AMATH 482/582 HOMEWORK 4 - CLASSIFYING POLITICIANS

PHILIP WESTPHAL

Department of Applied Mathematics, University of Washington, Seattle, WA
philw16@uw.edu

Abstract. In this report, the performance of an unsupervised Spectral Clustering method and semi-supervised linear regression models on a real world voting history dataset were explored. The performance of the Spectral Clustering method was tested using varying values of a parameter σ in a Gaussian weight function, and the optimal σ values identified. Semi-supervised learning models were built using varying numbers of rows and columns from the eigenvector matrix found during Spectral Clustering for each optimal σ. The performance of these models was compared.

1. Introduction and Overview

The goal of this problem was to apply Spectral Clustering and semi-supervised learning methods to classify political party affiliation based on voting history. A dataset representing the voting history for all 435 members of the US House of Representatives from 1984 was provided, giving the party affiliation and voting history on 16 specific bills. The party affiliation was broken off to represent the target variables y, and the voting information was taken as the input features X. Each vote on the bills was given as 'y' for yes, 'n' for no, and '?' in all other cases such as absence or abstaining. To convert the text data to a numeric representation, a value of -1 was assigned for 'n' votes, a value of 1 for 'y' votes, and 0 for the '?' values. Similarly, the targets were converted by assigning a value of -1 to Republican party affiliation and 1 to Democratic party affiliation.

Spectral Clustering is a useful unsupervised learning methodology with applications in image segmentation, entity resolution, and speech separation [1]. For binary classification cases, training a model on the data is not even required as all the information can be obtained from the breakdown of the Graph Laplacian matrix described in 2.1. For non-binary problems, the breakdown of the Graph Laplacian matrix can reduce the dimensionality of the dataset and identify the number of clusters to use [2].

Semi-supervised learning methods are incredibly important as labeling data is often expensive and time consuming in real world applications. As a result, it is advantageous to fit models on a small sample of labeled data points and use those results to generate predictions for the entire dataset. This can be tricky though, as with small samples of labeled data there may be insufficient information for the model to fit well on the input features. However, utilizing the decomposition of the Graph Laplacian matrix or other preprocessing methods to reduce the input dimensionality can lead to better performing models.

2. Theoretical Background

2.1. Graph Laplacians

A weighted, undirected graph for a dataset $X \in \mathbb{R}^d$ is defined as

$$G = \{X, W\}, \ W \in \mathbb{R}^{N,N} \tag{1}$$

where X_j are the vertices of G and W_{ij} represents the weights associating two edges X_i, X_j [2]. For the proximity graphs considered in this problem, a weight value of zero indicates no connection between two edges, while large values indicate a strong connection. The weight matrix of proximity graph G can generated by applying a non-negative, non-increasing, and continuous at zero function η to a pairwise distance matrix

Date: March 12, 2022.

for X calculated using a norm of the dataset [2]. Here X' signifies that we are taking the distance from X with respect to all points in X

$$W = \eta(\|X - X'\|_p),\ 1 \leq p \leq \infty \tag{2}$$

In this problem, the function η used is a Gaussian,

$$\eta(t) = \exp(\frac{-t^2}{2\sigma^2}) \tag{3}$$

Utilizing W the Graph Laplacian Matrix L can then be defined. The vector $d \in \mathbb{R}^N$ represents the row sums of W, and the matrix $D \in \mathbb{R}^{N,N}$ is the diagonalization of these sums

$$D = \text{diag}(d),\ d_j = \sum_{j=0}^{N-1} W_{ji} \tag{4}$$

With W and D, the unnormalized Graph Laplacian Matrix is calculated as [2]

$$L = D - W \tag{5}$$

The Graph Laplacian Matrix has a lot of properties that are useful for clustering. First, the matrix L is non-definite symmetric (NDS) and thus has real eigenvalues ($\lambda_j \geq 0$) and eigenvectors. Therefore, an eigendecomposition of L exists as

$$L = Q\Lambda Q^T \tag{6}$$

where Λ is a diagonal matrix with the eigenvalues of L, and the columns of Q are the corresponding eigenvectors [3]. The matrix L also has at least one eigenvalue $\lambda = 0$, and the total number of zero valued eigenvalues represents the number of disconnected components in a graph. Geometrically, this represents the number of distinct subgraphs that cannot be connected by any combination of edges in G. Therefore for clustering applications, the number of zero valued eigenvalues in L determines the number of disconnected subgraphs in G and the number of distinct clusters in X [2].

2.2. Spectral Clustering

Using the eigenvector matrix Q of equation 6, a feature map $F(X)$ can be defined as

$$F(X) = (Q_{jm})_{j=1,\ldots,J,\ m=1,\ldots,M} \tag{7}$$

where M represents the total number of eigenvectors to keep and J represents the number of observations in X. The initial eigenvector is ignored as it is always related to a zero valued eigenvalue and thus doesn't provide useful information [4]. Clustering algorithms such as K-means can then be applied to $F(X)$ to cluster the data points, where the number of clusters is chosen by the number of zero eigenvalues in L. One important note is that the feature map data should be normalized to the range $[-1, 1]$ to prevent issues that occur in certain clustering algorithms [2]

$$F(X_j) = \frac{F(X_j)}{\max(|F(X_j)|)} \tag{8}$$

The Fiedler Vector is defined as the eigenvector corresponding to the second smallest eigenvalue of L. As every L contains at least one eigenvalue of zero, the Fiedler Vector will correspond to the eigenvector of the first non-zero eigenvalue. Mathematically, the Fiedler Vector can be calculated as [5]

$$\begin{aligned} \min_x \quad & x^T L x \\ s.t. \quad & x^T \hat{\mathbf{1}} = 0 \\ & x^T x = c,\ c \in \mathbb{R} \end{aligned} \tag{9}$$

and partitions the vertices of the graph G into distinct subgraphs [6]. From the first constraint in equation 9, it can be seen that for a binary problem the cluster predictions $\hat{y}$ are simply determined by the sign of the Fiedler Vector

$$\hat{y} = \text{sign}(x_{fied}) \tag{10}$$

This is an unsupervised learning method, meaning that the outputs are determined without considering or learning from any target variables. In this problem, the outputs can be validated using the target vector y, however this is not the case in many real world problems.

2.3. Semi-Supervised Learning

From [7], the Least Squares Linear Regression model involves finding a β vector that optimizes

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \frac{1}{2\sigma^2} \left\| A\beta - y \right\|_2^2 \tag{11}$$

However, this model assumes that all the rows in $A \in \mathbb{R}^{N,M}$ have a corresponding target value in $y \in \mathbb{R}^N$. In many cases, all the data will not have corresponding targets and the model needs to be adjusted to generate the coefficients using only the labeled data points. Considering K labeled data points in y with $K << N$, the coefficient vector $\hat{c}$ is generated for considering only those rows

$$\hat{c} = \underset{c \in \mathbb{R}^M}{\operatorname{argmin}} \frac{1}{2\sigma^2} \sum_{k=0}^{K-1} |A_k^* c_k - y_m|_2^2, \ A^* \in \mathbb{R}^{K,M}, y \in \mathbb{R}^K \tag{12}$$

The coefficient vector can then be used to predict on the full dataset

$$\hat{y} \approx A\hat{c}, \ A \in \mathbb{R}^{N,M} \tag{13}$$

This is the general semi-supervised linear regression problem. Combining this method with spectral clustering as a regularization method can also be considered, and is known as a manifold regularization of the data [8]. This involves generating $\hat{c}$ using the labeled data points and a subset of columns from the Graph Laplacian eigenvector matrix Q. Considering K labeled points in the dataset, the following optimization problem exists using the first J eigenvectors of Q

$$\hat{c} = \underset{c \in \mathbb{R}^J}{\operatorname{argmin}} \frac{1}{2\sigma^2} \sum_{k=0}^{K-1} |\sum_{j=0}^{J-1} c_k Q_{kj}^* - y_k|_2^2, \ Q^* \in \mathbb{R}^{K,J} \tag{14}$$

The full dataset predictions can then be generated by applying the coefficients to all rows of Q with the first J eigenvectors. To measure the performance of the binary classifications in both the Spectral Clustering and semi-supervised learning cases, the accuracy was calculated as follows

$$\text{Accuracy} = 1 - \frac{\#\text{ of misclassified points in sign}(\hat{y})}{length(y)} \tag{15}$$

where y represents the target variables.

3. Algorithm Implementation and Development

Algorithm 1 describes the creation the of the unnormalized Graph Laplacian Matrix L and the corresponding eigenvector matrix Q. Note that input matrix X should be a pairwise distance matrix as specified in 2. This can be generated using the `distance_matrix()` function from `scipy`, and for this problem the value $p = 2$ was used.

Algorithm 1 Generate Eigenvector Matrix

```
function GENERATEUNNORMALIZEDLAPLACIAN(X, $\sigma$)
    Weights = $\eta(X, \sigma)$                    ▷ Using equation 3
    D = diag(rowsums of weights)
    L = D - W
    $\lambda$, Q = eigendecomposition(L)
    Sort $\lambda$, Q                              ▷ numpy by default doesn't use ascending order
    return $\lambda$, Q
end function
```

Algorithm 1 was ran for a range of $\sigma \in [0.1, 4)$ to find the optimal value(s) by classification accuracy. Equation 15 is calculated using algorithm 2 for each value of σ. Because this is an unsupervised learning task, the predictions may be off by a negative sign and this function accounts for that fact.

Algorithm 2 Calculate Accuracy

```
function calculateAccuracy(Q, y)
    predictions = sign(second column of Q)
    Calculate accuracy using equation 15
    final accuracy = max(accuracy, 1 - accuracy)                ▷ Account for potential -1 offset
    return final accuracy
end function
```

The optimal values found for σ (by algorithm 2) and their corresponding eigenvector matrices Q were then used in a semi-supervised learning model. Each Q matrix was fit using a `LinearRegression()` model from `sklearn` [9] keeping a varying number of rows and columns. The σ that provided the highest classification accuracy across all scenarios was determined to be the optimal value σ^*. This process is detailed in algorithm 3

Algorithm 3 Find Optimal Sigma & Accuracy

```
J vals $\in [5, 10, 20, 40]$                                    ▷ Rows to keep
M vals $\in [2, 3, 4, 5, 6]$                                    ▷ Cols to keep
for $\sigma$ in best sigmas do
    for j in J vals do
        for m in M vals do
            Q = subset of Q with j, m
            LinearRegression() fit (Q, labeled y vals)
            Predict using Q with all rows, m cols
            Compute accuracy of predictions using equation 15
            if accuracy < best accuracy then                    ▷ Update best values
                best accuracy = accuracy, $\sigma^* = \sigma$
            end if
        end for
    end for
    results = accuracy $\forall$ m, j
    if best accuracy in results then                            ▷ Update final result set with best results
        best results = results
    end if
end for
```

4. Computational Results

Figure 1 shows the classification accuracy measured using algorithm 2 for each value of $\sigma \in [0.1, 0.11, ..., 3.99]$. It can be seen that small values of σ yield accuracy values around 50%, indicating the Fiedler Vector is not providing enough information to perform better than a 50/50 guess. The values increase fairly rapidly, but then flatten out and plateau right around the first optimal σ value found. Additional values of σ past the initial optimal value reach the peak accuracy but do not exceed it, indicating that testing further σ beyond 4 would likely provide no accuracy gain.

Similarly from figure 1, it can be seen the that for the first optimal σ value most of the party affiliations are clustered well on one side of the decision boundary. Items that are closer to the decision boundary could be due to missing votes, and items on the opposite side of the decision boundary likely indicate voting alignment with the opposite party on a number of bills. One important note is that the clusters are of the opposite sign as the initial targets, with the Republican cluster being positive and the Democrat cluster being negative. This is due to the fact that the Spectral Clustering method is unsupervised, and the results can thus be off by a negative sign.

The optimal value σ^* from algorithm 3 was found to be 2.25. Table 1 shows the classification accuracy of the linear regression model predictions keeping a varying number of rows j and columns m from the eigenvector matrix Q

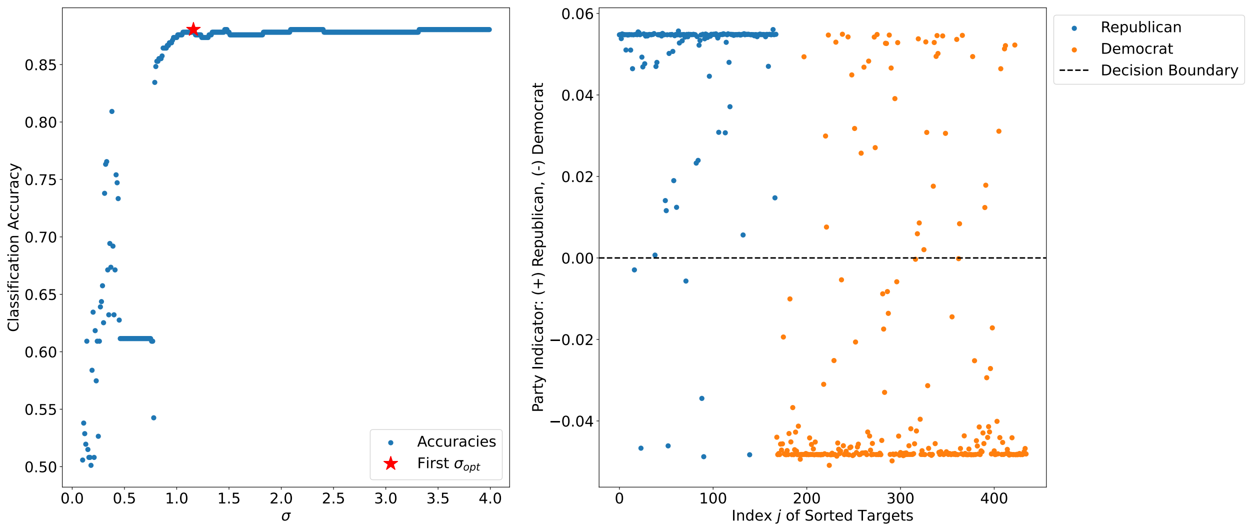

FIGURE 1. Classification Accuracy by σ & Classification Performance of $\sigma = 1.16$ (First Optimal)

	$m = 2$	$m = 3$	$m = 4$	$m = 5$	$m = 6$
$j = 5$	0.8736	0.8966	0.40	0.4115	0.8598
$j = 10$	0.8713	0.8598	0.9172	0.9287	0.8046
$j = 20$	0.8805	0.8805	0.9011	0.8621	0.8897
$j = 40$	0.8759	0.8759	0.8966	0.8713	0.8690

TABLE 1. Classification Accuracy of Linear Regression Models with varying rows, columns of Q

The data show that the optimal model had a classification accuracy of 92.87%, and was found keeping $M = 5$ columns and $J = 10$ rows of Q. From the first column of the table, we can see that the models using only the first two columns were able to correctly classifying nearly 90% of the politicians regardless of the number of rows kept. This makes sense given what is shown in figure 1, as in an unsupervised case the Fiedler Vector had a similar classification accuracy using equation . This indicates a large amount of information needed for the classification task is contained in the Fiedler Vector, and thus a linear regression model fit on the first two columns should reach similar results. Similarly, this is also likely why the data show adding additional columns did not yield significant accuracy improvements and in many cases hurt accuracy. The model has most of the information needed to predict the outputs from the Fiedler Vector, and adding columns could lead the model to assign unnecessary weight to them.

Interestingly, the number of rows generally did not seem to have a large impact on the overall classification accuracy. This again could be due to the fact that the model is able to learn enough from the sign of the Fiedler Vector as long as it sees points from both classes. However, there is more variation observed in the classification accuracy based on the number of rows kept for $m > 3$, and the two large decreases in classification accuracy for $j = 5$ and $m \in 4, 5$ especially stand out as being inconsistent with the rest of the data. This could also be due to the model overweighting the additional columns added, as with a small number of labeled rows it can't learn the additional columns are less useful. It could be useful to compile the results for all optimal values of σ to see if this sharp decrease is observed for other values.

5. Summary and Conclusions

In this problem, the application of spectral clustering and semi-supervised learning to classify political parties based on voting history was explored. It was found that using the sign Fiedler Vector of the eigenvector matrix Q of the Graph Laplacian served as a strong classifier, reaching a classification accuracy of nearly 90%. Fitting a linear regression model to the Q matrix with varying numbers of rows and columns kept in a semi-supervised learning scenario produced only marginally better results, reaching a maximum of 93% classification accuracy across all cases. The lack of significant improvement is likely due to the strength of the sign of the Fiedler Vector as a classifier on its own. In future experiements, it would be interesting to see the impact of a regularized model such as a Ridge or Lasso Regression in a semi-supervised learning scenario to see if similar results are observed. Additionally since in this case there are labels for all the data points, comparing the results of the semi-supervised cases to a full fit of the data could also be explored.

Acknowledgements

The author would like to thank Professor Bamdad Hosseini for covering the background concepts of Graph Theory, Spectral Clustering, and Semi-Supervised. Avi Joshi provided assistance with troubleshooting code issues and insight into looking at accuracy results for multiple optimal σ. Ryan Mouw also provided useful info about the variations in classification accuracy of the semi-supervised learning methods. The numpy [10] and matplotlib [11] packages in python were also useful for processing data and generating visualizations.

References

[1] S. V. Suryanarayana, D. G. V. Rao, and D. G. V. Swamy, "A survey : Spectral clustering applications and its enhancements," 2015.

[2] B. Hosseini, "Introduction to graph laplacians," University of Washington, Feb 2022, aMATH 482/582.

[3] R. Mittal, "Lecture 7: Positive semide nite matrices - cse - iit kanpur." [Online]. Available: https://cse.iitk.ac.in/users/rmittal/prev_course/s18/reports/7psdmatrices.pdf

[4] B. Hosseini, "Spectral clustering," University of Washington, Feb 2022, aMATH 482/582.

[5] A. Benson, "Cs224w: Graph clustering." [Online]. Available: https://snap.stanford.edu/class/cs224w-2020/

[6] B. Slininger, "Fiedler's theory of spectral graph partitioning."

[7] P. Westphal, "Amath 482/582 homework 2 - classifying digits," Feb 2022.

[8] B. Hosseini, "Semi-supervised learning," University of Washington, Feb 2022, aMATH 482/582.

[9] 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.

[10] 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*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[11] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.