Research Report - Spectral Clustering

Sourabh Antani

1 Introduction

Clustering is a cornerstone of exploratory data analysis. Today, clustering is applied in a wide range of fields ranging from population analysis, consumer segmentation, to detection and molecular profiling of diseases. Spectral clustering is especially useful when the structure of the individual clusters is non-convex or when the 'shape' of the data is not suitable to be described by a measure of center and spread of clusters in euclidean space. However, spectral clustering also faces a few practical challenges since it requires computation of eigenvectors and application of k-means. In this paper, we examine some of the proposed ways of alleviating these challenges and explore the application of a Chebyshev-Davidson method, to calculate the required smallest eigenvalues and corresponding eigenvectors without the need for full eigen-decomposition, as a means to accelerate spectral clustering.

2 The main framework of spectral clustering

The main structure of a spectral clustering algorithm includes three parts:

1. From the affinity matrix and the Laplacian matrix.
2. Compute the outer eigenvalues and their associated eigenvectors of the Laplacian.
3. Apply k-means to the normalized rows of the eigenvectors for clustering.

Assuming the goal is to cluster n data points $S = \{s_1, \ldots, s_n\} \subset \mathbb{R}^L, n \geq L$ into k clusters. A more detailed spectral clustering algorithm is presented in Algorithm **??**, it summarizes the three spectral clustering algorithms surveyed in [1]. A slight difference from [1] is that the normalization of the rows of eigenvectors as proposed in [2] are applied to all three Laplacians.

Algorithm 1 Spectral Clustering Algorithm

1: Form the affinity matrix $A \in \mathbb{R}^{n \times n}$, one common way is to set $A_{ij} = e^{-||s_i - s_j||^2}/\mu^2$ for a chosen $\mu \neq 0$, and $A_{ii} = 0$
2: Compute a Laplacian matrix from A, such as the random walk Laplacian $I - D^{-1}A$, or the symmetric Laplacian $I - D^{-1/2}AD^{-1/2}$, or simply $D - A$. Here D is the diagonal degree matrix, its i-th diagonal is the sum of degrees at vertex $i : d_i = \sum_{j=1}^{n} A_{ij}$.
3: Compute the largest k eigenvalues and their associated eigenvectors of the chosen Laplacian.
4: Normalize the n rows of the eigenvector matrix to unit length, treat the n rows as n points on the unit hyper-sphere in $\mathbb{R}^k$, apply k-means to cluster these points into k clusters.
5: Assign the same grouping of the rows of the eigenvectors to the original data. i.e., s_i is assigned to cluster j if and only if row-i of the eigenvector matrix is assigned so.

In [2], the scaled affinity matrix $D^{-1/2}AD-1/2$ is used. thus the smallest k eigenvalues and their associated eigenvectors are computed for clustering.

There exists rather extensive literature that tries to extend the power of spectral clustering to larger data. They can be grouped into two categories: 1. Building the affinity matrix utilizing some KNN techniques; 2. Computing the eigenvectors only approximately, either by filtering or utilizing some localized techniques. We focus on the second category. Our method can utilize any advancements that have been made for the first category.

3 Existing acceleration schemes

For large dataset, the adjacency matrix as well as the Laplacians are of high dimension. Computing the eigenvectors using standard eigen-algorithm would suffer the $O(n^3)$ complexity. Hence, methods of lower than cubic-order complexity are needed.

Fowlkes *et al.* [3] proposed the Nyström method to accelerate the computation of eigenvectors.

Tremblay *et al.* [4] proposed applying polynomials that approximate a low-pass step function to filter out unwanted spectrum of the adjacency matrix. Note that the lower part of the spectrum of the adjacency matrix would translate to the higher part of the spectrum of a Laplacian.

Wu *et al.* [**?**] proposed used of random binning to reduce the computational coast and memory footprint by selecting a representative element from each bin for clustering and finally extrapolating the cluster assignment to the full dataset.

4 Proposed acceleration scheme

As shown in Proposition 1 and Proposition 3 of [1], the Laplacian associated with the spectral clustering problem is a positive semi-definite symmetric matrix with real-valued eigenvalues. This means that we can employ a method that takes advantage of this structure to compute the eigenpairs of the Laplacian. However, most eigenvalue computation algorithms compute the largest eigenvalues first. This means that full eigendecomposition will be needed to compute the smallest k eigenvalues of the Laplacian. Chebyshev filtering can be applied as described below to directly compute only the required smallest eigenpairs directly.

Chebyshev polynomials of the first kind are defined as (see [**?**], Chapter 4, section 4.4)

$$C_k(t) = \begin{cases} \cos(k\cos^{-1}(t)), & -1 \le t \le 1, \\ \cosh(k\cosh^{-1}(t)), & |t| > 1. \end{cases}$$

Note that $C_0(t) = 1, C_1(t) = t$ and the three-term recurrence, $C_{k+1}(t) = 2tC_k(t) - C_{k-1}(t), k > 1, t \in \mathbb{R}$. Thus, $C_k(t)$ is a polynomial of t with degree k. A noteworthy property of Chebyshev polynomial is that its value $C_m(t)$ oscillates between -1 and 1 for $t \in [-1, 1]$ and its value increases in magnitude very rapidly outside $[-1, 1]$ as illustrated in Figure 1.

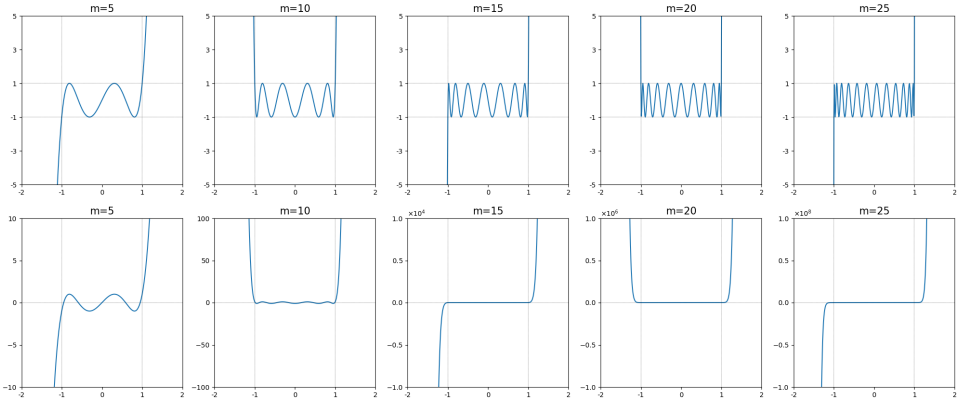

Figure 1: Behavior of Chebyshev polynomial of degree m within and outside [-1,1]

Given a Matrix A, let (λ, x) be one of its eigenpairs, i.e. $Ax = \lambda x$. If $p_m(x)$ is a polynomial of degree m, $p_m(A)x = p_m(\lambda)x$ since $\alpha Ax = \alpha\lambda x$ and $A^\beta x = \lambda^\beta x$, for $\alpha, \beta \in \mathbb{R}$.

Let $\lambda_0 \le \lambda_1 \le \cdots \le \lambda_k < \lambda_{k+1} \le \cdots \le \lambda_n$ be the eigenvalues of the matrix A. Since we need to find the smallest k eigenvalues, we would need to magnify the eigenvalues in $[\lambda_0, \lambda_k]$ and dampen the eigenvalues $\ge \lambda_{k+1}$. This can be achieved by mapping the interval $[a, b]$, such that $\lambda_k < a \le \lambda_{k+1}$ and $\lambda_n \le b$, to

$[-1, 1]$ by an affine mapping $x \to \frac{(x-c)}{e}$ where $c = \frac{a+b}{2}$ is the center of interval and $e = \frac{b-a}{2}$ is its half width. Thus,

$$y = p_m(A)x, \quad \text{where } p_m(t) = C_m\left(\frac{t-c}{e}\right)$$
$$x_{j+1} = \frac{2}{e}(A - cI)x_j - x_{j-1}, \quad j = 1, 2, \ldots, m-1.$$

It is crucial that the upper-bound b bounds the eigen values from above. Otherwise the filtering will magify the largest eigenvalues as well, defeating the purpose. The upper bound of the interval b can be computed using Gershgorin circle theorem which states that each eigenvalue must lie in a disc with radius R_i where,

$$\begin{aligned} R_i = |\lambda - a_{ii}| &\leq \sum_{j \neq i} |a_{ij}| \\ |\lambda| - |a_{ii}| \leq |\lambda - a_{ii}| &\leq \sum_{j \neq i} |a_{ij}| \\ \implies |\lambda| &\leq \sum_{j} |a_{ij}| \\ \implies |\lambda_n| &\leq \max \sum_{j} |a_{ij}| \end{aligned}$$

Since the Laplacian is a positive semi-definite symmetric matrix with positive real-valued eigenvalues, $\lambda_n \leq ||A||_1$. Alternately k-step Lanczos method described in Algorithm 4.4 of [**?**] can also be used. The lower-bound can be any value greater than λ_k. In the algorithm **??**, a is initialized as Rayleigh quotient with a random unit vector and subsequently approximated as Ritz value from previous step.

Here we present a brief overview of the method and refer to [5] for a detailed discussion of this method.

Algorithm 2 Chebyshev polynomial filtering of degree m

function CHEBYSHEV_FILTER(A, x, m, a, b, a_0)
  $e = (b-a)/2; c = (b+1)/2; \sigma = e/(a_0 - c); \sigma_1 = \sigma$
  $y = (Ax - cx)\sigma_1/e$
  for $i = 2 : m$ **do**
    $\sigma_{new} = \frac{1}{(a/\sigma_1 - \sigma)}$
    $y_{new} = 2(Ay - cy)\sigma_{new}/e - \sigma\sigma_{new}x$
    $x = y; y = y_{new}; \sigma = \sigma_{new}$
  end for
  return y
end function

For numerical stability, DGKS method is used for orthogonalization. The deflation of the eigenvectors is achived by indexing the columns of the projection basis V.

5 Experiments

To test our python implementation of the Chebyshev-Davidson method, we generated two toy-datasets using scikit python learn library and used normalized spectral clustering by replacing the standard numpy eigenvalue function call with our Chebyshev-Davidson function call. As shown in Figure 2 the two clusters were identified appropriately.

Next we used the Spectral clustering algorithm on two graphs generated using Networkx package [**?**], Gaussian random partition graph [**?**] and LFR Benchmark Graph [**?**].

Algorithm 3 Chebyshev-Davidson method for computing k_{want} smallest eigenpairs

Input: x - initial unit vector, can be initialized as a random vector; m-degree of polynomial; k_{keep}-number of vectors to keep during restart; dim_{max}-maximum subspace dimension; τ - tolerance
Output: k_c converged smallest eigenpairs: $eval[1:k_c], V[1:k_c]$.
Initialize $V = [x]$
$w = Ax$, $W = [w]$, $\mu = x^T w$, $H = [\mu]$
residual vector: $r = w - \mu x$
if $||r|| \leq \tau$ **then**
    $eval[1] = \mu$, $k_c = 1$, $H = []$
else
    $k_c = 0$
end if
Estimate the upper-bound $upperb$ using minimum of k-step Lanczos and $||A||_1$
Set lower-bound $lowerb = (uppeprb + \mu)/2$, $a_0 = lowerb$
Current subspace dimension $k_sub = 1$
while $iter \leq iter_{max}$ **do**
    $[t] = CHEBYSHEV\_FILTER(A, x, m, lowerb, upperb, a_0)$
    Orthonormalize t against first k_{sub} columns of V to get v and append to V. $k_{sub} = k_{sub}+1; k_{old} = k_{sub}$
    $w = Av$. Append w to W
    Compute last column of symmetric Rayleigh-Quotient Matrix $H[1:k_{sub}-k_c, k_{sub}-k_c] = V[:, k_c+1:k_{sub}]^T w$
    Compute eigendecomposition of $HY = YD$ with $diag(D)$ sorted in non-increasing order. $\mu = D[1,1]$
    if $k_{sub} \geq dim_{max}$ **then**
        restart: $k_{sub} = k_c + k_{keep}$
    end if
    $V[:, k_c+1:k_{sub}] = V[:k_c+1:k_{old}]Y[:, 1:k_{sub}-k_c]$; $W[:, k_c+1:k_{sub}] = W[:k_c+1:k_{old}]Y[:, 1:k_{sub}-k_c]$;
    residual vector: $r = W[:, k_c+1] - \mu V[:, k_c+1]$
    $iter = iter + 1$
    if $||r|| \leq \tau max(diag(D))$ **then**
        $k_c = k_c + 1; eval(kc) = \mu$
        sort converged eigenvalues in non-increasing and set $swap = 0$ if swap happens.
    end if
    if $k_c \geq k_{want}$ and no swap was needed **then**
        return $eval[1:k_c]$ and $V:, 1:k_c$
    end if
    update bounds for next iteration $lowerb = median(diag(D))$;
    if $a_0 > min(diag(D))$ **then**
        $a_0 = min(diag(D))$
    end if
    Ritz-vector for next iteration: $x = V[:, k_c+1]$
    $H = D[k_c+1:k_{sub}, k_c+1:k_{sub}]$
end while
return y

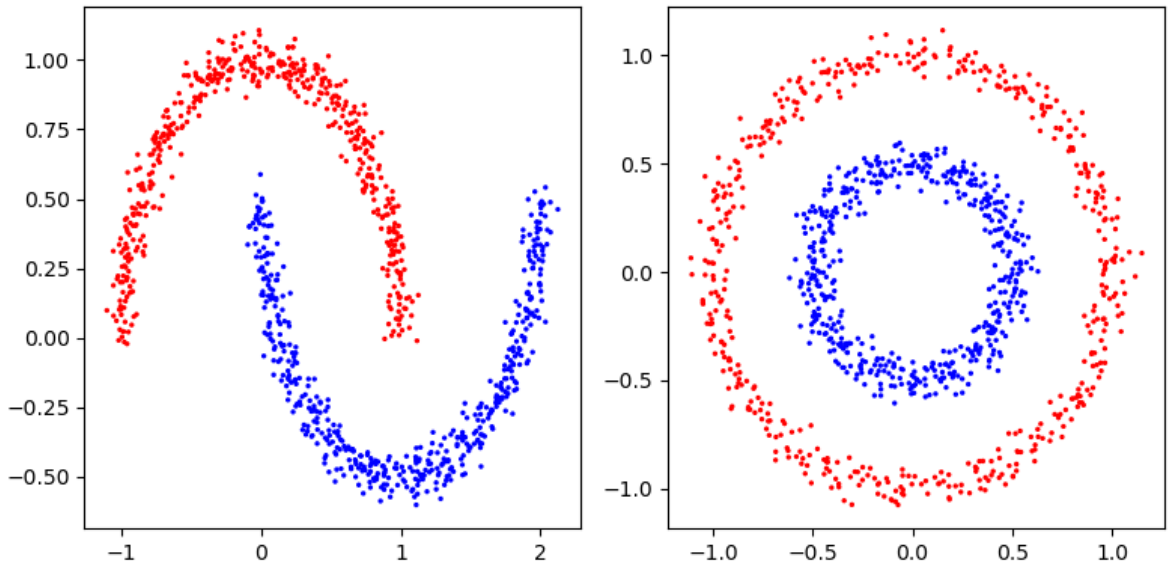

Figure 2: Spectral clustering of moons and concentric circles

6 Results

Our goal for this experiment was to improve the efficiency of spectral clustering process without sacrificing the clustring quality. The use of Chebyshev-Davidson method offers a two-fold advantage, we only need to compute the required set of eigen-vectors there by eliminating computation of full set of eigen-vectors, and by increasing the eigen-gap between the eigenvalues of interest to speed up the convergence. We chose the following metrics to compare the clusters detected with Chebyshev-Davidson method with the clusters detected using default algorithm employed by numpy.linalg package in python 3.12.3.

F1-Score:: F1-score is a measure of predictive performance. When the true labels (or clusters) are known, precision is defined as the ratio of samples correctly assigned to a label to the total number of samples assigned to the label. Also, recall is defined as ration of samples correctly assigned to a label to the total number of samples that actually belong to the label. Finally, F1-score is defined as the harmonic mean of precision and recall. In case of binary classification, where the labels can be Positive and Negative, the following formula shows computation of F1-score. In our case, we use the clusters identified by default numpy implementation as 'truth' to how well our results match. The range of F1-score is $[0, 1]$ where 1 would indicate perfect match.

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = \frac{2TP}{2TP + FP + FN}$$

Normalized Mutual Information (NMI): NMI is a metric that yields a value from 0 to 1 and indicates the similarity between two clusters. The following formula shows the computation

$$NMI\,(A, B) = \frac{-2\sum_{i=1}^{S}\sum_{j=1}^{R} C_{ij}\log\frac{C_{ij}N}{C_{i.}C_{.j}}}{\sum_{i=1}^{S} C_{i.}\log\frac{C_{i.}}{N} + \sum_{j=1}^{R} C_{.j}\log\frac{C_{.j}}{N}}$$

where C_i and C_J are the number of samples in clusters i and j from two clustering outcomes, while C_{ij} is the number of samples that are common between those clusters.

After initial experiments with datasets generated from scikit-learn and networkx libraries, we also tested the performance with some of the datasets from SNAP [6].

The results of our experiments are shown below. All the experiments reported below were performed on shared compute environment. Each experiment was repeated 3 times and median value was taken to reduce the variation due to concurrent processes and process, network and storage load. All the

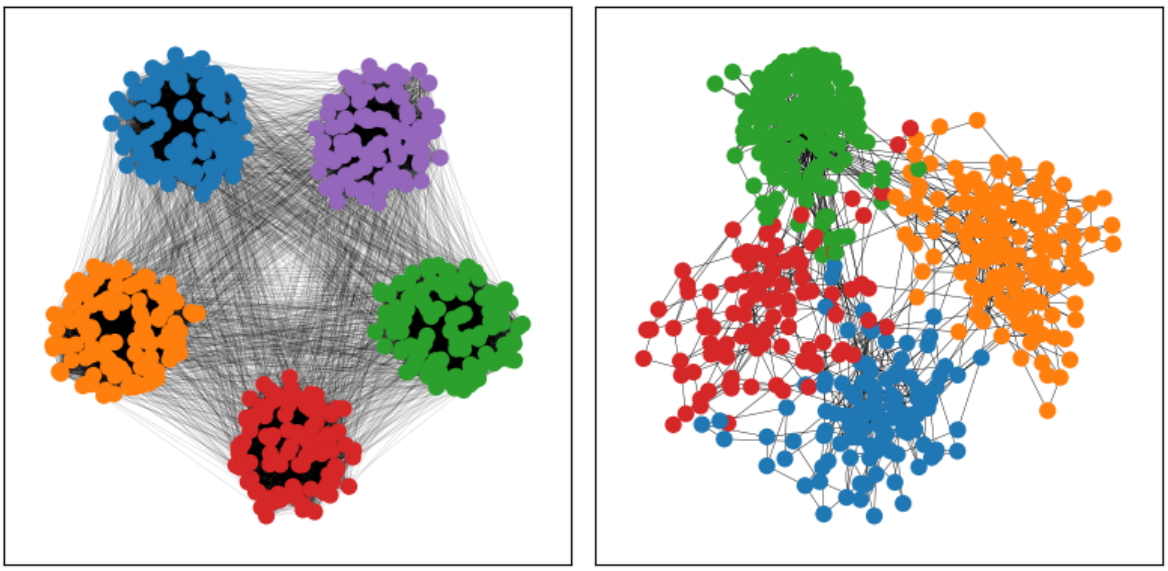

Figure 3: Spectral clustering of Gaussian random partition graph and LFR Benchmark community detection graphs

Dataset	Cheb-Dav	Default	F1-score	NMI Score
NetworkX Gaussian random partitions	0.27 s	0.37 s	1.0	1.0
NetworkX LFR benchmark graph	0.27 s	0.38 s	1.0	1.0
SNAP: email-Eu-core network	2.47 s	1.98 s	1.0	1.0
SNAP: Astro Physics collaboration network*	4 m 22 s	1 hr 5 m 34 s	0.98	0.92

Table 1: Clustering efficiency and similarity to default implementation

measurements shown are median values of the 3 readings taken. All values less than 10 are rounded to 2 digits after decimal point. [One exception to this was the Astro Physics collaboration network which was only performed once only. I will send an update with multiple runs and also add more datasets]

7 Concluding Remarks

References

[1] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[2] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Proc. NIPS*, pages 849–856, 2001.

[3] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[4] Nicolas Tremblay, Gilles Puy, Remi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering, arXiv:1602.02018, 2016.

[5] Y. Zhou and Y. Saad. A Chebyshev-Davidson algorithm for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):954–971, 2007.

[6] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.