HW4: Classifying Politicians

Jessica Jones

Mar. 11, 2022

Abstract

We are tasked with performing spectral clustering and semi-supervised regression algorithm on voting data of the House of Representatives from 1984.

1. Introduction and Overview

The data set consists of voting records of 435 members of the House on 16 bills, such as whether to keep religious groups in schools, whether to freeze physician fees, whether to give aid to Nicaraguan contras, and crime. There are 267 members of the democratic party and 168 members of the republican party. The voting record of each house member on the 16 bills will be our input x while the corresponding output/class y is that members party affiliation (republican or democrat embedded as ± 1). We are tasked with performing spectral clustering to classify politicians based on their voting behavior and create an algorithm to see whether we can predict future voting behavior from House members.

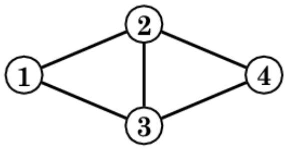
2. Theoretical Background

2.1. Spectral Clustering

Spectral clustering corresponds to a computationally tractable solution to the graph partitioning problem. It's used to identify communities of nodes in a graph based on the edges connecting them, which is very useful when clustering non-graph data.

2.2. Graph Laplacian

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix}$$

Figure 1: Laplacian matrix L with n = 4 nodes.

The Laplacian matrix is defined by node degrees and adjacency relationships between nodes Consider this Laplacian matrix L in **Figure 1.** The **A** matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph, and is computed by the weight function:

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

(1)

with variance parameter σ. This equation is similar to the Gaussian kernel similarity function, although in that case, $\sigma^2 \to \infty$ results in an adjacency matrix.

The **D** matrix is the sum of the adjacency matrix **A** that computes the number of edges attached to each vertex. The graph Laplacian is then the difference between the **D and A** matrices, such that:

$$L = D - A \tag{2}$$

L is symmetric, where the rows and columns sum to 0. Since it sums to 0, its implied that its eigenvalue λ_1 is also 0 with an eigenvector of $1 \in \mathbb{R}^n$.

2.3. Fiedler Vector

The Fiedler vector is the eigenvector corresponding to the second smallest eigenvalue $\lambda_2 > 0$ (i.e., the algebraic connectivity) of the Laplacian matrix of a graph. It minimizes the distance between the connected vertices in the original graph. Negative values indicate poor connectivity in a graph or, in this case, a cluster of politicians.

2.3. Laplacian embedding

Laplacian embedding focuses on preserving the local geometry in which nearby points in the original space remain nearby in the reduced space.

$$F(xj) = \big((q_0)j, (q_1)j, \dots, (q_{M-1})j\big) \in R^M \tag{3}$$

where q_j denote the eigenvectors of the Laplacian matrix. We write $F(X) \in R^{435\ X\ M}$ for the Laplacian embedding of X, i.e., the matrix whose j-th row is $F(\mathrm{x}j)$.

2.4. Least Squares of a graph

Given a set of alternatives to be ranked, and some pairwise comparison data, ranking is a least squares computation on a graph. The vertices are the alternatives, and the edge values comprise the comparison data. We use the equation:

$$\hat{\beta} = argmin_{\beta \in R^M} \|A\beta - b\|_2^2 \tag{4}$$

to find the least square error, then take:

$$\hat{y} = sign(F(X)\hat{\beta}) \tag{5}$$

as the predictor of classes of all points in X.

3. Algorithm Implementation and Development

3.1. Preprocessing

Our first step is to load the voting datasets and assign labels {-1,+1} to members of different parties. Then I constructed the input vectors xj corresponding to the voting records of each member by replacing 'y' votes with +1, 'n' votes with −1 and '?' with 0. This was done via **numpy.loadtxt()** function. I assigned -1 to republicans and 'n' voting decisions, and +1 to democrats and 'y' voting decisions. I then indexed the appropriate X and Y datasets. This gave matrices:

$$x \in \mathbb{R}^{435\,X\,16},\ y \in \mathbb{R}^{435} \tag{6}$$

to denote the input matrix of x features of voting categories and the vector of y party affiliations *respectively*

3.2. Spectral Clustering

In this step we worked with the matrix X and used y for validation of our clustering algorithm. To do this I wrote several functions to accomplish this:

- To compute the unnormalized graph Laplacian matrix on X, I created the function **unnorm_lap()** that takes in the distance matrix X (since the Graph Laplacian is symmetric) and sigma values. It first creates an adjacency matrix **A** using **Equation 1**. I then compute the degree matrix **D** using **numpy.diag().** I then constructed the Graph Laplacian **(Equation 3)** and computed its second eigenvector (i.e., the Fiedler vector) which we denote as **Q** using **numpy.linalg.eigh()** of **L**. It's helpful to plot **Q** by re-ordering **X** according to the original party affiliations to visualize the behavior of Q on the two clusters, so I sorted them.
- To assess accuracy of spectral clustering performance as I iterated through sigmas [0,4), I created the function **accuracy_()** that takes **sign(Q)** as your classifier and computes its clustering accuracy after comparison with vector **y** using the equation:

$$clustering\ accuracy = 1 - (\frac{1}{435}) * \#\ of\ missclassified\ members \tag{7}$$

  "misclassified" here means a member who should have belonged to the opposite party/cluster. Since this is an unsupervised learning task, **sign(Q)** might disagree with y by a negative sign, that is, the +1 class may be assigned to −1 and vice versa. We accounted for this by **"max(accuracy, 1 – accuracy)".**
- To assess the best accuracy I created the function **acc_best()** that accepts a list of accuracies for each q1 compared to y and I plotted them with **plot_acc()** that takes in the sigmas, the accuracies, the best sigma value and the best accuracy that corresponds to. $\sigma *$ denotes the optimal variance parameter achieving maximum clustering accuracy.

3.3. Semi-supervised Learning

We conducted semi-supervised learning via linear regression using our unnormalized Laplacian and the best sigma value we found from spectral clustering. To accomplish this, I created the function **ssl_acc().**

- Given an integer J ≥1 consider the submatrix A ∈ R $^{J \times M}$ and vector b ∈ R ^{J} consisting of the first J rows of $F(X)$ and y

$$A_{ij} = F(X)_{ij},\ i = 0, \dots, J-1,\ j = 0, \dots, M-1,$$
$$b_i = y_i,\ i = 0, \dots, J-1. \tag{8}$$

- This function uses linear regression (least squares) to find Beta **b** using **Equation 4**, then take y as predictor of classes of all points in X using **Equation 5**. It also provides a table summarizing the accuracy of $\hat{y}$ as our classifier

4. Computational Results

4.1. Results from iterating through sigma ranges and the best accuracy

We can see that the accuracy of the spectral clustering increases after sigma ≈ 0.8 The accuracies fall dramatically from (0.8, 0]. Accuracies stabilize (0.8, 4.0). The maximum accuracy occurs when sigma = 2.1. The accuracy at sigma = 2.1 is 88.05%, which indicates that even though our spectral clustering seems to be unstable its still good.

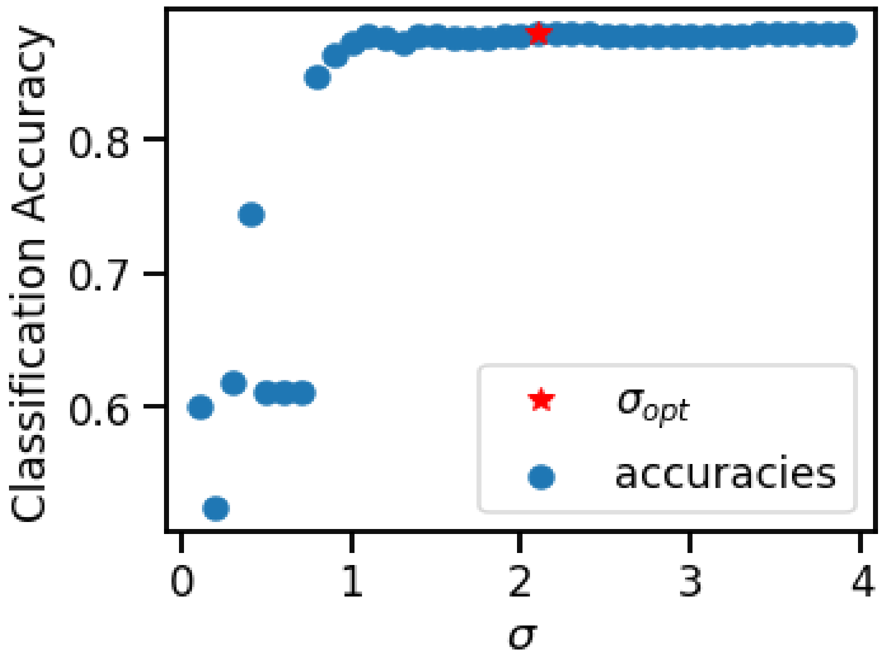


Figure 2: Graphical Representation of Classification Accuracies Based on Spectral Clustering

4.2. Results from Prediction Accuracies

Accuracies	M_2	M_3	M_4	M_5	M_6
J_5	0.86436788	0.90574713	0.68045977	0.68735632	0.69885057
J_{10}	0.86436782	0.86666667	0.92183908	0.88275862	0.67356322
J_{20}	0.87586207	0.88735632	0.91494253	0.90804598	0.89195402
J_{40}	0.87816092	0.88965517	0.90114943	0.92873563	0.91724138

Table 1: Table summary of accuracies of $\hat{y}$ as a classifier for M = 2,3,4,5,6 and J = 5,10,20,40.

Taking Table 1 into consideration, with the green box labeling the lowest rate of mislabeled politicians and orange box labeling the highest rate of mislabeled politicians with $\sigma_{2.1}$, our semi-supervised learning session utilizing linear regression performed relatively well predicting y.

5. Summary and Conclusions

Using Graph theory, we can cluster politicians and assess their voting behavior, while also comparing groups of politicians and discovering how similar each group behaves. Our spectral clustering algorithm utilizing Laplacian embedding does a good job of initial classifying politicians, and, using the best accuracies and sigma corresponding to it, we can further create a semi-supervised algorithm using simple linear regression to create a model to future voting behavior.

6. Acknowledgements

References

Graphs, L.A., Dept. of Computer Science, University of Western Australia 1984, Department of Computer Science, Monash University 1986, and (HTML) at csse, Monash 1999, https://users.monash.edu/~lloyd/tildeAlgDS/Graph/

M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, Computational Geometry, Algorithms and Applications, 2nd edition, Springer.
http://math.sfsu.edu/beck/teach/870/brendan.pdf