Pass-Efficient Algorithms for Graph Spectral Clustering (Student Abstract)

Boshen Yan[*1,2], **Guihong Wan**[*1,3], **Haim Schweitzer**[4], **Zoltan Maliga**[3],
Sara Khattab[1], **Kun-Hsing Yu**[2], **Peter K. Sorger**[3], **Yevgeniy R. Semenov**[1,3]
[1]Department of Dermatology, Massachusetts General Hospital, Harvard Medical School, MA, USA
[2]Department of Biomedical Informatics, Harvard Medical School, MA, USA
[3]Laboratory of Systems Pharmacology, Department of Systems Biology, Harvard Medical School, MA, USA
[4]Department of Computer Science, University of Texas at Dallas, Texas, USA
boshenyan@hms.harvard.edu, guihong_wan@hsph.harvard.edu

Abstract

Graph spectral clustering is a fundamental technique in data analysis, which utilizes eigenpairs of the Laplacian matrix to partition graph vertices into clusters. However, classical spectral clustering algorithms require eigendecomposition of the Laplacian matrix, which has cubic time complexity. In this work, we describe pass-efficient spectral clustering algorithms that leverage recent advances in randomized eigendecomposition and the structure of the graph vertex-edge matrix. Furthermore, we derive formulas for their efficient implementation. The resulting algorithms have a linear time complexity with respect to the number of vertices and edges and pass over the graph constant times, making them suitable for processing large graphs stored on slow memory. Experiments validate the accuracy and efficiency of the algorithms.

Introduction

Let $G{=}(V,W)$ be an undirected graph, where V is the n-vertex set, and W is the $n{\times}n$ weighted adjacency matrix. Let w_{ij} be the non-negative weight between nodes i,j. An edge exists between nodes i and j if $w_{ij}{>}0$. For an unweighted graph, $w_{ij}{=}1$ if there is an edge; $w_{ij}{=}0$ otherwise.

Let m be the number of edges. The edge $\mathbf{e}$ between nodes i,j is represented as an n-dimensional vector: $\mathbf{e} = \sqrt{w_{ij}}[0,\ldots,0,1,0,\ldots,0,-1,0,\ldots,0]^T$, where the ith location is 1, the jth location is -1, and other locations are 0. Let $E = [\mathbf{e}_0,\ldots,\mathbf{e}_{m-1}]$ be the vertex-edge matrix of size $n \times m$ obtained by taking the m edge vectors as its columns. Let D be the degree matrix of the graph with the ith diagonal element: $d_i = \sum_{j=0}^{n-1} w_{ij}$. The graph Laplacian matrix is defined as: $L = D - W$, which can be computed as follows:

$$L = EE^T = \sum_{t=0}^{m-1} \mathbf{e}_t \mathbf{e}_t^T. \tag{1}$$

The normalized Laplacian matrix is: $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$. Let $\mathcal{E} = [\epsilon_0,\ldots,\epsilon_{m-1}]$ be the normalized vertex-edge matrix with each n-dimensional edge ϵ vector normalized: $\epsilon = \sqrt{w_{ij}}[0,\ldots,0,\sqrt{d_i},0,\ldots,0,-\sqrt{d_j},0,\ldots,0]^T$. The normalized graph Laplacian matrix can be computed as:

$$\mathcal{L} = \mathcal{E}\mathcal{E}^T = \sum_{t=0}^{m-1} \epsilon_t \epsilon_t^T. \tag{2}$$

The classical spectral clustering algorithms take the (normalized) Laplacian matrix and the desired number of clusters k as input. They mainly consist of two steps, as summarized in Algorithm 1 (Von Luxburg 2007). We focus on the normalized variant, since it is straightforward to apply the proposed techniques to the unnormalized variant.

Spectral clustering has many applications in data analysis and computer vision. However, the eigendecomposition in Step 1 has cubic time complexity, making it computationally expensive, especially for large graphs. Recent advances in randomized eigendecomposition, e.g., Halko et al. (2011), have enabled the efficient computation of the k largest eigenvectors (corresponding to the k largest eigenvalues).

In this work, we propose a $T{+}1$-pass algorithm and a one-pass algorithm for normalized spectral clustering, leveraging the structure and sparsity of the vertex-edge matrix and utilizing randomized eigendecomposition. They run significantly faster than the classical normalized spectral clustering algorithm. Even with a small value of T, the $T{+}1$-pass algorithm achieves the same level of accuracy as the classical spectral clustering algorithm. Additionally, we derive formulas to further improve their runtime while maintaining the same level of accuracy.

The Proposed Algorithms

Algorithm 2 presents the randomized $T{+}1$-pass algorithm for spectral clustering. The randomized eigendecomposition, e.g., Halko et al. (2011), computates the k largest eigenpairs. However, for spectral clustering, we need the k smallest eigenpairs of $\mathcal{L}$. We addressed this by calculating the k largest eigenpairs of a shifted Laplacian matrix: $\mathcal{L}_s{=}2\mathbf{I}{-}\mathcal{L}$.

```
Algorithm 1: Classical spectral clustering algorithm.
Input: $L$ or $\mathcal{L}$, and $k$.
Output: $k$ clusters of the graph vertices.
1. Compute the $k$ smallest eigenvectors $U_{n\times k}$ of $L$ or $\mathcal{L}$,
   corresponding to the $k$ smallest eigenvalues.
2. Run a $k$-means algorithm on (row-normalized) $U_{n\times k}$.
```

*These authors contributed equally.

Algorithm 2: Randomized T+1-pass algorithm.

```
Input: $\mathcal{E}$=[$\epsilon_0, \dots, \epsilon_{m-1}$], $k$, $T \geq 1$, and $p$=10 (default).
Output: $k$ clusters of the graph vertices.
1. $l = k + p$; $\Omega_{n \times l} \leftarrow$ a random matrix; $Q = \text{orth}(\Omega)$.
2. For $\tau = 1 \cdots T$ do:                    // $Q$ phase: $T$ passes.
   $Y_{n \times l} = 2Q - \sum_{t=0}^{m-1} \epsilon_t \epsilon_t^T Q$;  $Q_{n \times l} = \text{orth}(Y)$.
3. $W_{l \times l} = 2\mathbf{I} - Q^T \sum_{t=0}^{m-1} \epsilon_t \epsilon_t^T Q$.      //$W$ phase: 1 pass.
4. $\Lambda, U \leftarrow \text{Eigendecomposition}(W), U$=$QU$.
5. $U_{n \times k} \leftarrow k$ eigenvectors from $U$ corresponding to the $k$
   largest eigenvalues in $\Lambda$.
6. Run a $k$-means algorithm on row-normalized $U_{n \times k}$.
```

Algorithm 3: Efficient Randomized T+1-pass algorithm.

```
Input: $\mathcal{N}$=[$\mathbf{v}_0, \dots, \mathbf{v}_{n-1}$], $k$, $T \geq 1$, and $p$=10 (default).
Output: $k$ clusters of the graph vertices.
1. $l = k + p$; $\Omega_{n \times l} \leftarrow$ a random matrix; $Q = \text{orth}(\Omega)$.
   $d_{n \times 1} \leftarrow$ a zero vector for storing vertex degrees.
2. For $\tau = 1 \cdots T$ do:                    // $Q$ phase: $T$ passes.
     $Y = Q$.
     For $\mathbf{v}_i \in \mathcal{N}$ do:
       For $(v_j, w_{ij}) \in \mathbf{v}_i$ do: $d_i = d_i + w_{ij}$.
       For $(v_j, w_{ij}) \in \mathbf{v}_i$ do:
         if $d_j > 0$:
           $Y[j,:] = Y[j,:] - d_j^{-\frac{1}{2}} d_i^{-\frac{1}{2}} w_{ij} Q[i,:]$;
           $Y[i,:] = Y[i,:] - d_j^{-\frac{1}{2}} d_i^{-\frac{1}{2}} w_{ij} Q[j,:]$.
     $Y = 2Q - Y; Q = \text{orth}(Y)$.
3. $W_{l \times l} \leftarrow$ a zero matrix.               // $W$ phase: 1 pass.
   For $\mathbf{v}_i \in \mathcal{N}$ do:
       For $(v_j, w_{ij}) \in \mathbf{v}_i$ do:
         $\mathbf{t} = Q[i,:]^T d_i^{-\frac{1}{2}} w_{ij}^{\frac{1}{2}} - Q[j,:]^T d_j^{-\frac{1}{2}} w_{ij}^{\frac{1}{2}}$.
         $W = W + \mathbf{t}\mathbf{t}^T$.
   $W = 2\mathbf{I} - W$.
4. $\Lambda, U \leftarrow \text{Eigendecomposition}(W), U = QU$.
5. $U_{n \times k} \leftarrow k$ columns from $U$ corresponding to the $k$
   largest eigenvalues in $\Lambda$.
6. Run a $k$-means algorithm on row-normalized $U_{n \times k}$.
```

One limitation of Algorithm 2 is that it assumes knowledge of $\mathcal{E}$, which may not hold true in real applications. Also, $\mathcal{E}$ cannot be computed with a single pass over the graph because calculating any edge vector $\epsilon_t \in \mathcal{E}$ requires knowing both d_i and d_j for the nodes i, j that bound the edge. In Algorithm 3, we propose to relax this assumption by employing the vertex-neighbors representation of the graph: $\mathcal{N}$=[$\mathbf{v}_0, \dots, \mathbf{v}_{n-1}$], where $\mathbf{v}_i$=[$(v_j, w_{ij}), \dots, (v_k, w_{ik})$] is a list containing its neighbors and the corresponding weights. The orth() denotes the orthonormalization of a given matrix. **Algorithm 3 Complexity.** Time complexity: $O(k^2(m{+}n))$; Memory complexity: $O(kn)$: Number of passes: T+1.

The key difference of the one-pass variants from Algorithms 2 and 3 is in the computation of W. By setting $T = 1$ in the Q phase and modifying the W phase with $W = 2\mathbf{I} - Q^T Y (Q^T \text{orth}(\Omega))^{-1}$ that does not require passing over the graph, we can have the one-pass variants.

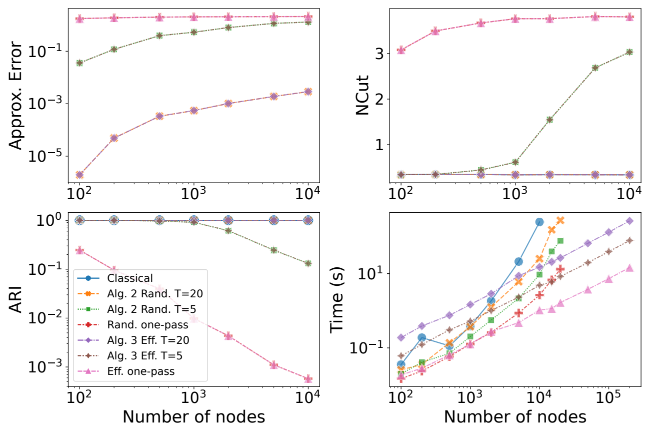

Figure 1: Performance of the proposed algorithms. Alg. 2 and Alg. 3 produce the same clusters at any given T, but Alg. 3 scales much better. Quality of clusters scales with T.

Experiments

We validated the algorithms by generating synthetic datasets with known cluster assignments and comparing to the classical algorithm (Ng et al. 2001). All experiments were carried out on a MacBook Air with 8 CPUs and 24 GB RAM.
Accuracy. We calculated the approximation error of an algorithm as the Frobenius norm between the computed eigenvectors and the ground truth eigenvectors of $\mathcal{L}$. Clustering quality was assessed using the normalized cut criterion (NCut) (Von Luxburg 2007) and the adjusted Rand score (ARI) (Hubert and Arabie 1985) with the ground truth cluster labels. Figure 1 presents the experimental results.

As expected, Algorithm 2 and Algorithm 3 output identical results with any T. (Results are overlapped in Figure 1.) When $T \geq 20$, the proposed Algorithm 3 produces clusters concordant with the classical algorithm (Ng et al. 2001).
Runtime. The results are shown in the last plot of Figure 1. The proposed Algorithm 3 scales linearly with the size of graphs and is more than **15X** faster than the classical algorithm in graphs with 10^4 nodes. It is also memory efficient and runs on large graphs. When the number of nodes is greater than 10^4, other algorithms failed to produce results.

Future Work

An important step in modern single-cell analysis workflows is the identification of distinct cell types via clustering algorithms. The most popular method is the Leiden algorithm, which efficiently identifies cell clusters through modularity maximization (Traag, Waltman, and Van Eck 2019). We observed that Algorithm 3 achieves similar efficiency, identifying cell clusters in a dataset of 200,000 cells within five minutes. Additionally, Algorithm 3 can sometimes identify small but distinct clusters of cells that were overlooked by the Leiden algorithm even when the same number of clusters is obtained. These results demonstrate the potential utility of spectral clustering as an alternative clustering algorithm in single-cell data analysis. In the future, we will conduct more detailed experiments on the application to single-cell data.

References

Halko, N.; et al. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2): 217–288.

Hubert, L.; and Arabie, P. 1985. Comparing partitions. *Journal of classification*, 2: 193–218.

Ng, A.; et al. 2001. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Traag, V. A.; Waltman, L.; and Van Eck, N. J. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1): 5233.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17: 395–416.