ACTIVE POLLING: IMPROVING PRESIDENTIAL ELECTION PREDICTIONS BY GEOGRAPHICALLY TARGETING POLLS WITH ACTIVE LEARNING

Raymond Gilmartin
CS 229 — Spring 2019
Stanford University

Sean Roelofs
CS 229 — Spring 2019
Stanford University

June 13, 2019

1 Introduction and Motivation

Most publicly-available poll data comes from large survey research companies, like Pew and Gallup, which frequently give away election poll data for free online in order to promote their brand [1]. However, since this data is collected and given away for free, it can't be targeted or specific. This is why most publicly-available polls are taken at the state- or national-level. Even when campaigns or political parties pay for targeted election research, they cannot poll every county in the United States. However, they can poll a small subset of counties which are of particular political interest.

Active learning offers a possible way to reduce the cost of targeted polling and improve election predictions. Once a survey research company finishes collecting poll data for a county, active learning could be used to choose which counties should be targeted next for polling. Once pollsters have exhausted their funds and cannot conduct any more polls, machine learning can be used to predict results for the counties that weren't polled. The final result would be a polled or predicted election result for every county in the United States. In theory, this result could be obtained at a lower cost than an equally accurate prediction found without active polling. In this report, we simulate polls for the 2016 presidential election to test whether active learning improves predictions from poll results.

2 Related Work

Much of the existing work in the field of active learning focuses on classification problems. However, since regression and random forests are important and useful types of learning algorithms, there is new work being done to determine effective querying methods for active learning in regression settings. In particular, recent work has focused on optimizing expected model change in random forests [2], [3]. Other approaches include training separate regression models to predict expected error reduction in tandem with the training of the overall model [4].

Much of the existing research in the field of political polling focuses on understanding the existing ability of polls to predict or not predict elections [5]. We hope to contribute to the literature by applying active learning to polling as a potential means of improving the predictive power of polling.

3 Data

Ideally, data for all predictors should be free and publicly available. The two categories of data that will be most relevant for predicting election results are demographic data and historical election results.

3.1 Sources

For demographic data, we use a dataset aggregated by a Kaggle user based on information from the 2015 United States Census American Community Survey [6]. The dataset contains a demographic profile of each county in the United States. General demographics in the dataset include total county population and a breakdown by sex and race. Economic demographics include median income, poverty rate, unemployment rate, and employment by industry.

Historical election data was aggregated by the MIT Election Science Data Lab based on county-level election return data from the Secretaries of State for each of the fifty states [7]. The complete dataset, which was downloaded from the

Harvard Dataverse, contains county-level election results for every presidential election from 2000 through 2016. The 2004-2012 results were used as predictors and the 2016 results were used for simulating the response.

3.2 Response Variable: Simulating Poll Data

Since polling is expensive, simulated polling data was used to train the model. Studies on polling have shown that there is justification for simulating polls results as a normal distribution around the true election results [5].

Each true election result, $y^{(i)}$ had two elements: $y_1^{(i)}$, the percentage of the county population who voted in that year's presidential election, and $y_2^{(i)}$, the percentage of votes for the Democratic candidate out of the total votes.

Simulate polling data as: $p_1^{(i)} \sim y_1^{(i)} + N(0, 0.04)$ and $p_2^{(i)} \sim y_2^{(i)} + N(0, 0.02)$. Using 2 percentage points as the variance for the Democratic candidate's polling share is motivated by the aforementioned literature on election polling. A more conservative 4 percentage points was used for turnout.

When fitting the model, polling data was used as the response variable. That is, each observation's response was represented as $(p_1^{(i)}, p_2^{(i)})$. For the purposes of testing the model and computing mean squared error (MSE), true election results were used. For a single data point, $MSE = ((y_1^{(i)} - \hat{y}_1^{(i)})^2 + (y_2^{(i)} - \hat{y}_2^{(i)})^2)/2$.

4 Methods

The active learning algorithm begins by fitting a model based on a randomly selected subset of the data. The size of this initial subset is a hyperparameter (See section 5). For the initial subset of counties, the model receives the predictors and response variables (the simulated polling data). The algorithm proceeds by alternately fitting an iteration of the model based on the counties for which it has simulated polling data, then using a querying method to choose which counties it will receive response data for next.5 querying methods and 2 model types were tested, in various combinations.

4.1 Querying Methods

1. **Random selection:** Choose random counties to poll next. This querying method is used as the baseline.
2. **Committee selection:** Train m "student" models on random independent subsets of counties polled so far. Each student's subset represents a fraction of the training data, randomly selected without replacement. Then, for each student, predict on the entire training dataset. For each observation (county), take the variance of the m student predictions for that observation. Poll the counties with the highest variance.
3. **K-means:** Fit a k-means clustering model on the predictors for all the counties. This is possible because k-means clustering is an unsupervised learning algorithm, so it can be fit without knowing any of the simulated poll data. Poll 1 county from each of the k clusters to ensure a diverse sample of counties.
4. **Entropy selection:** (random forests only) Use each of the forest's m trees to predict a response for each county. For each county, take the variance of the m tree predictions. Poll the counties with the highest variance.
5. **Committee with k-means initialization:** Choose the initial sample using the k-means approach outlined above instead of by randomly sampling 10 data points, then proceed by using committee selection.

The rationale behind entropy selection and committee selection is that the querying method should poll the counties for which the model is least certain about its prediction because these counties will generate the largest improvement in predictions if the value of their response variable (simulated poll data) becomes known [8]. The rationale behind k-means clustering is that the querying method should attempt to gather information about a diverse range of counties [9]. Combining committee selection with k-means clustering for initialization is an attempt to initially get information about a wide range of counties and then investigate the counties which will be most helpful to the development of the model.

4.2 Model 1: Linear Regression

The first model we tested was was a linear regression: $(p_1^{(i)}, p_2^{(i)}) = \sum_{j \in predictors} X_i(\beta_{j,1}, \beta_{j,2}$. Each $\beta_{j,k}$ is a scalar coefficient relating predictor j to the k'th element of the response. Define X to be the the matrix of all predictors for all counties. X has dimensions equal to the number of counties (3141) by the number of predictors (90). β is the 90×2 matrix of parameters. P is the 3141×2 matrix of polling data, where the i'th row of P contains the two values $p_1^{(i)}$ and $p_2^{(i)}$. So the model can be rewritten in matrix form as: $P = X\beta$.

The model was fit using `sklearn.linear_model.Ridge`, which incorporates $\ell 2$ regularization. This regression technique, ridge regression, minimizes the objective function: $||P - X\beta||_2^2 + \alpha||\beta||_2^2$. Regularization was important in order to deal with cases where the model was underdetermined (i.e., # of predictors $>$ # of counties polled).

The advantages of linear regression models are that they are computationally easy to fit and make it is easy to interpret the relationship between predictors and election results using β values. The disadvantages of linear regression models are that they are not flexible (can't model complex, nonlinear relationships) and may predict values outside of the range $[0, 1]$ for the response. Since responses are percentages, we remedy the second problem by changing predictions greater than 1 or less than 0 to be 1 or 0, respectively.

4.3 Model 2: Random Forests

The second model we tested was a random forest for regression. A regression forest consists of a set number of trees, each of which is capable of predicting the response from the predictors. We used `sklearn.ensemble.RandomForestRegressor` as the regression forest implementation.

Each tree in the forest is grown using a bootstrapped sample of the training data (randomly selected, with replacement) with size equal to that of the complete training dataset. Trees are grown by repeatedly splitting the feature space based on a single predictor variable in order to create two partitions with different mean responses. For example, if `RandomForestRegressor` determines that poll results vary based on the condition $\%white < 50\%$, the algorithm would partition the feature space based on that condition. `RandomForestRegressor` takes as a parameter `max_depth`, which indicates the maximum number of times the tree will split. The fitted tree's partitions of the feature space are called leaves. Each leaf has an associated response value equal to the mean of the responses for the training data points which fall into that leaf.

A fitted forest can be used to predict the response for a test observation. Each tree in the forest predicts the response by using the predictors to categorize the test observation into one of the tree's leaves and predicting the value associated with that leaf as the response. The forest's prediction is the mean of the predictions from each tree.

The advantages of random forests are that they are flexible (can model complex, nonlinear relationships) and won't predict values outside of [0, 1] because trees predict values for test observations based on the mean of the training observations in the same leaf, so they will never predict a response value that is more extreme than the training data. The disadvantages of random forests are that they are more computationally intense to fit than linear regression models and that they have no coefficients, so interpreting the effect of predictors on election results may be difficult.

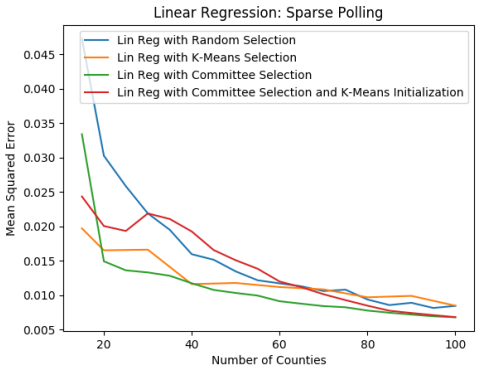
5 Results and Discussion

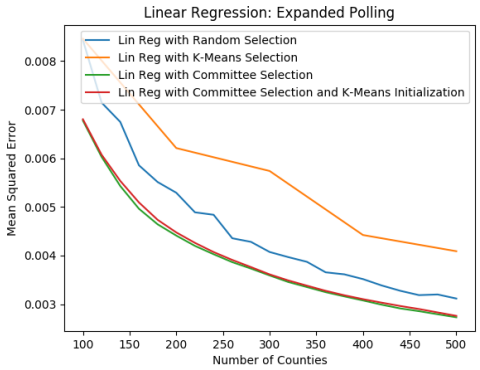
We use two categories for visualizations: "Sparse" for < 100 counties polled and "Expanded" for ≥ 100 counties.

All model-querying method pairs share 2 hyper parameters: size of initial sample and mini-batch size for querying. After experimenting with different values, initial sample size was set to 10—small enough for active learning to begin quickly, but not so small that students in committee selection or trees in entropy selection will receive identical training sets. Batch size was set to 5 for "Sparse Polling" and 20 for "Expanded Polling" because these batch sizes made visualization possible and allowed the active polling algorithm to iterate many times without being too slow.

Each of the lines in the graphs presented below represents multiple iterations of the active learning algorithm. This was done in order to minimize the impact of random variation on the results and to make the plots smoother and easier to read. The number of repetitions was chosen to be the maximum number possible within reasonable time constraints and ranged from 5 (for random forests with committee selection) to 100 (for all linear regression models).

5.1 Linear Regression





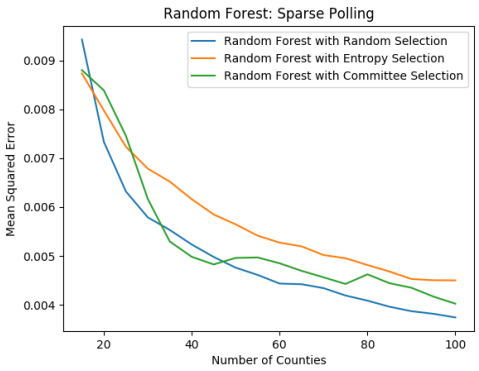
The above plots show the results from regularized linear regression models paired with four querying methods. The regularization term is proportional to $\frac{1}{|X_{train}|}$. This regularization term is used because sparse polling requires a large regularization term to avoid overfitting, especially when the model is underdetermined, but expanded polling requires a smaller regularization term to fit the data. **The baseline, shown in blue, used random selection as its querying method.**

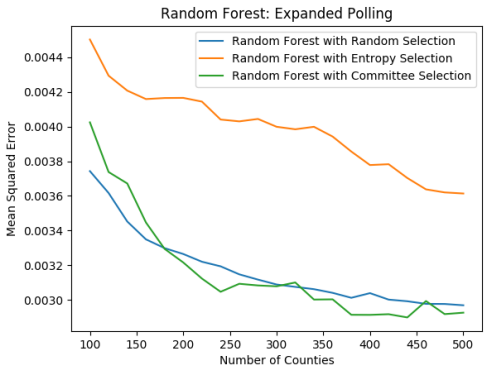
The orange line indicates the results using K-Means as the querying method. To select n counties to train on, we use a k-means clustering model with $k = n$. For very low n ($n < 20$) this method performed the best because the algorithm received a diverse group of counties as training data, but did not overfit on extreme cases due to the high regularization term. As n increased, this method performed worse than the baseline. This may be because K-Means increased the likelihood that outliers would be selected, which significantly altered the distribution of the data.

The green line indicates the results using committee selection as the querying method. With committee selection, there were 2 hyperparameters not shared by other querying methods: number of students and fraction of data seen by each student. We experimented with these parameters and chose 8 students, each receiving 50% of the training data. 8 students is small enough to be computationally feasible, but large enough to yield similar variance estimates on repeat runs. After just 20 counties were polled (10 from the initial random sample plus 10 from 2 iterations of committee selection), committee selection outperformed all other querying methods tested. Committee selection continued to outperform other methods in the "Expanded Polling" range. This indicates that the goal of committee selection was likely achieved: choosing the points for which the model had the highest variance (the points for which the model's predictions were least certain) was a good approximation for choosing the points which would most improve predictions.

The red line indicates the results using committee selection as the querying method and using K-Means to choose the initial 10 **data points.** This model performed worse than regular committee selection initially. This may be because, with such a small initial sample size, k-means chooses too many extreme points, causing the training dataset to be unrepresentative and the model to be incorrect. However, after about 100 counties, it begins to match the results from committee selection with random initialization because the initial 10 counties no longer matter as much.

5.2 Random Forest



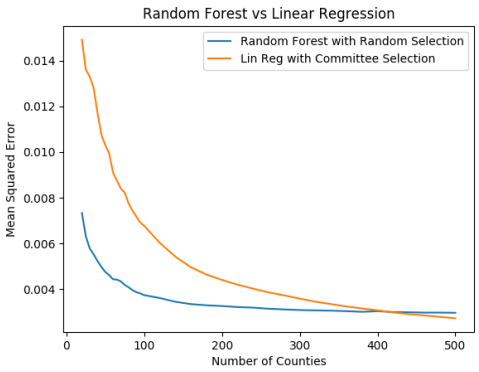


The above plots show the results from random forest models paired with three querying methods. Forests were fit using 50 trees and a maximum depth of 4. The number of trees was chosen because models with 50 trees could be fit in a reasonable amount of time and because models with more trees did not offer significantly improved predictions. **The baseline, shown in blue, used random selection as its querying method.**

The orange line indicates the results using entropy selection as the querying method. When a very small number of counties were chosen for polling (< 10), the performance was comparable to or better than that of the baseline. When more counties were chosen, performance was significantly worse than the baseline. This may be because random forests rely on bootstrapped samples for fitting. Bootstrapped samples can be treated as independent samples of the true distribution of data when they are drawn from a large random sample of that distribution. However, with entropy selection, points are selected based on the variance of their predicted values among the trees in the forest. This may cause outliers or extreme values to be overrepresented in the training dataset, which would skew the bootstrapped samples used to fit the forest. As the dataset gets larger, this problem would grow increasingly more pronounced.

The green line indicates the results using committee selection as the querying method. The performance of this method was comparable to that of the baseline. This may be because random forests are already a committee-based method of prediction. Taking the result from each forest to be the average of the results from 50 trees reduces the variance of the forest's predictions. It is possible that predictions from the student models have so little variance that choosing the counties with the highest variance among the student predictions is essentially the same as choosing counties randomly.

5.3 Comparing Random Forests and Linear Regression



The above plot compares the best random forest results with the best linear regression results. The random forest using random selection as the querying method outperforms linear regression at all training dataset sizes up to 400. After this point, linear regression begins to perform very slightly better, but this region is not of interest because it would be prohibitively expensive to poll in that many counties. Even though random forests trained on randomly-selected datasets out-predict any of the active learning methods tested here, there are still cases where linear regression would be preferred to random forests and active learning by committee selection might be useful. If the goal of polling is to explain the results of elections, linear regression is helpful because it offers clear relationships between predictors (characteristics of counties) and responses (election results).

6 Conclusions and Future Work

We were unable to find a method of predicting elections using active learning and geographically-targeted polling that could out-perform a random forest applied to a randomly-selected set of county-level polls. However, we did find several querying methods which could be used to improve predictions from polling and linear regression which have possible applications.

There may still be ways to improve predictions beyond the quality of those from random forests and random selection. In addition to the querying methods used here, random forests can use querying methods based on choosing observations whose predictions are well-distributed throughout the leaves of the forest's trees. This is similar to the k-means querying method applied to linear regression and offers a possible way to get a diverse sample of counties for training a random forest without making the training dataset unrepresentative. A mixed k-means and entropy selection querying method could also be attempted. Such a method would be analogous to the committee selection method with k-means initialization for linear regression.

7 Contributions

Raymond and Sean contributed equally to this project overall. Both Raymond and Sean did equal amounts of work on the initial project proposal and milestone. For the substance of the project, Sean did the majority of the work on the linear regression models and Raymond did the majority of the work on the random forest models. Additionally, Sean wrote much of the code within the `utils.py` file, which was used for graphing, error calculation, and file reading/writing. Raymond did the majority of the work on the poster, since he had experience making project posters. Both Raymond and Sean did an equal amount of work on the final report.

8 Code

All code used here can be found on the Github repository for this project. The url is: `https://github.com/rugilmartin/active_polling/settings`

References

[1] M. Fahey and E. Chemi, "Most Election Pollsters Aren't Really in it for the Money," *CNBC*, October 18, 2016. [Online], Available: https://www.cnbc.com/2016/10/18/most-election-pollsters-are-not-really-in-it-for-the-money.html. [Accessed May 24, 2019].

[2] J. Goetz, A. Tewari, and P. Zimmerman, "Active Learning for Non-Parametric Regression Using Purely Random Trees," [Online], Available: https://papers.nips.cc/paper/7520-active-learning-for-non-parametric-regression-using-purely-random-trees.pdf. [Accessed June 6, 2019].

[3] W. Cai, M. Zhang, and Y. Zhang, "Batch Mode Active Learning for Regression With Expected Model Change," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, No. 7, July, pp. 1668-1681, 2017.

[4] K. Konyushkova, R. Sznitman, and P. Fua, "Learning Active Learning from Data," July 14, 2017. [Online], Available: https://arxiv.org/pdf/1703.03365.pdf. [Accessed June 6, 2019].

[5] H. Shirani-Mehr, D. Rothschild, S. Goel, and A. Gelman, "Disentangling Bias and Variance in Election Polls," *Journal of the American Statistical Association*, vol. 113, no. 522, Feb., pp. 607-614, 2018.

[6] Kaggle user MuonNeutrino, "US Census Demographic Data," *kaggle.com*, March, 2019. [Online]. Available: https://www.kaggle.com/muonneutrino/us-census-demographic-data. [Accessed May 16, 2019].

[7] MIT Election Science Data Lab, "County Presidential Election Returns," *Harvard Dataverse*, October 11, 2018. [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/VOQCHQ. [Accessed May 16, 2019].

[8] H. Seung, M. Opper, and H. Sompolinsky. "Query by committee," In Proceedings of the Fifth Workshop on Computational Learning Theory, 1992, pp. 287–294.

[9] O. Cohen, "Active Learning Tutorial," *Towards Data Science*, April 19, 2018. [Online], Available: https://towardsdatascience.com/active-learning-tutorial-57c3398e34d. [Accessed June 6, 2019].