CLASSIFYING POLITICIANS

ABHISHEK SHARMA

Mechanical Engineering Department, University of Washington, Seattle, WA
as711@uw.edu

Abstract. In this work, we group politicians based on their voting records using spectral clustering. We construct a graph with edge weights signifying closeness of pairs of data points. Then, we cluster points using the second eigenvector of the graph Laplacian.
We also investigate the possibility of generating class labels for unlabelled samples in the data by combining clustering and a classifier learned using the labelled subset of the data.

1. Introduction and Overview

We have a dataset of voting records from 453 politician affiliated with the Democratic (267) and Repulicans (168) party on 16 bills in the 1984 congressional voting. The congressmen voted with either yes or no to each of the bill. Some attributes are missing and are tagged with a ?, implying that the information is not available. We assign 'yes' votes with a value of 1 and no votes with a value of -1. Missing votes are assigned a value of 0. Thus, we have 435 samples with 16 features each as the input $X \in R^{435\times 16}$. Our overall goal is to identify the party affiliation from the voting data. Thus, our labels are party affiliation. We assign a value of -1 to the Republican party and 1 to the Democratic party.

Our first task is to cluster the politicians according to the party affiliation using the voting data. We use spectral clustering to this end. We report the rate of misclassifications using this methods. "Misclassified" here means a member who should have belonged to the opposite party/cluster.

Next, we test if we can generate labels for politicians using clustering, assuming only a few samples in the data had been originally labelled. This explores the possibility of semi-supervised learning.

2. Theoretical Background

Identifying clusters of similar entities is a fundamental problem in machine learning. Samples are clustered together using some measure of distance e.g. euclidean distance, manhattan distance, cosine distance etc. Closer points are expected to be in the same cluster. Clustering can be performed using various algorithms like k-Means [1], spectral clustering [2], DBSCAN [3]. Clustering algorithms like k-Means require guessing or identifying how many clusters might be present in the data. Even, with a correct guess k-means assumes a flat geometry i.e. each cluster can be lies on a flat manifold (see Fig 1C). k-Means fails if the geometry of the data resembles that shown in Fig 1A and Fig 1B.

Spectral clustering [2] can alleviate this problem by constructing a similarity graph of weighted edges, where edge weights are a function of distance between the given pair of points in the original feature space. A common weight matrix (W) used in spectral clustering is shown in eq. (1)

Date: March 12, 2022.

(A) Non-flat geometry (B) Non-flat geometry (C) Flat geometry

FIGURE 1. Cluster examples

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

where W_{ij} is the entry corresponding to the i^{th} row and j^{th} column of W, $||x_i - x_j||_2$ is the euclidean distance between points x_i and x_j and σ is a parameter to be tuned.

Then, the degree matrix D and the Laplacian (unnormalized) L for the similarity graph is given by eq (2) and eq (4)

$$d_j = \sum_{i=0}^{N-1} W_{ji} \tag{2}$$

$$L = D - W \tag{3}$$

where d_j is the j^{th} diagonal entry of the degree matrix D. This graph construction performs a non-linear transformation on the original input space and allows for clustering in the case of non-flat geometry.

The eigendecomposition of the graph Laplacian L reveals the number of clusters present in the data. Ideally the number of zero eigenvalues of L is equal to the number of clusters in the data. The first eigenvalue of L is always zero and the corresponding eigenvector $\mathbf{q_0}$ is $\mathbf{1}$. Following eigenvalues need not be zero but would still be very small. The first big jump in eigenvalues indicates the number of clusters in the data. For example, if n^{th} eigenvalue jumps significantly compared to $n-1^{th}$ eigenvalue, then there are $n-1$ clusters in the data. In case there are two clusters in the data, then sign of the entries of the second eigenvector denoted by $\mathbf{q_1}$, contains information about the identity of the cluster. In other words, if points j and k are in the same cluster, then j^{th} and k^{th} entries of the vector $\mathbf{q_1}$ are more likely to have the same sign. We use this to cluster politicians in our dataset. The clustering performance is assessed using the number of "misclassified" points. "Misclassified" here means a member who should have belonged to the opposite party/cluster.

$$clustering\,accuracy = 1 - \frac{number\,of\,misclassified\,members}{Total\,number\,of\,members} \tag{4}$$

In real world datasets, often there are a few examples with labels assigned to them. Labeling is a time consuming process, thus there is a vast majority of data that is unlabelled. *Semi-supervised learning* refers to a set of methods that can be used to generate proxy labels for the unlabelled samples using information from labelled samples.

One way to do accomplish this is *manifold regularization* or *Laplacian regularization*, since the eigenvectors of the Laplacian preserve meaningful information about the clusters in the data. This approach first constructs feature maps given by eigendecomposition of the graph Laplacian, L (eq (5)).

$$F(x_j) = ((\mathbf{q_0})_j, (\mathbf{q_1})_j, ..., (\mathbf{q_{M-1}})_j) \in R^M \tag{5}$$

where $\mathbf{q_i}$ denote the eigenvectors of the Laplacian matrix and $F(x_j)$ is the j^{th} row of the Laplacian embedding $F(X) \in R^{435\times M}$, of the input matrix $X \in R^{435\times 16}$

Then, these features are used to learn a map between $A \in R^{J\times M}$ i.e. the labelled subset of $F(X)$ and the corresponding labels $y \in R^J$, where J is the number of labelled examples and M is the number of features.

In this report, we assume that the first J samples are labelled and rest are unlabelled. We use *Linear Regression* [4] to a learn classifier on the labelled data. The sign of the output of the regressor is used as the predicted label of the data. *Linear regression* fits a linear model to the data i.e. the outputs are the weighted sum of the input features. The weights are the coefficients ($\hat{\beta} \in R^k$) of the linear model learned by minimizing the error between the predictions and the desired label, as shown in eq 6.

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \, ||A\beta - y||^2 \tag{6}$$

where $A \in R^{m\times k}$ is the input matrix with m training examples and n features, $y \in R^m$ are the associated ground truth.

Classification labels $\hat{y}$ for the entire dataset is then generated as shown in eq (7) and classification accuracy is given by eq (8)

$$\hat{y} = sign(F(X)\hat{\beta}) \tag{7}$$

$$SSL\ accuracy = 1 - \frac{number\ of\ misclassified\ members}{Total\ number\ of\ members} \tag{8}$$

3. Algorithm Implementation and Development

All the algorithms were implemented in Python [5]. Numpy package [6] was used for data array manipulations, sklearn [7] was used for Linear Regression. All the visualizations and figures were generated using matplotlib [8].

For the optimal weight matrix W parameter σ, we used grid search in the range $(0, 4]$ with 100 equidistant points. For each value of σ, we performed *30 trials*. The optimal value (σ^*) was chosen as the one with maximum mean classification accuracy and minimum variance in classification accuracy over 30 trials. This σ^* was later chosen to construct Laplacian embedding $F(X)$ for the Semi-supervised Learning part.

4. Computational Results

Fig 2 show the eigenvalues of the graph Laplacian L. We see that there is large jump in eigenvalue after second eigenvalue λ_1, indicating that there are two clusters in the data. Fig 3 shows the visualization of cluster assignment for different members using $sign(\mathbf{q_1})$ for $\sigma = 2.2$.

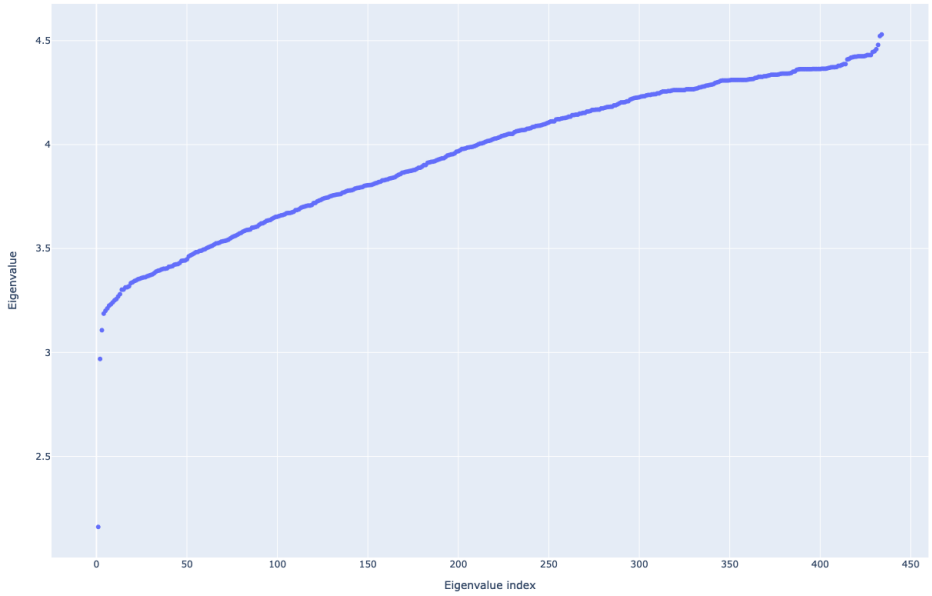

Figure 2. Eigenvalues of the Laplacian.

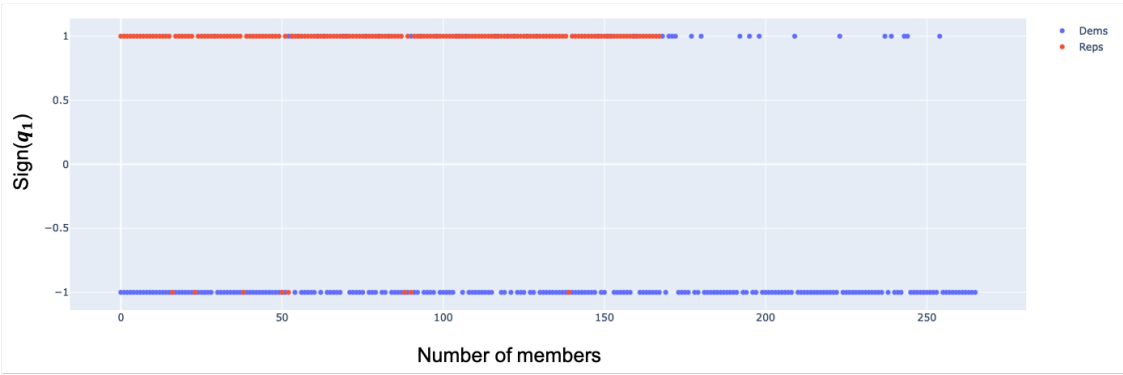

Figure 3. Cluster Assignment.

Fig 3 show the clustering accuracy as a function of σ. We find that for large values of σ i.e > 1.16, the clustering accuracy is relatively stable at 0.880. Large values of σ promote greater connectivity of the boundary points to the clusters. This prevents forming of "outlier" clusters with single points representing an entire cluster. Stable values of accuracy for large σ indicate that most of the sample in each of the clusters lie far inside the cluster boundaries. Thus, relative connectivity across clusters doesn't change much even with large values of σ. So, we randomly chose $\sigma^* = 2.2$ for the semi-supervised learning.

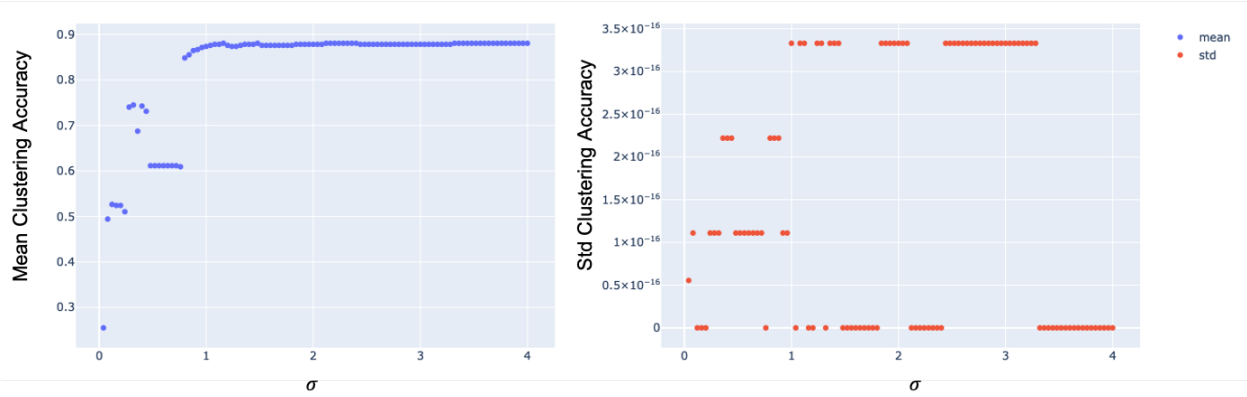

FIGURE 4. Clustering Accuracy vs σ.

Table 1 shows the classification performance of the semi-supervised learning algorithm, for different combination of number of features used (M) and number of labelled samples (J). We see that for a given number of labelled samples, the performance improves at first as we increase the number of features and then starts decreasing. In general, we observe that we need more number of samples to generalize well with greater number of features. This is expected from the bias-variance tradeoff.

	M = 2	M = 3	M = 4	M = 5	M = 6
J = 5	0.873	0.896	0.588	0.588	0.788
J = 10	0.871	0.862	0.919	0.827	0.804
J = 20	0.880	0.880	0.901	0.866	0.892
J = 40	0.878	0.876	0.903	0.882	0.871

TABLE 1. Classification Accuracy ratio for different number of features i.e. eigenvectors (M) and number of training samples (J).

However, for the semi-supervised learning part the classification performance is sensitive to the choice of σ. The optimal σ for Semi-supervised Learning depends on the value of J. So, our choice of σ^* need not be optimal. We did not perform joint optimization over values of σ, M and J in this work.

5. Summary and Conclusions

In this report we used spectral clustering to group politician based on their voting records. We saw that approximately 88 % of the politician are grouped with the members of their own party, if we use just the second eigenvector of Laplacian to assign the cluster. We also experimented with generating labels for held-out members using data labels from a few members and featurization using Laplacian embeddings. We found the labels to be highly accurate (greater than85%), even with small number of samples example ($J = 5$,$M = 3$). This indicated the potential of Laplacian embeddings in generating proxy labels for binary class dataset.

Acknowledgements

The author is thankful to Prof. Bamdad Hosseini for explaining the basics of Graph Theory, Spectral Clustering and Semi-supervised Learning, and providing useful examples with code. We are also thankful to Katherine Owens for promptly responding to the queries. Furthermore, the discussion thread on canvas was extremely helpful.

References

[1] H.-H. Bock, "Clustering methods: a history of k-means algorithms," *Selected contributions in data analysis and classification*, pp. 161–172, 2007.
[2] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
[3] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.
[4] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012.
[5] G. Van Rossum *et al.*, "Python programming language.," in *USENIX annual technical conference*, vol. 41, p. 36, 2007.
[6] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.
[7] 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*, vol. 12, pp. 2825–2830, 2011.
[8] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 03, pp. 90–95, 2007.