# AMATH 482/582: HOME WORK 4

SHENGBO JIN

*CFRM Program, University of Washington, Seattle, WA*
Shengboj@uw.edu

Abstract. This report aims at testing the performance of spectral clustering and a simple semi-supervised regression algorithm with different parameters. Firstly, in spectral clustering, test different variance parameters and try to show the relationship between clustering accuracy and variance parameters. Then, using the above optimal parameters, test different dimensions of Laplacian embedding and the used number of labeled data. At last, summarize the testing results.

## 1. Introduction and Overview

Our goal is to test the performance of spectral clustering and a simple semi-supervised regression algorithm on the 1984 house voting records dataset. The data set consists of voting records of 435 members of the House on 16 bills. There are 267 members of the democratic party and 168 members of the republican party. The performance of the classifier is determined by multiple factors like the variance parameter in graph Laplacian matrix, the dimension of the Laplacian embedding.

In this report, under Python environment, spectral clustering and semi-supervised learning are realized by Numpy[1], Scipy[4], and Sklearn[3]. Matplotlib[2] is used for visualization.

## 2. Theoretical Background

**Spectral clustering**

Given an enumerated set of data points, the similarity matrix may be defined as a symmetric matrix A, where $A_{ij}$ represents a measure of the similarity between data points with indices i and j. The general approach to spectral clustering is to use a standard clustering method on relevant eigenvectors of a Laplacian matrix of A. The eigenvectors that are relevant are the ones that correspond to smallest several eigenvalues of the Laplacian except for the smallest eigenvalue which will have a value of 0.

The graph Laplacian matrix is defined as

L:= D - A

where D is the diagonal matrix

$D_{ii} = \sum_j A_{ij}$

Knowing the n-by-k matrix V of selected eigenvectors, mapping — called spectral embedding — of the original n data points is performed to a k-dimensional vector space using the rows of V. Now the analysis is reduced to clustering vectors with k components, which may be done in various ways.

In the simplest case k=1, the selected single eigenvector v, called the Fiedler vector, corresponds to the second smallest eigenvalue. Using the components of v, one can place all points whose component in v is positive in the set $B_+$ and the rest in $B_-$, thus bi-partitioning the graph and labeling the data points with two labels.

*Date*: March 11, 2022.

In the general case k>1, any vector clustering technique can be used.

**Semi-supervised learning**

Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. Semi-supervised learning falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). It is a special instance of weak supervision.

Unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent (e.g. to transcribe an audio segment) or a physical experiment (e.g. determining the 3D structure of a protein or determining whether there is oil at a particular location). The cost associated with the labeling process thus may render large, fully labeled training sets infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value. Semi-supervised learning is also of theoretical interest in machine learning and as a model for human learning.

A set of $l$ independently identically distributed examples $x_1, x_2, x_3...x_l \in X$ with corresponding labels $y_1, y_2, y_3...y_l \in Y$ and $u$ unlabeled examples $x_{l+1}, x_{l+2}, x_{l+3}...x_{l+u} \in X$ are processed. Semi-supervised learning combines this information to surpass the classification performance that can be obtained either by discarding the unlabeled data and doing supervised learning or by discarding the labels and doing unsupervised learning.

## 3. Algorithm Implementation and Development

**Step 1 Data preprocessing**

Use *np.loadtxt()* to load the data, then apply boolean array and slicing to get matrix $X$ and vector $y$.

**Step 2 Spectral Clustering**

1. Define *eta()* with *np.exp()* as the weight function. Then, define *classifier()* with *eta()*, *np.diag()*, *np.linalg.eigh()*, *np.sign()* as Spectral Clustering function to return classified results and eigenvectors of the unnormalized graph Laplacian matrix. Note: both *np.sign()* and *-np.sign()* will be used to adjust for the unsupervised learning.
2. Use *scipy.spatial.distance_matrix()* with $X$ to compute the distance matrix. With different variance parameter $\sigma$ in (0, 4], input the distance matrix to the *classifier()* to obtain classified results. Then, calculate clustering accuracy by *np.mean()*.
3. Find best accuracy and optimal $\sigma^*$ by *np.max()*. Then, use *plt.plot()* to plot accuracy as a function of $\sigma$.
4. For visualizing the behavior of $\mathbf{q}_1$ on the two clusters, plot $\mathbf{q}_1$ by re-ordering $X$ according to the original party affiliations using *plt.plot()* and *argsort()*.

**Step 3 Semi-supervised Learning**

1. Initialize *LinearRegression(fit_intercept=False)* as *lin*. Under different optimal $\sigma^*$, use *classifier()* to get the eigenvectors of the unnormalized graph Laplacian matrix.
2. For each $M$ and J, use the eigenvectors matrix to get the submatrix $\mathbf{A}$ as the predictor variables, and use subvector $\mathbf{b}$ as the response variable to fit linear regression with *lin.fit()*. Then, by *lin.coef_* to obtain $\hat{\beta}$, with *np.dot()* and *np.sign()* to get $\hat{y}$. Finally, calculate SSL accuracy by *np.mean()* .
3. Find best accuracy and global optimal $\sigma^*$ by *np.max()*.
4. For visualizing the behavior of $\mathbf{F(X)}\hat{\beta}$ on the two clusters, plot $\mathbf{F(X)}\hat{\beta}$ by re-ordering $\chi$ according to the original party affiliations using *plt.plot()* and *argsort()*.

## 4. Computational Results

**Task 1&2**

Figure1 plots accuracy as a function of $\sigma$, which shows the relationship between clustering accuracy and variance parameter.

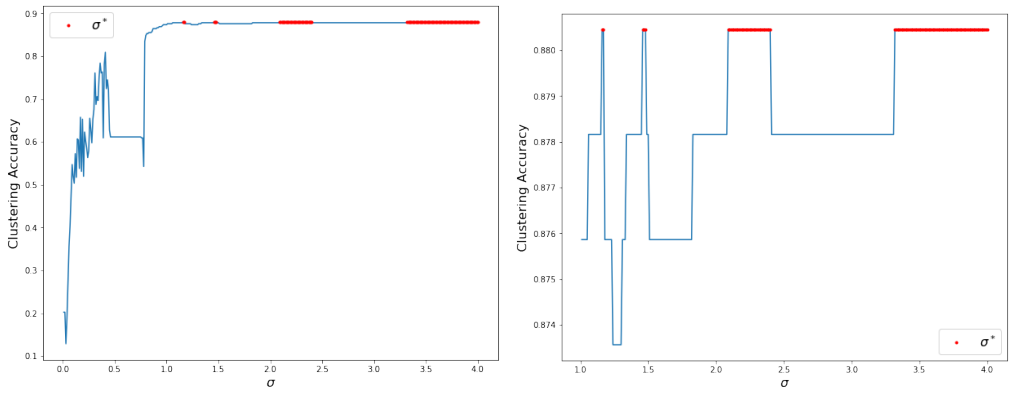
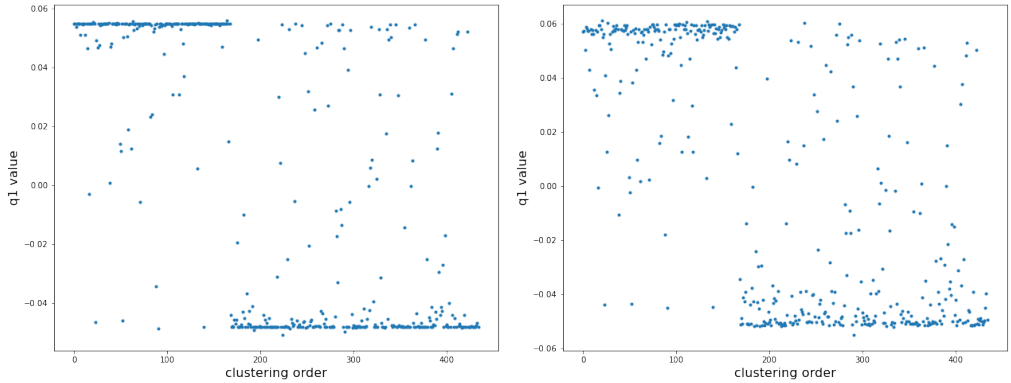



FIGURE 1. Clustering Accuracy v.s $\sigma$

Based on the computational results and plots, to a large extent, clustering accuracy increases as variance parameter increases, but after about 1, the accuracy becomes stable basically. (When the sigma is too small, lots of elements in q1 will be 0, thus the accuracy is so low at the beginning.) The best accuracy is about 88.046%, which is reached with multiple optimal $\sigma$* values.

For different optimal$\sigma$*, select $\sigma$*=1.16 and $\sigma$*=2.11 as examples, visualize the behavior of q1 on the two clusters, plot q1 by re-ordering $X$ according to the original party affiliations.




(A) $\sigma$*=1.16 (B) $\sigma$*=2.11

FIGURE 2. q1 values under $\sigma$*=1.16 and $\sigma$*=2.11

From the above plots, we know that, although the clustering accuracy is the same under both variance parameters, the q1 value has a looser distribution under a bigger $\sigma$*=2.11 rather than focuses on the same value under $\sigma$*=1.16.

**Task 3**

Use the above optimal $\sigma$* to construct different eigenvectors of the unnormalized graph Laplacian matrix. Under these different conditions, for **M** = 2, 3, 4, 5, 6 and **J** = 5, 10, 20, 40, the semi-supervised learning accuracy gets highest when $\sigma$* is around 3.43, M=4, J=10, which is equal 93.563%

| | J = 5 | J = 10 | J = 20 | J = 40 |
|---|---|---|---|---|
| M = 2 | 0.86436782 | 0.86436782 | 0.87816092 | 0.87816092 |
| M = 3 | 0.90114943 | 0.85517241 | 0.89655172 | 0.88965517 |
| M = 4 | 0.44137931 | 0.93563218 | 0.90344828 | 0.91034483 |
| M = 5 | 0.51494253 | 0.66666667 | 0.90114943 | 0.91954023 |
| M = 6 | 0.47816092 | 0.54022989 | 0.88965517 | 0.90344828 |

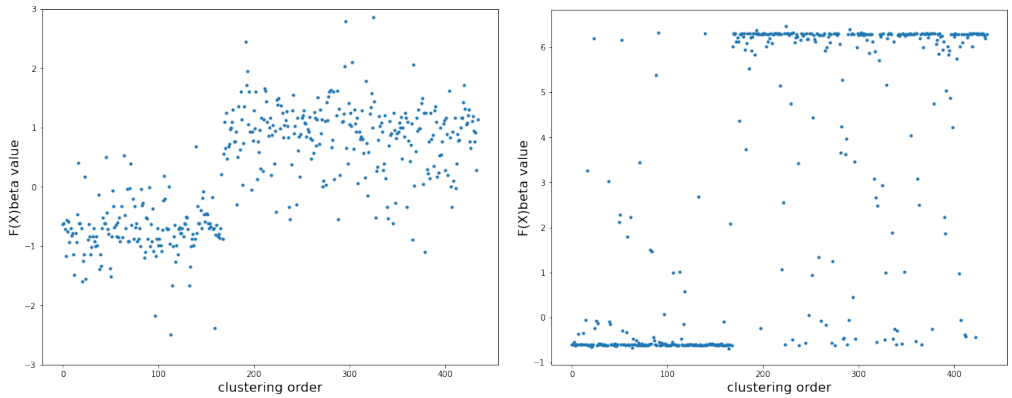
TABLE 1. SSL accuracy under $\sigma$*=3.43

When $\sigma$*=1.16, the best SSL accuracy is 88.966% with M=2 and J=5.

| | J = 5 | J = 10 | J = 20 | J = 40 |
|---|---|---|---|---|
| M = 2 | 0.88965517 | 0.88735632 | 0.88275862 | 0.88045977 |
| M = 3 | 0.88735632 | 0.81609195 | 0.82068966 | 0.83678161 |
| M = 4 | 0.83908046 | 0.85057471 | 0.86436782 | 0.87586207 |
| M = 5 | 0.86896552 | 0.71034483 | 0.83908046 | 0.88045977 |
| M = 6 | 0.86896552 | 0.68505747 | 0.87356322 | 0.86436782 |

TABLE 2. SSL accuracy under $\sigma$*=1.16

In general, we can find that, a bigger size submatrix $A \in R^{JM}$ in semi-supervised learning maybe not mean better classification accuracy.




(A) $\sigma$*=3.43 (B) $\sigma$*=1.16

FIGURE 3. F(X) $\hat{\beta}$ values under $\sigma$*=3.43 and $\sigma$*=1.16

According to the above figures, using the eigenvectors of the unnormalized graph Laplacian matrix with bigger $\sigma$ in SSL will lead the predicted value to be more widely distributed. In addition, when $\sigma$=3.43, SSL accuracy reaches the best performance with M=4, J=10 than $\sigma$=1.16.

## 5. SUMMARY AND CONCLUSIONS

Unlike supervised learning, the performance of unsupervised learning may be hard to evaluate. Thus, simply using the evaluating tools in supervised learning may lead to incorrect analysis results. In addition, from this simple semi-supervised learning example, we find that, using more labeled data may not improve the classifier performance. Another question is, the mysterious variance parameters in this problem, which influence the behavior of the predicted value. The best parameters with same performance in spectral clustering may exhibit different performances in the later semi-supervised learning. But the more detailed relationship needs further work to research.

## Acknowledgements

The author is thankful to Prof. Hosseini for lectures about spectral clustering and semi-supervised learning. Furthermore, my peers Xuqing Hu is helpful in the use of Latex.

## References

[1] 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*, 585(7825):357–362, Sept. 2020.

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[3] 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.

[4] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.