CS 292F Project
Spectral Theory on Different Representations of a Graph

Mert Kosan

June 10, 2021

1 Introduction

Graph representation is a well-known problem for storing graph data in a memory of a computer. There are several ways to represent a graph, where the easiest way is adjacency matrix representation. In this report, I will work on 3 different representations adjacency, Laplacian, and normalized Laplacian matrix.

Calculating the laplacian of an adjacency matrix enables many useful properties of a graph such as the number of spanning-tree calculations with Kirchhoff's theorem [5], or graph cut with the second smallest eigenvalue (i.e. Fiedler value) by Cheeger's inequality [1]. With Spectral theory [2], it can produce low dimensional embeddings that could be used in various machine learning applications such as clustering, classification.

In this project, I will analyze the computational cost of calculating embeddings from adjacency, Laplacian, and normalized Laplacian form of a matrix. Then, I will show the effectiveness of embeddings calculated by the different forms in different downstream machine learning tasks such as clustering and classification. I have the following contributions:

- I analyze how different representations of a graph behaves in terms of calculation cost of the smallest eigenvectors and performing in clustering and classification tasks in 3 different datasets.
- I propose a new algorithm to visualize the predicted node labels along with their original labels.
- I show the effect of the largest and the smallest eigenvectors of graph representations in clustering.

2 Short Literature Review

Graph representations.

A is an adjacency matrix, where entry i-j represents if the edge weight between node i and j. 0 means no edge. L and N are defined as the following, where D is a diagonal degree matrix.

$$L = D - A$$

$$N = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

Eigenvectors and Eigenvalues.

The embeddings of matrix A will be the smallest k eigenvectors, so each node in the graph will be represented with a vector of size k. The representation of eigenvectors and eigenvalues will be the following:

$$Av = \lambda v$$

where λ and v are eigenvalue and corresponding eigenvector.

$$AW = W\Lambda$$

is a combined form of eigenvectors and eigenvalues where Λ is a diagonal matrix with eigenvalues on the diagonal, and $W = [v_1, \ldots, v_n]$.

Similarity Graph.

Some datasets are not represented as a graph, but points in the space. However, they can be still represented as a graph using similarity measurement between each node.

One famous similarity measurement is the Radial basis function kernel (RBF), commonly using in Support Vector Machines.

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

Clustering Metric.

As clustering is an unsupervised learning method, accuracy or F1 score may not be directly applied to calculate the performance of clustering. I use NMI (Normalized Mutual Information) for a performance metric of clustering.

$$NMI(Y, C) = \frac{2 \times I(Y; C)}{H(Y) + H(C)}$$

where Y, C are ground labels, predicted clusters respectively. $I(Y; C)$ is mutual information between Y and C, and $H(.)$ calculates entropy.

3 Downstream Tasks

When I have node embeddings (top eigenvectors in our case), I can perform downstream tasks like clustering or classification. I explore K-Means algorithm in clustering, Neural networks, Support Vector Machines, and Random Forest for classification.

Clustering.

Clustering or community detection in graphs is a task for grouping objects into similar meanings or representations. In other words, the object in the same cluster will be similar to each other compare to an object from another cluster. Clustering algorithms are unsupervised meaning the label of the objects won't be used to learn clusters. I implemented K-Means algorithm to detect clusters. In K-Means, the number of clusters is predefined.

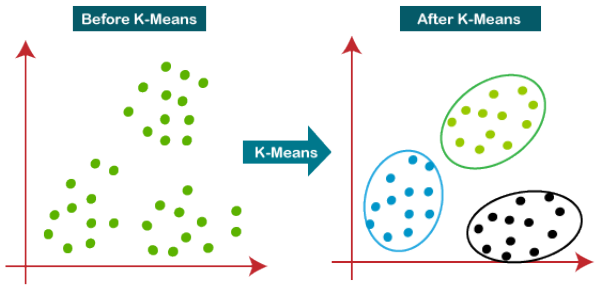
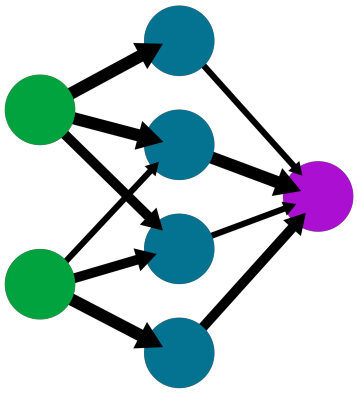


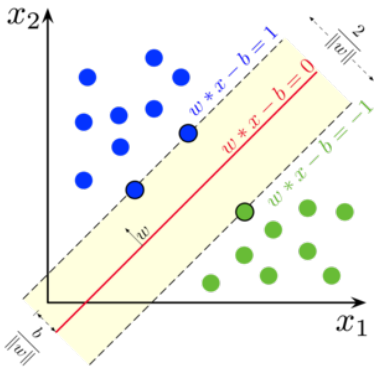
Figure 1: K-Means [6]

Classification.

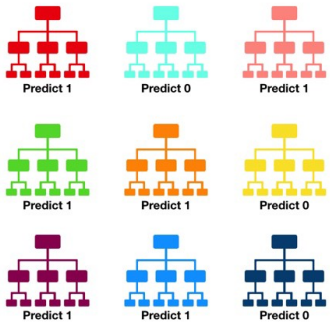
Classification is another task to detect the label or a property of an object. Compare to clustering, the classification methods are supervised, meaning there will label of the objects. As I am dealing with the graphs, I will perform node classification using 3 different famous methods: Neural networks, support vector machines (SVM), random forests.



(a) Neural Networks[8]



(b) Support Vector Machines[7]



(c) Random Forest[10]

4 Datasets

In our experiments, I have used 3 different datasets, which are explained here.

Cora [4]

The Cora citation networks include 2708 research papers (nodes) classified into one of the seven classes, i.e. Cased Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, Theory. The network consists of 5429 links (edges) between publications. Each publication has a binary word vector as an attribute, however, in our problem I don't use node attributes. So, I omit them. The network is unweighted and directed in nature, however, for simplicity, I consider directed edges undirected. The original dataset has isolated nodes, so I selected the largest component.

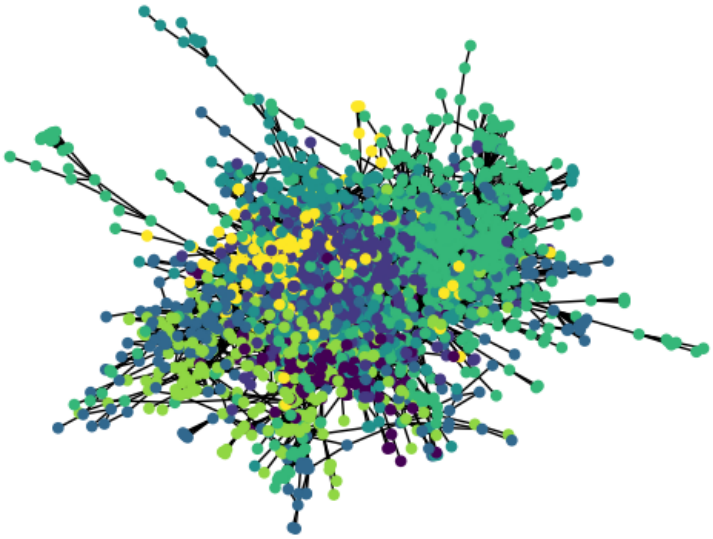


Figure 3: Cora

Email [9]

The network consists of email transfer between people from a European research institution. In the binary network, if a person sends an email to another person, there will be an edge between them. There 42 communities (i.e. clusters) representing the departments at the research institute. The original dataset is directed, I made it undirected by simply ignoring direction and weights. It has also isolated nodes, so I selected the largest component. The largest component has 986 people (i.e. nodes), and 25552 emails (i.e. edges).

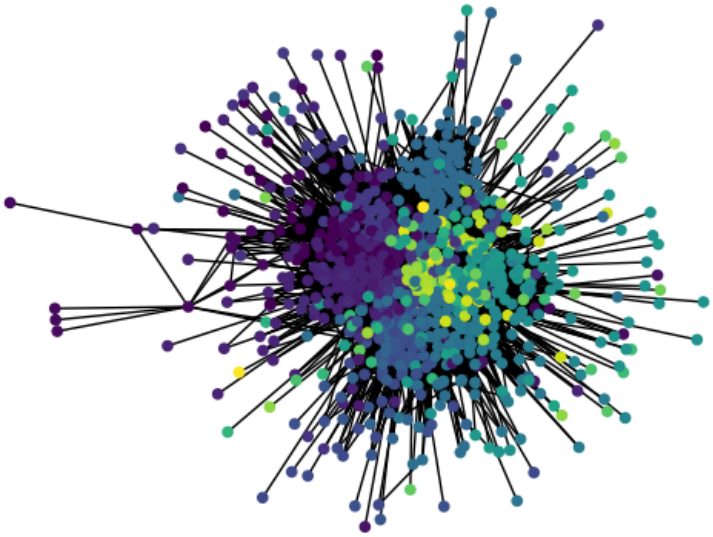
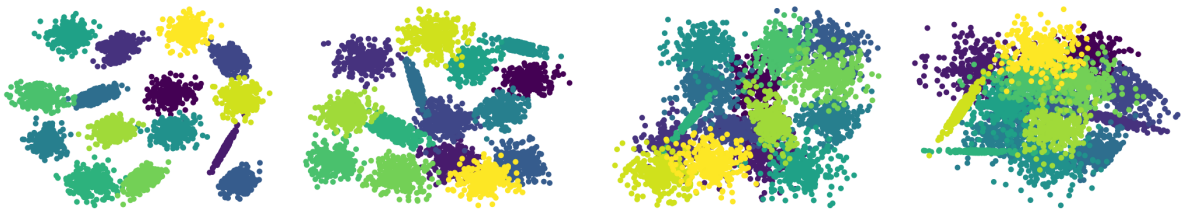


Figure 4: Email

S-sets [3]

Synthetic dataset including 5000 2d points with 15 different clusters. There are 4 versions where the noise degree is changing. I will use the first and the fourth one for our experiments. I created a similarity graph using the Radial-basis function kernel (RBF).



(a) S-sets v1 **(b)** S-sets v2 **(c)** S-sets v3 **(d)** S-sets v4

Figure 5: S-sets different versions

5 Experiments

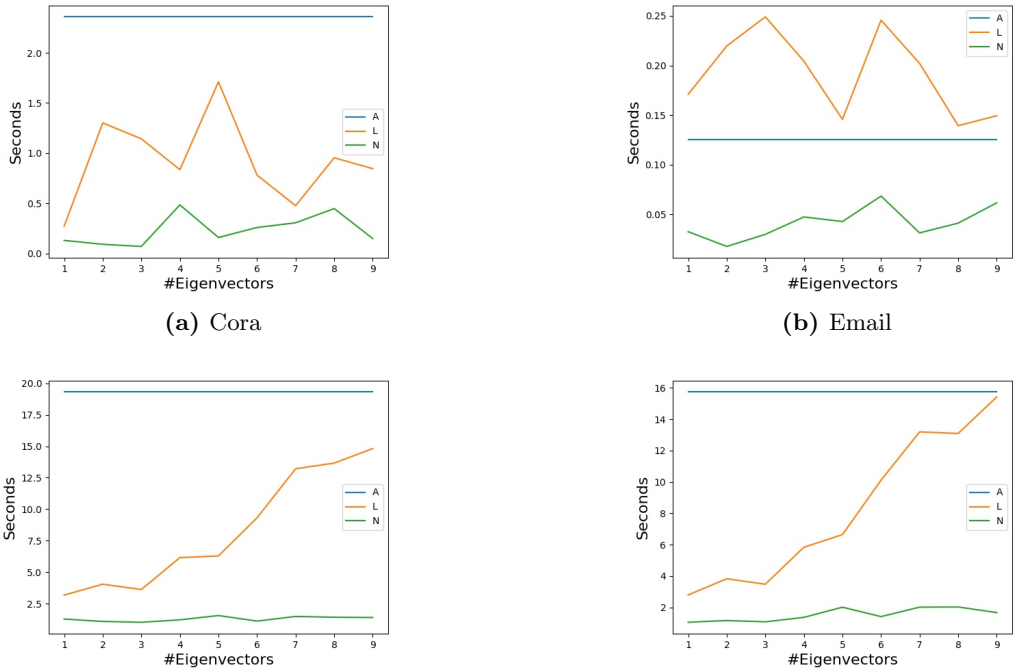
The project will consist of the quality and computational cost comparison of the spectral algorithm on the different representations of graphs. I use L, N, and A (Laplacian form, Normalized Laplacian Form, and Adjacency Matrix Form respectively) as different representations for each dataset.

The implementation[1] is using Python 3, and I used *Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz* to run experiments.

Computational costs in calculating eigenvectors of A, L, and N.

Figure 6 suggests the following outcomes:

- Calculating eigenvectors from A is the same across eigenvectors. They are calculated from full eigendecomposition since when A is sparse, finding eigenvalues takes longer. Scipy doesn't converge with more than 10k numbers of iterations. Even this may suggest L or N is in better form to calculate eigenvalues.
- Calculating eigenvectors from Normalized Laplacian matrix (i.e. N) is faster than L and A.
- Calculating eigenvectors from L is faster than A, except Email dataset. The possible reason is that Email dataset is not sparse enough for Scipy solver. So it won't converge fast.



(a) Cora **(b)** Email

(c) S-sets v1 **(d)** S-sets v4

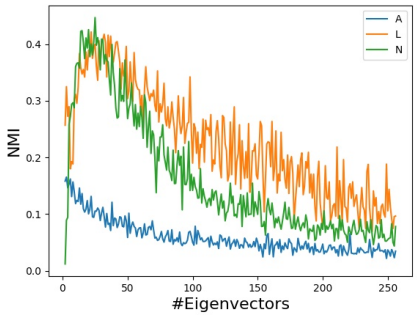
Figure 6: Computational time calculating the smallest eigenvectors.

[1] https://github.com/mertkosan/spectral-clustering

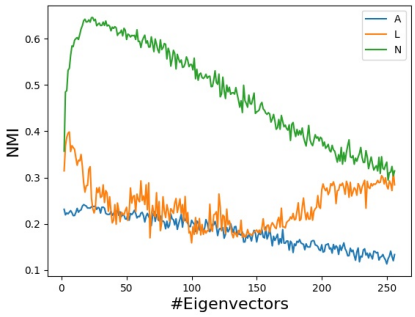
Spectral clustering performance difference for A, L, and N.

I applied K-Means clustering on eigenvector embeddings of A, L, and N. The results are shown in Figure 7. I experimented with up to 256 smallest eigenvalues for Cora and Email, and up to 16 smallest eigenvalues for S-sets datasets. I have the following outcomes from the plots.

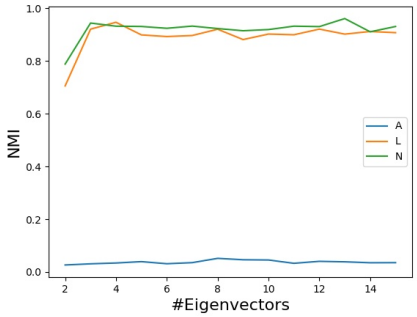
- For all datasets, embeddings from N are more effective than other representations for peak NMI score. However, L is giving better results than N with the increased number of eigenvectors for Cora dataset.
- For S-sets dataset, the first version v1 has a better result than v4. This is expected since v4 has more noise.
- The number of eigenvectors helps increase in NMI score up to some point, then the performance decays with a large number of eigenvectors. The possible reason is that K-Means is becoming harder. The exception is L for Email, the NMI starts to increase after 150 smallest eigenvectors.
- For N representation, the best number of eigenvectors is 25, 24, 13, 8 for Cora, Email, S-sets v1, and S-sets v4 respectively.



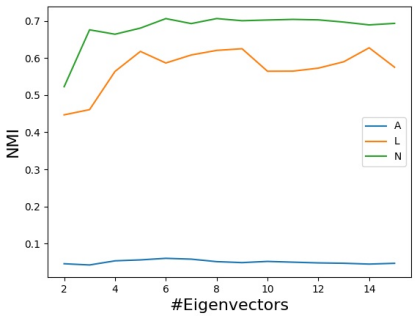
(a) Cora



(b) Email



(c) S-sets v1



(d) S-sets v4

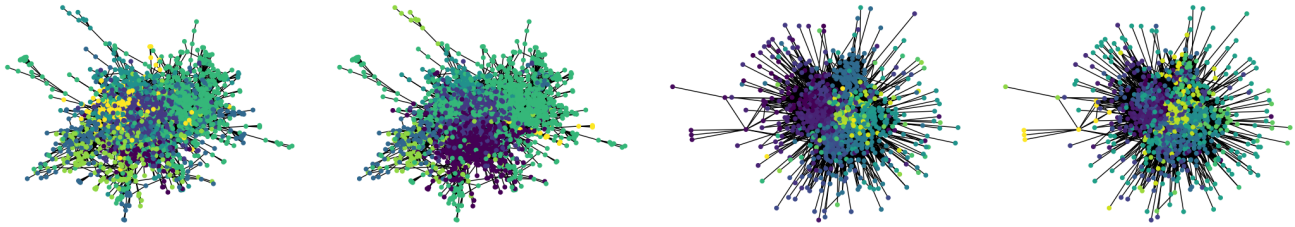
Figure 7: NMI vs # Eigenvectors relationship and comparison of performance in A, L, N representation embeddings.

Spectral Clustering Visuals

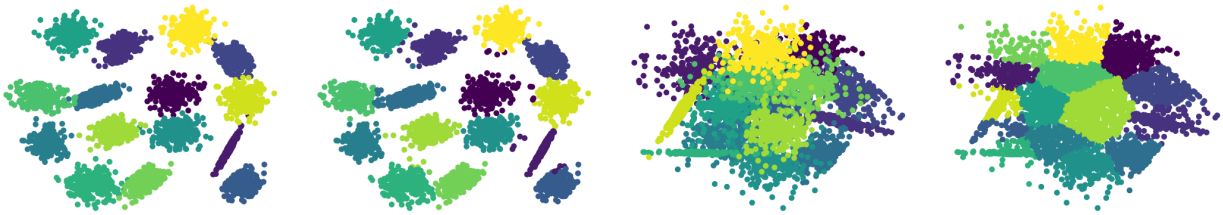
The clustering algorithm will return random cluster labels, that's why I have to map the best possible labels to the predicted ones. For this purpose, I propose a greedy method to match the ground truth and the predicted labels in Algorithm 1.

Algorithm 1: Matching predicted clusters to the ground truth

```
   Input: ground, predicted
   Output: shuffled
1  remained-classes ⟵ unique(ground) // a set of remained classes from all classes
2  shuffled ⟵ predicted
3  while remained-classes ≠ ∅ do
4      predicted ⟵ shuffled
5      best-accuracy ⟵ accuracy(ground, predicted)
6      for trials do
           /* Shuffle classes of data in a random way if they are in remained classes   */
7          temp ⟵ shuffle-clusters(predicted, remained-classes)
8          temp-accuracy ⟵ accuracy(ground, temp)
           /* Update best accuracy and shuffled if temp accuracy is bigger than best accuracy */
9          best-accuracy, shuffled ⟵ update(best-accuracy, temp-accuracy, temp)
       /* It find most matched classes between ground and shuffled, and remove from remained
          classes, the most matched class can be -1, then it means there is no matching from
          remaining classes.                                                          */
10     if most-matched-class = -1 then
11         break
12     most-matched-class ⟵ class-match(ground, shuffled)
13     remained-classes ⟵ remained-classes - {most-matched-class}
14
```



(a) Cora Original **(b)** Cora Prediction **(c)** Email Original **(d)** Email Prediction



(e) S-sets v1 Original **(f)** S-sets v1 Prediction **(g)** S-sets v4 Original **(h)** S-sets v4 Prediction

Figure 8: Clustering (predicted) visual for each dataset along with the original labels (ground truth). The coloring matching is established with Algorithm 1. Our predictions for all datasets are quite good, and our matching algorithm works well.

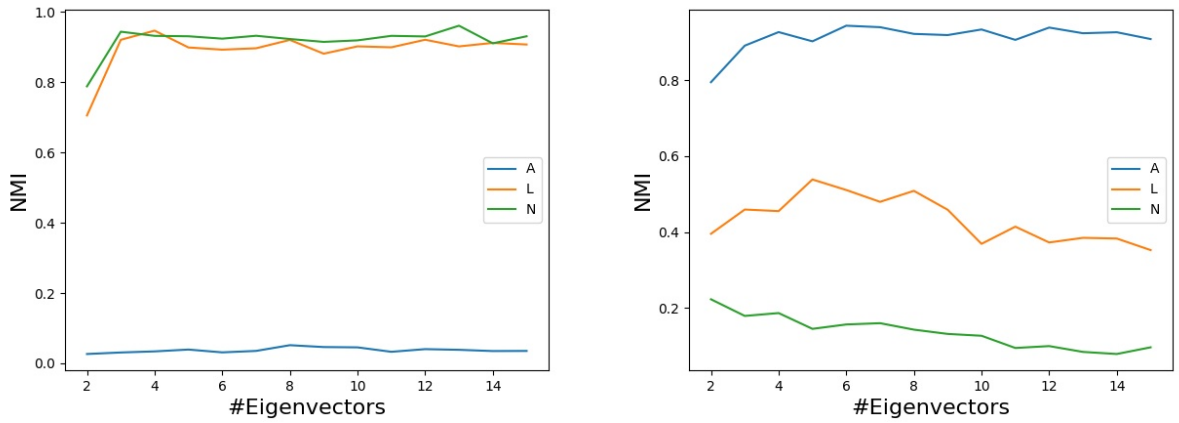
Smallest vs largest eigenvectors effect.

I compare the results if I select the largest k eigenvalues instead of the smallest k. Figure 9 suggests the following outcomes:

- With the largest eigenvectors, embeddings from A are more effective than L and N.
- The largest eigenvalue embeddings from L and N give much worse results than the smallest ones. However, while L still gives some information, N decays to very deep.

The possible reasons of the strange outcomes:

- The eigenvalues from the Laplacian matrices represent the connectivity of a graph starting from the smallest ones. So, for clustering applications, the smallest will be more effective than the largest ones for Laplacian matrices (i.e. L and N).
- The largest eigenvalue of an adjacency matrix gives an upper bound on an average degree in the graph, and also it preserves the similarity of nodes. That's why the largest eigenvectors could be better for adjacency matrix in clustering.



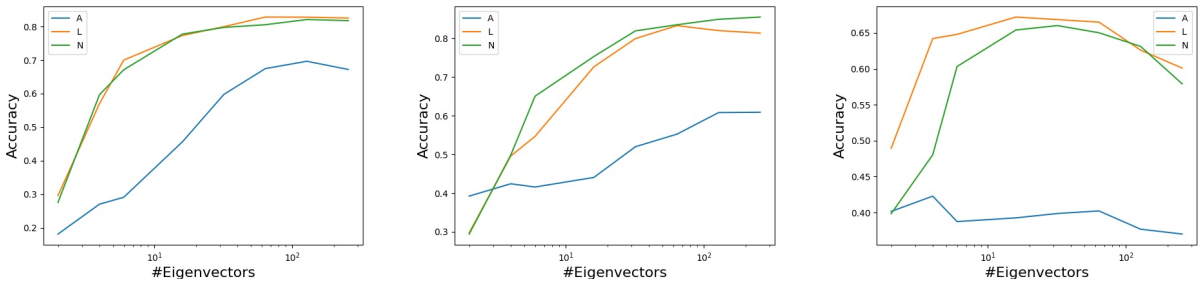
(a) Smallest **(b)** Largest

Figure 9: The smallest and the largest eigenvectors analysis for S-sets v1 dataset.

Node classification.

I also conduct a node classification task to see the effect of different classification methods on the different representations of a graph for Cora dataset using the smallest k eigenvectors. Figure 10 and 11 suggests the following outcomes:

- Laplacian representation (L or N) of a graph creates better embeddings than adjacency matrix A for all methods. While L and N have similar results, L is slightly better with Neural Networks and Random Forest, whereas N is better with SVM.
- Neural networks and SVM classify nodes better than random forest. Neural networks are better with the smaller number of eigenvectors, while SVM becomes better when the number increases. The possible reason is that Neural Networks starting to overfit as it needs to learn more parameters.



(a) Neural Networks **(b)** Support Vector Machines **(c)** Random Forest

Figure 10: A, L, N embeddings effect on different classification methods for Cora dataset.

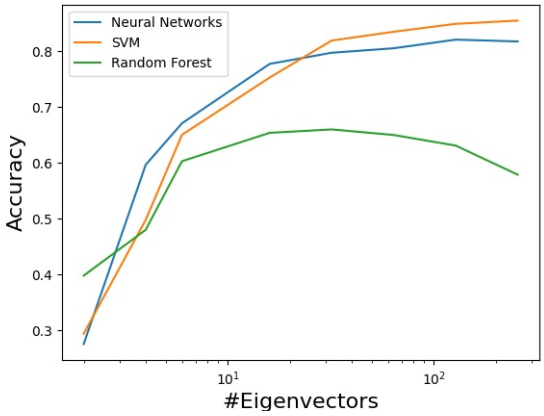


Figure 11: Classification method comparison with N representation of a matrix.

6 Conclusion

I analyze the computational difference between calculating the smallest eigenvectors from A, L, N representation of a graph. I show representation N is more useful in calculating embedding and performing better in clustering and classification. Summary results would be following:

- N representation converges faster than A and L.
- For most datasets, N is giving better clustering results with its embeddings.
- The number of eigenvectors helps up to some point, then performance starts to decay.
- I propose a new algorithm to visualize the predicted clusters for a better comparison with the original graph labels.
- For an adjacency matrix, the largest eigenvalues perform better in clustering.
- SVM and Neural networks perform better than Random forest in node classification, and Laplacian representations (L and N) have better results than adjacency representation.

References

[1] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Problems in analysis*, pages 195–200. Princeton University Press, 2015.

[2] F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[3] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–765, 2006.

[4] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[5] G. Kirchhoff. Ueber die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148(12):497–508, 1847.

[6] PRANSHU0. K means clustering simplified in python, 2021. [Online; accessed June 10, 2021].

[7] Wikipedia. File:svm margin.png. [Online; accessed June 10, 2021].

[8] Wikipedia. Neural network. [Online; accessed June 10, 2021].

[9] H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 555–564, 2017.

[10] T. Yiu. Understanding random forest, 2019. [Online; accessed June 10, 2021].