COMPUTATIONAL METHODS FOR DATA ANALYSIS CLASSIFYING POLITICIANS

AVERY LEE

Applied Mathematics Department, University of Washington, Seattle, WA
leeave22@uw.edu
03/11/2022

Abstract. Classifying Politicians creates a machine learning (ML) regression model that predicts politicians' political standing (Democrat or Republican) based on their votes (yes, no, or NA) for 16 different bills. This is done with spectral clustering and semi-supervised machine learning techniques to find the optimal variables and study the accuracies of the prediction model using those variables. This paper will go through the mathematical theories behind the calculation methods, the algorithms used to create the ML model, the final computational results, and conclusions.

1. Introduction and Overview

Machine Learning is a field of study that has been expansively growing due to its endless applications; it can be used to predict, classify, approximate, or even cluster any type of dataset [1]. In this paper, a semi-supervised regression model will be created to predict a politician's political standing based on their vote by studying a given dataset containing politicians' votes on different bills and their political standing between Republican and Democrat.

Section 2 of this paper will go over the mathematical background necessary to understand the computations. Section 3 will cover the algorithms and Python software packages. Then Section 4 will summarize the results from the computations, and Section 5 will summarize the learnings.

2. Theoretical Background

2.1. **Linear Regression.**

Linear regression (least squares) is one of the simplest ML models that can be created. It assumes a linear relationship between the input variables and output values. Then, it finds the coefficients b_i that give the smallest squared error for this equation where x_i are the independent variables, y is the dependent variable, and b_i are the coefficients:

$$y = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n$$

Another way to write the optimization problem, which we will use for our problem, is

$$\hat{\beta} = argmin_{\beta \in \mathbb{R}^M} ||A\beta - b||_2^2$$

where $\hat{\beta}$ is the coefficients of the linear regression model found.

2.2. **Spectral Clustering.**

Spectral clustering is a technique used to identify clusters of nodes in a graph based on the edges connecting them [2]. This method is very flexible and less restrictive, and can be useful when the dataset has a more unusual cluster shape that can be difficult to identify.

First, the Laplacian can be found with this eigendecomposition equation. For this report, the unnormalized Laplacian will be used.

$$L = Q\Lambda Q$$

where Q contains eigenvectors q, and Λ contains eigenvalues. Q is constructed like below

$$Q = \begin{pmatrix} q_0 & q_1 & \dots & q_{N-1} \end{pmatrix}$$

Often q_1 is a good representation of the clusters. Note that this is the second column of eigenvectors, not first. This will be shown more in detail in Section 4.

The unnormalized graph Laplacian matrix can be constructed on input data X with the weight function where σ represents the variance parameter. It is important to choose an appropriate σ as outputs can highly vary:

$$\eta(t) = exp(-\frac{t^2}{2\sigma^2})$$

The feature map F: $\mathbb{R}^d \to \mathbb{R}^M$ where $M \geq 1$ can be defined as:

$$F(x_j) = \begin{pmatrix} q_{1j} \\ q_{2j} \\ \dots \\ q_{Mj} \end{pmatrix}$$

In this example, the classes, or votes, will be represented as -1, 0, and +1. So the sign of the prediction categorizes which class it falls into. Since this portion is an unsupervised learning task, the model only clusters, not identifies the prediction, meaning all the signs might be flipped in some instances. If the accuracy (clustering accuracy = 1 - $\frac{\text{number of misclassified members}}{\text{number of members}}$) is less than 50%, then the predictions (signs) can be flipped because it was likely oppositely clustered.

2.3. **Semi-Supervised Machine Learning.**

Machine learning models can commonly be categorized into either supervised or unsupervised learning.

Supervised machine learning is when input data and labels are studied to make a prediction on a new input. For example, input images of dogs and cats and their true labels can be studied through the model, and the model will predict if a new input image is a dog or a cat. They can have classification or regression techniques.

Unsupervised machine learning studies a dataset and groups the categories into separate classes. For example, if there is a dataset of dogs and cats, the model will predict that there are 2 groups. However, it will not know that one is a dog and one is a cat, it will just know that they are different things. This has clustering techniques.

Now, semi-supervised machine learning is a mix of them both, where it studies both labelled and unlabelled datasets, and groups the separate groups together [3]. The assumed structure is that there are inputs $X = \{x_0, x_1, \ldots, x_{N-1}\} \in \mathbb{R}^{d\times N}$ with outputs $Y = \{y_0(x_0), y_1(x_1), \ldots, y_{M-1}(x_{M-1}),\} \in \mathbb{R}^M$ that are only available for a subset $M \leq N$ of the inputs. This means only a portion is truly labelled. The goal is to predict the outputs of the unlabelled set using both the labelled and unlabelled datasets.

Now, graph Laplacian embedding can be used as briefly mentioned in the Spectral Clustering section. For our purposes, the graph Laplacian is created using the weight function $\eta(t)$ mentioned in the previous section. Again, for our purposes, the eigenvector of the graph Laplacian using the weights will be used as it tells a lot about the politician's political standing.

It is not always necessary to use all of the rows and columns in the Laplacian embedding, so only a top left chunk of it can be used. Of course, varying number of rows and columns will create varying accuracies, so the number of rows and columns can also be tested.

To be more specific, a submatrix A and vector b can be defined as below:

$$A_{ij} = F(X)_{ij}, \quad i = 0, \ldots, J-1, \quad j = 0, \ldots, M-1$$

$$b_i = y_i, \quad i = 0, \ldots, J-1$$

3. Algorithm Implementation and Development

3.1. **Python Packages and Notable Functions.**

Python was used to compute the algorithms and produce plots. Below is a list of packages used.

Python Package or Function	Description
`google.colab, drive`	Gives access to the user's Google Drive account where they can access the data.
`scikit-learn [4]`	Provides tools for machine learning algorithms including linear_model.
`scipy.spatial [5]`	Helps perform distance matrix computation.
`numpy [6]`	Allows usage of multidimensional arrays and functions to manipulate those arrays.
`matplotlib.pyplot [7]`	Useful for graphics manipulation.
`LinearRegression() and fit()`	Fits the X and Y values into a linear regression model.

Table 1. Python Packages and Functions

3.2. **Setup.**

The given data is a 435x17 dataset that includes 435 politicians' votes for 16 different bills. Of the 435 politicians, 267 are Democrats and 168 are Republicans.

The first column represents their political party between Republican and Democrat. This column containing political party information will be called y. Since there are only 2 parties, Republicans will be marked -1 and Democrats will be marked +1 in y.

This column will be removed to create 435x16 matrix X, where the vote "y" becomes +1, "n" becomes -1, and "?" becomes 0.

3.3. **Spectral Clustering.**

For each sigma between 0.1 and 4 with a step size of 0.01, the accuracy of the clustering model will be calculated. First, the spatial distances of the X is calculated, where this matrix will be the t in the $\eta(t)$ equation. Let's call these weights W. The diagonal of the sum of this matrix D is calculated.

Now, to calculate the unnormalized Laplacian L, we can simply do $D - W$. The eigenvalues and eigenvectors of L can be found with the built-in *np.linalg.eigh*() function. Now we sort the eigenvectors based on the eigenvalues. As we talked about in Section 2, the classifier will be the sign (-, +) of the $q1$, which is the second column of the matrix of sorted eigenvectors.

The accuracy can then be calculated using the equation above. Again, this algorithm only clusters the data, not fully identify them. This means that instead of Republicans being categorized into -1 and Democrats +1, the Republicans can be categorized into +1 and Democrats -1. We can fix this issue by swapping the signs if the accuracy is less than 50%.

After calculating the clustering accuracy for each of the sigmas using this process, we can see for which sigma the accuracy is highest. This is the optimal sigma we can use for the semi-supervised learning model which comes next. This optimal sigma is called σ^*

3.4. **Semi-Supervised Learning.**

As stated in Section 2, not all of the rows and columns of the Laplacian embedding need to be used for the prediction model. However, since it is unknown how many rows and columns to actually use, we can test this out by testing out combinations for J rows 5, 10, 20, 40, and M columns 2, 3, 4, 5, 6.

The exact same process mentioned above is done with the optimal sigma value σ^*, except this time, the submatrix A with the first J rows and first M columns of the eigenvectors matrix and vector b is calculated. A linear regression model is fit using A and b. The coefficients of this linear regression model $\hat{\beta}$ is found. Now, the classifier will be the sign (-, +) of $F\hat{\beta}$, which we can now call the $\hat{y}$. We can calculate the accuracy of the model with the same equation used for cluster accuracy comparing true labels y and predicted labels $\hat{y}$.

After doing this process for all combinations of the J's and M's mentioned above, we can now find the J and M values where the accuracy is highest.

4. Computational Results

4.1. **Spectral Clustering.**

The sigmas are evenly spaced by 0.01 from 0.1 to 4. In the graph below, we can see that after around $\sigma = 1$, the clustering accuracy levels out at around 0.88. From these very similar values, the highest accuracy found is 0.88046 with a σ^* of about 1.17. This is marked as a red star in the graph. Before around $\sigma = 1$, the clustering accuracy highly varies depending on the σ value. This is why it is important to make sure that an appropriate σ is used to calculate the weights.

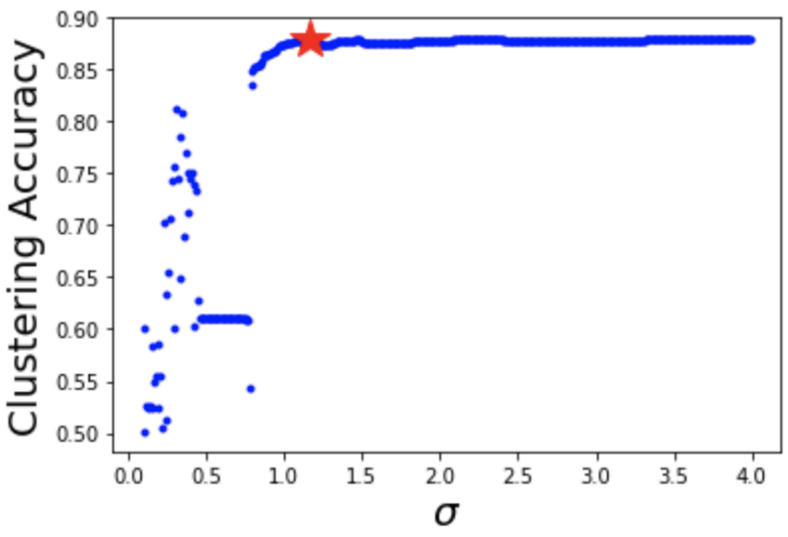

Figure 1. Clustering Accuracy by Sigma

We can also plot the q_1 using the optimal σ^*. To visualize the importance of the 2nd eigenvector effectively, we can sort the values by Republican first, then Democrats. The red points represent Republicans, and blue points represent Democrats. The results of this plot makes sense as the majority of the red points are towards the negative values, and the majority of the blue points are towards the positive values. Since there is a clear distinction between the two clusters, we can again say that q_1 are appropriate values to use for classification.

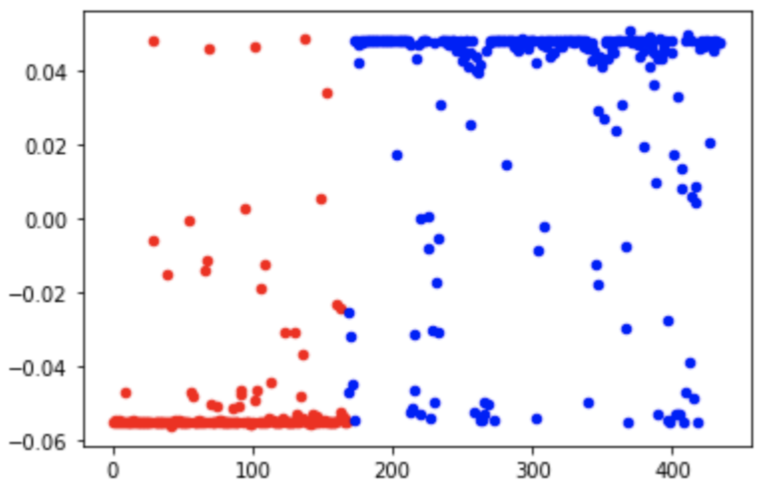

Figure 2. q_1 Predictors (Republican vs Democrat)

4.2. **Semi-Supervised Machine Learning.**

Using the M and J combinations mentioned above, we get these accuracies.

M \ J	5	10	20	40
2	0.8896	0.8873	0.8827	0.8804
3	0.8873	0.8160	0.8206	0.8367
4	0.8390	0.8505	0.8643	0.8758
5	0.8689	0.7103	0.8390	0.8804
6	0.8689	0.6850	0.8735	0.8643

Table 2. SSL Accuracy based on M and J

The highest accuracy is at 0.8896 when $M = 2, J = 5$, but it is not a specially high value compared to others, so we can look at the general trend of the accuracies. The accuracy is pretty consistent around 0.88 for all $M = 2$ values, regardless of the J value. The accuracy gets lower for $J = 10$ as M increases. Most of the other accuracies are still pretty consistent between 0.8 and 0.9.

It is important to also note that the nature of the bills and therefore the votes can impact the results here. If different bills were considered, there could be stronger or weaker deviations between the political parties.

5. Summary and Conclusions

In this paper, a semi-supervised machine learning model using spectral clustering was created to study the voting patterns of Republican and Democratic politicians. The importance of choice of σ was observed through trial and error of different values. In the future, this model could be slightly modified to perform similar tasks. For example, more bills can be added or subtracted to the dataset, or more defining bills can be observed. This model can also be used for other things than just political classification, as long as the data follows a similar structure.

Acknowledgements

The author is thankful to TA Katherine Owens for the general rundown and clarifications on calculations and code, as well as classmates such as Pranav Natarajan that reconfirmed my data results and Python syntax. Last but not least, the author is thankful to Professor Bamdad Hosseini for teaching most of the mathematical concepts mentioned in the paper.

References

[1] Bamdad Hosseini. Introduction to machine learning. University of Washington (LOW 216), Jan 2022.

[2] Bamdad Hosseini. Spectral clustering. University of Washington (LOW 216), Feb 2022.

[3] Bamdad Hosseini. Semi-supervised learning. University of Washington (LOW 216), Feb 2022.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.