using the kernel trick (i.e. by mapping input data via a non-linear function). The SVM algorithm can perform this computation efficiently because it considers a small number of training points and ignores all training points that are close (within a threshold ϵ) to the model prediction.

The primal optimization problem is given by:

$$\min \ \frac{1}{2}\|w^2\| + C\sum_{i=1}^{m}(\xi_i + \xi_i{}^*)$$

$$\text{subject to: } \begin{cases} y^{(i)} - < \mathrm{w},\ x^{(i)} > -b \le \epsilon + \xi_i \\ < \mathrm{w},\ x^{(i)} > + b - y^{(i)} \le \epsilon + \xi_i{}^* \\ \xi_i,\ \xi_i{}^* \ge 0 \end{cases}$$

The norm $\|w^2\|$ measures the flatness of the proxy, and the constraints force the model to approximate all training points within an absolute margin ϵ. ξ_i, $\xi_i{}^*$ are slack variables that allow for compliance with the ϵ margin constraints and the flatness of the proxy. C is the penalty for violating the constraints.

The corresponding dual problem is given by:

$$\max \ -\frac{1}{2}\sum_{i,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\left(< x^{(i)}, x^{(j)} >\right) - \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) - y^{(i)}\sum_{i=1}^{m}(\alpha_i - \alpha_i^*)$$

$$\text{subject to: } \begin{cases} \sum_{i=1}^{m}(\alpha_i - \alpha_i^*) = 1 \\ 0 \le \alpha_i, \alpha_i^* \le C \end{cases}$$

The dual optimization is convex and can easily be solved with optimization software. We use LIBSVM to implement SVM classification with a Gaussian kernel function.

Results for SVM Classification

(Training Data Set)

Years Trained Upon	N	N Correctly classified	Training Error
1998-2008	183	179	2.19%
2000-2010	183	180	1.64%
2002-2012	182	178	2.20%

(Test Data Set)

Year (Using Training Data from)	N	N Correctly classified	Test Error
2010 (1998-2008)	30	27	**10.00 %**
2012 (2000-2010)	30	26	**13.33%**
2014 (2002-2012)	30	27	**10.00%**

Discount-Weighted Least-Means Square Regression

Once again, we solve three regression problems for the years 2010, 2012 and 2014.

Given that the composition and voting intentions of a state evolve rapidly, we thought it would be beneficial to give less weight to earlier training data as compared to later ones.

The basic premise of this *time discount rate* algorithm, which has been adapted from Harrison and Johnston [5], is to use a 'discount factor' which conveys the rate of decay of the information content of an observation.

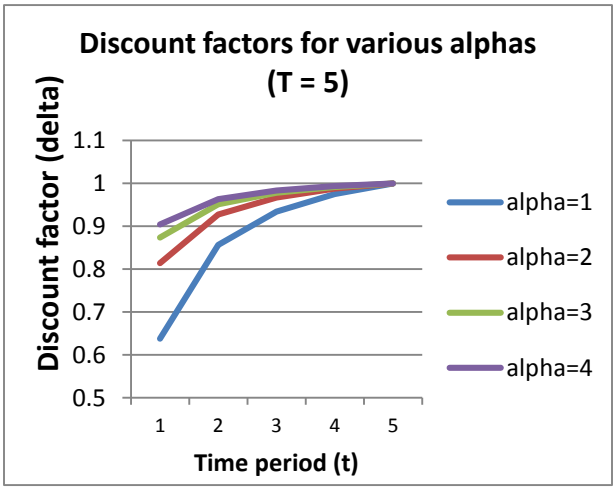
The discount-weighted LMS algorithm had a lower generalization error than a standard LMS algorithm when forward-chaining cross-validation was used.

We used a discount factor of the form:

$$\delta_t = \sqrt{\frac{2\alpha t - 1}{2\alpha t + 1}} \Bigg/ \sqrt{\frac{2\alpha T - 1}{2\alpha T + 1}}$$

Where δ_1 is the discount factor for the earliest time period and T is the number of time periods (i.e. $t = 1, \ldots, T$). α is a parameter than can be optimized. Clearly, δ_T is always = 1.

Discount factor for various alphas is shown below:



Results for Discount Weighted LMS

(Training Data Set)

Years Trained Upon	N	Mean margin of error	N Correctly classified	Training Classification Error
1998-2008	183	4.20%	179	2.19%
2000-2010	183	4.61%	180	1.64%
2002-2012	182	4.43%	178	2.20%

(Test Data Set)

Year (Using Training Data from)	N	Mean margin of error	N Correctly classified	Test Classification Error
2010 (1998-2008)	30	7.48%	27	**10.00 %**
2012 (2000-2010)	30	6.52%	26	**13.33%**
2014 (2002-2012)	30	7.10%	27	**10.00%**

Random Forests

We also implemented Random Forests classification on the original data set.

Random forests use decision trees as the basic building block to enable prediction. A decision tree uses a tree-like graph or model of decisions to split up the feature space into separate regions. Each data point falls into exactly one region, and in the case of classification, the most common class is the predicted class.

Random forests use multiple decision trees, and the reasoning behind this is to reduce the chances of overfitting to the data. Each tree is built on a separate dataset where each dataset is sampled from the original distribution. However, since we do not know, or have access to, the original distribution, we build each dataset by sampling with replacement using the original dataset. This is known as *bootstrap aggregation*, since we now have multiple decision trees which are all fit to an approximation of the original distribution. By using multiple trees we can lower the variance of the model at the cost of increasing the bias.

Although bootstrap aggregation helps to reduce the variance of the model, it does not fix an important problem which is that every tree may be highly correlated to each other. In that case, it does not matter how many trees we average our predictions over if each tree is exactly the same, since the variance of the model will not decrease at all. In order to prevent highly similar trees, we will only consider a random subset of the features at each split. Often the number of features considered, m, is much lower than p, where p is the original number of predictors.

There are two parameters to tune over in random forests: B, the number of decision trees to create, and m, the number of predictors to consider at each split. Increasing B will prevent the model from overfitting, but may also prevent accurately capturing the relationship between the training data and the output. Increasing m will increase the chances of overfitting, but may allow a better fit to the training data. Appropriate choices for B and m can be selected by using cross validation. Choices for B and m that were optimal in our three tests hovered **around $B \approx 100$ and $m \approx p/7 \approx 10$.**

Results for Random Forests

(Test Data Set)

Year (Using Training Data from)	N	N Correctly classified	Test Error
2010 (1998-2008)	30	28	**6.67 %**
2012 (2000-2010)	30	27	**10.00%**
2014 (2002-2012)	30	27	**10.00%**

Conclusions

Random Forests clearly works better than the SVM classifier while attempting binary classification with a small number of data points (and hence a high possibility of over-fitting). The average classification test error rate for Random Forests is 8.9%, while for the other two algorithms it is 11.1%.

Most importantly, we conclude that we predicted the 2010, 2012 and 2014 Senate elections with a reasonable amount of accuracy ***with data that was mostly available at least two years in advance of those elections.*** That is, except for unemployment statistics (for which we can use forecasts), we have enough data to predict the 2016 election too (we do just that in the Appendix)!

While a lot of attention is directed towards Presidential Elections, individual Senators have tremendous power over legislation. Therefore, we believe that *having a bird's eye estimation of what the Senate might shape up to be two years in the future could be very useful for a lot of stakeholders*, such as:

- Stakeholders in key bills: If Senator X loses, will the 'AJKL bill' fail in the next Congress?
- Lobbyists: Can the threat of being vulnerable help persuade Senator X to support Z?
- Speculators: Can I shape my investments with a reasonable amount of confidence in having a Republican/ Democratic Senate 2 years from now?
- Party machinery: Senator Z is vulnerable. We must begin directing resources towards his/her campaign IMMEDIATELY.

And therein lies the practical utility of our exercise. We're excited that we were able to get reasonably good results with publicly available data and machine-learning approaches – clearly, elections can be predictable! We're eager to build on some of these approaches, especially Random Forest, and explore new techniques as well.

Data Sources

All data is publicly available:

- Election Results are sourced from the Federal Election Commission website (www.fec.gov)
- Unemployment Rate Statistics are sourced from the Bureau of Labor Statistics (www.bls.gov)
- Demographic Statistics are sourced from the United States Census bureau (www.census.gov)

References

[1] Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. Advances in neural information processing systems, 9, 155-161.

[2] Basak, D., Pal, S., & Patranabis, D. C. (2007). Support vector regression. Neural Information Processing-Letters and Reviews, 11(10), 203-224.

[3] J. Smolatand Bernhard Scholkof. A tutorial on support vector regression. 2004

[4] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. New York: Springer Series in Statistics, 2001.

[5] Harrison, P. J., and F. R. Johnston. "Discount weighted regression." Journal of the Operational Research Society (1984): 923-932.

Appendix: Our Prediction for the 2016 Senate Elections

The Republicans lose two seats, but hold on to the Senate, 52-48!