Optimization Problems in Spectral Clustering

April 26, 2023

Yufei Ma
yufeima@link.cuhk.edu.cn

Xiangyi Li
xiangyili@link.cuhk.edu.cn

1 Introduction

Clustering has been an established unsupervised learning method to group similar data without requiring labeling the training data. In most scientific fields that require dealing with empirical data, clustering is one of the most used exploratory data analysis techniques. Since [1], K-means clustering has been widely used. The K-means algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-square. However, [2] has shown that traditional clustering algorithms have some fundamental limitations, such as the inability not to cluster non-convex and high-dimensional data well. The spectral clustering algorithm, especially the one proposed by [3] is not only easy to implement but also outperforms K-means and other clustering algorithms shown by comparing different clustering algorithms from [4]. The spectral clustering algorithm considers good clusters as graphs that maximizes in-cluster edges and minimize inter-cluster cuts. It applies clustering to a projection of the normalized Laplacian. Below is a visualization on a toy dataset:

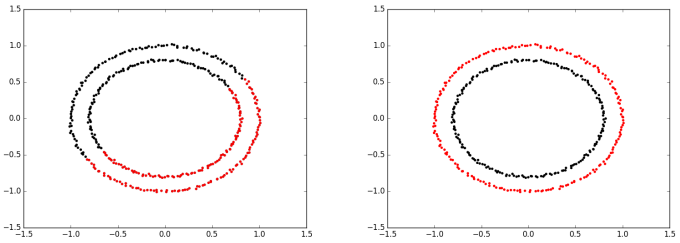

Figure 1: Left: K-means clustering; right: specral clustering

2 Spectral Clustering

Given a dataset $X = \{x_1, ..., x_n\}$, assume the similarity between x_i and x_j is $s_{ij} \geq 0$. To construct spectral clustering, we need to embed data points as nodes in a graph. A cluster is considered good if points within the same cluster are similar, and point wheras points in different clusters are dissimilar. Here we use similarity graph $G = (V, E)$ to represent the dataset.

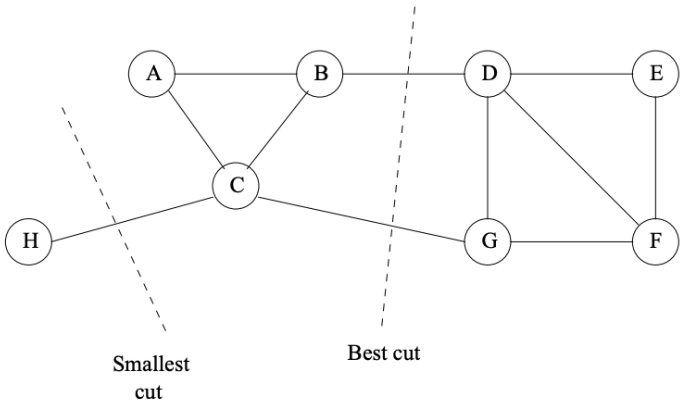

Figure 2: Figure 10.11 in [5]

2.1 Graph Cut

Let $G = (V, E)$ be the similarity graph, where $V = \{v_1, ..., v_n\}$ and each vertex v_i denotes a data point. We assume the graph is weighted and each edge between v_i and v_j carries a non-negative weight w_{ij}. Then the weighted adjacency matrix is constructed as

$$W := \begin{cases} w_{ij} \text{ if } v_i \text{ and } v_j \text{ are connected} \\ 0 \text{ else} \end{cases}$$

.

The degree matrix of G is defined as:

$$D = \text{diag}(d_1, ..., d_n)$$

$$d_i = \sum_{j=1}^{N} w_{ij}$$

To cluster the graph into several clusters, we want to minimize the weight of edges between different clusters. In a k-cluster problem, the min-cut approach is to choose a partition $A_1, ..., A_k$ which minimizes

$$\text{cut}(A_1, ..., A_k) := \frac{1}{2} \sum_{i=1}^{k} W\left(A_i, \overline{A}_i\right)$$

and the optimization problem can be written as

$$\min_{A_i} \quad \text{cut}(A_1, ..., A_k)$$

Although min-cut is a relatively easy problem to solve, it often does not lead to satisfactory partitions, for the reason that in many cases, it simply separates one individual vertex from the rest of the graph. To solve this problem, we want to constrain the size of the clusters so that they are "reasonably large." Normalized cut (Ncut) is one of the most common objective functions to solve the problem.

2.2 Normalized cut for k-cluster problem

Normalized cut (Ncut) provides a balanced graph partition by constraining the size of the clusters to be similar. Let $\text{vol}(A_i) = \sum_{i \in A_i} d_i$ denotes the size of edges in cluster A_i. Normalized cut for k clusters is defined as:

$$\text{Ncut}(A_1, ..., A_k) := \frac{1}{2} \sum_{i=1}^{k} \frac{W\left(A_i, \overline{A}_i\right)}{\text{vol}(A_i)} = \sum_{i=1}^{k} \frac{\text{cut}\left(A_i, \overline{A}_i\right)}{\text{vol}(A_i)}$$

where $\overline{A}_i$ stands for the complement of cluster A_i

Then the optimization problem constructed by normalized cut is

$$\min_{A_i} \text{Ncut}(A_1, ..., A_k)$$

The minimum of $\sum_{i=1}^{k} \left(\frac{1}{\text{vol}(A)}\right)$ is achieved if all $\text{vol}(A_i)$ coincide. However, solving this problem is NP-hard. Thus we use spectral clustering to solve the relaxed version of the problem.

The objective function can be rewritten using graph Laplacian. Define the cluster inidcator vector $h_j = \left(h_{1,j}, ..., h_{n,j}\right)^T$ as

$$h_{i,j} = \begin{cases} \frac{1}{\sqrt{\text{vol}(A_j)}} \text{ if } v_i \in A_j \\ 0 \text{ otherwise} \end{cases}$$

Set the matrix H as the matrix containing those k indicator vectors as columns. We have

$$H^T H = I$$

$$h_i^T D h_i = 1$$

$$h_i^T L h_i = \frac{\text{cut}\big(A_i, \overline{A}_i\big)}{\text{vol}(A_i)}$$

where L is the Laplacian matrix.

Thus the problem can be rewritten as:

$$\min_{A_1,...,A_k} \quad \text{Tr}\big(H^T L H\big)$$
$$\text{s.t.} \quad H^T D H = I$$

Now we relax the problem and substituting $T = D^{\frac{1}{2}} H$, and the problem becomes

$$\min_{T \in \mathbb{R}^{n \times k}} \quad \text{Tr}\Big(T^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} T\Big)$$
$$\text{s.t.} \quad T^T T = I$$

where $L_{\text{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ is the normalized graph Laplacian matrix. By Rayleigh-Ritz theorem, the solution T should contain the first k eigenvectors of L_{sym} as columns.

3 The Spectral Clustering Algorithm

Now that we have relaxed the NP-hard graph min-cut problem with the graph Laplacian, we will continue to formulate the algorithm that takes the $n \times m$ dataset, where m is the feature number. According to [6], the approximation does not guarantee the optimal cut.

3.1 Classic Spectral Clustering Algorithm

NgJordanWeiss Normalized Spectral Clustering(Data $X \in \mathbb{R}^{n \times m}$, K clusters):

Input: data $X = \{x_1, x_2, ..., x_N\}$, number K of clusters

Step 1: Construct a simiarity matrix W

Step 2: Compute the Laplacian matrix L_{sym}

$$L_{\text{sym}} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

Step 3: Perform eigenvalue decomposition on L_{sym} and use the first K eigenvectors to form matrix Z

$$\hat{L} = V \Sigma V^T, \quad Z = [v_1, ..., v_k]^T \in \mathbb{R}^{K \times N}$$

Step 4: Normalize the columns of Z to unit L_2 norm

$$z_i \leftarrow \frac{z_i}{\|z_i\|}, \quad i = 1, ..., N$$

Step5: Perform K-means on $\{z_1, z_2, ..., z_N\}$ Output: K of clusters of Z

According to [7], there are several ways of constructing a similarity matrix, such as the K-nearest neighbor graph and ε-neighborhood graph. We implemented the K-nearest neighbor similarity graph and the fully-connected graph.

3.2 Attempts to accelerate the NgJordanWeiss algorithm

Previously, the authors have tried applying ADMM [8] to solve for the graph embedding adaptively. Theoretically, it should be more performant than directly computing the closed-form for step 2 and step 3 since EVD and matrix inversion are both computationally expensive despite optimal algorithms developed by [9] [10] [11]. However, the empirical performance of the ADMM is much worse than direct solving matrices using packages like NumPy, SciPy, and scikit-learn. The source code for [12] [13] [11] [4] implemented sophisticated optimization by using the C++ and C interface with Python binding. One [12] went as far as building an LLVM-compliant compiler to speed up matrix computation using parallelism. To make a meaningful comparison between the speed of the closed-form approach and the adaptive approach, we must deploy similar Matlab-level matrix computation optimization techniques to our algorithm, which would go beyond the focus of this project. It is also shown from our experiments with ADMM that the initialization would also significantly impact the convergence speed, which meant that with data amount comparable to MNIST (70,000), there's no compelling reason to use adaptive algorithms due to the instability even with a possible solution to the initialization problem.

The authors moved on to improve step 5 using the accelerated K-means algorithm with triangular inequality [14]. Overall, there is certain speedup on the MNIST and Fashion MNIST datasets, but not always notable. This is explainable since after reducing the dimension of the Laplacian, the K in the K-means would not be large enough for the acceleration to be significant.

3.3 Python Implementation

Thanks to [15], we can extend some APIs to run experiments with different weight matrix constructing methods, metrics, and spectral clustering algorithm implementation. The code is available at https://github.com/l1xiangyi/GraphLearning.

4 Experimentation Results

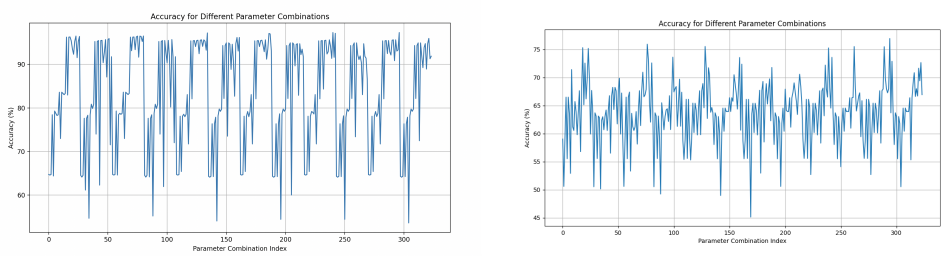

Figure 3: Left: K-means clustering; right: specral clustering

5 Conclusion

In this article, we present a comprehensive derivation of the spectral clustering algorithm from scratch and explore the formulation of related optimization problems. In the derivation, we emphasize the role of the spectral graph Laplacian in formulating and solving a relaxed version of the graph-normalized minimum cut problem. We also conduct extensive experiments to investigate the effects of modifying various aspects of spectral clustering algorithms. By examining the theoretical properties of these modifications and analyzing their performance on diverse datasets, we aim to deepen our understanding of optimization problems within the context of spectral clustering. This study could provide a gentle entry point for the examing of optimization problems in spectral clustering.

Reference

[1] S. P. Lloyd, "Least squares quantization in pcm.," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–136, 1982. [Online]. Available: http://dblp.uni-trier.de/db/journals/tit/tit28.html#Lloyd82

[2] U. von Luxburg, "A tutorial on spectral clustering.," *Stat. Comput.*, vol. 17, no. 4, pp. 395–416, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/sac/sac17.html#Luxburg07

[3] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: analysis and an algorithm," in *Advances Neural Inf. Process. Syst.*, 2001, pp. 849–856. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.8100

[4] F. Pedregosa, G. Varoquaux, et al., "Scikit-learn: machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[5] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*, Second, Cambridge University Press, 2014. [Online]. Available: http://mmds.org/

[6] S. Guattery, and G. L. Miller, "On the quality of spectral separators," *SIAM J. Matrix Anal. Appl.*, vol. 19, no. 3, pp. 701–719, 1998, doi: 10.1137/S0895479896312262.

[7] L. Huang, D. Yan, N. Taft, and M. Jordan, "Spectral clustering with perturbed data," in *Advances Neural Inf. Process. Syst.*, vol. 21, 2008. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2008/file/185e65bc40581880c4f2c82958de8cfe-Paper.pdf

[8] F. Zhang, J. Zhao, X. Ye, and H. Chen, "One-step adaptive spectral clustering networks," *IEEE Signal Process. Lett.*, vol. 29, no. , pp. 2263–2267, 2022, doi: 10.1109/LSP.2022.3217441.

[9] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: locally optimal block preconditioned conjugate gradient method," *SIAM J. Scientific Comput.*, vol. 23, no. 2, pp. 517–541, 2001, doi: 10.1137/S1064827500366124.

[10] G. H. Golub, and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 2013.

[11] P. Virtanen, R. Gommers, et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[12] S. K. Lam, A. Pitrou, and S. Seibert, "Numba: a llvm-based python jit compiler," in *Proc. Second Workshop LLVM Compiler Infrastructure Hpc* in Llvm '15, Austin, Texas, 2015, doi: 10.1145/2833157.2833162. [Online]. Available: https://doi.org/10.1145/2833157.2833162

[13] C. R. Harris, K. J. Millman, et al., "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[14] C. Elkan, "Using the triangle inequality to accelerate k-means," in *Proc. Twentieth Int. Conf. Int. Conf. Mach. Learn.* in Icml'03, Washington, DC, USA, 2003, p. 147.

[15] J. Calder, "Graphlearning python package," Zenodo, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5850940