CLASSIFYING POLITICAL PARTY MEMBERSHIP WITH SPECTRAL CLUSTERING AND SEMI-SUPERVISED MACHINE LEARNING ALGORITHMS

RYAN MOUW

Applied Mathematics Department, University of Washington, Seattle, WA
`rmouw@uw.edu`

Abstract. The task is to group the members of the 1984 U.S. House of Congress via voting record sample. The goal is to simply distinguish two groups of voters, corresponding to Democrats and Republicans. It is known that there are 267 members of the Democratic Party and 168 members of the Republican party at that time. The voting record consists of 16 arbitrary bills. Semi-supervised and spectral clustering algorithms and techniques are utilized, resulting in almost 90% accurate classification model.

1. Introduction and Overview

The goal is to differentiate between Democrats and Republicans in the 1984 U.S. House of Congress, given a subset of the member's voting record. In total there are 435 members of the 1984 House of Congress. 267 members were part of the Democratic party, and 168 members were part of the Republican party. In this problem, we are given a subset of the member's voting record. The subset includes the member's votes on 16 bills. The data consists of a ”yes”, ”no”, or ”?”. Spectral clustering was first used to assign group labels, followed by a semi-supervised machine learning algorithm.

2. Theoretical Background

Clustering is an example of unsupervised machine learning and in its simplest form is grouping data points based upon each point's distance from a shared center. Spectral clustering takes this idea further. In spectral clustering, the data is embedded in a special type of graph, called the graph Laplacian. To understand the graph Laplacian, graphs, the Adjacency matrix, and the Degree matrix need to be defined.

A graph is a collection of nodes, which may or may not have edges connecting them. Sometimes these edges can have values associated with them. These values are known as weights. The graph then encodes each point of data that is being considered, along with a weight that is associated between any two points. This weight is undirected and can often be interpreted as the distance between two points. We refer to the matrix representation of a weighted, undirected graph as an Adjacency matrix. Each data point in the adjacency matrix has, what is called, a degree. Traditionally, the degree of a node is how many edges are connected to it. In the case of a weighted graph, the degree of a node will just be the value in the corresponding index, after the rows of the adjacency matrix, and thus the weights, are summed. The degree matrix, (D) is then a diagonal matrix where the diagonal entry (i, i) is the degree of the i^{th} node. [2]

The weights of an adjacency matrix can be constructed in a number of ways. The weights can have a binary nature, (I.e. they have a value of 1 if the edge exists, and 0 if the edge does not

Date: March 11, 2022.

exist) but this does not necessarily need to be the case. A Gaussian weighting, or even a threshold weighting scheme are both acceptable in an adjacency matrix and will give rise to different values in the associated degree matrix. The Gaussian weighting is defined for points x_i and x_j in an Adjacency matrix w_{ij} as;

$$w_{ij} = \exp \frac{-||x_i - x_j||_2^2}{2\sigma^2} \tag{1}$$

[7]

In this weighting scheme, the value of σ becomes a hyperparameter that needs to be tuned. The Graph Laplacian, L, is now just the Adjacency matrix (w_{ij}) subtracted from the Degree matrix ($diag()$;

$$L = D - w_{ij} \tag{2}$$

[2]

The diagonal of the Graph Laplacian is now the degree of the nodes, while the values off the diagonal are the negative edge weights.

The Eigenvalues and Eigenvectors of the Graph Laplacian matrix provide valuable information about the structure and groupings of the data in question.[2] Namely, the number of zero valued Eigenvalues of the Graph Laplacian indicate the number of distinct clusters in the data. As clusters of data points become more connected, the Eigenvalues increase. Smaller weights connecting nodes corresponds to smaller increases in the Eigenvalues of the Graph Laplacian. The first Eigenvalue of the Graph Laplacian is always zero, because at both extremes, either there is one connected graph, in which case the first Eigenevalue is zero, or there are N disconnected points, in which case all N Eigenevalues are zero.

The second Eigenvalue and Eigenvector of the graph Laplacian is called the Fiedler value and the Fiedler vector. The Fiedler vector is very useful, as its entries encode the grouping of the nodes based upon the sign of the value in the Fiedler vector. I.e., if index k of the Fiedler vector is negative, and index j of the Fiedler vector is positive, then the node k and node j are in different groups.

Semi-supervised machine learning combines spectral clustering with ridge or kernel regression. Given $\{(x_0, y_0), ..., (x_{M-1}, y_{M-1})\}$ as the labeled dataset and $\{x_M, ..., x_{N-1}\}$ as the unlabelled data subset, the goal is then to predict the outputs $\{y_M(x_M), ..., y_{N-1}(x_{N-1})\}$.[3] Just as in kernel regression, this is accomplished by finding an $f(x)$ such that;

$$f(x) = \sum_{j=0}^{J} c_j \Psi_j(x) \qquad s.t. \qquad f(x) \approx y(x) \quad for \quad M \leq j \leq N-1 \tag{3}$$

[3]

Where $\Psi(x)$ is a function with Reproducing Kernel Hilbert Space (RKHS) constraints (A feature map) and c_j are the coefficients. The above formulation constrains $f(x)$ in such a way that it approximates the unlabelled dataset well, but not the new data points, as in regression. The Graph Laplacian Eigenvectors can then be used as a feature map in which regression is performed.

3. Algorithm Implementation and Development

The following is a overview of the pseudo-algorithm that was used in the semi-supervised learning process to categorize politicians by voting record. The matrix creation, manipulation, Eigen decomposition, and linear regression were accomplished using Scikit-learn [5], Numpy [1], and Scipy [6].

(1) Create $X \in \mathbb{R}^{435\times16}$ (# of House members $\times$ sample of voting record)
(2) Create $Y \in \mathbb{R}^{435}$ (The labels of -1 for Democrat, 1 for Republican for each House member)
(3) Construct the Adjacency matrix using the Gaussian weighting scheme $w_{ij} = \{X, W\}$.
(4) Construct the Degree matrix, D.
(5) Construct the Graph Laplacian, $L = D - w_{ij}$.
(6) Compute the K Eigenvectors of L in ascending order, $\{q_m\}_{m=0}^{M-1}$.
(7) Solve the Ridge regression problem $\hat{c} = argmin \sum_{j=0}^{J-1} |\sum_{m=0}^{M-1} c_m q_{jm} - y_j|^2 + \lambda||c||_2^2$ [4]

Where J and M in Step 7 are the number of voting records being sampled and the number of Eigenvectors of L being used to train the model, respectively.

There are now four hyperparameters to consider; σ, λ, J (The number of voting records to use as the labelled set), and M (The number of Eigenvectors of L to use as feature maps.) In this problem, Least Squares regression is used. This removes the need to tune λ.

To tune σ, the Fiedler vector of the Graph Laplacian is utilized. Values of σ ranging from (0,4], in steps of .01, were used to create the Adjacency matrix, and then the resulting Degree Matrix and Graph Laplacian, $\hat{\text{L}}$. The Fiedler vector of each $\hat{\text{L}}$ was then used to predict labels based upon the sign of each entry. The classification accuracy for each value of σ is shown in Figure 2. Figure 2 shows that the accuracy seems to stabilize at values of $\sigma > 1.25$. For this reason, we chose the $\sigma^\star = 1.47$ as the optimal value because it had the highest accuracy after stabilization occurred.

Using $\sigma^\star = 1.47$, we then trained a least squares regression model on varying values of M (Number of Eigenvectors of L) and J (Number of labeled samples of voting records). The classification accuracy for each combination of the M and J hyperparameters are in Table 1.

4. Computational Results

The first thing constructed was the Laplacian matrix. The unnormalized Eigenvalues of the matrix are shown in Figure 1. The first few Eigenvalues are very small, then they grow rapidly. This is indicative of a few number of clusters.

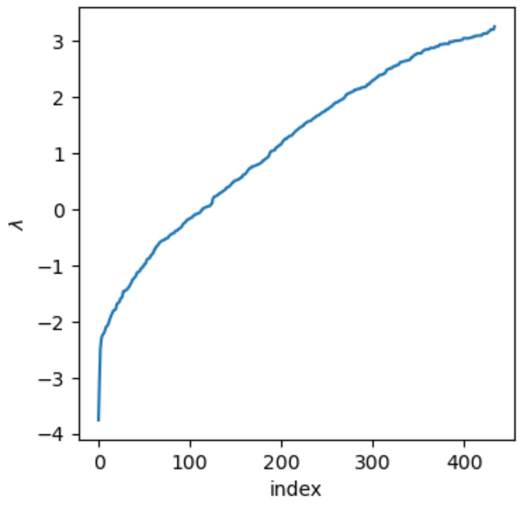

Figure 1. Unnormalized log of Eigenvalues in ascending order

The optimal value for σ was found to be $\sigma^\star = 1.47$. This is shown in Figure 2. What Figure 2 also reveals is that beyond a value of $\sigma \approx 1.25$, the classification accuracy is fairly stable. This may suggest that there is some threshold that needs to be overcome to distinguish between the two groups, but once the threshold is met, the increasing value of σ does little to further differentiate between the groups.

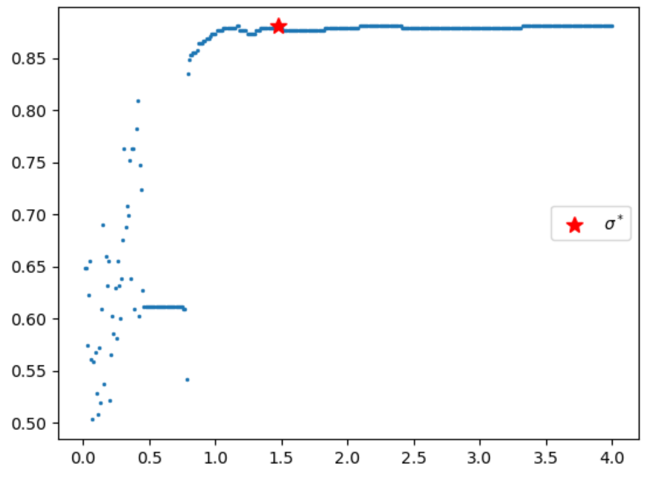

FIGURE 2. Optimal σ value

The Laplacian embedding of the first 3 Eigenvectors is shown in Figure 3. This data is unnormalized, but even so, it is clear that the majority of the variation in data is along the q_1 axis.

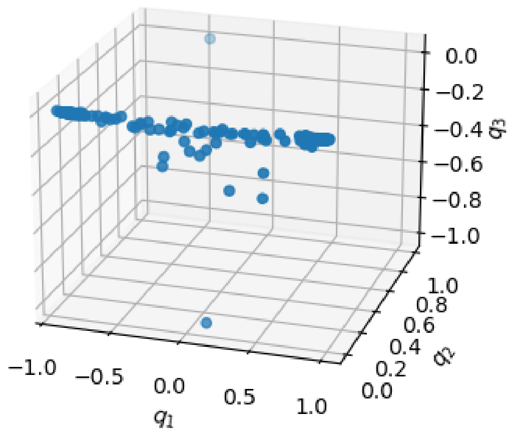

FIGURE 3. Laplacian embedding with first 3 eigenvectors

Figure 4 further demonstrates what was found in Figure 3. Figure 4 shows the Laplacian embedding on only the q_1 axis. It is clear that the majority of the Republican and Democrat politicians can be grouped by the sign of q_1.

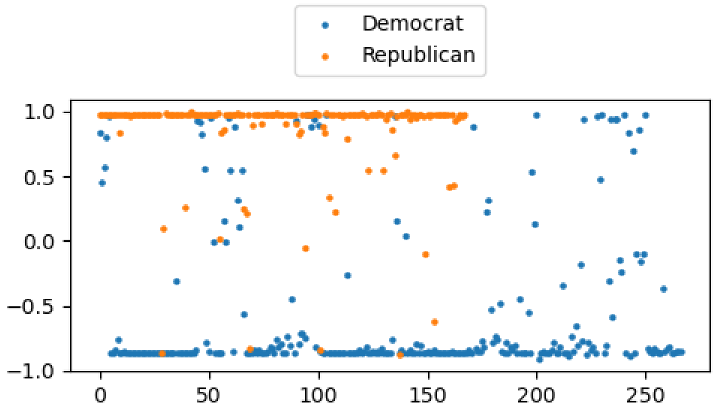

FIGURE 4. Laplacian embedding of q_1

Table 1 shows the classification accuracy given different values of M and J, where M is the number of Laplacian Eigenvectors and J is the number of politician voting records used in the training of a Ridge regression model (with $lambda = e^{-8}$). The best classification accuracy was with $M = 3$ and $J = 5$.

J \ M	2	3	4	5	6		mean	std
5	0.8759	**0.8897**	0.5839	0.8598	0.8782		0.784	0.184
10	0.8782	0.7701	0.8529	0.7747	0.6966		0.794	0.065
20	0.8805	0.8437	0.8759	0.8253	0.8092		0.847	0.028
40	0.8828	0.8736	0.8644	0.8805	0.8828		0.877	0.007
mean	0.879	0.844	0.752	0.835	0.817			
std	0.003	0.046	0.194	0.040	0.075			

TABLE 1. Classification Accuracy for values of M and J

5. Summary and Conclusions

The Fiedler vector did an excellent job categorizing the House members after finding $\sigma^{\star}$. Interestingly, the accuracy was fairly consistent with increasing values of roughly $\sigma > 1.25$. This could be an indication that some sort of threshold needed to be met to distinguish the two groups, but beyond that threshold no further useful differentiation could be made.

Also worth noting is that while $M = 3$ and $J = 5$ seems to be the winning combination in this circumstance, in general, the $M = 2$ column and $J = 40$ row have the highest mean accuracy and potentially more importantly, relatively low variance in the associated classification error as compared to other rows and columns. This is perhaps an intuitive combination in that the higher number of voting records used to train could fit the data well, but a low number of Eigenvectors could introduce enough bias as to avoid overfitting.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[2] B. Hosseini. Introduction to graph laplacians, Feb 2022.

[3] B. Hosseini. Semi-supervised learning, Feb 2022.

[4] B. Hosseini. Ssl demo, Feb 2022.

[5] 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.

[6] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[7] U. von Luxburg. A tutorial on spectral clustering. *CoRR*, abs/0711.0189, 2007.