AMATH 582: HOMEWORK 4

CASSIA CAI

School of Oceanography, University of Washington, Seattle, WA
`fmc2855@uw.edu`

Abstract. We test the performance of spectral clustering and a semi-supervised regression algorithm to predict the political orientation of House members based on their voting records on 16 bills. Data used is the 1984 house voting records. To apply spectral clustering, we construct the unnormalized graph Laplacian matrix on the voting records using a weight function. We compute the Fiedler vector and compute its classification accuracy using the Fielder vector as our classifier. σ in the weight function is a parameter, and we determine the optimal σ to be 1.16 (accuracy of 0.88). We next apply semi-supervised learning, using the unnormalized Laplacian with the $\sigma = 1.16$. We then test combinations of different numbers of eigenvectors to keep and and numbers of rows corresponding to the labeled data points we want to keep. We find that the classification accuracy ranges from 0.729 to 0.892 with the highest accuracy with 6 eigenvectors and 5 rows of data. The mean classification accuracy is 0.853 ± 0.043. However, it is difficult to determine whether these accuracy values are statistically significant. This is because the semi-supervised regression algorithm is very sensitive to the choice of σ. For example, when we set $\sigma = 3.38$, the accuracy ranges from 0.520 to 0.926 with the highest accuracy at 6 eigenvectors and 40 rows of data. An extension is to test different σs with different combinations of numbers of eigenvectors and data points.

1. Introduction

Machine learning techniques can be applied to predict political orientation. The goal of this study is to test the performance of spectral clustering and a simple semi-supervised regression algorithm on the 1984 house voting records data. This data set, which consists of the voting records of 435 members of the House on 16 bills, has been widely used to develop and validate techniques like clustering aggregation, categorical clustering, and decision trees, and feature selection. Of the 435 members of the House, 267 are members of the democratic party and 168 are members of the republican party. We first apply spectral clustering to the data set, working mainly with the votes on the 16 bills to cluster the data into (1) democratic members and (2) republican members. We then validate with the party orientation from the 1984 data set. We also test a simple semi-supervised regression algorithm.

This kind of study is relevant today as the current political climate is divided between the liberal (the Democratic party) and conservative (the Republican party) ideologies. These two parties have different opinions on a number of social issues and have different priorities in terms of socio-economic goals. An extension, which involves compiling a new data set, is to evaluate the types of content of each member's social media feeds to predict political orientation, exploring the concept of an ideological bubble.

2. Theoretical Background

We have the choice to use supervised learning (which involves training the model on both features and labels) and unsupervised learning (which involves dividing the features into groups or clusters).

Date: March 9 2022.

For supervised learning, we can use regression methods (ex. linear regression, ridge regression, and kernel regression). For unsupervised learning, we can use spectral clustering. Another choice is to use semi-supervised learning (SSL).

SPECTRAL CLUSTERING PART 1: Spectral clustering uses the spectrum of the similarity matrix for dimensionality reduction. Clustering is defined simply as the unsupervised task of dividing the input data X into meaningful groups with the goal of finding meaningful structure in the data. One way to cluster is to use K-means. One caveat to using K-means is that it does not see low-dimensional structures in the data. In this study, we use spectral clustering.

The idea of spectral clustering begins by letting $X = \left\{\underline{x}_0, \ldots, \underline{x}_{N-1}\right\} \in \mathbb{R}^d$ be our dataset [1]. We want to find a feature map so that instead of using X, we use $F(X)$, which can lead to better clustering. Spectral clustering finds this mapping as the feature map of an implicit kernel that is not explicity computed. Key to understanding how spectral clustering works is the similarity graph.

Similarity graphs: If we have a dataset $\mathbf{X} = \{x_0, \ldots, x_{N-1}\} \in \mathbb{R}^d$ and a symmetric matrix $W \in \mathbb{R}^{N\times N}$ with non-negative entries $w_{ij} \geq 0$, then we define a (weighted undirected) graph $G = \{X, W\}$ where the $\underline{x}_j \in \mathbb{R}^d$ are the vertices of G and the entries w_{ij} of W denote weights that are associated to edges that connect $\underline{x}_i$ to $\underline{x}_j$. In this study, we consider proximity graphs, which is a family of weighted graphs. We define our weight functon as $\eta : [0, +\infty) \to [0, +\infty)$, a non-negative, non-increasing and continuous function at zero. $w_{ij} = \eta(\|\underline{x}_i - \underline{x}_j\|_p)$ for $1 \leq p \leq +\infty$ In this study, we choose:

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

$$w_{ij} = \exp\left(-\frac{\|\underline{x}_i - \underline{x}_j\|_2^2}{2\sigma^2}\right)$$

These are similarity graphs because the weights w_{ij} encode the similarily/closeness of the vertices.

Graph Laplacians: We define the graph Laplacian matrix of G by first defining the degree vector $\underline{d} \in \mathbb{R}^N$, $\quad d_j = \sum_{i=0}^{N-1} W_{ji}$, then the diagonal matrix:

$$= \text{diag}(\underline{d}) = \begin{bmatrix} d_0 & & & \\ & d_1 & & \\ & & \ddots & \\ & & & d_{N-1} \end{bmatrix}$$

and finally the unnormalized graph Laplacian.

$$\tilde{L} = D - W$$

We can also define the normalized graph Laplacian. The graph Laplacian matrices are non-negative definite and symmetric. They have have real eigenvalues and eigenvectores $\{\lambda_j, \underline{u}_j\}_{j=0}^{N-1}$ and at least one non-zero eigenvalue. If $\underline{u}$ is an eigenvector of $\tilde{L}$ then $D^{\frac{1}{2}}\underline{u}$ solves the generalized eigenvalue problem. Moreover, if the graph G has k-disconnected components, then

$$0 = \lambda_0 = \lambda_1 = \ldots = \lambda_{k-1} < \lambda_k \leq \lambda_{k+1} \leq \ldots$$

Thus, the number of zero eigenvalues of L, correspond to clusters in the graph [2, 3, 4]. Note that here we denote L, and $\tilde{L}$ as the normalized and unnormalized graph Laplacian respectively.

$$\tilde{L}_1 = D_1 - W_1, \quad \tilde{L}_1\mathbf{1} = \underline{d}_1 - \underline{d}_1 = 0$$

$$\tilde{L}_2 = D_2 - W_2, \quad \tilde{L}_2\mathbf{2} = \underline{d}_2 - \underline{d}_2 = 0$$

$$W = \begin{bmatrix} W_1 & | & 0 \\ 0 & | & W_2 \end{bmatrix} \quad \tilde{L} = \begin{bmatrix} \tilde{L}_1 & | & 0 \\ 0 & | & \tilde{L}_2 \end{bmatrix}$$

SPECTRAL CLUSTERING PART 2: The goal of spectral clustering is to construct a feature map $F : \mathbb{R}^N \to \mathbb{R}^M$ so that clustering the transformed data $F(X)$ is easier/more effective than directly clustering X.

Our first task is to construct the graph Laplacian on X and compute its eigen decomposition using the weight function defined above.

$$\tilde{L} = \tilde{Q}\Lambda\tilde{Q}^T$$
$$\tilde{Q} = \begin{bmatrix} \tilde{\underline{q}}_0 & \tilde{\underline{q}}_1 & \cdots & \tilde{\underline{q}}_{N-1} \end{bmatrix}, \quad \tilde{\underline{q}}_j \in \mathbb{R}^N$$

We define the feature map

$$M \geq 1 \quad \tilde{F}(\underline{u}_j) = \begin{bmatrix} \tilde{\underline{q}}_{1j} \\ \tilde{\underline{q}}_{2j} \\ \vdots \\ \tilde{\underline{q}}_{Mj} \end{bmatrix}, F(\underline{u}_j) = \begin{bmatrix} \underline{q}_{1j} \\ \underline{q}_{2j} \\ \vdots \\ \underline{q}_{Mj} \end{bmatrix}$$
$$\tilde{F} : \mathbb{R}^d \to \mathbb{R}^M$$

We then use its second eigenvector (the Fiedler vector) as our classifier. We can compute the classification accuracy by comparing the predicted labels with the provided labels. From previous studies, we know that the number of eigenvectors kept is the number of clusters in the dataset. The q_i represent eigenvectors of the Laplacian matrix. For binary clustering, we relabel $\mathbf{Y}$ to $-1, 1$. We extract the sign of the Fielder vector as our classifier. L_0, L_1 have 0 eigenvalues with corresponding constant eigenvectors. L has two 0 eigenvalues, i.e. $\lambda_0 = \lambda_1 = 0$,

$$L = \begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix} \begin{bmatrix} \mathbf{1}_{M_0} \\ 0 \end{bmatrix} = \begin{bmatrix} L_0 & 0 \\ 0 & L_1 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{1}_{N_1} \end{bmatrix} = 0$$

where $\begin{bmatrix} \mathbf{1}_{M_0} \\ 0 \end{bmatrix} = \tilde{\underline{q}}_0$ and $\begin{bmatrix} 0 \\ \mathbf{1}_{N_1} \end{bmatrix} = \tilde{\underline{q}}_1$
$\tilde{\underline{q}}_0$ and $\tilde{\underline{q}}_1$ are the indicators of the X_0 and X_1. In this study, we use the Fiedler vector, which is that second vector.

Accuracy: Accuracy $= 1 - (\frac{\#misclassified}{\#points})$

SEMI-SUPERVISED LEARNING: SSL uses a sub-set of features and labels.

$$\mathbf{X} = \{x_0, \ldots, x_{N-1}\} \in \mathbb{R}^d \quad , \mathbf{Y} = \{y_0(x_0), \ldots, y_{M-1}(x_{M-1})\} \in \mathbb{R}^M$$

where M is less than or equal to N. We therefore only have a subset of inputs and outputs. We want to use SSL to predict the outputs $y_0(x_0), \ldots, y_{M-1}(x_{M-1})$ given a subset of labelled data and features, essentially predicting the "unseen" labels [5].

We use linear regression (least squares) on the labelled dataset given $J \geq 1$ (number of samples in the model training).

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

A is the feature matrix or the Laplacian embedding $f(X_{ij})$ and b is the output of Y, which is the party affiliation. In this study, we consider the case where we choose M ≥ 1, where M is the number of eigenvectors we wish to keep from the graph Laplacian. We also consider an integer J ≥ 1, which represent the first J rows of F(X). With this, we apply linear regression to find $\hat{\beta}$. We can then find the predictor for each combination.

The predictor is: $\hat{y} = sign(F(X)\hat{\beta})$

We can then evaluate the accuracy of the predictor described in the clustering accuracy section.

3. Algorithm Implementation and Development

We first preprocessed the data by relabelling the labelled data to -1 or 1 for democrats and republicans. We then relabel votes to either -1, 0, or 1 depending on whether the vote was yes, no, or not available. We used python, and the package used is scikit-learn's LinearRegression. We first defined the weight function, constructed the graph Laplacian, computed the eigendecomposition, used the Fiedler vector as the predictor, and calculated the clustering accuracy.

I coded Python and used NumPy and scikit-learn for mathematical calculations and Matplotlib for visualizations [6, 7].

4. Computational Results

For spectral clustering, we construct the graph Laplacian, computed the eigendecomposition, and took the sign of the second eigenvector (the Fiedler vector) as our classifier. In our weight matrix, σ is a parameter. We test σ values ranging from 0 to 4. We found the optimal $\sigma = 1.16$ (Figure 1). A σ of 1.16 achieves an clustering accuracy of approximately 0.88. From Figure 1, we observe that past some threshold σ value, the accuracy plateaus. We also plot the Fiedler vector by re-ordering according to the original party affiliations (Figure 2). The helps visualize the behavior of the Fiedler vector on the two clusters. The entries of the Fiedler vector indicate connectivity strength in a the graph and is often used in spectral graph partitioning.

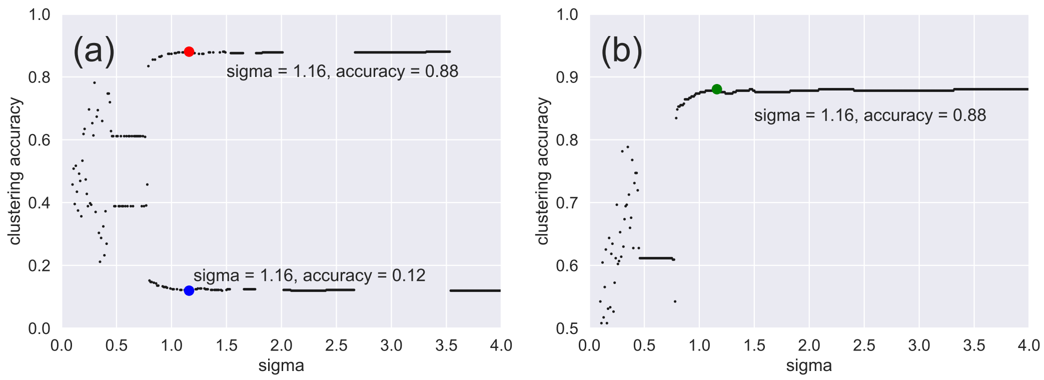

Figure 1. (a) clustering accuracy versus sigma values, (b) clustering accuracy versus modified sigma values. Sigma values now range from 0.5 to 1.0.

For SSL, we computed the graph Laplacian with the σ determined in the spectral clustering step. We set $\sigma = 1.16$. We then compute the eigendecomposition. We then apply Laplacian embedding, choosing to use different combinations of numbers of eigenvectors and different numbers of rows of data. Note that M in Table 1 represents the number of eigenvectors we consider while J represents the number of rows of data. From Table 1, we observe that the range of accuracies is rather small, ranging from 0.729 to 0.892, and that the most accurate combination is M = 6 and J = 5. However, we also noticed that this is very sensitive to the choice of σ. A different σ works better for different combinations of M and J. Also, note that we are tuning all these parameters at once. So the optimal σ value determined before is not necessarily optimal in all these combinations. For example, if we use $\sigma = 1.05$, then we can get a higher accuracy of 0.93 for the combination M = 6 and J = 40. If we use $\sigma = 3.43$, then we get an accuracy of 0.90 for the combination M = 6 and J = 40. This goes to show that the optimal σ determined previously is not necessarily the best σ to use here for the different M and J combinations Table 2.

An extension to this would be to test different values of σ for the different combinations to see what is the optimal σ and to decide what is the best combination to achieve the highest accuracy given that the SSL is sensitive to the choice of σ.

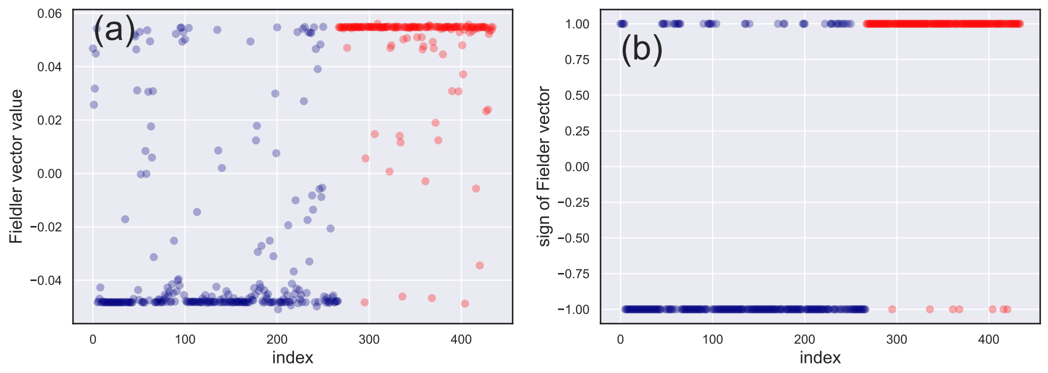

FIGURE 2. (a) the Fielder vector values versus party member index, (b) sign of Fielder vector vs. party member index. Red indicates republicans. Blue indicates democrats

J \ M	2	3	4	5	6
5	0.8897	0.8897	0.8391	0.8667	0.8920
10	0.8874	0.8207	0.8575	0.7540	0.7287
20	0.8828	0.8230	0.8644	0.8414	0.8759
40	0.8805	0.8391	0.8759	0.8782	0.8644

TABLE 1. Table summarizing the accuracy (in percentage) of $\hat{\mathbf{y}}$ as our classifier for M = 2, 3, 4, 5, and 6 (representing different numbers of eigenvectors) and J = 5, 10, 20, and 40 (representing the number of rows corresponding to the labeled data points we keep) using $\sigma = 1.16$.

J \ M	2	3	4	5	6
5	0.8644	0.9011	0.5586	0.5333	0.5218
10	0.8644	0.8552	0.9356	0.6667	0.5425
20	0.8782	0.8966	0.9034	0.9011	0.8897
40	0.8782	0.8897	0.9103	0.9195	0.9034

TABLE 2. Same as Table 1 but with $\sigma = 3.43$.

5. SUMMARY AND CONCLUSIONS

We tested the performance of spectral clustering and a simple semi-supervised regression algorithm on the 1984 house voting records data. For both methods, we constructed the graph Laplacian, computed the eigendecompositions, and used the Fiedler vector as our classifier. From spectral clustering, we found the first optimal σ to be 1.16. We also observed that past a certain threshold σ value, the accuracy seems to stabilize in the high 0.80s. For SSL, we used what we determined to be the optimal $\sigma = 1.16$, computed the graph Laplacian, use graph Laplacian embedding considering different numbers of eigenvectors and different numbers of rows of dataset. We then applied linear regression. With $\sigma = 1.16$, we find that the accuracies have a small range (0.729 to 0.892), with the highest accuracy for M = 6 and J = 5. However, when we use a different σ value, we are able to get a higher accuracy of 0.92 for M = 5 and J = 40. This means that our simple semi-supervised regression algorithm is sensitive to our choice of σ. A natural extension is to test different values of σ and their effects on the accuracies of different combinations of M and J. This way, we can then decide which combination results in the highest accuracy.

Acknowledgements and Code Availability Statement

The author is thankful for the algorithm-implementation suggestions from peers in AMATH 482-582. The code is publicly available on this Github repository after the submission date.

References

[1] Bamdad Hosseini. Spectral clustering. University of Washington-Seattle (LOW 216), March 2022. AMATH 482/582.

[2] Bamdad Hosseini. Introduction to graph laplacians. University of Washington-Seattle (LOW 216), March 2022. AMATH 482/582.

[3] M. Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 25:57–70, 1989.

[4] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 12:298–305, 1973.

[5] Bamdad Hosseini. Semi-supervised learning. University of Washington-Seattle (LOW 216), March 2022. AMATH 482/582.

[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] 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.