Compressive Spectral Clustering

Nicolas Tremblay[*†] NICOLAS.TREMBLAY@INRIA.FR
Gilles Puy[§*] GILLES.PUY@TECHNICOLOR.COM
Rémi Gribonval[*] REMI.GRIBONVAL@INRIA.FR
Pierre Vandergheynst[†*] PIERRE.VANDERGHEYNST@EPFL.CH

[*] INRIA Rennes - Bretagne Atlantique, Campus de Beaulieu, FR-35042 Rennes Cedex, France
[†] Institute of Electrical Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
[§] Technicolor, 975 Avenue des Champs Blancs, 35576 Cesson-Sévigné, France

Abstract

Spectral clustering has become a popular technique due to its high performance in many contexts. It comprises three main steps: create a similarity graph between N objects to cluster, compute the first k eigenvectors of its Laplacian matrix to define a feature vector for each object, and run k-means on these features to separate objects into k classes. Each of these three steps becomes computationally intensive for large N and/or k. We propose to speed up the last two steps based on recent results in the emerging field of graph signal processing: graph filtering of random signals, and random sampling of bandlimited graph signals. We prove that our method, with a gain in computation time that can reach several orders of magnitude, is in fact an approximation of spectral clustering, for which we are able to control the error. We test the performance of our method on artificial and real-world network data.

1. Introduction

Spectral clustering (SC) is a fundamental tool in data mining (Nascimento & de Carvalho, 2011). Given a set of N data points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, the goal is to partition this set into k weakly inter-connected clusters. Several spectral clustering algorithms exist, *e.g.*, (Belkin & Niyogi, 2003; Ng et al., 2002; Shi & Malik, 2000; Zelnik-Manor & Perona, 2004), but all follow the same scheme. First, compute weights $\mathsf{W}_{ij} \geqslant 0$ that model the similarity between pairs of data points $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This gives rise to a graph $\mathcal{G}$ with N nodes and adjacency matrix $\mathsf{W} = (\mathsf{W}_{ij})_{1 \leqslant i,j \leqslant N} \in \mathbb{R}^{N \times N}$. Second, compute the first k eigenvectors $\mathsf{U}_k := (\boldsymbol{u}_1, \dots, \boldsymbol{u}_k) \in \mathbb{R}^{N \times k}$ of the Laplacian matrix $\mathsf{L} \in \mathbb{R}^{N \times N}$ associated to $\mathcal{G}$ (see Sec. 2 for L's definition). And finally, run k-means using the rows of U_k as feature vectors to partition the N data points into k clusters. This k-way scheme is a generalisation of Fiedler's pioneering work (1973).

SC is mainly used in two contexts: 1) if the N data points show particular structures (*e.g.*, concentric circles) for which naive k-means clustering fails; 2) if the input data is directly a graph $\mathcal{G}$ modeling a network (White & Smyth, 2005), such as social, neuronal, or transportation networks. SC suffers nevertheless from three main computational bottlenecks for large N and/or k: the creation of the similarity matrix W; the partial eigendecomposition of the graph Laplacian matrix L; and k-means.

1.1. Related work

Circumventing these bottlenecks has raised a significant interest in the past decade. Several authors have proposed ideas to tackle the eigendecomposition bottleneck, *e.g.*, via the power method (Boutsidis & Gittens, 2015; Lin & Cohen, 2010), via a careful optimisation of diagonalisation algorithms in the context of SC (Liu et al., 2007), or via matrix column-subsampling such as in the Nyström method (Fowlkes et al., 2004), the nSPEC and cSPEC methods of (Wang et al., 2009), or in (Chen & Cai, 2011; Sakai & Imiya, 2009). All these methods aim to quickly compute feature vectors, but k-means is still applied on N feature vectors. Other authors, inspired by research aiming at reducing k-means complexity (Jain, 2010), such as the line of work on coresets (Har-Peled & Mazumdar, 2004), have proposed to circumvent k-means in high dimension by subsampling a few data points out of the N available ones, applying SC on its reduced similarity graph, and interpolating the results back on the complete dataset. One can find similar methods in (Yan et al., 2009) and (Wang et al., 2009)'s eSPEC proposition, where two different interpolation methods are used. Both methods are heuristic:

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

there is no proof that these methods approach the results of SC. Also, let us mention (Dhillon et al., 2007) that circumvents both the eigendecomposition and the k-means bottlenecks: the authors reduce the graph's size by successive aggregation of nodes, apply SC on this small graph, and propagate the results on the complete graph using kernel k-means to control interpolation errors. The kernel is computed so that kernel k-means and SC share the same objective function (Filippone et al., 2008). Finally, we mention works (Boutsidis et al., 2011; Cohen et al., 2015) that concentrate on reducing the feature vectors' dimension in the k-means problem, but do not sidestep the eigendecomposition nor the large N issues.

1.2. Contribution: compressive clustering

In this work, inspired by recent advances in the emerging field of graph signal processing (Sandryhaila & Moura, 2014; Shuman et al., 2013), we circumvent SC's last two bottlenecks and detail a fast approximate spectral clustering method for large datasets, as well as the supporting theory. We suppose that the Laplacian matrix $\mathsf{L} \in \mathbb{R}^{N \times N}$ of $\mathcal{G}$ is given. Our method is made of two ingredients.

The first ingredient builds upon recent works (Ramasamy & Madhow, 2015; Tremblay et al., 2016) that avoid the costly computation of the eigenvectors of L by filtering $O(\log(k))$ random signals on $\mathcal{G}$ that will then serve as feature vectors to perform clustering. We show in this paper how to incorporate the effects of non-ideal, but computationally efficient, graph filters on the quality of the feature vectors used for clustering.

The second ingredient uses a recent sampling theory of bandlimited graph-signals (Puy et al., 2015) to reduce the computational complexity of k-means. Using the fact that the indicator vectors of each cluster are approximately bandlimited on $\mathcal{G}$, we prove that clustering a random subset of $O(k \log(k))$ nodes of $\mathcal{G}$ using random features vectors of size $O(\log(k))$ is sufficient to infer rapidly and accurately the cluster label of all N nodes of the graph. Note that the complexity of k-means is reduced to $O(k^2 \log^2(k))$ instead of $O(Nk^2)$ for SC. One readily sees that this method scales easily to large datasets, as will be demonstrated on artifical and real-world datasets containing up to $N = 10^6$ nodes.

The proposed compressive spectral clustering method can be summarised as follows:

- generate a feature vector for each node by filtering $O(\log(k))$ random Gaussian signals on $\mathcal{G}$;
- sample $O(k \log(k))$ nodes from the full set of nodes;
- cluster the reduced set of nodes;
- interpolate the cluster indicator vectors back to the complete graph.

2. Background

2.1. Graph signal processing

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ be an undirected weighted graph with $\mathcal{V}$ the set of N nodes, $\mathcal{E}$ the set of edges, and $\mathbf{W}$ the weighted adjacency matrix such that $W_{ij} = W_{ji} \geqslant 0$ is the weight of the edge between nodes i and j.

The graph Fourier matrix. Consider the graph's normalized Laplacian matrix $\mathsf{L} = \mathsf{I} - \mathsf{D}^{-1/2}\mathsf{W}\mathsf{D}^{-1/2}$ where I is the identity in dimension N, and D is diagonal with $\mathsf{D}_{ii} = \sum_{j \neq i} \mathsf{W}_{ij}$. L is real symmetric and positive semi-definite, therefore diagonalizable as $\mathsf{L} = \mathsf{U}\Lambda\mathsf{U}^\intercal$, where $\mathsf{U} := (\boldsymbol{u}_1 | \boldsymbol{u}_2 | \dots | \boldsymbol{u}_N) \in \mathbb{R}^{N \times N}$ is the orthonormal basis of eigenvectors and $\Lambda \in \mathbb{R}^{N \times N}$ the diagonal matrix containing its sorted eigenvalues : $0 = \lambda_1 \leqslant \dots \leqslant \lambda_N \leqslant 2$ (Chung, 1997). By analogy to the continuous Laplacian operator whose eigenfunctions are the classical Fourier modes and eigenvalues their squared frequencies, the columns of U are considered as the graph's Fourier modes, and $\{\sqrt{\lambda_l}\}_l$ as its set of associated "frequencies" (Shuman et al., 2013). Other types of graph Fourier matrices have been proposed, *e.g.*, (Sandryhaila & Moura, 2013), but in order to exhibit the link between graph signal processing and SC, the Laplacian-based Fourier matrix appears more natural.

Graph filtering. The graph Fourier transform $\hat{\boldsymbol{x}}$ of a signal $\boldsymbol{x}$ defined on the nodes of the graph (called a *graph signal*) reads: $\hat{\boldsymbol{x}} = \mathsf{U}^\intercal \boldsymbol{x}$. Given a continuous filter function h defined on $[0, 2]$, its associated graph filter operator $\mathsf{H} \in \mathbb{R}^{N \times N}$ is defined as $\mathsf{H} := h(\mathsf{L}) = \mathsf{U} h(\Lambda) \mathsf{U}^\intercal$, where $h(\Lambda) := \mathrm{diag}(h(\lambda_1), h(\lambda_2), \cdots, h(\lambda_N))$. The signal $\boldsymbol{x}$ filtered by h is $\mathsf{H}\boldsymbol{x}$. In the following, we consider ideal low-pass filters, denoted by h_{λ_c}, that satisfy, for all $\lambda \in [0, 2]$,

$$h_{\lambda_c}(\lambda) = 1, \text{ if } \lambda \leqslant \lambda_c, \text{ and } h_{\lambda_c}(\lambda) = 0, \text{ if not.} \quad (1)$$

Denote by H_{λ_c} the graph filter operator associated to h_{λ_c}.

Fast graph filtering. To filter a signal by h without diagonalizing L, one may approximate h by a polynomial $\tilde{h}$ of order p satisfying $\tilde{h}(\lambda) := \sum_{l=0}^{p} \alpha_l \lambda^l \simeq h(\lambda)$ for all $\lambda \in [0, 2]$, where $\alpha_1, \dots, \alpha_p \in \mathbb{R}$. In matrix form, we have $\tilde{\mathsf{H}} := \tilde{h}(\mathsf{L}) = \sum_{l=0}^{p} \alpha_l \mathsf{L}^l \simeq \mathsf{H}$. *Let us highlight that we never compute the potentially dense matrix* $\tilde{\mathsf{H}}$ *in practice*. Indeed, we are only interested in the result of the filtering operation: $\tilde{\mathsf{H}}\boldsymbol{x} = \sum_{l=0}^{p} \alpha_l \mathsf{L}^l \boldsymbol{x} \approx \mathsf{H}\boldsymbol{x}$ for $\boldsymbol{x} \in \mathbb{R}^N$, obtainable with only p successive matrix-vector multiplications with L. The computational complexity of filtering a signal is thus $O(p \# \mathcal{E})$, where $\# \mathcal{E}$ is the number of edges of $\mathcal{G}$.

2.2. Spectral clustering

We choose here Ng et al.'s method (2002) based on the normalized Laplacian as our standard SC method. The input

Algorithm 1 Spectral Clustering (Ng et al., 2002)

Input: The Laplacian matrix L, the number of clusters k

1· Compute $\mathsf{U}_k \in \mathbb{R}^{N\times k}$, L's first k eigenvectors: $\mathsf{U}_k = (\boldsymbol{u}_1|\boldsymbol{u}_2|\cdots|\boldsymbol{u}_k)$.

2· Form the matrix $\mathsf{Y}_k \in \mathbb{R}^{N\times k}$ from U_k by normalizing each of U_k's rows to unit length: $(\mathsf{Y}_k)_{ij} = (\mathsf{U}_k)_{ij} \,/\sqrt{\sum_{j=1}^{k} \mathsf{U}_{ij}^2}$.

3· Treat each node i as a point in $\mathbb{R}^k$ by defining its feature vector $\boldsymbol{f}_i \in \mathbb{R}^k$ as the transposed i-th row of Y_k:

$$\boldsymbol{f}_i := \mathsf{Y}_k^\mathsf{T}\boldsymbol{\delta}_i,$$

where $\delta_i(j) = 1$ if $j = i$ and 0 otherwise.

4· To obtain k clusters, run k-means with the Euclidean distance:

$$D_{ij} := \|\boldsymbol{f}_i - \boldsymbol{f}_j\| \tag{2}$$

is the adjacency matrix W representing the pairwise similarity of all the N objects to cluster[1]. After computing its Laplacian L, follow Alg. 1 to find k classes.

3. Principles of CSC

Compressive spectral clustering (CSC) circumvents two of SC's bottlenecks, the partial diagonalisation of the Laplacian and the high-dimensional k-means, thanks to the following ideas.

1) Perform a controlled estimation $\tilde{D}_{ij}$ of the spectral clustering distance D_{ij} (see Eq (2)), *without* partially diagonalizing the Laplacian, by fast filtering a few random signals with the polynomial approximation $\tilde{h}_{\lambda_k}$ of the ideal low pass filter h_{λ_k} (see Eq. (1)). A theorem recently published independently by two teams (Ramasamy & Madhow, 2015; Tremblay et al., 2016) shows that this is possible when there is no normalisation step (step 2 in Alg. 1) and when the order p of the polynomial approximation tends to infinity, *i.e.*, when $\tilde{h}_{\lambda_k} = h_{\lambda_k}$. In Sec. 3.1, we provide a first extension of this theorem that takes into account normalisation. A complete extension that also takes into account the polynomial approximation error is presented in Sec. 4.2.

2) Run k-means on n randomly selected feature vectors out of the N available ones - thus clustering the corresponding n nodes into k groups - and interpolate the result back on the full graph. To guarantee robust reconstruction, we take advantage of our recent results on random sampling of k-bandlimited graph signals. In Sec. 3.2, we explain why these results are applicable to clustering and show that it is sufficient to sample $n = O(k \log k)$ features only! Note that to cluster data into k groups, one needs *at least* k samples. This result is thus optimal up to the extra $\log k$ factor.

[1] In network analysis, the raw data is directly W. In the case where one starts with a set of data points $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, the first step consists in deriving W from the pairwise similarities $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$. See (von Luxburg, 2007) for several choices of similarity measure s and several ways to create W from the $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

3.1. Ideal filtering of random signals

Definition 3.1 (Local cumulative coherence)**.** *Given a graph $\mathcal{G}$, the local cumulative coherence of order k at node i is*[2] $v_k(i) := \|\mathsf{U}_k^\mathsf{T}\boldsymbol{\delta}_i\| = \sqrt{\sum_{j=1}^{k} \mathsf{U}_{ij}^2}$.

Let us define the diagonal matrix: $\mathsf{V}_k(i,i) = 1/v_k(i)$. Note that we assume that $v_k(i) > 0$. Indeed, in the pathologic cases where $v_k(i) = 0$ for some nodes i, step 2 of the standard SC algorithm cannot be run either. Now, consider the matrix $\mathsf{R} = (\boldsymbol{r}_1|\boldsymbol{r}_2|\cdots|\boldsymbol{r}_d) \in \mathbb{R}^{N\times d}$ consisting of d random signals $\boldsymbol{r}_i$, whose components are independent Bernouilli, Gaussian, or sparse (as in Theorem 1.1 of (Achlioptas, 2003)) random variables. To fix ideas in the following, we consider the components as independent random Gaussian variables of mean zero and variance $1/d$. Consider the coherence-normalized filtered version of R, $\mathsf{V}_k\mathsf{H}_{\lambda_k}\mathsf{R} \in \mathbb{R}^{N\times d}$, and define node i's new feature vector $\tilde{\boldsymbol{f}}_i \in \mathbb{R}^d$ as the transposed i-th line of this filtered matrix:

$$\tilde{\boldsymbol{f}}_i := (\mathsf{V}_k\mathsf{H}_{\lambda_k}\mathsf{R})^\mathsf{T}\boldsymbol{\delta}_i.$$

The following theorem shows that, for large enough d,

$$\tilde{D}_{ij} := \left\|\tilde{\boldsymbol{f}}_i - \tilde{\boldsymbol{f}}_j\right\| = \|(\mathsf{V}_k\mathsf{H}_{\lambda_k}\mathsf{R})^\mathsf{T}(\boldsymbol{\delta}_i - \boldsymbol{\delta}_j)\|$$

is a good estimation of D_{ij} with high probability.

Theorem 3.2. *Let $\epsilon \in]0,1]$ and $\beta > 0$ be given. If d is larger than*

$$\frac{4+2\beta}{\epsilon^2/2 - \epsilon^3/3}\log N,$$

then with probability at least $1 - N^{-\beta}$, we have

$$(1-\epsilon)D_{ij} \leqslant \tilde{D}_{ij} \leqslant (1+\epsilon)D_{ij}.$$

for all $(i,j) \in \{1,\ldots,N\}^2$.

The proof is provided in the supplementary material.

In Sec. 4.2, we generalize this result to the real-world case where the low-pass filter is approximated by a finite order polynomial; we also prove that, as announced in the introduction, one only needs $d = O(\log k)$ features when using the downsampling scheme that we now detail.

3.2. Downsampling and interpolation

For $j = 1, \ldots, k$, let us denote by $\boldsymbol{c}_j \in \mathbb{R}^N$ the ground-truth indicator vector of cluster $\mathcal{C}_j$, *i.e.*,

$$(\boldsymbol{c}_j)_i := \begin{cases} 1 & \text{if } i \in \mathcal{C}_j, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall i \in \{1,\ldots,N\}.$$

[2] Throughout this paper, $\|.\|$ stands for the usual ℓ_2-norm.

To estimate $\boldsymbol{c}_j$, one could run k-means on the N feature vectors $\{\tilde{\boldsymbol{f}}_1, \dots, \tilde{\boldsymbol{f}}_N\}$, as done in (Ramasamy & Madhow, 2015; Tremblay et al., 2016). Yet, this is still inefficient for large N. To reduce the computational cost further, we propose to run k-means on a small subset of n feature vectors only. The goal is then to infer the labels of all N nodes from the labels of the n sampled nodes. To this end, we need 1) a low-dimensional model that captures the regularity of the vectors $\boldsymbol{c}_j$, 2) to make sure that enough information is preserved after sampling to be able to recover the vectors $\boldsymbol{c}_j$, and 3) an algorithm that rapidly and accurately estimates the vectors $\boldsymbol{c}_j$ by exploiting their regularity.

3.2.1. The low-dimensional model

For a simple regular (with nodes of same degree) graph of k disconnected clusters, it is easy to check that $\{\boldsymbol{c}_1, \dots, \boldsymbol{c}_k\}$ form a set of orthogonal eigenvectors of L with eigenvalue 0. All indicator vectors $\boldsymbol{c}_j$ therefore live in $\mathrm{span}(\mathsf{U}_k)$. For general graphs, we assume that the indicator vectors $\boldsymbol{c}_j$ live close to $\mathrm{span}(\mathsf{U}_k)$, *i.e.*, the difference between any $\boldsymbol{c}_j$ and its orthogonal projection onto $\mathrm{span}(\mathsf{U}_k)$ is small. Experiments in Section 5 will confirm that it is a good enough model to recover the cluster indicator vectors.

In graph signal processing words, one can say that $\boldsymbol{c}_j$ is approximately k-bandlimited, *i.e.*, its k first graph Fourier coefficients bear most of its energy. There has been recently a surge of interest around adapting classical sampling theorems to such bandlimited graph signals (Anis et al., 2015; Chen et al., 2015; Marques et al., 2015; Tsitsvero et al., 2015). We rely here on the random sampling strategy proposed in (Puy et al., 2015) to select a subset of n nodes.

3.2.2. Sampling and interpolation

The subset of feature vectors is selected by drawing n indices $\Omega := \{\omega_1, \dots, \omega_n\}$ uniformly at random from $\{1, \dots, N\}$ without replacement. Running k-means on the subset of features $\{\tilde{\boldsymbol{f}}_{\omega_1}, \dots, \tilde{\boldsymbol{f}}_{\omega_n}\}$ thus yields a clustering of the n sampled nodes into k clusters. We denote by $\boldsymbol{c}_j^r \in \mathbb{R}^n$ the resulting low-dimensional indicator vectors. Our goal is now to recover $\boldsymbol{c}_j$ from $\boldsymbol{c}_j^r$.

Consider that k-means is able to correctly identify $\boldsymbol{c}_1, \dots, \boldsymbol{c}_k \in \mathbb{R}^N$ using the original set of features $\{\boldsymbol{f}_1, \dots, \boldsymbol{f}_N\}$ with the SC algorithm (otherwise, CSC is doomed to fail from the start). Results in (Ramasamy & Madhow, 2015; Tremblay et al., 2016) show that k-means is also able to identify the clusters using the feature vectors $\{\tilde{\boldsymbol{f}}_1, \dots, \tilde{\boldsymbol{f}}_N\}$. This is explained theoretically by the fact that the distance between all pairs of feature vectors is preserved (see Theorem 3.2). Then, as choosing a subset $\{\tilde{\boldsymbol{f}}_{\omega_1}, \dots, \tilde{\boldsymbol{f}}_{\omega_n}\}$ of $\{\tilde{\boldsymbol{f}}_1, \dots, \tilde{\boldsymbol{f}}_N\}$ does not change the distance between the feature vectors, we can hope that k-means correctly clusters the n sampled nodes, provided that each cluster is sufficiently sampled. Experiments in Sec. 5 will confirm this intuition. In this ideal situation, we have

$$\boldsymbol{c}_j^r = \mathsf{M}\, \boldsymbol{c}_j, \tag{3}$$

where $\mathsf{M} \in \mathbb{R}^{n \times N}$ is the sampling matrix satisfying:

$$\mathsf{M}_{ij} := \begin{cases} 1 & \text{if } j = \omega_i, \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

To recover $\boldsymbol{c}_j$ from its n observations $\boldsymbol{c}_j^r$, Puy et al. (2015) show that the solution to the optimisation problem

$$\min_{\boldsymbol{x} \in \mathbb{R}^N} \left\| \mathsf{M}\boldsymbol{x} - \boldsymbol{c}_j^r \right\|_2^2 + \gamma\, \boldsymbol{x}^\mathsf{T} g(\mathsf{L}) \boldsymbol{x}, \tag{5}$$

is a faithful [3] estimation of $\boldsymbol{c}_j$, provided that $\boldsymbol{c}_j$ is close to $\mathrm{span}(\mathsf{U}_k)$ and that M satisfies the restricted isometry property (discussed in the next subsection). In (5), $\gamma > 0$ is a regularisation parameter and g a positive non-decreasing polynomial function (see Section 2.1 for the definition of $g(\mathsf{L})$). This reconstruction scheme is proved to be robust to: 1) observation noise, *i.e.*, to imperfect clustering of the n nodes in our context; 2) model errors, *i.e.*, the indicator vectors do not need to be exactly in $\mathrm{span}(\mathsf{U}_k)$ for the method to work. Also, the performance is shown to depend on the ratio $g(\lambda_k)/g(\lambda_{k+1})$. The smaller it is, the better the reconstruction. To decrease this ratio, we decide to approximate the ideal high-pass filter $g_{\lambda_k}(\lambda) = 1 - h_{\lambda_k}(\lambda)$ for the reconstruction. Remark that this filter favors the recovery of signals living in $\mathrm{span}(\mathsf{U}_k)$. The approximation $\tilde{g}_{\lambda_k}$ of g_{λ_k} is obtained using a polynomial (as in Sec. 2.1), which permits us to find fast algorithms to solve (5).

3.2.3. How many features to sample?

We terminate this section by providing the theoretical number of features n one needs to sample in order to make sure that the indicator vectors can be faithfully recovered. This number is driven by the following quantity.

Definition 3.3 (Global cumulative coherence)**.** *The global cumulative coherence of order k of the graph $\mathcal{G}$ is* $\nu_k := \sqrt{N} \cdot \max_{1 \leqslant i \leqslant N} \{v_k(i)\}$.

It is shown in (Puy et al., 2015) that $\nu_k \in [k^{1/2}, N^{1/2}]$.

Theorem 3.4 ((Puy et al., 2015))**.** *Let M be a random sampling matrix constructed as in (4). For any $\delta, \epsilon \in]0, 1[$,*

$$(1 - \delta) \left\|\boldsymbol{x}\right\|_2^2 \leqslant \frac{N}{n} \left\|\mathsf{M}\,\boldsymbol{x}\right\|_2^2 \leqslant (1 + \delta) \left\|\boldsymbol{x}\right\|_2^2 \tag{6}$$

for all $\boldsymbol{x} \in \mathrm{span}(\mathsf{U}_k)$ with probability at least $1 - \epsilon$ provided that

$$n \geqslant \frac{6}{\delta^2}\, \nu_k^2\, \log\left(\frac{k}{\epsilon}\right).$$

[3] precise error bounds are provided in (Puy et al., 2015).

The above theorem presents a sufficient condition on n ensuring that M satisfies the restricted isometry property (6). This condition is required to ensure that the solution of (5) is an accurate estimation of $\boldsymbol{c}_j$. The above theorem thus indicates that sampling $O(\nu_k^2 \log k)$ features is sufficient to recover the cluster indicator vectors.

For a simple regular graph $\mathcal{G}$ made of k disconnected clusters, we have seen that $\mathsf{U}_k = (\boldsymbol{c}_1, \dots, \boldsymbol{c}_k)$ up to a normalisation of the vectors. Therefore, $\nu_k = N^{1/2} / \min_i\{N_i^{1/2}\}$, where N_i is the size of the i^{th} cluster. If the clusters have the same size $N_i = N/k$ then $\nu_k = k^{1/2}$, the lower bound on ν_k. In this simple optimal scenario, sampling $O(\nu_k^2 \log k) = O(k \log k)$ features is thus sufficient to recover the cluster indicator vectors.

The attentive reader will have noticed that for graphs where $\nu_k^2 \approx N$, no downsampling is possible. Yet, a simple solution exists in this situation: variable density sampling. Indeed, it is proved in (Puy et al., 2015) that, whatever the graph $\mathcal{G}$, there always exists an optimal sampling distribution such that $n = O(k \log k)$ samples are sufficient to satisfy Eq. (6). This distribution depends on the profile of the local cumulative coherence and can be estimated rapidly (see (Puy et al., 2015) for more details). In this paper, we only consider uniform sampling to simplify the explanations, but keep in mind that in practice results will always be improved if one uses variable density sampling. Note also that one cannot expect to sample less than k nodes to find k clusters. Up to the extra $\log(k)$, our result is optimal.

4. CSC in practice

We have detailed the two fundamental theoretical notions supporting our algorithm, presented in Alg. 2. However, some steps in Alg. 2 still need to be clarified. In particular, Sec. 4.2 provides an extension of Theorem 3.2 that takes into account the use of a non-ideal low-pass filter (to handle the practical case where the order of the polynomial approximation is finite). This theorem *in fine* explains and justifies step 4 of Alg. 2. Then, in Sec. 4.3, important details are discussed such as the estimation of λ_k (step 1) and the choice of the polynomial approximation (step 2). We finish this section with complexity considerations.

4.1. The CSC algorithm

As for SC (see Sec. 2.2), the algorithm starts with the adjacency matrix W of a graph $\mathcal{G}$. After computing its Laplacian L, the CSC algorithm is summarized in Alg. 2. The output $\tilde{\boldsymbol{c}}_j^*(i)$ is not binary and in fact quantifies how much node i belongs to cluster j, useful for fuzzy partitioning. To obtain an exact partition of the nodes, we normalize each indicator vector $\tilde{\boldsymbol{c}}_j^*$, and assign node i to the cluster j for which $\tilde{\boldsymbol{c}}_j^*(i) / \left\|\tilde{\boldsymbol{c}}_j^*\right\|$ is maximal.

Algorithm 2 Compressive Spectral Clustering

Input: The Laplacian matrix L, the number of clusters k; and parameters typically set to $n = 2k \log k$, $d = 4 \log n$, $p = 50$ and $\gamma = 10^{-3}$.

1· Estimate L's k-th eigenvalue λ_k as in Sec. 4.3.
2· Compute the polynomial approximation $\tilde{h}_{\lambda_k}$ of order p of the ideal low-pass filter h_{λ_k}.
3· Generate d random Gaussian signals of mean 0 and variance $1/d$: $\mathsf{R} = (\boldsymbol{r}_1 | \boldsymbol{r}_2 | \cdots | \boldsymbol{r}_d) \in \mathbb{R}^{N \times d}$.
4· Filter R with $\tilde{\mathsf{H}}_{\lambda_k} = \tilde{h}_{\lambda_k}(\mathsf{L})$ as in Sec. 2.1 and define, for each node i, its feature vector $\tilde{\boldsymbol{f}}_i \in \mathbb{R}^d$:

$$\tilde{\boldsymbol{f}}_i = \left[\left(\tilde{\mathsf{H}}_{\lambda_k}\mathsf{R}\right)^\mathsf{T} \boldsymbol{\delta}_i\right] \Big/ \left\|\left(\tilde{\mathsf{H}}_{\lambda_k}\mathsf{R}\right)^\mathsf{T} \boldsymbol{\delta}_i\right\|.$$

5· Generate a random sampling matrix $\mathsf{M} \in \mathbb{R}^{n \times N}$ as in Eq. (4) and keep only n feature vectors: $(\tilde{\boldsymbol{f}}_{\omega_1} | \dots | \tilde{\boldsymbol{f}}_{\omega_n})^\mathsf{T} = \mathsf{M}(\tilde{\boldsymbol{f}}_1 | \dots | \tilde{\boldsymbol{f}}_N)^\mathsf{T}$.
6· Run k-means on the reduced dataset with the Euclidean distance: $\tilde{D}_{ij}^r = \left\|\tilde{\boldsymbol{f}}_{\omega_i} - \tilde{\boldsymbol{f}}_{\omega_j}\right\|$ to obtain k reduced indicator vectors $\boldsymbol{c}_j^r \in \mathbb{R}^n$, one for each cluster.
7· Interpolate each reduced indicator vector $\boldsymbol{c}_j^r$ with the optimisation problem of Eq. (5), to obtain the k indicator vectors $\tilde{\boldsymbol{c}_j}^* \in \mathbb{R}^N$ on the full set of nodes.

4.2. Non-ideal filtering of random signals

In this section, we improve Theorem 3.2 by studying how the error of the polynomial approximation $\tilde{h}_{\lambda_k}$ of h_{λ_k} propagates to the spectral distance estimation, and by taking into account the fact that k-means is performed on the reduced set of features $(\tilde{\boldsymbol{f}}_{\omega_1} | \dots | \tilde{\boldsymbol{f}}_{\omega_n})^\mathsf{T} = \mathsf{M}(\tilde{\boldsymbol{f}}_1 | \dots | \tilde{\boldsymbol{f}}_N)^\mathsf{T}$. We denote by $\mathsf{MY}_k \in \mathbb{R}^{n \times k}$ the ideal reduced feature matrix. We have $(\boldsymbol{f}_{\omega_1} | \cdots | \boldsymbol{f}_{\omega_n})^\mathsf{T} = \mathsf{M}(\boldsymbol{f}_1 | \cdots | \boldsymbol{f}_N)^\mathsf{T} = \mathsf{MY}_k$. The actual distances we want to estimate using random signals are thus, for all $(i,j) \in \{1, \dots, n\}^2$

$$D_{ij}^r := \left\|\boldsymbol{f}_{\omega_i} - \boldsymbol{f}_{\omega_j}\right\| = \left\|\mathsf{Y}_k^\mathsf{T}\mathsf{M}^\mathsf{T}(\boldsymbol{\delta}_i^r - \boldsymbol{\delta}_j^r)\right\|,$$

where the $\{\boldsymbol{\delta}_i^r\}$ are here Diracs in n dimensions.

Consider the random matrix $\mathsf{R} = (\boldsymbol{r}_1 | \boldsymbol{r}_2 | \cdots | \boldsymbol{r}_d) \in \mathbb{R}^{N \times d}$ constructed as in Sec. 3.1. Its filtered, normalized and reduced version is $\mathsf{MV}_k\tilde{\mathsf{H}}_{\lambda_k}\mathsf{R} \in \mathbb{R}^{n \times d}$. The new feature vector $\tilde{\boldsymbol{f}}_{\omega_i} \in \mathbb{R}^d$ associated to node ω_i is thus

$$\tilde{\boldsymbol{f}}_{\omega_i} = (\mathsf{MV}_k\tilde{\mathsf{H}}_{\lambda_k}\mathsf{R})^\mathsf{T}\boldsymbol{\delta}_i^r.$$

The normalisation of Step 4 in Alg. 2 approximates the action of V_k in the above equation. More details and justifications are provided in the "Important remark" at the end of this section. The distance between any two features reads

$$\tilde{D}_{ij}^r := \left\|\tilde{\boldsymbol{f}}_{\omega_i} - \tilde{\boldsymbol{f}}_{\omega_j}\right\| = \left\|\mathsf{R}^\mathsf{T}\tilde{\mathsf{H}}_{\lambda_k}^\mathsf{T}\mathsf{V}_k^\mathsf{T}\mathsf{M}^\mathsf{T}(\boldsymbol{\delta}_i^r - \boldsymbol{\delta}_j^r)\right\|.$$

We now study how well $\tilde{D}_{ij}^r$ estimates D_{ij}^r.

Approximation error. Denote $e(\lambda)$ the approximation error of the ideal low-pass filter:

$$\forall \lambda \in [0,2], \qquad e(\lambda) := \tilde{h}_{\lambda_k}(\lambda) - h_{\lambda_k}(\lambda).$$

In the form of graph filter operators, one has

$$\tilde{h}_{\lambda_k}(\mathsf{L}) = \tilde{\mathsf{H}}_{\lambda_k} = \mathsf{H}_{\lambda_k} + \mathsf{E} = h_{\lambda_k}(\mathsf{L}) + e(\mathsf{L}).$$

We model the error e using two parameters: e_1 (resp. e_2) the maximal error for $\lambda \leqslant \lambda_k$ (resp. $\lambda > \lambda_k$). We have

$$e_1 := \sup_{\lambda \in \{\lambda_1, \dots, \lambda_k\}} |e(\lambda)|, \qquad e_2 := \sup_{\lambda \in \{\lambda_{k+1}, \dots, \lambda_N\}} |e(\lambda)|.$$

The resolution parameter. In some cases, the ideal reduced spectral distance D^r_{ij} may be null. In such cases, approximating $D^r_{ij} = 0$ using a non-ideal filter is not possible. In fact, non-ideal filtering introduces an irreducible error on the estimation of the feature vectors that is not possible to compensate in general. We thus introduce a resolution parameter D^r_{min} below which the distances D^r_{ij} do not need to be approximated exactly, but should remain below D^r_{min} (up to a tolerated error).

Theorem 4.1 (General norm conservation theorem)**.** *Let $D^r_{min} \in \left]0, \sqrt{2}\right]$ be a chosen resolution parameter. For any $\delta \in]0,1]$, $\beta > 0$, if d is larger than*

$$\frac{16(2+\beta)}{\delta^2 - \delta^3/3} \log n,$$

then, for all $(i,j) \in \{1, \dots, n\}^2$,

$$(1-\delta) D^r_{ij} \leqslant \tilde{D}^r_{ij} \leqslant (1+\delta) D^r_{ij}, \;\; \textit{if} \;\; D^r_{ij} \geqslant D^r_{min},$$

and

$$\tilde{D}^r_{ij} < (1+\delta) D^r_{min}, \;\; \textit{if} \;\; D^r_{ij} < D^r_{min},$$

with probability at least $1 - 2n^{-\beta}$ provided that

$$\sqrt{|e_1^2 - e_2^2|} + \frac{\sqrt{2}\, e_2}{D^r_{min} \min_i\{v_k(i)\}} \leqslant \frac{\delta}{2+\delta}. \qquad (7)$$

The proof is provided in the supplementary material.

Consequence of Theorem 4.1. All distances smaller (resp. larger) than the chosen resolution parameter D^r_{min} are estimated smaller than $(1+\delta)D^r_{min}$ (resp. correctly estimated up to a relative error δ). Moreover, for a fixed distance estimation error δ, the lower we decide to fix D^r_{min}, the lower should also be the errors e_1 and/or e_2 to ensure that Eq. (7) still holds, which implies an increase of the order p of the polynomial approximation of the ideal filter h_{λ_k}, and ultimately, that means a higher computation time for the filtering operation of the random signals.

Important remark. The feature matrix $\mathsf{V}_k \tilde{\mathsf{H}}_{\lambda_k} \mathsf{R}$ can be easily computed if one knows the cut-off value λ_k and the local cumulative coherences $v_k(i)$. Unfortunately, this is not the case in practice. We propose a solution to estimate λ_k in Sec. 4.3. To estimate $v_k(i)$, one can use the results in Sec. 4 of (Puy et al., 2015) showing that $v_k(i) = \|\mathsf{U}_k^\intercal \boldsymbol{\delta}_i\| \approx \|(\mathsf{H}_{\lambda_k}\mathsf{R})^\intercal \boldsymbol{\delta}_i\|$. Thus, a practical way to estimate $\mathsf{V}_k \tilde{\mathsf{H}}_{\lambda_k} \mathsf{R}$ is to first compute $\tilde{\mathsf{H}}_{\lambda_k}\mathsf{R}$ and then normalize its rows to unit length, as done in Step 4 of Alg. 2.

4.3. Polynomial approximation and estimation of λ_k

The polynomial approximation. Theorem 4.1 uses a separate control on $e(\lambda)$ below λ_k (with e_1) and above λ_k (with e_2). To have such a control in practice, one would need to use rational filters (ratio of two polynomials) to approximate h_{λ_k}. Such filters have been introduced in the graph context (Shi et al., 2015), but they involve another optimisation step that would burden our main message. We prefer to simplify our analysis by using polynomials for which only the maximal error can be controlled. We write

$$e_m := \max(e_1, e_2) = \sup_{\lambda \in \{\lambda_1, \dots, \lambda_N\}} |e(\lambda)|. \qquad (8)$$

In this easier case, one can show that Theorem 4.1 is still valid if Eq. (7) is replaced by

$$\frac{\sqrt{2}\, e_m}{D^r_{min} \min_i\{v_k(i)\}} \leqslant \frac{\delta}{2+\delta}. \qquad (9)$$

In our experiments, we could follow (Shuman et al., 2011) and use truncated Chebychev polynomials to approximate the ideal filter, as these polynomials are known to require a small degree to ensure a given tolerated maximal error e_m. We prefer to follow (Napoli et al., 2013) who suggest to use Jackson-Chebychev polynomials: Chebychev polynomials to which are added damping multipliers to alleviate the unwanted Gibbs oscillations around the cut-off frequency λ_k.

The polynomial's order p. For a fixed δ, D^r_{min}, and $\min_i\{v_k(i)\}$, one should use the Jackson-Chebychev polynomial of smallest order p^* ensuring that e_m satisfies Eq. (9), in order to optimize the computation time while making sure that Theorem 4.1 applies. Studying p^* theoretically without computing the Laplacian's complete spectrum (see Eq. (8)) is beyond the scope of this paper. Experimentally, $p = 50$ yields good results (see Fig. 1c).

Estimation of λ_k. The fast filtering step is based on the polynomial approximation of h_{λ_k}, which is itself parametrized by λ_k. Unless we compute the first k eigenvectors of L, thereby partly loosing our efficiency edge on other methods, we cannot know the value of λ_k with infinite precision. To estimate it efficiently, we use eigencount techniques (Napoli et al., 2013): based on low-pass filtering with a cut-off frequency at λ of random signals, one obtains an estimation of the number of enclosed eigenvalues in the interval $[0, \lambda]$. Starting with $\lambda = 2$ and proceeding

by dichotomy on λ, one stops the algorithm as soon as the number of enclosed eigenvalues equals k. For each value of λ, in order to have a proper estimation of the number of enclosed eigenvalues, we choose to filter $2 \log N$ random signals with Jackson-Chebychev polynomial approximation of the ideal low-pass filters.

4.4. Complexity considerations

The complexity of steps 2, 3 and 5 of Alg. 2 are not detailed as they are insignificant compared to the others. First, note that fast filtering a graph signal costs $O(p \, \#\mathcal{E})$.[4] Therefore, Step 1 costs $O(p \, \#\mathcal{E} \log N)$ per iteration of the dichotomy, and Step 4 costs $O(p \, \#\mathcal{E} \log n)$ (as $d = O(\log n)$). Step 7 requires to solve Eq. (5) with the polynomial approximation of $g_{\lambda_k}(\lambda) = 1 - h_{\lambda_k}(\lambda)$. When solved, *e.g.*, by conjugate gradient or gradient descent, this step costs a fast filtering operation per iteration of the solver and for each of the k classes. Step 7 thus costs $O(p \, \#\mathcal{E} k)$. Also, the complexity of k-means to cluster Q feature vectors of dimension r into k classes is $O(kQr)$ per iteration. Therefore, Step 6 with $Q = n$ and $r = d = O(\log(n))$ costs $O(kn \log n)$. CSC's complexity is thus $O\left(kn \log n + p \, \#\mathcal{E} \left(\log N + \log n + k\right)\right)$. In practice, we are interested in sparse graphs: $\#\mathcal{E} = O(N)$. Using the fact that $n = O(k \log k)$, CSC's complexity simplifies to

$$O\left(k^2 \log^2 k + pN \left(\log N + k\right)\right).$$

SC's k-means step has a complexity of $O(Nk^2)$ per iteration. In many cases[5] this sole task is more expensive than the CSC algorithm. On top of this, SC has the additional complexity of computing the first k eigenvectors of L, for which the cost of ARPACK - a popular eigenvalue solver - is $O(k^3 + Nk^2)$ (see, *e.g.*, Sec. 3.2 of (Chen et al., 2011)).

This study suggests that CSC is faster than SC for large N and/or k. The above algorithms' number of iterations are not taken into account as they are difficult to predict theoretically. Yet, the following experiments confirm the superiority of CSC over SC in terms of computational time.

5. Experiments

We first perform well-controlled experiments on the Stochastic Block Model (SBM), a model of random graphs with community structure, that was showed suitable as a benchmark for SC in (Lei & Rinaldo, 2015). We also show performance results on a large real-world network. Implementation was done in Matlab R2015a, using the built-in function `kmeans` with 20 replicates, and the function `eigs` for SC. Experiments were done on a laptop with a 2.60 GHz Intel i7 dual-core processor running OS Fedora release 22 with 16 GB of RAM. The fast filtering part of CSC uses the `gsp_cheby_op` function of the `GSP` toolbox (Perraudin et al., 2014). Equation (5) is solved using Matlab's `gmres` function. All our results are reproducible with the CSCbox downloadable at `http://cscbox.gforge.inria.fr/`.

[4] Recall that p is the order of the polynomial filter.

[5] Roughly, all cases for which $k^2 > p(\log N + k)$.

5.1. The Stochastic Block Model

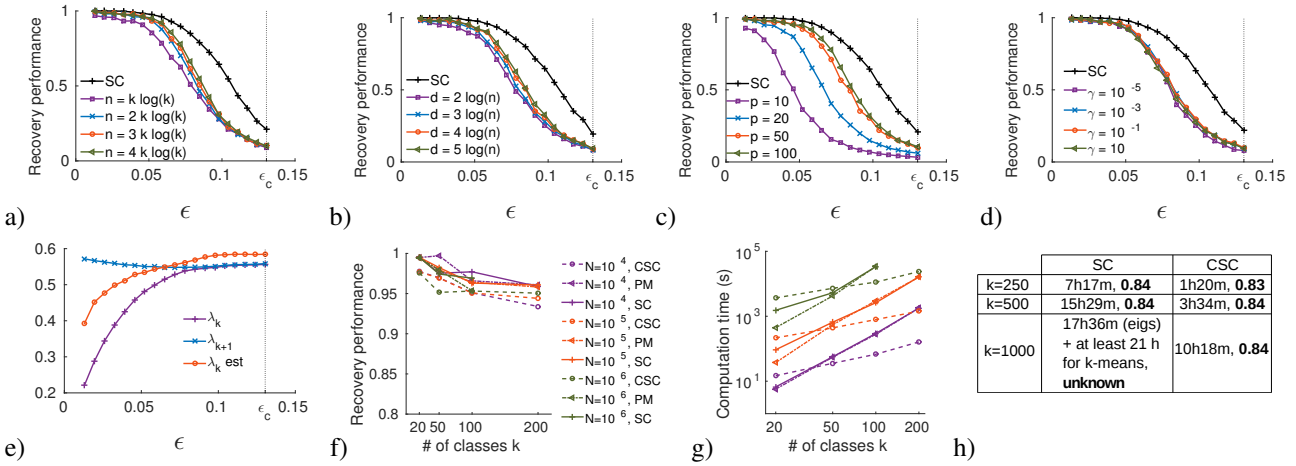
What distinguishes the SBM from Erdos-Renyi graphs is that the probability of connection between two nodes i and j is not uniform, but depends on the community label of i and j. More precisely, the probability of connection between nodes i and j equals q_1 if they are in the same community, and q_2 if not. In a first approach, we look at graphs with k communities, all of same size N/k. Furthermore, instead of considering the probabilities, one may fully characterize a SBM by providing their ratio $\epsilon = \frac{q_2}{q_1}$, as well as the average degree s of the graph. The larger ϵ, the more difficult the community structure's detection. In fact, Decelle et al. (2011) show that a critical value ϵ_c exists above which community detection is impossible at the large N limit: $\epsilon_c = (s - \sqrt{s})/(s + \sqrt{s}(k-1))$.

5.2. Performance results

In Figs. 1 a-d), we compare the recovery performance of CSC versus SC for different parameters. The performance is measured by the Adjusted Rand similarity index (Hubert & Arabie, 1985) between the SBM's ground truth and the obtained partitions. It varies between -1 and 1. The higher it is, the better is the reconstruction. These figures show that the performance of CSC saturates at the default values of n, d, p and γ (see top of Alg. 2). Experiments on the SBM with heterogeneous community sizes are provided in the supplementary material and show similar results.

Fig. 1 e) shows the estimation results of λ_k for different values of ϵ : it is overestimated in the SBM context. As long as the estimated value stays under λ_{k+1}, this overestimation does not have a strong impact on the method. On the other hand, as ϵ becomes larger than ~ 0.06, our estimation of λ_k is larger than λ_{k+1}, which means that our feature vectors start to integrate some unwanted information from eigenvectors outside of $\mathrm{span}(\mathsf{U}_k)$. Even though the impact of this additional information is application-dependent and in some cases insignificant, further efforts to improve the estimation of λ_k would be beneficial to our method.

In Figs. 1 f-g) we fix ϵ to $\epsilon_c/4$, n, d, p and γ to the values given in Alg. 2, and vary N and k. We compare the recovery performance and the time of computation of CSC, SC and Boutsidis' power method (Boutsidis & Gittens, 2015). The power method (PM), in a nutshell, 1) applies the Laplacian matrix to the power r to k random signals, 2) computes the left singular vectors of the $N \times k$ obtained matrix, to ex-



h)

	SC	CSC
k=250	7h17m, **0.84**	1h20m, **0.83**
k=500	15h29m, **0.84**	3h34m, **0.84**
k=1000	17h36m (eigs) + at least 21 h for k-means, **unknown**	10h18m, **0.84**

Figure 1. (a-d): recovery performance of CSC on a SBM with $N = 10^3, k = 20, s = 16$ versus ϵ, for different n, d, p, γ. Default is $n = 2k \log k$, $d = 4 \log n$, $p = 50$ and $\gamma = 10^{-3}$. All results are averaged over 20 graph realisations. e) Estimation of λ_k ('λ_k est') and the true values of λ_k and λ_{k+1}, versus ϵ on the same SBM. f-g) Performance and time of computation on a SBM with $\epsilon = \epsilon_c/4$ and different values of N and k; for CSC, PM (Power Method) and SC. For $N = 10^6$ and $k = 200$, we stopped SC (and PM) after 20h of computation. Figures f and g are averaged over 3 graph realisations. N.B.: Fig. f is zoomed around high values of the recovery score, and Fig. g is plotted in log-log. h) Time of computation (in hours) and modularity (in bold) of the obtained partitions for SC and CSC on the Amazon graph. For $k = 1000$, SC's eigendecomposition converges in 17h, and we stopped k-means after 21 hours of computation.

tract feature vectors, 3) applies k-means in high-dimension (like SC) with these feature vectors. In our experiments, we use $r = 10$. The recovery performances are nearly identical in all situations, even though CSC is only a few percents under SC and PM (Fig. f is zoomed around the high values of the recovery score). For the time of computation, the experiments confirm that all three methods are roughly linear in N and polynomial in k (Fig. g is plotted in log-log), with a lower exponent for CSC than for SC and PM; such that SC and PM are faster for $k = 20$ but CSC becomes up to an order of magnitude faster as k increases to 200. Note that the SBM is favorable to SC as Matlab's function `eigs` converges very fast in this case, *e.g.*, for $N = 10^5$, it finds the first $k = 200$ eigenvectors in less than 2 minutes! PM sidesteps successfully the cost of `eigs`, but the cost of k-means in high-dimension is still a strong bottleneck.

We finally compare CSC and SC on a real-world dataset: the Amazon co-purchasing network (Yang & Leskovec, 2015). It is an undirected connected graph comprising $N = 334\,863$ nodes and $\#\mathcal{E} = 925\,872$ edges. The results are presented in Fig.1 h) for three values of k. As there is no clear ground truth in this case, we use the modularity (Newman & Girvan, 2004) to measure the algorithm's clustering performance, a well-known cost function that measures how well a given partition separates a network in different communities. Note that the 20 replicates of k-means would not converge for SC with the default maximum number of iterations set to 100. For a fair comparison with CSC, we used only 2 replicates with a maximum number of iterations set to 1000 for SC's k-means step. We see that for the same clustering performance, CSC is much faster than SC, especially as k increases. The PM algorithm on this dataset does not perform well: even though the features are estimated quickly, they apparently do not form clear classes such that its k-means step takes even longer than SC's. For the three values of k, we stopped the PM algorithm after a time of computation exceeding SC's.

6. Conclusion

By graph filtering $O(\log k)$ random signals, we construct feature vectors whose interdistances approach the standard SC feature distances. Then, building upon compressive sensing results, we show that one can sample $O(k \log k)$ nodes from the set of N nodes, cluster this reduced set of nodes and interpolate the result back to the whole graph. If the low-dimensional k-means result is correct, *i.e.*, if Eq. (3) is verified, we guarantee that the interpolation is a good approximation of the SC result. To improve the clustering result of the reduced set of nodes, one could consider the concept of community cores (Seifi et al., 2013). In fact, as the filtering and the low-dimensional clustering steps are fairly cheap to compute, one could repeat these steps for different random signals, keep the sets of nodes that are always classified together and use only these stable "cores" for interpolation. Our experiments show that even without such potential improvements, CSC proves efficient and accurate in synthetic and real-world datasets; and could be preferred to SC for large N and/or k.

Acknowledgements

This article was submitted when G. Puy was with INRIA Rennes - Bretagne Atlantique, France. This work was partly funded by the European Research Council, PLEASE project (ERC-StG-2011-277906), and by the Swiss National Science Foundation, grant 200021-154350/1 - Towards Signal Processing on Graphs.

References

Achlioptas, D. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671 – 687, 2003.

Anis, A., Gadde, A., and Ortega, A. Efficient sampling set selection for bandlimited graph signals using graph spectral proxies. *arXiv*, abs/1510.00297, 2015.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Boutsidis, C., Zouzias, A., Mahoney, M. W, and Drineas, P. Stochastic dimensionality reduction for k-means clustering. *arXiv, abs/1110.2897*, 2011.

Boutsidis, C., Kambadur P. and Gittens, A. Spectral clustering via the power method - provably. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15), Lille, France*, pp. 40–48, 2015.

Chen, S., Varma, R., Sandryhaila, A., and Kovacevic, J. Discrete signal processing on graphs: Sampling theory. *Signal Processing, IEEE Transactions on*, 63(24):6510–6523, 2015.

Chen, W.-Y., Song, Y., Bai, H., C.-J, Lin., and Chang, E.Y. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586, 2011.

Chen, X. and Cai, D. Large scale spectral clustering with landmark-based representation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011.

Chung, F.R.K. *Spectral graph theory*. Number 92. Amer Mathematical Society, 1997.

Cohen, M. B., Elder, S., Musco, C., Musco, C., and Persu, M. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing*, pp. 163–172. ACM, 2015.

Decelle, A., Krzakala, F., Moore, C, and Zdeborová, L. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Phys. Rev. E*, 84:066106, 2011.

Dhillon, I.S., Guan, Y., and Kulis, B. Weighted graph cuts without eigenvectors a multilevel approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(11):1944–1957, 2007.

Fiedler, M. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

Filippone, M., Camastra, F., Masulli, F., and Rovetta, S. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176 – 190, 2008.

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

Har-Peled, Sariel and Mazumdar, Soham. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 291–300. ACM, 2004.

Hubert, L. and Arabie, P. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Jain, Anil K. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666, 2010.

Lei, J. and Rinaldo, A. Consistency of spectral clustering in stochastic block models. *Ann. Statist.*, 43(1):215–237, 2015.

Lin, F. and Cohen, W. W. Power iteration clustering. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel*, pp. 655–662, 2010.

Liu, T.-Y., Yang, H.-Y., Zheng, X., Qin, T., and Ma, W.-Y. Fast large-scale spectral clustering by sequential shrinkage optimization. In *Advances in Information Retrieval*, pp. 319–330. 2007.

Marques, A., Segarra, S., Leus, G., and Ribeiro, A. Sampling of graph signals with successive local aggregations. *Signal Processing, IEEE Transactions on*, PP(99): 1–1, 2015.

Napoli, Edoardo Di, Polizzi, Eric, and Saad, Yousef. Efficient estimation of eigenvalue counts in an interval. *arXiv*, abs/1308.4275, 2013.

Nascimento, M.C.V. and de Carvalho, A.C.P.L.F. Spectral methods for graph clustering a survey. *European Journal of Operational Research*, 211(2):221 – 231, 2011.

Newman, M. E. J. and Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69: 026113, 2004.

Ng, A.Y., Jordan, M.I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In Dietterich, T.G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*, pp. 849–856. MIT Press, 2002.

Perraudin, N., Paratte, J., Shuman, D., Kalofolias, V., Vandergheynst, P., and Hammond, D.K. Gspbox: A toolbox for signal processing on graphs. *arXiv*, abs/1408.5781, 2014.

Puy, G., Tremblay, N., Gribonval, R., and Vandergheynst, P. Random sampling of bandlimited signals on graphs. *arXiv*, abs/1511.05118, 2015.

Ramasamy, D. and Madhow, U. Compressive spectral embedding: sidestepping the SVD. In *Advances in Neural Information Processing Systems 28*, pp. 550–558. 2015.

Sakai, Tomoya and Imiya, Atsushi. Fast spectral clustering with random projection and sampling. In *Machine Learning and Data Mining in Pattern Recognition*, pp. 372–384. 2009.

Sandryhaila, A. and Moura, J.M.F. Discrete signal processing on graphs. *Signal Processing, IEEE Transactions on*, 61(7):1644–1656, 2013.

Sandryhaila, A. and Moura, J.M.F. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *Signal Processing Magazine, IEEE*, 31(5):80–90, 2014.

Seifi, M., Junier, I., Rouquier, J.-B., Iskrov, S., and Guillaume, J.-L. Stable community cores in complex networks. In *Complex Networks*, pp. 87–98. 2013.

Shi, J. and Malik, J. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

Shi, X., Feng, H., Zhai, M., Yang, T., and Hu, B. Infinite impulse response graph filters in wireless sensor networks. *Signal Processing Letters, IEEE*, 22(8):1113–1117, 2015.

Shuman, D.I., Vandergheynst, P., and Frossard, P. Chebyshev polynomial approximation for distributed signal processing. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), International Conference on*, pp. 1–8, 2011.

Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *Signal Processing Magazine, IEEE*, 30(3):83–98, 2013.

Tremblay, N., Puy, G., Borgnat, P., Gribonval, R., and Vandergheynst, P. Accelerated spectral clustering using graph filtering of random signals. In *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*, 2016. accepted.

Tsitsvero, M., Barbarossa, S., and Lorenzo, P. Di. Signals on graphs: Uncertainty principle and sampling. *arXiv*, abs/1507.08822, 2015.

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

Wang, L., Leckie, C., Ramamohanarao, K., and Bezdek, J. Approximate spectral clustering. In *Advances in Knowledge Discovery and Data Mining*, pp. 134–146. 2009.

White, S. and Smyth, P. A spectral clustering approach to finding communities in graph. In *SDM*, volume 5, pp. 76–84. SIAM, 2005.

Yan, D., Huang, L., and Jordan, M.I. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pp. 907–916, New York, NY, USA, 2009.

Yang, J. and Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

Zelnik-Manor, L. and Perona, P. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pp. 1601–1608, 2004.