AMATH 482: Home Work 4

Avinash Joshi
bwaseous@uw.edu
AMATH 482 Wi 22
11 March, 2022

Abstract

Supervised learning and related advancements have brought about a world wide boom in innumerable fields, but one part of the process, processing and labelling information, is a costly yet mandatory part of the this advancement. This assignment employs spectral clustering and semi-supervised learning methods on a set of voting records of sixteen bills for the 435 representatives of the 1984 House of Representatives in order to predict political party membership - Republican or Democrat. Using spectral clustering methods, numerous optimal σ^* values were found that produced the best classifying results, Figure 1, and, using an arbitrary σ^*, produced the results in Figure 2 with about 11.5%[1] of representatives mislabelled. The entire set of σ^*'s were then employed in manifold regularization or graph Laplacian regularization using various subsets of the entire Laplacian embedding of the voting data. Using $\sigma^* = 3.60$, $M = 5$, and $J = 40$, the minimum mislabelled percentage was 7.13%, with the full results outlined in Table 1..

1 Introduction and Overview

The purpose of this assignment is to predict the party affiliation of the 267 members of the democratic party and 168 members of the republican party for the 435 representatives of the 1984 House of Representative given their voting records on sixteen (16) different bills. The voting data is 435×17 with the first column specifying their party as either republican or democrat, and the following 16 labelled either "y" for yay, "n" for nay, and "?" for no vote. Before any analysis and clustering can be done, the data is transformed into numbers where each parameter is encoded as a specified value. The parameters are encoded as follows: Republican = +1, democrat = -1, "y" = +1, "n" = -1, and "?" = 0. The data is then divided into the labels $y = \{+1, -1\} \in R^{435\times 1}$ and $X = \{+1, 0, -1\} \in R^{435\times 16}$. An unnormalized graph Lapalcian, $\tilde{L}$, is constructed using a Gaussian weight matrix, W, defined by the Cartesian distance between any two of the voting logs of a representative,

[1]Percentages will be used when in analysis for ease of reading but will be represented as decimals in all figures and tables.

and various values of the separating parameter $\sigma \in (0, 4]$. Since there is known to be two classes, republican and democrat, it is expected that there will be two clusters. By taking the eigenvalue decomposition of $\tilde{L} = Q\Lambda Q^T$, and the sign of the first eigenvector, q_1, the party of each representative can be predicted. The cluster accuracy for some σ is then computed by taking the ratio of the mislabelled representatives by the total number of representatives.

Using this optimal σ^* found before, a semi-supervised learning task can be tackled. In this semi-supervised learning task, the labels of the entire House of Representatives will be predicted using a subset of the representatives and using only a few number of bills. In this problem, the Laplacian embedding of X, called $\tilde{F}(X) \in R^{435\times M}$, where M is the number of eigenvectors chosen, or the eigenvectors of the results of each bill. A subset of $\tilde{F}(X)$ is taken, called A, with only the first J representatives chosen to train the classifier on. The resulting $A \in R^{J\times M}$ is used to then train a classifier, $\hat{\beta}$, using least squares. This classifier can then be multiplied with $\tilde{F}(X)$ to predict party membership, $\hat{y}$, based on its sign. The accuracy is evaluated using the same technique as specified before.

2 Theoretical Background

The heart of spectral clustering comes from facts associated the graph Laplacians which in this case is the unnormalized variant, $\tilde{L}$. This graph Laplacian is a similarity graph which defines the connectivity of various points by some weight matrix, W. To construct W, a Gaussian weight function, $w_{ij} = \exp(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2})$, is employed (the norm can be changed to any value in $1 \leq p \leq +\infty$). From W, the diagonal degree matrix, D, can be constructed by taking the sum of the rows of W as the diagonal of D, or $D = \text{diag}(\sum_{i=0}^{N-1} w_{ji})$. Then, the unnormalized graph Laplacian is then defined as $\tilde{L} = D - W$. For graph Laplacians, they are all non-negative, definite, and symmetric (NDS) with eigenvalue and eigenvector pairs defined by $\{\lambda_j, q_j\}_{j=0}^{N-1}$. The eigenvectors and values can be computed using the eigenvalue decomposition

$$\tilde{L} = Q\Lambda Q^T$$

where Q is an orthonormal basis of $\tilde{L}$ and the eigenvectors of Λ. Q must be normalized for other spectral clustering methods such as K-means. A unique property of graph Laplacians is that the first eigenvalue, λ_0, is always 0 and its corresponding eigenvector, q_0, is constant. Also, if there are k-disconnected groups in the graph, then $0 = \lambda_0 = \lambda_1 = \ldots = \lambda_{k-1} < \lambda_k \leq \lambda_{k+1} \leq \ldots$. Knowing that there are only two clusters, however, allows the use of the $sign$[2] function on the Fielder vector, q_1, as a classifier. This is true because, as for most solvers, q_0 is a constant vector and q_1 must be orthogonal to q_0, so, if there are N_0 points in the first cluster and N_1 points in the second cluster, then q_1 must have N_1 normalized negative entries and N_0 positive entries so that $q_0^T q_1 = 0$.

This problem, however, is only if all the points in X have labels y, or the the entire graph Laplacian is not cumbersome to compute. For a subset of labels, however, the Laplacian embedding of X, $\tilde{F}(X)$, can computed which takes only M eigenvectors of Q so that $\tilde{F}(X) \in$

[2]The classifiers might be off by the opposite label, or $-1\times$label, so adding a -1 out front of the result should be tested.

$R^{J\times M}$, where J is the number of known labels. Using $\tilde{F}(X)$ can then be used to train a classifier, $\hat{\beta}$, to predict the labels, $\hat{y}$. This is done through linear regression where

$$\hat{\beta} = [\tilde{F}(X)^T \tilde{F}(X)]^{-1} \tilde{F}(X)^T b$$

where $b \in R^{N\times 1}$ is the vector of known labels. The labels of the entire set is then predicted as $\hat{y} = \text{sign}(\tilde{F}(X)\hat{\beta})$.

For both spectral clustering and Laplacian embedding, the clustering or semi-supervised learning (SSL) accuracy is defined as

$$\text{accuracy} = \frac{1}{N} \times \text{number of misclassified labels}$$

with lowers values corresponding to higher accuracy.

3 Algorithm Implementation and Development

Before any of the methods are used, the data is read and converted into numbered data. The voting data is loaded using *np.loadtext*[3] with *delimiter = ',', dtype = object* and transposed. The object data is then converted into integer values using a function called *numberer* that loops through each row and replaces the object with the appropriate number as outlined in the Introduction and Overview section. The function returns X and y. From here, a uniform array of values of σ is generated using *np.arange* and sliced to remove 0. Next, a *for* loop is utilized to calculate the Euclidean distance between the values of X, using *scipy.spatial.distance_matrix*. In the for loop, W is calculated using a function called *eta* that takes in the Euclidean distance and the value of σ and then returns the Gaussian weight matrix W. D is then constructed as using *np.sum* along *axis = 1* and *np.diag* on that. $\tilde{L}$ is implemented as outlined in the Theoretical Background section. *np.linalg.eigh* computes the eigenvalue decomposition which is then sorted using the indices returned by calling *.argsort()* on the eigenvalues. From here calling $\pm$*np.sign* on the second, sorted eigenvector will return the predicted party membership and placed into two separate $\hat{y}$s. To count the number of correct predictions, *np.count_nonzero($\hat{y}$ == y)* is used to count the number of True values. The accuracy is then simply taking the difference between the total y and the number of Trues divided by the total y. To find the minimum σ for mislabelling, using *np.where(accuracy == np.min(accuracy))* will return the indices to pluck it out. The middle value of all the σ^* is chosen if there are many σ^* values.

Everything for the semi-supervised learning portion is identical to the spectral clustering method up to ordering the eigenvalue decomposition. The process of choosing different values of M and J is automated by creating nested *for* loops that go through each combination of M and J. To generate $\tilde{F}(X)$, M columns are chosen; to construct A, J rows of $\tilde{F}(X)$ are chosen; and, to construct b, J rows of y are kept. From here *sklearn.linear_model.LinearRegression*, with *set_intercept == False*, is called inside all the for loops and *.fit()* is called in conjunction with A and b. $\hat{y}$ is then predicted using *np.sign()* on *np.dot* between $\tilde{F}(X)$ and $\hat{\beta}$. The accuracy is evaluated the same way spectral clustering is done and inserted into an array.

[3]Numpy is written as np

4 Computational Results

By computing the clustering accuracy for various values of σ, many values of σ^* that corresponded to the positive branch of the *sign* function where found. $\sigma_+^* = 3.60$ was found to be the middle value of that set σ^*s. Likewise $\sigma_-^* = 3.42$ corresponds to the chosen optimal negative branch value of the *sign* function. Both branches had the optimal result of 11.95% mislabelled representatives. The results for a uniform set of σ can be found in Figure 1, and the results of the classification task using σ^* are found in Figure 2.

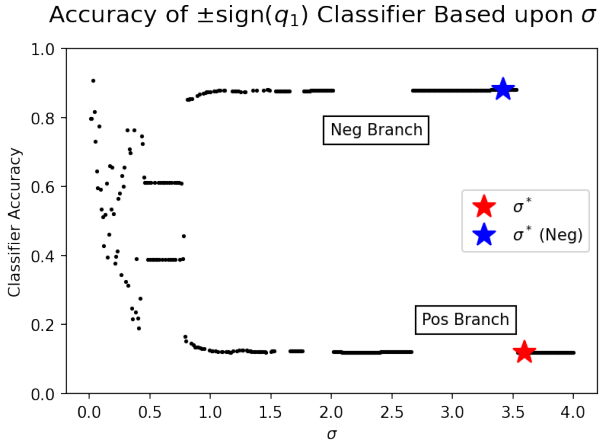

Figure 1: The red star corresponds to $\sigma_+^* = 3.60$ and the blue star corresponds to $\sigma_-^* = 3.42$. These values are the middle σ^* values of each branch.

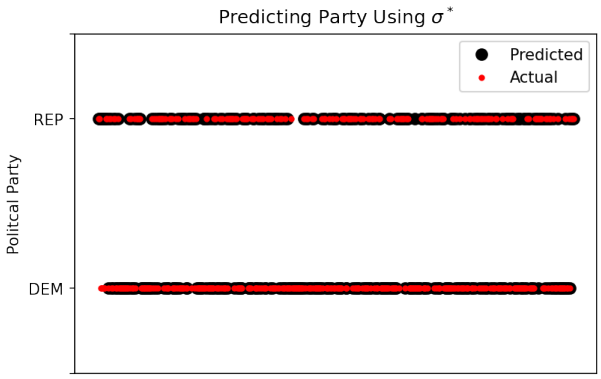

Figure 2: Dots that don't have the opposite color inside them means that the representative was mislabeled as the opposite party. Using $\sigma_+^* = 3.60$ resulted in an accuracy of 11.95%.

Following the steps outlined in the Algorithm Implementation and Development, $\sigma_+^* = 3.60$ was chosen to be used in the semi-supervised learning task and the associated values for each combination of M and J are reported in Table 1. The lowest mislabelling rate was 7.13% with $M = 5$ and $J = 40$.

5 Summary and Conclusions

When the accuracy of the *sign* classifier as a function of σ is plotted, there is an initial region of high instability corresponding to $\sigma < 0.5$ and should be ignored. At $0.5 < \sigma < 1.75$, the accuracy oscillates between the positive branch and negative branch of *sign* corresponding to when the classifier mislabels -1 as $+1$ and vis-versa. This region should likewise be ignored because of the instability brought about by this oscillatory behavior. This leaves $\sigma > 2$ as the acceptable region for σ^* though which branch of the *sign* function the values correspond to must be noted for accuracy computations. The optimal accuracy for both branches was found to be 11.95% which means that about 1 in 10 representatives was thought to be a member of the opposite party they are actually affiliated with.

Accuracy		M				
		2	3	4	5	6
J	5	0.1356	0.0943	0.4736	0.3908	0.4713
	10	0.1356	0.1402	0.0851	0.2000	0.3862
	20	0.1218	0.1011	0.0920	0.0989	0.1057
	40	0.1218	0.1149	0.0966	0.0713	0.0851

Table 1: The green box corresponds to the lowest rate of mislabelling while red corresponds to the highest rate of mislabelling for $\sigma_+^* = 3.60$.

For the semi-supervised learning task, the highest J value, $J = 40$, always corresponded to the best across- the-board performance of any M. Increasing M by itself, however, was found it increase mislabelling. This is because the matrix becomes more over-determined as the classifier tries to fit too many parameters, M, to too little data, J. Keeping M on the lower end with lower values of J produces better results than higher values of M for lower J. The best accuracy using $\sigma_+^* = 3.60$ was found to be 7.13% by using $M = 5$ and $J = 40$. This implies that using the bottom middle combinations of Table 1 will produce the best results for various σ^* values.

Using semi-supervised learning almost had a 50% accuracy increase over spectral clustering. This revelation implies that just using the $\text{sign}(q_1)$ is not enough to completely describe the dynamics of the clustering. This means that certain representatives are more likely to vote across party lines on bills that other members of their party vote unanimously on. Evaluating which representatives are mislabelled over different values of σ^* would then imply who more likely to break from party lines and which party is likely to be more bi-partisan.

6 Acknowledgements

The author would like to graciously thank his classmates for allowing him to discuss the problem with them as a means to help them and test his own understanding of the material. The author would also like to thank Professor Hosseini for his demonstrations of spectral clustering which were used as the basis for their implementation in this assignment.