Approximating Spectral Clustering via Sampling: a Review

Nicolas Tremblay, Andreas Loukas

Approximating Spectral Clustering via Sampling: a Review

Nicolas Tremblay and Andreas Loukas

Abstract Spectral clustering refers to a family of well-known unsupervised learning algorithms. Rather than attempting to cluster points in their native domain, one constructs a (usually sparse) similarity graph and computes the principal eigenvectors of its Laplacian. The eigenvectors are then interpreted as transformed points and fed into a k-means clustering algorithm. As a result of this non-linear transformation, it becomes possible to use a simple centroid-based algorithm in order to identify non-convex clusters, something that was otherwise impossible. Unfortunately, what makes spectral clustering so successful is also its Achilles heel: forming a graph and computing its dominant eigenvectors can be computationally prohibitive when dealing with more that a few tens of thousands of points. In this chapter, we review the principal research efforts aiming to reduce this computational cost. We focus on methods that come with a theoretical control on the clustering performance and incorporate some form of sampling in their operation. Such methods abound in the machine learning, numerical linear algebra, and graph signal processing literature and, amongst others, include Nyström-approximation, landmarks, coarsening, coresets, and compressive spectral clustering. We present the approximation guarantees available for each and discuss practical merits and limitations. Surprisingly, despite the breadth of the literature explored, we conclude that there is still a gap between theory and practice: the most scalable methods are only intuitively motivated or loosely controlled, whereas those that come with end-to-end guarantees rely on strong assumptions or enable a limited gain of computation time.

Nicolas Tremblay
CNRS, Univ. Grenoble Alpes, GIPSA-lab, France, e-mail: nicolas.tremblay@grenoble-inp.fr

Andreas Loukas
Ecole Polytechnique Fédérale de Lausanne, Switzerland, e-mail: andreas.loukas@epfl.ch

1 Introduction

Clustering is a cornerstone of our learning process and, thus, of our understanding of the world. Indeed, we can all distinguish between a rose and a tulip precisely because we have learned what these flowers *are*. Plato would say that we learned the Idea –or Form [120]– of both the rose and the tulip, which then enables us to recognize all instances of such flowers. A machine learner would say that we learned two *classes*: their most discriminating features (shape, size, number of petals, smell, etc.) as well as their possible intra-class variability.

Mathematically speaking, the first step on the road to classifying objects (such as flowers) is to create an abstract representation of these objects: with each object i we associate a feature vector $\mathbf{p}_i \in \mathbb{R}^d$, where the dimension d of the vector corresponds to the number of features one chooses to select for the classification task. The space $\mathbb{R}^d$ in this context is sometimes called the *feature space*. The choice of representation will obviously have a strong impact on the subsequent classification performance. Say that in the flower example we choose to represent each flower by only $d = 3$ features: the average color of each RGB channel (level of red, green and blue) of its petals. This choice is not fail-proof: even though the archetype of the rose is red and the archetype of the tulip is yellow, we know that some varieties of both flowers can have very similar colors and thus a classification solely based on the color will necessarily lead to confusion. In fact, there are many different ways of choosing features: from features based on the expert knowledge of a botanist, to features learned by a deep learning architecture from many instances of labeled images of roses and tulips, via features obtained by hybrid methods more-or-less based on human intelligence (such as the first few components of a Principal Component Analysis of expert-based features).

The second step on the road to classifying n objects is to choose a machine learning algorithm that groups the set of n points $\mathsf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ in k classes (k may be known in advance or determined by the algorithm itself). Choosing an appropriate algorithm depends on the context:

- **Availability of pre-labeled data.** Classifying the points P in k classes may be seen as assigning a label (such as “rose” or ”tulip” in our $k = 2$ example) to each of the points. If one has access to some pre-labeled data, we are in the case of *supervised learning*: a more-or-less parametrized model is first learned from the pre-labeled data and then applied to the unlabeled points that need classification. If one does not have access to any pre-labeled data, we are in the case of *unsupervised learning* where classes are typically inferred only via geometrical consideration of the distribution of points in the feature space. If one has only access to a few labeled data, we are in the in-between case of *semi-supervised learning* where the known labels are typically propagated in one form or another in the feature space.
- **Inductive vs transductive learning.** Another important characteristic of a classification algorithm is whether it can be used to classify only the set of points P

at hand (transductive), or if it can also be directly used to classify any never-seen data point $\mathbf{p}_{n+1}$ (inductive).

This chapter focuses on the family of algorithms jointly referred to as *spectral clustering*. These algorithms are unsupervised and transductive: no label is known in advance and one may not naturally[1] extend the results obtained on P to never-seen data points. Another particularity of spectral clustering algorithms is that the number of classes k is known in advance.

Spectral clustering algorithms have received a large attention in the last two decades due to their good performance on a wide range of different datasets, as well as their ease of implementation. In a nutshell, they combine three steps:

1. **Graph construction.** A sparse similarity graph is built between the n points.
2. **Spectral embedding.** The first k eigenvectors of a graph representative matrix (such as the Laplacian) are computed.
3. **Clustering.** k-means is performed on these spectral features, to obtain k clusters.

For background information about spectral clustering, such as several justifications of its performance, out-of-sample extensions, as well as comparisons with local methods, the interested reader is referred to the recent book chapter [144].

One of the drawbacks of spectral clustering is its computational cost as n, d, and/or k become large (see Sec. 2.3 for a discussion on the cost). Since the turn of the century, a large number of authors have striven to reduce the computational cost while keeping the high level of classification performance. The majority of such accelerating methods are based on sampling: they reduce the dimension of a subproblem of spectral clustering, compute a low-cost solution in small dimension, and lift the result back to the original space.

The goal of this chapter is to review existing sampling methods for spectral clustering, focusing especially on their approximation guarantees. Some of the fundamental questions we are interested in are: *where is the sampling performed and what is sampled precisely? how should the reduced approximate solutions be lifted back to the original space? what is the computational gain? what is the control on performances—if it exists?* Given the breadth of the literature on the subject, we do not try to be exhaustive, but rather to illustrate the key ways that sampling can be used to provide acceleration, paying special attention on recent developments on the subject.

Chapter organization. We begin by recalling in Sec. 2 the prototypical spectral clustering algorithm. We also provide some intuitive and formal justification of why it works. The next three sections classify the different methods of the literature depending on where the sampling is performed with respect to the three steps of spectral clustering:

- Sec. 3 details methods that sample directly in the original feature space.

[1] Out-of-sample extensions of spectral clustering do exist (see for instance Section 5.3.6 of [144]), but they require additional work.

- Sec. 4 assumes that the similarity graph is given and details methods that sample nodes and/or edges to approximate the spectral embedding.
- Sec. 5 assumes that the spectral embedding is given and details methods to accelerate the k-means step.

Finally, Sec. 6 gives perspective on the limitations of existing works and discusses key open problems.

Notation. Scalars, such as λ or d, are written with low-case letters. Vectors, such as $\mathbf{u}$, $\mathbf{z}$ or the all-one vector $\mathbf{1}$, are denoted by low-case bold letters. Matrices, such as $\mathbf{W}$ or $\mathbf{L}$ are denoted with bold capital letters. Ensembles are denoted by serif font capital letters, such as C or X. The "tilde" will denote approximations, such as in $\tilde{\mathbf{z}}$ or $\tilde{\mathbf{U}}_k$. We use so-called Matlab notations to slice matrices: given a set of indices S of size m and an $n \times n$ matrix $\mathbf{W}$, $\mathbf{W}(\mathsf{S},:) \in \mathbb{R}^{m \times n}$ is $\mathbf{W}$ reduced to the lines indexed by S, $\mathbf{W}(:,\mathsf{S}) \in \mathbb{R}^{n \times m}$ is $\mathbf{W}$ reduced to the columns indexed by S, and $\mathbf{W}(\mathsf{S},\mathsf{S}) \in \mathbb{R}^{m \times m}$ is $\mathbf{W}$ reduced to the lines and columns indexed by S. The equation $\mathbf{U}_k = \mathbf{U}(:,:k)$ defines $\mathbf{U}_k$ as the reduction of $\mathbf{U}$ to its first k columns. Also, $\mathbf{C}^\top$ is the transpose of matrix $\mathbf{C}$ and $\mathbf{C}^+$ its Moore-Penrose pseudo-inverse. The operator $\mathbf{X} = \mathrm{diag}(\mathbf{x})$ takes as an input a vector $\mathbf{x} \in \mathbb{R}^n$ and returns an $n \times n$ diagonal matrix $\mathbf{X}$ featuring $\mathbf{x}$ in its main diagonal, i.e., $\mathbf{X}(i,j) = \mathbf{x}(i)$ if $i = j$ and $\mathbf{X}(i,j) = 0$, otherwise. Finally, we will consider graphs in a large part of this paper. We will denote by $G = (\mathsf{V}, \mathsf{E}, \mathbf{W})$ the undirected weighted graph of $|\mathsf{V}| = n$ nodes interconnected by $|\mathsf{E}| = e$ edges: $e_{ij} \in \mathsf{E}$ is the edge connecting nodes v_i and v_j, with weight $\mathbf{W}(i,j) \geq 0$. Matrix $\mathbf{W}$ is the adjacency matrix of G. As G is considered undirected, $\mathbf{W}$ is also symmetric. In general, $\mathbf{W}$ can be any symmetric matrix with positive entries, but we usually prefer to work with sparse graphs without self-loops, in which case the matrix is also sparse and has a zero diagonal.

2 Spectral clustering

The input of spectral clustering algorithms consists of (i) a set of points $\mathsf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ (also called featured vectors) representing n objects in a feature space of dimension d, and (ii) the number of classes k in which to classify these objects. The output is a partition of the n objects in k disjoint clusters. The prototypical spectral clustering algorithm [121, 102], dates back in fact to fundamental ideas by Fiedler [46] and entails the following steps:

Algorithm 1. The prototypical Spectral Clustering algorithm
Input. A set of n points $\mathsf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ in dimension d and a number of desired clusters k.

1. Graph construction (optional)
   a. Compute the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$: $\forall (i,j)$, $\mathbf{K}(i,j) = \kappa(\|\mathbf{p}_i - \mathbf{p}_j\|_2)$.

b. Compute $\mathbf{W} = s(\mathbf{K})$, a sparsified version of $\mathbf{K}$.
c. Interpret $\mathbf{W}$ as the adjacency matrix of a weighted undirected graph G.

2. Spectral embedding
   a. Compute the eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k$ associated with the k smallest eigenvalues of a graph representative matrix $\mathbf{R}$ (usually a Laplacian) computed from $\mathbf{W}$.
   b. Set $\mathbf{U}_k = [\,\mathbf{u}_1 | \,\mathbf{u}_2 | \cdots | \,\mathbf{u}_k\,] \in \mathbb{R}^{n \times k}$.
   c. Embed the i-th node to $\mathbf{x}_i = \frac{\mathbf{U}_k(i,:)^\top}{q(\|\mathbf{U}_k(i,:)\|_2)}$, with $q(\cdot)$ a normalizing function.
3. Clustering
   a. Use k-means on $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in order to identify k centroids $\mathbf{c}_1, \ldots, \mathbf{c}_k$.
   b. Voronoi tesselation: construct one cluster per centroid $\mathbf{c}_\ell$ and assign each object i to the cluster of the centroid closest to $\mathbf{x}_i$.

Output: A partition of the n points in k clusters.

A few comments are in order:

- A common choice of kernel in step 1a is the radial basis function (RBF) kernel $\kappa(\|\mathbf{p}_i - \mathbf{p}_j\|_2) = \exp\left(-\|\mathbf{p}_i - \mathbf{p}_j\|_2^2/\sigma^2\right)$ for some user-defined sparsification function σ (step 1b). The sparsification s of $\mathbf{K}$ usually entails setting the diagonal to 0 and keeping only the k largest entries of each column (i.e., set all others to 0). The obtained matrix $\mathbf{K}_{\text{sp}}$ is not symmetric in general and a final "symmetrization" step $\mathbf{W} = \mathbf{K}_{\text{sp}} + \mathbf{K}_{\text{sp}}^\top$ is necessary to obtain a matrix $\mathbf{W}$ interpretable as the adjacency matrix of a weighted undirected graph[2] $G = (\mathsf{V}, \mathsf{E}, \mathbf{W})$. This graph is called the k nearest neighbour (k-NN) similarity graph (note that the k used in this paragraph has nothing to do with the number of clusters). Other kernel functions κ and sparsification methods are possible (see Section 2 of [138] for instance).
- There are several possibilities for choosing the graph representative matrix $\mathbf{R}$ in step 2a. We consider three main choices [138]: Let us denote by $\mathbf{D}$ the diagonal degree matrix such that $\mathbf{D}(i,i) = \sum_j \mathbf{W}(i,j)$ is the (weighted) degree of node v_i. We define the *combinatorial* graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, the *normalized* graph Laplacian matrix $\mathbf{L}_n = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, and the *random walk* Laplacian $\mathbf{L}_{rw} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$. Other popular choices include[3] the non-backtracking matrix [73], degree-corrected versions of the modularity matrix [2], the Bethe-Hessian matrix [114] or similar deformed Laplacians [34].
- The normalizing function $q(\cdot)$ used in step 2c depends on which representative matrix is chosen. In the case of the Laplacians, experimental evidence as well as some theoretical arguments [138] support using a unit norm normalization for

[2] Each node v_i of V represents a point $\mathbf{p}_i$, an undirected edge exists between nodes v_i and v_j if and only if $\mathbf{W}(i,j) \neq 0$, and the weight of that connection is $\mathbf{W}(i,j)$.

[3] In some of these examples, the k largest eigenvalues (instead of the k lowest in the Laplacian cases) of the representative matrix, and especially their corresponding eigenvectors, are of interest. This is only a matter of sign of the matrix $\mathbf{R}$ and has no impact on the general discussion.

the eigenvectors of $\mathbf{L}_n$ (i.e. q is the identity function), and no normalization for the eigenvectors of $\mathbf{L}$ and $\mathbf{L}_{rw}$ (i.e. $q(\cdot) = 1$).

- Step 1 of the algorithm is "optional" in the sense that in some cases the input is not a set of points but directly a graph. For instance, it could be a graph representing a social network between n individuals, where each node is an individual and there is an edge between two nodes if they know each other. The weight on each edge can represent the strength of their relation (for instance close to 0 if they barely know each other, and close to 1 if they are best friends). The goal is then to classify individuals based on the structure of these social connections and is usually referred to as *community detection* in this context [47]. Given the input graph, and the number k of communities to identify, one can run spectral algorithms starting directly at step 2. Readers only interested in such applications can skip Sec. 3, which is devoted to sampling techniques designed to accelerate step 1.

After the spectral embedding $\mathsf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ has been identified, spectral clustering uses k-means in order to find the set of k centroids $\mathsf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ that best represents the data. Formally, the k-means cost function to minimize reads:

$$f(\mathsf{C};\mathsf{X}) = \sum_{\mathbf{x}\in\mathsf{X}} \min_{\mathbf{c}\in\mathsf{C}} \|\mathbf{x}-\mathbf{c}\|_2^2. \tag{1}$$

We would ideally hope to identify the set of k centroids C^* minimizing $f(\mathsf{C};\mathsf{X})$. Solving exactly this problem is NP-hard [41], so one commonly resorts to approximation and heuristic solutions (see for instance [128] for details on different such heuristics). The most famous is the so-called Lloyd-Max heuristic algorithm:

Algorithm 2. The Lloyd-Max algorithm [87]
Input. Set of n points $\mathsf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ and number of desired clusters k.

1. Start from an initial guess C_{ini} of k centroids
2. Iterate until convergence:
   a. Assign each point $\mathbf{x}_i$ to its closest centroid to obtain a partition of X in k clusters.
   b. Move each centroid $\mathbf{c}_\ell$ to the average position of all points in cluster ℓ.

Output: A set of k centroids $\mathsf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$.

When the clusters are sufficiently separated and C_{ini} is not too far from the optimal centroids, then the Lloyd-Max algorithm converges to the correct solution [75]. Otherwise, it typically ends up in a local minimum.

A remark on notation. Two quantities of fundamental importance in spectral clustering are the eigenvalues λ_i and especially the eigenvectors $\mathbf{u}_i$ of the graph Laplacian matrix. We adopt the graph theoretic convention of sorting eigenvalues in non-decreasing order: $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. Also, for reasons of brevity, we overload

notation and use the same symbol for the spectrum of the three Laplacians $\mathbf{L}$, $\mathbf{L}_n$ and $\mathbf{L}_{rw}$. Thus, we advise the reader to rely on the context in order to discern which Laplacian gives rise to the eigenvalues and eigenvectors. Finally, the reader should keep in mind that the largest eigenvalue if always bounded by 2 for $\mathbf{L}_n$ and $\mathbf{L}_{rw}$.

2.1 An illustration of spectral clustering

The first two steps of the algorithm can be understood as a non-linear transformation from the initial feature space to another feature space (that we call spectral feature space or spectral embedding): a transformation of features $\mathbf{p}_i$ in $\mathbb{R}^d$ to spectral features $\mathbf{x}_i$ in $\mathbb{R}^k$. The first natural question that arises is why do we run k-means on the spectral features $\mathsf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ that are subject to parameter tuning and costly to compute, rather than directly run k-means on the original P? Figures 1 and 2 illustrate the answer.

In Figure 1, we show the result of k-means directly on a set of artificial features P known as the two-half moons dataset. In this example, the intuitive ground truth is that each half-moon corresponds to a class, that we want to recover. Running k-means directly in this 2D feature space will necessarily output a linear separation between the two obtained Voronoi cells and will thus necessarily fail, as no straight line can separate the two half-moons.

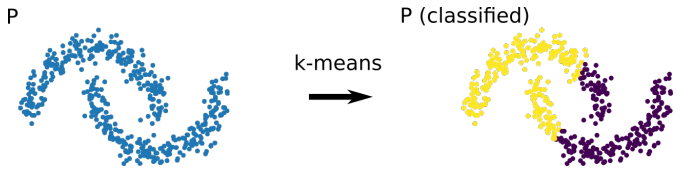

Fig. 1 Left: the two half-moons synthetic dataset ($n = 500$, $d = 2$, $k = 2$). Right: k-means with $k = 2$ directly on P is unsuccessful to separate the two half-moons.

Spectral clustering, via the computation of the spectral features of a similarity graph, transforms these original features P in spectral features X that are typically linearly separable by k-means: the two half-moons are successfully recovered! We illustrate this in Figure 2. In the next section, we will examine a theoretical argument aiming to justify this phenomenon.

2.2 Justification of spectral clustering

A popular approach –and by no means the only one, see Sec. 2.2.3– to justify spectral clustering algorithms stems from its connection to graph partitioning. Suppose

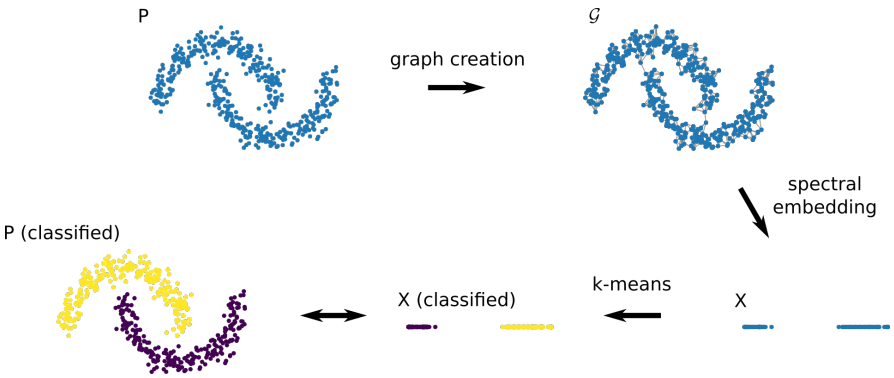

Fig. 2 Illustration of the spectral clustering algorithm on the two half-moons dataset ($n = 500$, $d = 2$, $k = 2$). The graph is created with a RBF kernel and via a sparsification done with k-nearest neighbours (with $k = 5$). The spectral embedding is done with the two eigenvectors associated to the two smallest eigenvalues of the combinatorial Laplacian matrix $\mathbf{L}$. The embedding X is here in practice in $1\mathbf{D}$ as the first eigenvector of $\mathbf{L}$ is always constant and thus not discriminative (to confirm this, first show that $\mathbf{L}$ is a PSD matrix and then prove that $\mathbf{L}\mathbf{c} = 0$ for any constant vector $\mathbf{c}$). Observe how the two clusters are now linearly separable in the spectral feature space. k-means on *these* features successfully recovers the two half-moons.

that the similarity graph $G = (\mathsf{V}, \mathsf{E}, \mathbf{W})$ has been obtained and we want to compute a partition[4] $\mathscr{P} = \{\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_k\}$ of the nodes V in k groups. Intuitively, a good clustering objective function should favor strongly connected nodes to end up in the same subset, and nodes that are far apart in the graph to end up in different subsets. This intuition can be formalized with graph cuts.

Considering two groups V_1 and V_2, define $w(\mathsf{V}_1, \mathsf{V}_2) = \sum_{i \in \mathsf{V}_1} \sum_{j \in \mathsf{V}_2} \mathbf{W}(i, j)$ to be the total weight of all links connecting V_1 to V_2. Also, denote by $\bar{\mathsf{V}}_\ell$ the complement of V_ℓ in V, such that $w(\mathsf{V}_\ell, \bar{\mathsf{V}}_\ell)$ is the total weight one needs to cut in order to disconnect V_ℓ from the rest of the graph. Given these definitions, the simplest graph cut objective function, denoted by `cut`, is:

$$\texttt{cut}\,(\mathscr{P} = \{\mathsf{V}_1, \ldots, \mathsf{V}_k\}) = \frac{1}{2} \sum_{\ell=1}^{k} w(\mathsf{V}_\ell, \bar{\mathsf{V}}_\ell). \tag{2}$$

The best partition according to the `cut` criterion is $\mathscr{P}^* = \operatorname{argmin}_{\mathscr{P}} \texttt{cut}(\mathscr{P})$. For $k = 2$, solving this problem can be done exactly in $O(ne + n^2 \log(n))$ amortized time using the Stoer-Wagner algorithm [126] and approximated in nearly linear time [68]. Nevertheless, this criterion is not satisfactory as it often separates an individual node from the rest of the graph, with no attention to the balance of the sizes or volumes of the groups. In clustering, one usually wants to partition into groups that are "large enough". There are two famous ways to balance the previous cost in the machine

[4] By definition, a *partition* $\mathscr{P} = \{\mathsf{V}_1, \mathsf{V}_2, \ldots, \mathsf{V}_k\}$ of the nodes V is such that $\cup_{\ell=1,\ldots,k} \mathsf{V}_\ell = \mathsf{V}$ and $\forall \ell \neq \ell', \mathsf{V}_\ell \cap \mathsf{V}_{\ell'} = \emptyset$

learning literature[5]: the *ratio cut* [143] and *normalized cut* [121] cost functions, respectively defined as:

$$\texttt{rcut}(\mathscr{P}) = \frac{1}{2}\sum_{\ell=1}^{k}\frac{w(\mathsf{V}_\ell,\bar{\mathsf{V}}_\ell)}{|\mathsf{V}_\ell|} \quad \text{and} \quad \texttt{ncut}(\mathscr{P}) = \frac{1}{2}\sum_{\ell=1}^{k}\frac{w(\mathsf{V}_\ell,\bar{\mathsf{V}}_\ell)}{\mathrm{vol}(\mathsf{V}_\ell)}, \tag{3}$$

where $|\mathsf{V}_\ell|$ is the number of nodes in V_ℓ and $\mathrm{vol}(\mathsf{V}_\ell) = \sum_{i\in\mathsf{V}_\ell}\sum_{j\in\mathsf{V}}\mathbf{W}(i,j)$ is the so-called volume of V_ℓ. The difference between them is that `ncut` favors clusters of large volume, whereas `rcut` only considers cluster size—though for a d-regular graph with unit weights the two measures match (up to multiplication by $1/d$). Unfortunately, it is hard to minimize these cost functions directly: minimizing these two balanced costs is NP-hard [139, 121] and one needs to search over the space of all possible partitions which is of exponential size.

A continuous relaxation. Spectral clustering may be interpreted as a continuous relaxation of the above minimization problems. Without loss of generality, in the following we concentrate on relaxing the `rcut` minimization problem (`ncut` is relaxed almost identically). Given a partition $\mathscr{P} = (\mathsf{V}_1,\ldots,\mathsf{V}_k)$, let us define

$$\mathbf{C} = \left(\frac{\mathbf{z}_1}{\sqrt{|\mathsf{V}_1|}}|\ldots|\frac{\mathbf{z}_k}{\sqrt{|\mathsf{V}_k|}}\right) \in \mathbb{R}^{n\times k}, \tag{4}$$

where $\mathbf{z}_\ell \in \mathbb{R}^n$ is the indicator vector of V_ℓ:

$$\mathbf{z}_\ell(i) = \begin{cases} 1 & \text{if node } i \in \mathsf{V}_\ell, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

It will prove useful in the following to remark that, independently of how the partitions are chosen, we always have that $\mathbf{C}^\top\mathbf{C} = \mathbf{I}$, the identity matrix in dimension k. With this in place, the problem of minimizing `rcut` can be rewritten as (see discussion in [138]):

$$\min_{\mathbf{C}\in\mathbb{R}^{n\times k}} \mathrm{tr}\Big(\mathbf{C}^\top\mathbf{L}\mathbf{C}\Big) \quad \text{s.t.} \quad \mathbf{C}^\top\mathbf{C} = \mathbf{I} \quad \text{and} \quad \mathbf{C} \text{ as in (4)} \tag{6}$$

To understand why this equivalence holds, one should simply note that

$$\begin{aligned}\mathrm{tr}\Big(\mathbf{C}^\top\mathbf{L}\mathbf{C}\Big) = \sum_{\ell=1}^{k}\frac{1}{|\mathsf{V}_\ell|}\mathbf{z}_\ell^\top\mathbf{L}\mathbf{z}_\ell &= \sum_{\ell=1}^{k}\frac{1}{|\mathsf{V}_\ell|}\sum_{i>j}\mathbf{W}(i,j)(\mathbf{z}_\ell(i)-\mathbf{z}_\ell(j))^2 \\ &= \sum_{\ell=1}^{k}\frac{w(\mathsf{V}_\ell,\bar{\mathsf{V}}_\ell)}{|\mathsf{V}_\ell|} = 2\,\texttt{rcut}(\mathscr{P}).\end{aligned}$$

[5] The reader should note that in the graph theory literature, the measure of conductance is preferred over `ncut`. Conductance is $\max_\ell w(\mathsf{V}_\ell,\bar{\mathsf{V}}_\ell)/w(\mathsf{V}_\ell)$. The two measures are equivalent when $k = 2$.

Solving (6) is obviously still NP-hard as the only thing we have achieved is to rewrite the `rcut` minimization problem in matrix form. Yet, in this form, it is easier to realize that one may find an approximate solution by relaxing the discreteness constraint "$\mathbf{C}$ as in (4)". In the absence of the hard-to-deal-with constraint, the relaxed problem is not only polynomially solvable but also possesses a closed-form solution! By the Courant–Fischer–Weyl (min-max) theorem, the solution is given by the first k eigenvectors $\mathbf{U}_k = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k]$ of $\mathbf{L}$:

$$\mathbf{U}_k = \underset{\mathbf{C}\in\mathbb{R}^{n\times k}}{\arg\min}\ \mathrm{tr}\Big(\mathbf{C}^\top \mathbf{L}\mathbf{C}\Big) \quad \text{subject to} \quad \mathbf{C}^\top\mathbf{C} = \mathbf{I}.$$

This relaxation is not unique to the combinatorial Laplacian. In the same spirit, the minimum `ncut` optimization problem can be formulated in terms of the normalized Laplacian matrix $\mathbf{L}_n$, and the relaxed problem's solution is given by the first k eigenvectors of $\mathbf{L}_n$.

A difficulty still lies before us: how do we go from a real-valued $\mathbf{U}_k$ to a partition of the nodes? The two next subsections aim to motivate the use of k-means as a rounding heuristic. The exposition starts from the simple case when there are only two clusters ($k = 2$) before considering the general case (arbitrary k).

2.2.1 The case of two clusters: thresholding suffices

For simplicity, we first consider the case of two clusters. If one constructs a partitioning $\mathscr{P}_t$ with $\mathsf{V}_1 = \{v_i : \mathbf{u}_2(i) > t\}$ and $\mathsf{V}_2 = \{v_i : \mathbf{u}_2(i) \leq t\}$ for every level set $t \in (-1,1)$, then it is a folklore result that

$$\texttt{rcut}(\mathscr{P}^*) \leq \min_t \texttt{rcut}(\mathscr{P}_t) \leq 2\sqrt{\texttt{rcut}(\mathscr{P}^*)\left(d_{\max} - \frac{\lambda_2}{2}\right)}, \tag{7}$$

with $\mathscr{P}^* = \arg\min_{\mathscr{P}} \texttt{rcut}(\mathscr{P})$ being the optimal partitioning, $d_{\max}$ is the maximum degree of any node in V, and λ_2 the second smallest eigenvalue of $\mathbf{L}$. The upper bound is achieved by the tree-cross-path graph constructed by Guattery and Miller [57]. In an analogous manner, if $\mathscr{P}^* = \arg\min_{\mathscr{P}} \texttt{ncut}(\mathscr{P})$ is the optimal partitioning w.r.t. the `ncut` cost and every $\mathscr{P}_t$ has been constructed by thresholding the second eigenvector of $\mathbf{L}_n$, then

$$\texttt{ncut}(\mathscr{P}^*) \leq \min_t \texttt{ncut}(\mathscr{P}_t) \leq 2\sqrt{\texttt{ncut}(\mathscr{P}^*)}. \tag{8}$$

Inequality (8) can be derived as a consequence of the *Cheeger inequality*, a key result of spectral graph theory [32], which for the normalized Laplacian reads:

$$\frac{\lambda_2}{2} \leq \texttt{ncut}(\mathscr{P}^*) \leq \min_{\mathsf{V}} \frac{w(\mathsf{V},\bar{\mathsf{V}})}{\min\{w(\mathsf{V}), w(\bar{\mathsf{V}})\}} \leq \min_t \texttt{ncut}(\mathscr{P}_t) \leq \sqrt{2\lambda_2}.$$

As a consequence, we have

$$\texttt{ncut}(\mathscr{P}^*) \leq \min_t \texttt{ncut}(\mathscr{P}_t) \leq \sqrt{2\lambda_2} \leq \sqrt{4\texttt{ncut}(\mathscr{P}^*)} = 2\sqrt{\texttt{ncut}(\mathscr{P}^*)},$$

as desired. The derivation of the `rcut` bound given in (7) follows similarly.

2.2.2 More than two clusters: use k-means

As the number of clusters k increases, the brute-force approach of testing every level set becomes quickly prohibitive. But why is k-means the right way to obtain the clusters in the spectral embedding? Though a plethora of experimental evidence advocate the use of k-means, a rigorous justification is still lacking. The interested reader may refer to [83] for an example of an analysis of spectral partitioning without k-means.

More recently, Peng et al. [107] came up with a mathematical argument showing that, if G is well clusterable and we use a k-means algorithm (e.g., [76]) which guarantees that the identified solution $\tilde{\mathsf{C}}$ abides to

$$f(\tilde{\mathsf{C}};\mathsf{X}) \leq (1+\varepsilon) f(\mathsf{C}^*;\mathsf{X}),$$

where C^* is the optimal solution of the k-means problem, then the partitioning $\tilde{\mathscr{P}}$ produced by spectral clustering when using $\mathbf{L}_n$ has `ncut` cost provably close to that of the optimal partitioning $\mathscr{P}^*$. In particular, it was shown that, as long as $\lambda_{k+1} \geq \mathrm{c}k^2\texttt{ncut}(\mathscr{P}^*)$, then

$$\texttt{ncut}(\mathscr{P}^*) \leq \texttt{ncut}(\tilde{\mathscr{P}}) \leq \zeta\,\texttt{ncut}(\mathscr{P}^*)\left(1+\varepsilon\,\frac{k^3}{\lambda_{k+1}}\right),$$

for some constants $\mathrm{c}, \zeta > 0$ that are independent of n and k (see also [71]). Note that, using the higher-order Cheeger inequality [83] $\lambda_k/2 \leq \texttt{ncut}(\mathscr{P}^*)$, the condition $\lambda_{k+1} \geq \mathrm{c}k^2\texttt{ncut}(\mathscr{P}^*)$ implies

$$\frac{\lambda_{k+1}}{\lambda_k} \geq \frac{\mathrm{c}k^2}{2} = \Omega(k^2).$$

Though hopefully milder than this one[6], such gap assumptions are very common in the analysis of spectral clustering. Simply put, the larger the gap $\lambda_{k+1} - \lambda_k$ is, the stronger the cluster structure and the easier it is to identify a good clustering. Besides quantifying the difficulty of the clustering problem, the gap also encodes the robustness of the spectral embedding to errors induced by approximation algorithms [36]. The eigenvectors of a *perturbed* Hermitian matrix exhibit an interesting property:

[6] To construct an example possibly verifying such a strong gap assumption, consider k cliques of size k connected together via only $k-1$ edges, so as to form a loosely connected chain. Even though this is a straightforward clustering problem known to be easy for spectral clustering algorithms, the above theorem's assumption implies $\lambda_{k+1} = \Omega(k^2\texttt{ncut}(\mathscr{P}^*)) = \Omega(k)$ which, independently of n, can only be satisfied when k is a small (recall that the eigenvalues of $\mathbf{L}_n$ are necessarily between 0 and 2).

instead of being arbitrary, induced changes are localized w.r.t. the eigenvalue axis, following an inverse square eigenvalue-distance law [89]. More precisely, if $\tilde{\mathbf{u}}_i$ is the i-th eigenvector after perturbation, then the inner products $(\tilde{\mathbf{u}}_i^\top \mathbf{u}_j)^2$ decrease proportionally with $|\lambda_i - \lambda_j|^2$. As such, demanding that $\lambda_{k+1} - \lambda_k$ is large is often helpful in the analysis of spectral clustering algorithms in order to ensure that the majority of useful information (contained within $\mathbf{U}_k$) is preserved (in $\tilde{\mathbf{U}}_k$) despite approximation errors[7].

2.2.3 Choice of relaxation

The presented relaxation approach is not unique and other relaxations could be equally valid (see for instance [17, 24, 112]). This relaxation has nevertheless the double advantage of being theoretically simple and computationally easy to implement. Also, justification of spectral clustering algorithms does not only come from this graph cut perspective and in fact encompasses several approaches that we will not detail here: perturbation approaches or hitting time considerations [138], a polarization theorem [23], consistency derivations [135, 84], etc. Interestingly, recent studies (for instance [18]) on the Stochastic Block Models have shown that spectral clustering (on other matrices than the Laplacian, such as the non-backtracking matrix [73], or the Bethe-Hessian matrix [114] or other similar deformed Laplacians [34]) perform well up to the detectability threshold of the block structure.

2.3 Computational complexity considerations

What is the computational complexity of spectral clustering as a function of the number of points n, their dimension d and the number of desired clusters k? Let us examine the three steps involved one by one.

The first step entails the construction of a sparse similarity graph from the input points, which is dominated by the kernel computation and costs $\mathcal{O}(dn^2)$. In the second step, given the graph G consisting of n nodes and e edges[8], one needs to compute the spectral embedding (step 2 of Algorithm 1). Without exploiting the special structure of a graph Laplacian —other than its sparsity that is— there are two main options:

- Using power iterations, one may identify sequentially each non-trivial eigenvector $\mathbf{u}_\ell$ in time $\mathcal{O}(e/\delta_\ell)$, where $\delta_\ell = \lambda_\ell - \lambda_{\ell-1}$ is the ℓ-th eigenvalue gap and e is the number of edges of the graph [136]. Computing the spectral embedding there-

[7] Usually, one needs to ensure that $\sum_{i\leq k, j>k}(\tilde{\mathbf{u}}_i^\top \mathbf{u}_j)^2/k$ remains bounded.

[8] with e of the order of n if the sparsification step was well conducted

fore takes $\mathscr{O}(ke/\delta)$ with $\delta = \min_\ell \delta_\ell$. Unfortunately, there exist graphs[9] such that $\delta = \mathscr{O}(1/n)$, bringing the overall worst-case complexity to $\mathscr{O}(kne)$.

- The Lanczos method can be used to approximate the first k eigenvectors in roughly $\mathscr{O}(ek + nk^2)$ time. This procedure is often numerically unstable resulting to a loss of orthogonality in the computed Krylov subspace basis. The most common way to circumvent this problem is by implicit restart [26], whose computational complexity is not easily derived. The number of restarts, empirically, depend heavily on the eigenvalue distribution in the vicinity of λ_k: if λ_k is in an eigenvalue bulk, the algorithms takes longer than when λ_k is isolated. We decide to write the complexity of restarted Arnoldi as $\mathscr{O}(t(ek + nk^2))$ with t modeling the number of restarts. Note that throughout this paper, t will generically refer to a number of iterations in algorithm complexities. We refer the interested reader to [13] for an in-depth discussion of Lanczos methods.

The third step entails solving the k-means problem, typically by using the Lloyd-Max algorithm to converge to a local minimum of $f(\mathsf{C};\mathsf{X})$. Since there is no guarantee that this procedure will find a good local minimum, it is usually rerun multiple times, starting in each case from randomly selected centroids C_{ini}. The computational complexity of this third step is $\mathscr{O}(tnk^2)$, where t is a bound on the number of iterations required until convergence multiplied by the number of retries (typically 10).

2.4 A taxonomy of sampling methods for spectral clustering

For the remainder of the chapter, we propose to classify sampling methods aiming at accelerating one or more of these three steps according to when they sample. If they sample before step 1, they are detailed in Sec. 3. Methods that assume that the similarity graph is given or well-approximated and sample between steps 1 and 2 will be found in Sec. 4. Finally, methods that assume that the spectral embedding has been exactly computed or well-approximated and sample before the k-means step are explained in Sec. 5. This classification of methods, like all classification systems, bears a few flaws. For instance, Nyström methods can be applied to both the context of Sections 3 and 4 and are thus mentioned in both. Also, we decided to include the pseudo-code of only a few chosen algorithms that we think are illustrative of the literature. This choice is of course subjective and debatable. Notwithstanding these flaws, we hope that this taxonomy clarifies the landscape of existing methods.

[9] The combinatorial Laplacian of a complete balanced binary tree on $k \geq 3$ levels and $n = 2^k - 1$ nodes has $\frac{1}{n} \leq \lambda_2 \leq \frac{2}{n}$ [56].

3 Sampling in the original feature space

This section is devoted to methods that ambitiously aim to reduce the dimension of the spectral clustering problem even before the graph has been formed. Indeed, the naive way of building the similarity graph (step 1 of spectral clustering algorithms) costs $\mathscr{O}(dn^2)$ and, as such, is one of the the main computational bottlenecks of spectral clustering. It should be remarked that the present discussion fits into the wider realm of kernel approximation, a proper review of which cannot fit in this chapter: we will thus concentrate on methods that were in practice used for spectral clustering.

3.1 Nyström-based methods

The methods of this section aim to obtain an approximation $\tilde{\mathbf{U}}_k$ of the exact spectral embedding $\mathbf{U}_k$ via a sampling procedure in the original feature space.

The Nyström method is a well known algorithm for obtaining a rough low rank approximation of a positive semi-definite (PSD) matrix $\mathbf{A}$. Here is a high level description of the steps entailed:

Algorithm 3. Nyström's method
Input. PSD matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, number of samples m, desired rank k

1. Let S be m column indices chosen by some sampling procedure.
2. Denote by $\mathbf{B} = \mathbf{A}(\mathsf{S},\mathsf{S}) \in \mathbb{R}^{m\times m}$ and $\mathbf{C} = \mathbf{A}(:,\mathsf{S}) \in \mathbb{R}^{n\times m}$ the sub-matrices indexed by S.
3. Let $\mathbf{B} = \mathbf{Q}\mathbf{\Sigma}\mathbf{Q}^\top$ be the eigen-decomposition of $\mathbf{B}$ with the diagonal of $\mathbf{\Sigma}$ sorted in decreasing magnitude.
4. Compute the rank-k approximation of $\mathbf{B}$ as $\mathbf{B}_k = \mathbf{Q}_k\mathbf{\Sigma}_k\mathbf{Q}_k^\top$, where $\mathbf{Q}_k = \mathbf{Q}(:,:k) \in \mathbb{R}^{n\times k}$ and $\mathbf{\Sigma}_k = \mathbf{\Sigma}(:k,:k)$.

Possible outputs:

- A low-rank approximation $\tilde{\mathbf{A}} = \mathbf{C}\mathbf{B}^{+}\mathbf{C}^\top \in \mathbb{R}^{n\times n}$ of $\mathbf{A}$
- A rank-k approximation $\tilde{\mathbf{A}}_k = \mathbf{C}\mathbf{B}_k^{+}\mathbf{C}^\top \in \mathbb{R}^{n\times n}$ of $\mathbf{A}$
- The top k eigenvectors of $\tilde{\mathbf{A}}_k$, stacked as columns in matrix $\tilde{\mathbf{V}}_k \in \mathbb{R}^{n\times k}$, obtained by orthonormalizing the columns of $\tilde{\mathbf{Q}}_k = \mathbf{C}\mathbf{Q}_k\mathbf{\Sigma}_k^{-1} \in \mathbb{R}^{n\times k}$

Various guarantees are known for the quality of $\tilde{\mathbf{A}}$ depending on the type of sampling utilized (i.e., how the indices in S are selected in step 1) and the preferred notion of error (spectral $\|.\|_2$ vs frobenius $\|.\|_F$ vs trace $\|.\|_*$ norm) [54, 77, 50, 148]. For instance:

Theorem 1 (Lemma 8 for $q = 1$ in [54]). *Let $\varepsilon \in (0,1)$ and $\delta \in (0,1)$ and suppose that S contains the indices of m columns drawn i.i.d. uniformly at random (with or without replacement). Then:*

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_2 \leq \left(1 + \frac{n}{(1-\varepsilon)m}\right) \|\mathbf{A} - \mathbf{A}_k\|_2$$

holds with probability at least $1 - 3\delta$, provided that $m \geq 2\varepsilon^{-2}\mu k \log(k/\delta)$; where

$$\mu = \frac{n}{k} \max_{i=1,\ldots,n} \|\mathbf{V}_k(i,:)\|_2^2$$

is the coherence associated with the first k eigenvectors $\mathbf{V}_k$ of $\mathbf{A}$, and $\mathbf{A}_k$ is the best rank-k approximation of $\mathbf{A}$.

Guarantees independent of the coherence can be obtained for more advanced sampling methods. Perhaps the most well known method is that of *leverage scores*, where one draws m samples independently by selecting (with replacement) the i-th column with probability $\mathbf{p}_i = \|\mathbf{V}_k(i,:)\|_2^2/k$.

Theorem 2 (Lemma 5 for $q = 1$ in [54]). *Let $\varepsilon \in (0,1)$ and $\delta \in (0,1)$ and suppose that S contains the indices of m columns drawn i.i.d. with replacement from such a probability distribution. Then:*

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_2 \leq \|\mathbf{A} - \mathbf{A}_k\|_2 + \varepsilon^2 \|\mathbf{A} - \mathbf{A}_k\|_*$$

holds with probability at least $0.8 - 2\delta$ provided that $m \geq \mathscr{O}(\varepsilon^{-2} k \log(k/\delta))$.

Computing leverage scores exactly is computationally prohibitive since it necessitates a partial SVD decomposition of $\mathbf{A}$, which we are trying to avoid in the first place. Nevertheless, it is possible to approximate all leverage scores with a multiplicative error guarantee in time roughly $\mathscr{O}(ek\log(e))$ if $\mathbf{A}$ has $O(e)$ non-zero entries. (see Algorithms 1 to 3 in [54]). Many variants of the above exist [77, 78], but to the best of our knowledge, the fastest current Nyström algorithm utilizes ridge leverage scores with a complex recursive sampling scheme and runs in time nearly linear in n [100].

Nyström for spectral clustering. Though initially conceived for low-rank approximation, Nyström's method can also be used to accelerate spectral clustering. The key observation is that $\mathbf{U}_k$, the tailing k eigenvectors of the graph representative matrix $\mathbf{R}$, can be interpreted as the top k eigenvectors of the PSD matrix $\mathbf{A} = \|\mathbf{R}\|_2\mathbf{I} - \mathbf{R}$. As such, the span of the k top eigenvectors of $\tilde{\mathbf{A}}_k$ obtained by running Algorithm 3 on $\mathbf{A}$ is an approximation of the span of the exact spectral embedding. Different variants of this idea have been considered for the acceleration of spectral clustering [48, 141, 85, 19, 97, 86].

Following our taxonomy, we hereby focus on the case where we have at our disposal n points $\mathbf{p}_i$ in dimension d, and the similarity graph has yet to be formed. The case where the graph is known is deferred to Sec. 4.

In this case, we cannot run Algorithm 3 on $\mathbf{A} = \|\mathbf{R}\|_2\mathbf{I} - \mathbf{R}$ as the graph, and *a fortiori* its representative matrix $\mathbf{R}$, has not yet been formed. What we *can* have access to *efficiently* is $\mathbf{B} = s(\mathbf{K}(\mathsf{S},\mathsf{S}))$ and $\mathbf{C} = s(\mathbf{K}(:,\mathsf{S}))$, as these require only a partial computation of the kernel and cost only $\mathscr{O}(dnm)$. Note that s is a sparsification function that is applied on a subset of the kernel matrix.

The following pseudo-code exemplifies how Nyström-based techniques can be used to approximate the first k eigenvectors $\mathbf{U}_k$ associated with the normalized Laplacian matrix (i.e., here $\mathbf{R} = \mathbf{L}_n$):

Algorithm 3b. Nyström for spectral clustering [85]
Input. The set of points P, the number of desired clusters k, a sampling set S of size $m \geq k$

1. Compute the sub-matrices $\mathbf{B} = s(\mathbf{K}(\mathsf{S},\mathsf{S})) \in \mathbb{R}^{m\times m}$ and $\mathbf{C} = s(\mathbf{K}(:,\mathsf{S})) \in \mathbb{R}^{n\times m}$, where s is a sparsification function.
2. Let $\mathbf{D}_r = \mathrm{diag}(\mathbf{B1})$ be the $m \times m$ degree matrix.
3. Compute the top k eigenvalues $\mathbf{\Sigma}_k$ and eigenvectors $\mathbf{Q}_k$ of $\mathbf{D}_r^{-1/2}\mathbf{B}\mathbf{D}_r^{-1/2}$.
4. Set $\tilde{\mathbf{Q}}_k = \mathbf{C}\mathbf{D}_r^{-1/2}\mathbf{Q}_k\mathbf{\Sigma}_k^{-1}$.
5. Let $\mathbf{D}_l = \mathrm{diag}(\tilde{\mathbf{Q}}_k\mathbf{\Sigma}_k\tilde{\mathbf{Q}}_k^\top\mathbf{1})$ be the $n \times n$ degree matrix.
6. Compute $\tilde{\mathbf{U}}_k$ obtained by orthogonalizing $\mathbf{D}_l^{-1/2}\tilde{\mathbf{Q}}_k$.

Output: $\tilde{\mathbf{U}}_k$, an approximation of the spectral embedding $\mathbf{U}_k$.

This algorithm runs in $\mathscr{O}(nm\max(d,k))$ time, which is small when m depends mildly on the other parameters of interest. Nevertheless, the algorithm (and others like it) suffers from several issues:

- Alg. 3b attempts to use Nyström's method on $\mathbf{A} = 2\mathbf{I} - \mathbf{L}_n = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}s(\mathbf{K})\mathbf{D}^{-\frac{1}{2}}$ via the exact computation of two sub-matrices of $\mathbf{K}$. In doing so, it makes two strong (and uncontrolled) approximations: First of all, the sparsification step (step 1 in Alg. 3b) is applied to the sub-matrices $\mathbf{K}(\mathsf{S},\mathsf{S})$ and $\mathbf{K}(:,\mathsf{S})$, deviating from the correct sparsification procedure that takes into account the entire kernel matrix $\mathbf{K}$. Second, the degree matrix $\mathbf{D}$ is never exactly computed as knowing it exactly would entail computing exactly $s(\mathbf{K})$, which is precisely what we are trying to avoid. Existing methods thus rely on heuristic approximations of the degree in order to bypass this difficulty (see steps 2 and 5 of Alg. 3b).
- Since we don't have direct access to the kernel matrix, we cannot utilize advanced sampling methods such as leverage scores to draw the sampling set S. This is particularly problematic if (due to sparsification), matrices $\mathbf{B}$ and $\mathbf{C}$ are sparse, as for sparse matrices uniform sampling is known to perform poorly [97]. Techniques that rely on distances between columns do not fair much better. Landmark-based approaches commonly perform better in simple problems but suffer when the clusters are non-convex [19]. We refer the reader to the work by Mohan et al. [97] for more information on landmark-based methods. The latter

work also describes an involved sampling scheme that is aimed at general (i.e., non-convex) clusters.

For the reasons highlighted above, the low-rank approximation guarantees accompanying the classical Nyström method cannot be directly used here. *A fortiori*, it is an open question how much the quality of the spectral clustering solution is affected by using the centroids obtained by running k-means on $\tilde{\mathbf{U}}_k$.

Column sampling. Akin in spirit to Nyström methods, an alternative approach to accelerating spectral clustering was inspired by column sampling low-rank approximation techniques [42, 37].

An instance of such algorithms was put forth under the name of cSPEC (column sampling spectral clustering) by Wang et al. [141]. Let $\mathbf{C} = \mathbf{U}_C \mathbf{\Sigma}_C \mathbf{V}_C^\top$ be the singular value decomposition of the $n \times m$ matrix $\mathbf{C} = s(\mathbf{K}(:, \mathsf{S}))$. Then, matrices

$$\tilde{\mathbf{\Sigma}} = \sqrt{\frac{n}{m}}\,\mathbf{\Sigma}_C \quad \text{and} \quad \tilde{\mathbf{U}} = \mathbf{C}\mathbf{V}_C\mathbf{\Sigma}_C^+$$

are interpreted as an approximation of the actual eigenvalues and eigenvectors of $\mathbf{K}$ and thus $\mathbf{U}_k$ can be substituted by the first k columns of $\tilde{\mathbf{U}}$. This algorithm runs in $\mathcal{O}(ndm + nm^2)$.

Authors in [30] propose a hybrid method, between column sampling and the representative-based methods discussed in Sec. 3.3, where they propose the following approximate factorization of the data matrix:

$$(\mathbf{p}_1 | \ldots | \mathbf{p}_n) \simeq \mathbf{F}\mathbf{Z} \in \mathbb{R}^{d \times n}, \tag{9}$$

where $\mathbf{F} \in \mathbb{R}^{d \times m}$ concatenates the feature vectors of m sampled points and $\mathbf{Z} \in \mathbb{R}^{m \times n}$ represents all unsampled points as approximate linear combinations of the representatives, computed via sparse coding techniques [82][10]. The SVD of $\tilde{\mathbf{D}}^{-1/2}\mathbf{Z}$, with $\tilde{\mathbf{D}}$ the row-sum of $\mathbf{Z}$, is then computed to obtain an approximation $\tilde{\mathbf{U}}_k$ of $\mathbf{U}_k$. The complexity of their algorithm is also $\mathcal{O}(ndm + nm^2)$.

In these methods, the choice of the sample set S is, of course, central and has been much debated. Popular options are uniformly at random or via better-tailored probability distributions, via a first k-means (with $k = m$) pass on P, or via other selective sampling methods. Also, as with most extensions of Nyström's method to spectral clustering, column sampling methods for spectral clustering do not come with end-to-end approximation guarantees on $\mathbf{U}_k$.

In the world of low-rank matrix approximation the situation is somewhat more advanced. Recent work in column sampling utilizes adaptive sampling with leverage scores in time $\mathcal{O}(e + n\text{poly}(k))$, or uniformly i.i.d. after preconditioning by a fast randomized Hadamard transform [145, 43]. Others have also used a correlated version called volume sampling to obtain column indices [37]. Nevertheless, this

[10] Authors in [116] have a very similar proposition as [30], adding a projection phase at the beginning to reduce the dimension d (see Sec. 3.4.2). Similar ideas may also be found in [137].

literature extends beyond the scope of this chapter and thus we invite the interested reader to consider the aforementioned references for a more in-depth perspective.

3.2 Random Fourier features

Out of several sketching techniques one could *a priori* use to accelerate spectral clustering, we focus on random Fourier features (RFF) [110]: a method that samples in the Fourier space associated to the original feature space. Even though RFFs have originally been developed to approximate a kernel matrix $\mathbf{K}$ in time linear in n instead of the quadratic time necessary for its exact computation, they can in fact be used to obtain an approximation $\tilde{\mathbf{U}}_k$ of the exact spectral embedding $\mathbf{U}_k$.

Let us denote by κ the RBF kernel, i.e., $\kappa(\mathbf{t}) = \exp(-\mathbf{t}^2/\sigma^2)$, whose Fourier transform is:

$$\hat{\kappa}(\omega) = \int_{\mathbb{R}^d} \kappa(\mathbf{t}) \exp^{-i\omega^\top \mathbf{t}} \mathrm{d}\mathbf{t}. \tag{10}$$

The above takes real values as κ is symmetric. One may write:

$$\kappa(\mathbf{p},\mathbf{q}) = \kappa(\mathbf{p}-\mathbf{q}) = \frac{1}{Z}\int_{\mathbb{R}^d} \hat{\kappa}(\omega) \exp^{i\omega^\top(\mathbf{p}-\mathbf{q})} \mathrm{d}\omega, \tag{11}$$

where, in order to ensure that $\kappa(\mathbf{p},\mathbf{p}) = 1$, the normalization constant is set to $Z = \int_{\mathbb{R}^d} \hat{\kappa}(\omega)\mathrm{d}\omega$. According to Bochner's theorem, and due to the fact that κ is positive-definite, $\hat{\kappa}/Z$ is a valid probability density function. $\kappa(\mathbf{p},\mathbf{q})$ may thus be interpreted as the expected value of $\exp^{i\omega^\top(\mathbf{p}-\mathbf{q})}$ provided that ω is drawn from $\hat{\kappa}/Z$:

$$\kappa(\mathbf{p},\mathbf{q}) = \mathbb{E}_\omega\left(\exp^{i\omega^\top(\mathbf{p}-\mathbf{q})}\right) \tag{12}$$

Drawing ω from the distribution $\hat{\kappa}/Z$ is equivalent to drawing independently each of its d entries according to the normal law of mean 0 and variance $2/\sigma^2$. Indeed: $\hat{\kappa}(\omega) = \pi^{d/2}\sigma^d \exp(-\sigma^2\omega^2/4)$ and $Z = \int_{\mathbb{R}^d} \hat{\kappa}(\omega)\mathrm{d}\omega = (2\pi)^d$, leading to

$$\frac{\hat{\kappa}(\omega)}{Z} = \left(\frac{\sigma}{2\sqrt{\pi}}\right)^d \exp^{-\sigma^2\omega^2/4}.$$

In practice, we draw independently m such vectors ω to obtain the set of sampled frequencies $\Omega = (\omega_1,\ldots,\omega_m)$. For each data point $\mathbf{p}_i$, and given this set of samples Ω, we define the associated random Fourier feature vector:

$$\psi_i = \frac{1}{\sqrt{m}}[\cos(\omega_1^\top\mathbf{p}_i)|\cdots|\cos(\omega_m^\top\mathbf{p}_i)|\sin(\omega_1^\top\mathbf{p}_i)|\cdots|\sin(\omega_m^\top\mathbf{p}_i)]^\top \in \mathbb{R}^{2m}, \tag{13}$$

and call $\Psi = (\psi_1|\cdots|\psi_n) \in \mathbb{R}^{2m\times n}$ the RFF matrix. Other embeddings are possible in the RFF framework, but this one was shown to be the most appropriate to

the Gaussian kernel [127]. As m increases, $\psi_i^\top \psi_j$ concentrates around its expected value $\kappa(\mathbf{p}_i, \mathbf{p}_j)$: $\psi_i^\top \psi_j \simeq \kappa(\mathbf{p}_i, \mathbf{p}_j)$. Proposition 1 of [127] states the tightness of this concentration: it shows that the approximation starts to be valid with high probability for $m \geq \mathscr{O}(d \log d)$. The Gaussian kernel matrix is thus well approximated as $\mathbf{K} \simeq \mathbf{\Psi}^\top \mathbf{\Psi}$. With such a low-rank approximation $\mathbf{\Psi}$ of $\mathbf{K}$, one can: estimate the degrees[11], degree-normalize $\mathbf{\Psi}$ to obtain a low-rank approximation of the normalized Laplacian $\mathbf{L}_n$ and perform an SVD to directly obtain an approximation $\tilde{\mathbf{U}}_k$ of the spectral embedding $\mathbf{U}_k$. The total cost to obtain this approximation is $\mathscr{O}(ndm + nm^2)$. These ideas were developed in Refs. [31, 146] for instance.

As in Nyström methods however, the concentration guarantees of RFFs for $\mathbf{K}$ do not extend to the degree-normalized case; moreover, the sparsification step 1b of spectral clustering is ignored. Note that improving over RFFs in terms of efficiency and concentration properties is the subject of recent research (see for instance [81]).

3.3 The paradigm of representative points

The methods detailed here sample in the original feature space and directly obtain a control on the misclustering rate due to the sampling process. They are based on the following framework:

1. Sample m so-called representatives.
2. Run spectral clustering on the representatives.
3. Lift the solution back to the entire dataset.

Let us illustrate this with the example of KASP:

Algorithm 4. KASP: k-means-based approximate spectral clustering [147]
Input. A set of n points $\mathsf{P} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$ in dimension d, a number of desired clusters k, and a number of representatives m.

1. Perform k-means with $k = m$ on P and obtain:
   a. the cluster centroids $\mathsf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$ as the m representative points.
   b. a correspondence table to associate each $\mathbf{p}_i$ to its nearest representative
2. Run spectral clustering on Y to get the cluster membership of each $\mathbf{y}_i$.
3. Lift the cluster membership to each $\mathbf{p}_i$ by looking up the cluster membership of its representative in the correspondence table.

Output: k clusters

[11] an approximation of the degree d_i of node v_i is $\psi_i^\top \bar{\psi}$ where $\bar{\psi} = \sum_j \psi_j$. All degrees can thus be estimated in time $\mathscr{O}(nm^2)$.

The complexity of KASP is bounded by[12] $\mathscr{O}(mdnt + m^3)$. For a summary of the analysis given in [147], let us consider the cluster memberships given by exact spectral clustering on P as well as the memberships given by exact spectral clustering on $\tilde{\mathsf{P}} = (\mathbf{p}_1 + \varepsilon_1, \ldots, \mathbf{p}_n + \varepsilon_n)$ where the ε_i are any small perturbations on the initial points. Let us denote by $\mathbf{L}_n$ (resp. $\tilde{\mathbf{L}}_n$) the normalized Laplacian matrix of the similarity graph on P (resp. $\tilde{\mathsf{P}}$). The analysis concentrates on the study of the miss-clustering rate ρ:

$$\rho = \frac{\text{\# of points with different memberships}}{n}. \tag{14}$$

The main result, building upon preliminary work in [63], stems from a perturbation approach and reads:

Theorem 3. *Under the assumptions of Theorem 3 in [147]:* $\rho \leq \mathscr{O}\left(\frac{k}{g_0^2}\|\mathbf{L}_n - \tilde{\mathbf{L}}_n\|_F\right)$, *where* g_0 *is a value depending on the spectral gap. Also, under the assumptions of Theorem 6 in [147], one has, with high probability:*

$$\|\mathbf{L}_n - \tilde{\mathbf{L}}_n\|_F \leq \mathscr{O}\left(\sigma_\varepsilon^{(2)} + \sigma_\varepsilon^{(4)}\right), \tag{15}$$

with $\sigma_\varepsilon^{(2)}$ *and* $\sigma_\varepsilon^{(4)}$ *the 2nd and 4th moments of the perturbation's norms* $\|\varepsilon_i\|_2$.

Combining both bounds, one obtains an upper bound on the misclustering rate that depends on the second and fourth moments of the perturbation's norms $\|\varepsilon_i\|_2$. The "collapse" of points onto the m representative points, interpreted as a perturbation on the original points, should thus tend to minimize these two moments, leading the authors to propose *distortion-minimizing* algorithms, such as KASP. A very similar algorithm, eSPEC, is described in [141].

3.4 Other methods

3.4.1 Approximate nearest neighbour search algorithms

The objective here is to approximate the nearest neighbour graph efficiently. Even though these methods are not necessarily based on sampling, we include them in the discussion as they are frequently used in practice.

Given the feature vectors $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$ and a query point $\mathbf{q} \in \mathbb{R}^d$, the exact nearest neighbour search (exact NNS) associated to P and $\mathbf{q}$ is $\mathbf{p}^* = \operatorname{argmin}_{\mathbf{p}\in\mathsf{P}} \operatorname{dist}(\mathbf{q}, \mathbf{p})$ where *dist* stands for any distance. Different distances are possible depending on the choice of kernel κ. We will here consider the Euclidean norm as it enters the definition of the classical RBF kernel. Computing the exact NNS costs $\mathscr{O}(dn)$. The goal

[12] It is in fact $\mathscr{O}(mdnt)$ for step 1, and bounded by $\mathscr{O}(dm^2 + m^2k + mk^2)$ for step 2. As $n \geq m$ and $m \geq k$, the total complexity is bounded by $\mathscr{O}(mdnt + m^3)$.

of the approximate NNS field of research is to provide faster algorithms that have the following control on the error.

Definition 1. Point $\mathbf{p}^*$ is an ε-approximate nearest neighbor of query $\mathbf{q} \in \mathbb{R}^d$, if

$$\forall \mathbf{p} \in \mathsf{P} \qquad \mathrm{dist}(\mathbf{q}, \mathbf{p}^*) \leq (1+\varepsilon)\,\mathrm{dist}(\mathbf{q}, \mathbf{p}).$$

For $\varepsilon = 0$, this reduces to exact NNS.

Extensions of this objective to the k-nearest neighbour goal are considered in the NNS literature. A k-nearest neighbour graph can then be constructed simply by running an approximate k-NNS query for each object $\mathbf{p}_i$. Thus, approximate NSS algorithms are interesting candidates to approximate the adjacency matrix of the nearest-neighbour affinity graph, that we need in step 1 of spectral clustering. Many algorithms exist, their respective performances depending essentially on the dimension d of the feature vectors. According to [9], randomized k-d forests as implemented in the library FLANN [98] are considered state-of-the-art for dimension of around 100, whereas methods based on Balanced Box Decomposition (BBD) [7, 4] are known to perform well for d roughly smaller than 100. In high dimensions, to avoid the curse of dimensionality, successful approaches are for instance based on hashing methods (such as Locality Sensitive Hashing (LSH) [5], Product Quantization (PQ) [66]) or k-d generalized random forests [9]. Finally, proximity graph methods, that sequentially improve over a first coarse approximation of the k-NN graph (or other graph structures such as navigable graphs) have received a large attention recently and are becoming state-of-the-art in regimes where quality of approximation primes (see for instance [94, 40, 51, 8]). Such tools come with various levels of guarantees and computation costs, the details of which are not in the scope of this chapter.

Experimentally, to obtain an approximate k-NN graph with a typical recall rate[13] of 0.9, these algorithms are observed to achieve a complexity of $\mathscr{O}(dn^{\alpha})$ with α close to 1 ($\alpha \simeq 1.1$ in [40] for instance).

3.4.2 Feature selection and feature projection

Some methods work on reducing the factor d of the complexity $\mathscr{O}(dn^2)$ of the kernel computation via feature selection, i.e., the sampling of features deemed more useful for the underlying clustering task, or feature projection, i.e., the projection on usually random subspaces of dimension $d' < d$. Feature selection methods are usually designed to *improve* the classification by removing features that are too noisy or useless for the classification. We thus do not detail further these methods as they are not approximation algorithms *per se*. The interested reader will find some entries in the literature via references [35, 60, 149, 25]. Projection methods use random projections of the original points P on spaces of dimension $d' \sim \log n$ in order to

[13] The recall rate for a node is the number of correctly identified k-NN divided by k. The recall rate for a k-NN graph is the average recall rate over all nodes.

take advantage of the Johnson-Lindenstrauss lemma of norm conservation: the kernel computed from the projected features in dimension d' is thus an approximation of the true kernel with high probability. We refer to the works [116, 64] for more details.

4 Sampling given the similarity graph

We now suppose that the similarity graph is either given (e.g., in cases where the original data *is* a graph) or has been well approximated (by approximate k-NN search for instance) and concentrate on sampling-based methods that aim to reduce the cost of computing the first k eigenvectors of $\mathbf{R}$.

These methods predominantly aim to approximate $\mathbf{R}$ by a smaller matrix $\tilde{\mathbf{R}}$ of size m. The eigen-decomposition is done in $\mathbb{R}^m$ which can be significantly cheaper when $m \ll n$. In addition, each method comes with a fast way of lifting vectors from $\mathbb{R}^m$ back to $\mathbb{R}^n$ (this is usually a linear transformation). After lifting, the eigenvectors of $\tilde{\mathbf{R}}$ are used as a proxy for those of $\mathbf{R}$.

Unlike the previous section where a strong approximation guarantee of the exact embedding $\mathbf{U}_k$ by an efficiently computed $\tilde{\mathbf{U}}_k$ was a distant and difficult goal to achieve in itself; we will see in this section that the knowledge of the similarity graph not only enables to obtain such strong approximation guarantees, but also enables to control how the error on $\mathbf{U}_k$ transfers as an error on the k-means cost.

To be more precise, recall (1) defining the k-means cost $f(\mathsf{C};\mathsf{X})$ associated to the n points $\mathsf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and a centroid set C. Now, suppose that we have identified a set of n points $\tilde{\mathsf{X}} = (\tilde{\mathbf{x}}_1 | \ldots | \tilde{\mathbf{x}}_n)$ that are meant to approximate the exact spectral embedding X. Moreover, let C^* (resp. $\tilde{\mathsf{C}}^*$) be the optimal set of k centroids minimizing the k-means cost on X (resp. $\tilde{\mathsf{X}}$). We will see that several (not all) approximation methods of this Section achieve an end-to-end approximation guarantee of the form

$$\left| f(\mathsf{C}^*;\mathsf{X})^{1/2} - f(\tilde{\mathsf{C}}^*;\mathsf{X})^{1/2} \right| \leq \varepsilon,$$

for some small ε with -at least- constant probability. Such an end-to-end guarantee is indeed more desirable than a simple guarantee on the distance between $\mathbf{U}_k$ and $\tilde{\mathbf{U}}_k$: it informs us on the approximation quality of the attained clustering.

4.1 Nyström-based methods

The Nyström-based methods discussed in Sec. 3.1 are also applicable here. Let us concentrate on the choice $\mathbf{R} = \mathbf{L}_n$ to illustrate the main ideas. As explained in Sec. 3.1, the tailing k eigenvectors $\mathbf{U}_k$ of $\mathbf{L}_n$, can be interpreted as the top k eigenvectors of the PSD matrix $\mathbf{A} = 2\mathbf{I} - \mathbf{L}_n$. As such, the span of the top-k eigenvectors of $\tilde{\mathbf{A}}_k$, span$(\tilde{\mathbf{U}}_k)$, obtained by running Algorithm 3 on $\mathbf{A}$ should approximate the

span of $\mathbf{U}_k$. Now, how does ones goes from Nyström theorems such as Theorem 2 to error bounds on the k-means cost function?

The first step towards an end-to-end guarantee relies on the following result:

Lemma 1 (see the proof of Theorem 6 in [21]). *Denote by $\tilde{\mathsf{C}}^*$ the optimal centroid set obtained by solving k-means on the rows of $\tilde{\mathbf{U}}_k$. It holds that*

$$\left| f(\mathsf{C}^*;\mathsf{X})^{1/2} - f(\tilde{\mathsf{C}}^*;\mathsf{X})^{1/2} \right| \leq 2\|\mathbf{E}\|_F, \tag{16}$$

where $\mathbf{E} = \mathbf{U}_k\mathbf{U}_k^\top - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^\top$.

This means that the error made by considering the optimal k-means solution based on $\tilde{\mathbf{U}}_k$ (instead of $\mathbf{U}_k$) is controlled by the Frobenius norm of the projector difference $\mathbf{E} = \mathbf{U}_k\mathbf{U}_k^\top - \tilde{\mathbf{U}}_k\tilde{\mathbf{U}}_k^\top$. Furthermore, since[14] $\|\mathbf{E}\|_F \leq \sqrt{2k}\|\mathbf{E}\|_2$ and $\|\mathbf{E}\|_2 = \|\sin(\Theta(\mathbf{U}_k, \tilde{\mathbf{U}}_k))\|_2$, we can apply the Davis-Kahan $\sin\Theta$ perturbation theorem (see for instance Section VII of [16]) and, provided that $\sigma_k - \tilde{\sigma}_{k+1} > 0$, obtain:

$$\|\mathbf{E}\|_F \leq \sqrt{2k}\|\mathbf{E}\|_2 \leq \sqrt{2k}\,\frac{\|\mathbf{A} - \tilde{\mathbf{A}}\|_2}{\sigma_k - \tilde{\sigma}_{k+1}},$$

where $\{\sigma_i\}$ (resp. $\{\tilde{\sigma}_i\}$) are the singular values of $\mathbf{A}$ (resp. $\tilde{\mathbf{A}}$) ordered decreasingly[15]. The final bound is obtained by combining the above with the leverage score sampling bound given by Theorem 2:

Theorem 4. *Let $\tilde{\mathbf{U}}_k$ be the eigenvectors obtained by running Alg. 3 on $\mathbf{A} = 2\mathbf{I} - \mathbf{L}_n$ (with the leverage score sampling scheme for the m samples S of step 1). Denote by $\tilde{\mathsf{C}}^*$ the optimal centroid set obtained by solving k-means on the rows of $\tilde{\mathbf{U}}_k$. Then, for some constant $C > 1$, we have*

$$\left| f(\mathsf{C}^*;\mathsf{X})^{1/2} - f(\tilde{\mathsf{C}}^*;\mathsf{X})^{1/2} \right| \leq 2\frac{\sqrt{2k}}{\sigma_k - \tilde{\sigma}_{k+1}} \left(\sigma_{k+1}(\mathbf{A}) + \frac{Ck\log(k/\delta)}{m} \sum_{j=k+1}^{n} \sigma_j \right)$$

with probability at least $0.8 - 2\delta$.

Examining the above bound one notices that $2\sqrt{2k}\frac{\sigma_{k+1}(\mathbf{A})}{\sigma_k - \tilde{\sigma}_{k+1}}$ is independent of the number of samples. The incompressibility of this error term emanates from $\mathbf{A}$ being (in general) different from its best low-rank approximation. On the other hand, all remaining error terms can be made independent of k and n by setting

$$m = \mathscr{O}\left(k\sqrt{k}\log k \sum_{j=k+1}^{n} \frac{\sigma_j}{\sigma_k - \tilde{\sigma}_{k+1}} \right).$$

[14] Based on three arguments: (i) for any two matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ of rank r_1 and r_2 it holds that $\mathrm{rank}(\mathbf{M}_1 + \mathbf{M}_2) \leq r_1 + r_2$, (ii) for any matrix $\mathbf{M}$ or rank r, $\|\mathbf{M}\|_F \leq \sqrt{r}\|\mathbf{M}\|_2$, and (iii) both $\mathbf{U}_k$ and $\tilde{\mathbf{U}}_k$ are of rank k.

[15] Note that, in our setting, $\mathbf{A} = 2\mathbf{I} - \mathbf{L}_n$ and $\sigma_k = 2 - \lambda_k$.

This end-to-end guarantee is not satisfactory for several reasons. First of all, it relies on the assumption $\sigma_k > \tilde{\sigma}_{k+1}$, which is not necessarily true. Moreover, the Davis-Kahan theorem could in theory guarantee $\|\mathbf{E}\|_2 \leq \|\mathbf{A}_k - \tilde{\mathbf{A}}_k\|_2/\sigma_k$ and $\|\mathbf{E}\|_2 \leq \|\mathbf{A} - \tilde{\mathbf{A}}_k\|_2/\sigma_k$, which are stronger than the bound depending on $\|\mathbf{A} - \tilde{\mathbf{A}}\|_2$ that we used. Unfortunately, Nyström approximation theorems do not give controls on $\|\mathbf{A}_k - \tilde{\mathbf{A}}_k\|_2$ nor on $\|\mathbf{A} - \tilde{\mathbf{A}}_k\|_2$, impeding tighter end-to-end bounds.

4.2 Graph coarsening

Inspired by the algebraic multi-grid, researchers realized early on that a natural way to accelerate spectral clustering is by graph coarsening [61, 69, 38]. Here, instead of solving the clustering problem directly on G, one may first reduce it to a coarser graph G_c consisting of $m \ll n$ nodes using a multi-level graph coarsening procedure. The expensive eigen-decomposition computation is done at a lower cost on the representative matrix of the small graph and the final spectral embedding is obtained by inexpensively lifting and refining the result.

In the notation of [92], coarsening involves a sequence of $c+1$ graphs

$$G = G_0 = (V_0, E_0, \mathbf{W}_0) \quad G_1 = (V_1, E_1, \mathbf{W}_1) \quad \cdots \quad G_c = (V_c, E_c, \mathbf{W}_c) \tag{17}$$

of decreasing size $n = n_0 > n_1 > \cdots > n_c = m$, where each vertex of G_ℓ represents one of more vertices of $G_{\ell-1}$. To express coarsening in algebraic form, we suppose that $\mathbf{L}(G_0) = \mathbf{L}$ is the combinatorial Laplacian associated with G. We then obtain $\mathbf{L}(G_c)$ by applying the following repeatedly

$$\mathbf{L}(G_\ell) = \mathbf{P}_\ell^{\mp} \mathbf{L}(G_{\ell-1}) \mathbf{P}_\ell^{+}, \tag{18}$$

where $\mathbf{P}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ is a matrix with more columns than rows, $\ell = 1, 2, \ldots, c$ is the level of the reduction and symbol $\mp$ denotes the transposed pseudoinverse. An eigenvector $\tilde{\mathbf{u}} \in \mathbb{R}^m$ of $\mathbf{L}(G_c)$ is lifted back to $\mathbb{R}^n$ by backwards recursion

$$\tilde{\mathbf{u}}_{\ell-1} = \mathbf{P}_\ell \tilde{\mathbf{u}}_\ell,$$

where $\tilde{\mathbf{u}}_c = \tilde{\mathbf{u}}$.

Matrices $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_c$ are determined by the transformation performed at each level. Specifically, one should define for each level a surjective map $\varphi_\ell : V_{\ell-1} \to V_\ell$ between the original vertex set $V_{\ell-1}$ and the smaller vertex set V_ℓ. We refer to the set of vertices $V_{\ell-1}^{(r)} \subseteq V_{\ell-1}$ mapped onto the same vertex v'_r of V_ℓ as a *contraction set*:

$$V_{\ell-1}^{(r)} = \{v \in V_{\ell-1} : \varphi_\ell(v) = v'_r\}$$

It is easy to deduce from the above that contraction sets induce a partitioning of $V_{\ell-1}$ into n_ℓ subgraphs, each corresponding to a single vertex of V_ℓ.

Then, for any $v'_r \in V_\ell$ and $v_i \in V_{\ell-1}$, matrices $\mathbf{P}_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $\mathbf{P}_\ell^+ \in \mathbb{R}^{n_{\ell-1} \times n_\ell}$ are given by:

$$\mathbf{P}_\ell(r,i) = \begin{cases} \frac{1}{|V_{\ell-1}^{(r)}|} & \text{if } v_i \in V_{\ell-1}^{(r)} \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \mathbf{P}_\ell^+(i,r) = \begin{cases} 1 & \text{if } v_i \in V_{\ell-1}^{(r)} \\ 0 & \text{otherwise.} \end{cases}$$

The preceding construction is the only one that guarantees that every $\mathbf{L}(G_\ell)$ will be the combinatorial Laplacian associated with G_ℓ [90].

Note that from a computational perspective the reduction is very efficient and can be carried out in linear time: each coarsening level entails multiplication by a sparse matrix, meaning that $\mathscr{O}(e)$ and $\mathscr{O}(n)$ operations suffice, respectively, to coarsen $\mathbf{L}$ and lift any vector (such as the eigenvectors of $\mathbf{L}(G_c)$) from $\mathbb{R}^m$ back to $\mathbb{R}^n$.

4.2.1 Coarsening for spectral clustering

Using coarsening effectively boils down to determining for each ℓ how to partition $G_{\ell-1}$ into n_ℓ contraction sets $V_\ell^{(1)}, \ldots, V_\ell^{(n_\ell)}$, such that, after lifting, the first k eigenvectors $\tilde{\mathbf{U}}_k$ of $\mathbf{L}(G_c)$ approximate the spectral embedding $\mathbf{U}_k$ derived from $\mathbf{L}$. Alternatively, one may also solve the k-means problem in the small dimension and only lift the resulting cluster assignments [38]. This scheme is computationally superior but we will not discuss it here as it does not come with any guarantees.

Perhaps the most *simple* (and common) method of forming contraction sets is by the *heavy edge matching heuristic*—originally developed in the multi-grid literature and first considered for graph partitioning in [69]. This method is derived based on the intuition that, the larger the weight of an edge, the less likely it will be that the vertices it connects will reside in different clusters. We should therefore aim to contract pairs of vertices connected by a heavy edge (i.e., of large weight) first. Let us consider this case further. By focusing on edges, we basically constrain ourselves by enforcing that every contraction set $V_\ell^{(r)}$ contains either two nodes connected by an edge, or a single node, signifying that said node is chosen to remain as is in the coarser graph. As such, we can reformulate the problem of selecting contraction sets at each level as that of selecting the largest number of edges (to attain the largest reduction), while also striving to make the cumulative sum of selected edge weights as large as possible (giving preference to heavy edges). This is exactly the *maximum weight matching problem*, which can be approximated in linear time [44].

A plethora of numerical evidence motivates the use of matching-based coarsening methods, such as the heavy-edge heuristic, for accelerating spectral clustering [69, 38, 115]. From a theoretical perspective, the approximation quality of matching-based methods was characterized in [92]. Therein, the matching was constructed in the following randomized manner:

Algorithm 5. Randomized edge contraction (one level) [92]
Input. A graph $G = (\mathsf{V}, \mathsf{E})$

1. Associate with each $e_{ij} \in \mathsf{E}$ a probability $p_{ij} > 0$.
2. While $|E| > 0$:
   a. Draw a sample e_{ij} from E with probability $\propto p_{ij}$.
   b. Remove from E both e_{ij} as well as all edges sharing a common endpoint with it.
   c. Construct contraction set (v_i, v_j).

Output: Contraction sets

The following approximation result is known:

Theorem 5 (Corollary 5.1 in [92]). *Consider a graph with bounded degrees $d_i \ll n$ and $\lambda_k \leq \min_{e_{ij} \in E} \left\{ \frac{d_i + d_j}{2} \right\}$. Suppose that the graph is coarsened by Algorithm 5, using a* heavy-edge potential *such that $p_{ij} \propto w_{ij}$. For sufficiently large n, a single level, and $\delta > 0$,*

$$\left| f(\mathsf{C}^*; \mathsf{X})^{1/2} - f(\tilde{\mathsf{C}}^*; \mathsf{X})^{1/2} \right| = \mathscr{O}\left(\sqrt{\frac{1 - \frac{m}{n}}{\delta} \frac{\sum_{\ell=2}^{k} \lambda_\ell}{\lambda_{k+1} - \lambda_k}} \right)$$

with probability at least $1 - \delta$. Above, $\tilde{\mathsf{C}}^*$ is the optimal k-means solution when using the lifted eigenvectors of $\mathbf{L}_c$ as a spectral embedding.

We deduce that coarsening works better when the spectral clustering problem is easy (as quantified by the weighted gap $\sum_{\ell=2}^{k} \lambda_\ell / (\lambda_{k+1} - \lambda_k)$) and the achieved error is linear on the reduction ratio $1 - m/n$.

There also exist more advanced techniques for selecting contraction sets that come with stronger guarantees w.r.t. the attained reduction and quality of approximation, but feature running time that is not smaller than that of spectral clustering [90]. In particular, these work also with the normalized Laplacian and can be used to achieve multi-level reduction. Roughly, their strategy is to identify and contract sets $\mathsf{S} \subset V$ for which $\mathbf{x}(i) \approx \mathbf{x}(j)$ for all vectors $\mathbf{x} \in \mathbf{U}_k$ and $v_i, v_j \in \mathsf{S}$. This strategy ensures that the best partitionings of G are preserved by coarsening. We will not expand on these methods here as they do not aim to improve the running time of spectral clustering.

4.3 Other approaches

In the following, we present two additional approaches for approximately computing spectral embeddings. The former can be interpreted as a sampling-based method (but in a different manner than the techniques discussed so far), whereas the latter

is only vaguely linked to sampling. Nevertheless, we find that both techniques are very interesting and merit a brief discussion.

4.3.1 Spectral sparsification

This approach is best suited for cases when the input of spectral clustering is directly a graph[16]. Differently from the methods discussed earlier, here the aim is to identify a Laplacian matrix $\tilde{\mathbf{L}}$ of the same size as $\mathbf{L}$ but with fewer entries. Additionally, it should be ensured that

$$\frac{1}{1+\varepsilon}\mathbf{x}^\top \mathbf{L}\mathbf{x} \leq \mathbf{x}^\top \tilde{\mathbf{L}}\mathbf{x} \leq (1+\varepsilon)\mathbf{x}^\top \mathbf{L}\mathbf{x} \quad \text{for all} \quad \mathbf{x} \in \mathbb{R}^n \tag{19}$$

for some small constant $\varepsilon > 0$ [125]. Most fast algorithms for spectral sparsification entail sampling $\mathcal{O}(n \log n)$ edges from the total edges present in the graph. Different sampling schemes are possible [124, 72], but the most popular ones entail sampling edges with replacement based on their effective resistance. It should be noted that though computing all effective resistances exactly can be computationally prohibitive, the effective resistance of edges can be approximated in nearly linear-time on the number of edges based on a Johnson-Lindenstrauss argument [124].

There are different ways to use sparsification in order to accelerate spectral clustering. The most direct one is to exploit the fact that the eigenvalues $\tilde{\lambda}_k$ and eigenvectors $\tilde{\mathbf{U}}_k$ of $\tilde{\mathbf{L}}$ approximate, respectively, the eigenvalues and eigenvectors of $\mathbf{L}$ up to multiplicative error. This yields the same flavor of guarantees as in graph coarsening and ensures that the computational complexity of the partial eigen-decomposition will decrease when $e = \omega(n \log n)$. A variation of this idea was considered in [142], though the latter did not provide a complete error and complexity analysis. Alternative approaches are also possible. We refer the interested reader to [136] for a rigorous argument that invokes a Laplacian solver.

Despite these exciting developments, we should mention that the overwhelming majority of graph sparsification algorithms remain in the realm of theory. That is, we are currently not aware of any practical and competitive implementation and thus retain a measure of skepticism with regards to their utility in the setting of spectral clustering.

4.3.2 Random eigenspace projection

There also exists approaches that do not explicitly rely on sampling. The key starting point here is that, with regards to spectral clustering, one does not need the eigenvectors exactly—any rotation of $\mathbf{U}_k$ suffices (indeed, k-means is an algorithm based on distances and rotations conserve distances). Even more generally, consider

[16] When one starts from a set of points, it is preferable to sparsify the graph by retaining a constant number of nearest neighbors for each point. The resulting nearest neighbor graph has already $O(n)$ edges, which is the smallest possible.

$\tilde{\mathbf{U}}_k \in \mathbb{R}^{n\times m}$ with $m \geq k$ and denote:

$$\varepsilon = \min_{\mathbf{Q}\in\mathscr{Q}} \|\mathbf{U}_k\mathbf{I}_{k\times m}\mathbf{Q} - \tilde{\mathbf{U}}_k\|_F,$$

where $\mathscr{Q}$ is the space of $m\times m$ unitary matrices and $\mathbf{I}_{k\times m}$ consists of the first k rows of an $m\times m$ identity matrix.

The following lemma (which is a generalization of Lemma 1) shows how ε can be used to provide control on the k-means error:

Lemma 2 (Lemma 3.1 in [95]). *Let $\tilde{\mathsf{C}}^*$ be the optimal solution of the k-means problem on $\tilde{\mathbf{U}}_k$. It holds that*[17]

$$\left| f(\mathsf{C}^*;\mathsf{X})^{1/2} - f(\tilde{\mathsf{C}}^*;\mathsf{X})^{1/2} \right| \leq 2\varepsilon. \tag{20}$$

There exists (at least) two approaches to efficiently compute $\tilde{\mathbf{U}}_k$ while controlling ε [21, 133] (see also related work in [58]). We will consider here a simple variant of the one proposed in [133] and further analyzed in [95]: Let $\mathbf{G} \in \mathbb{R}^{n\times m}$ be a random Gaussian matrix with centered i.i.d. entries, each having variance $\frac{1}{m}$. Further, suppose that we project $\mathbf{G}$ onto span$(\mathbf{U}_k)$ by multiplying each one of its columns by an ideal projector $\mathbf{P}_k$ defined as

$$\mathbf{P}_k = \mathbf{U}\begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix}\mathbf{U}^\top. \tag{21}$$

Theorem 6 ([133, 95]). *Let $\tilde{\mathsf{C}}^*$ be the optimal solution of the k-means problem on the rows of $\tilde{\mathbf{U}}_k = \mathbf{P}_k\mathbf{G}$. For every $\delta \geq 0$, one has*

$$\left| f(\mathsf{C}^*;\mathsf{X})^{1/2} - f(\tilde{\mathsf{C}}^*;\mathsf{X})^{1/2} \right| \leq 2\sqrt{\frac{k}{m}}\,(\sqrt{k}+\delta), \tag{22}$$

with probability at least $1-\exp(-\delta^2/2)$.

This result means that for an ideal projector $\mathbf{P}_k$, dimension $m = \mathscr{O}(k^2)$ suffices to guarantee good approximation (since the error becomes independent of k and n)! A similar argument also holds when the entries of $\mathbf{G}$, instead of being Gaussian,

[17] **A remark on the definition of the k-means cost**. Note that, here, the lines $\tilde{\mathbf{X}}$ of $\tilde{\mathbf{U}}_k$ are points in dimension $m \geq k$, such that the optimal centroid set $\tilde{\mathsf{C}}^*$ minimizing the k-means cost on $\tilde{\mathbf{X}}$ is a set of k points in dimension $m \geq k$. In this context, the notation $f(\tilde{\mathsf{C}}^*;\mathsf{X})$ is ill-defined: it is a sum of distances between points that do not necessarily have the same dimension. We abuse notations and give the following meaning to $f(\tilde{\mathsf{C}};\mathsf{X})$. First, consider the matrix form of the k-means cost, as used in the proofs of Lemmas 1 and 2: $f(\mathsf{C};\mathsf{X}) = \|\mathbf{X} - \mathbf{C}\mathbf{C}^\top\mathbf{X}\|_F^2$, where $\mathbf{X} = (\mathbf{x}_1|\ldots|\mathbf{x}_n)^\top \in \mathbb{R}^{n\times k}$ and $\mathbf{C} \in \mathbb{R}^{n\times k}$ is the (weighted) cluster indicator matrix associated to the Voronoi tesselation of X given C: $\mathbf{C}_{i\ell} = 1/\sqrt{s_\ell}$ if data point i belongs to cluster ℓ, and 0 otherwise; where s_ℓ is the size of cluster ℓ. Now, let $\tilde{\mathbf{C}} \in \mathbb{R}^{n\times k}$ be the cluster indicator matrix associated to the Voronoi tesselation of $\tilde{\mathsf{X}}$ given $\tilde{\mathsf{C}}$. One writes: $f(\tilde{\mathsf{C}};\mathsf{X}) = \|\mathbf{X} - \tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top\mathbf{X}\|_F^2$.

are selected i.i.d. from $\{-\sqrt{3},0,+\sqrt{3}\}$ with probabilities $\{1/6,2/3,1/6\}$, respectively [1]. This construction has the benefit of being sparser and, moreover, is reminiscent of sampling. It should be noted that in [133], $m=\mathscr{O}(\log n)$ was deemed enough because one only wanted that the distance between two rows of $\mathbf{U}_k$ was approximated by the distance between the same two rows of $\tilde{\mathbf{U}}_k$. There was in fact no end-to-end control on the k-means error.

The discussion so far assumed that $\mathbf{P}_k$ is an ideal projector onto span$(\mathbf{U}_k)$. However, in practice one does not have access to this projector as we are in fact in the process of *computing* $\mathbf{U}_k$. One may choose to approximate the action of $\mathbf{P}_k$ by an application of a matrix function h on the representative matrix $\mathbf{R}$ [132, 111]. Assuming a point λ_* in the interval $[\lambda_k,\lambda_{k+1})$ is known, one may select a polynomial [123] or rational function [65, 91] that approximates the ideal low-pass response, i.e., $h(\lambda)=1$ if $\lambda\leq\lambda_*$ and $h(\lambda)=0$, otherwise. The approximated projector $\tilde{\mathbf{P}}_k=h(\mathbf{R})$ can be designed to be very close to $\mathbf{P}_k$. For instance, in the case of Chebychev polynomials of order c using the arguments of [80, Lemma 1] it is easy to prove that w.h.p. using $h(\mathbf{R})$ instead of $\mathbf{P}_k$ does not add more than $\mathscr{O}(c^{-c}\sqrt{n})$ error in (20). Furthermore, the operation $\tilde{\mathbf{P}}_k\mathbf{G}$ can conveniently be computed in $\mathscr{O}(mce)$ time via this polynomial approximation.

The last ingredient needed for this approximation is λ_*, i.e., a point in the interval $[\lambda_k,\lambda_{k+1})$. Finding efficiently a valid λ_* is difficult. An option is to rely on eigencount techniques [39, 105, 109] to find one in[18] $\mathscr{O}(ck^2(\log n)(e+n\log(\lambda_n/(\lambda_{k+1}-\lambda_k))))$ time, which features similar complexity as the Lanczos method (see discussion in Sec. 2.3). Another option is to content oneself with values of λ_* known only to be close to the interval $[\lambda_k,\lambda_{k+1})$, but thereby loosing the end-to-end guarantee [133].

5 Sampling in the spectral feature space

Having computed (or approximated) the spectral embedding $\mathsf{X}=(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n)$, what remains is to solve the k-means problem on X, in order to obtain k centroids together with the associated k classes obtained after Voronoi tessellation.

The usual heuristic used to solve the k-means problem, namely the Lloyd-Max algorithm, is already very efficient as it runs in $\mathscr{O}(nk^2t)$ time as seen in Sec. 2.3. Nonetheless, this section considers ways to accelerate k-means even further. In the

[18] *Proof sketch*: Given $\lambda\in(0,\lambda_n]$, denote by j the largest integer such that $\lambda_j\leq\lambda$ and by $\mathbf{P}_j$ the orthogonal projector on $\mathbf{U}_j$. Let $\mathbf{G}\in\mathbb{R}^{n\times m'}$ be a random Gaussian matrix with centered i.i.d. entries, each having variance $\frac{1}{m'}$ and denote by $\hat{j}=\|\mathbf{P}_j\mathbf{G}\|_F$. Relying on Theorem 4.1 (and the following discussion in Section 4.2) of [109] with $\mathsf{E}_\lambda=\mathbf{0}$, one has with prob. at least $1-\varepsilon$ that $(1-\delta)j\leq\hat{j}\leq(1+\delta)j$ for all $j=1,\ldots,n$ provided $m'\geq\frac{1}{\delta^2}\log\frac{n}{\varepsilon}$. Setting $\delta=1/(2k+3)$, gives w.h.p. that $\frac{2k+2}{2k+3}j\leq\hat{j}\leq\frac{2k+4}{2k+3}j$ for all $j=1,\ldots,n$ provided $m'\geq\mathscr{O}(k^2\log n)$. This implies that w.h.p. for every $j\leq k+1$ it must be that round$(\hat{j})=j$, whereas when $j>k+1$ we have round$(\hat{j})>k+1$. Note that round$(\hat{j})$ is the closest integer to $\hat{j}$. By dichotomy on $\lambda\in(0,\lambda_n]$, one thus finds a λ_* in time $\mathscr{O}(ck^2(\log n)(e+n\log(\lambda_n/(\lambda_{k+1}-\lambda_k))))$.

following, we classify the relevant literature in five categories and point towards representative references for each case. In our effort to provide depth (as well as breadth) of presentation, the rest of the section details only methods that belong to the first and last categories.

- **Exact acceleration of Lloyd-Max.** There exists exact accelerated Lloyd-Max algorithms, some of them based on avoiding unnecessary distance calculations using the triangular inequality [59, 101], or on optimized data organization [67], and others concentrating on clever initializations [6, 104]. The latter concern sampling and are discussed in Sec. 5.1.
- **Approximate acceleration of Lloyd-Max.** Approximately accelerating the Lloyd-Max algorithm has also received attention for instance via approximate nearest neighbour methods [108], via cluster closure [140], or via applying Lloyd-Max hierarchically (in the large k context) [103]. An approach involving sampling is introduced in [119]: it is based on mini-batches sampled uniformly at random from $\mathbf{X}$. We will not discuss further this method as it does not come with guarantees on the cost of the obtained solution.
- **Methods involving sampling in the Fourier domain.** There are a few sampling-based heuristics to solve the k-means problem, that are different from the Lloyd-Max algorithm. For instance, the work in [70] proposes to sample in the frequency domain to obtain a sketch from which one may recover the centroids with an orthogonal matching pursuit algorithm specifically tailored to this kind of compressive learning task [55]. These methods are reminiscent of the random Fourier features sketching approach introduced in Sec. 3.2. We will not discuss them further.
- **Methods involving sampling features.** Similarly to ideas presented in Sec. 3.4.2 but here specific to the k-means setting, some works reduce the ambient dimension of the vectors, either by selecting a limited number of features [20, 3], or by embedding all points in a lower dimension using random projections [22, 33, 93]. The tightest results to day are a $(1+\varepsilon)$ multiplicative error on the k-means cost f either by randomly selecting $\mathscr{O}(\varepsilon^{-2}k\log k)$ features or by projecting them on a random space of dimension $\mathscr{O}(\varepsilon^{-2}\log(k/\varepsilon))$ (sublinear in k!). The sampling result is useless in the spectral clustering setting as the ambient dimension of the spectral features is already k. The projection result could in principle be applied in our setting, to reduce the cost of the k-means step to $\mathscr{O}(tnk\log k)$. We will nevertheless not discuss it further in this chapter.
- **Methods involving sampling points.** Finally, the last group of existing methods are the ones that solve k-means on a subset S of X, before lifting back the result on the whole dataset. We classify such methods in two categories. In Sec. 5.2, we detail methods that are graph-agnostic, meaning that they apply to *any* k-means problem; and in Sec. 5.3 we discuss methods that explicitly rely on the fact that the features $\mathbf{x}$ were in fact obtained from a known graph. We argue that the latter are better suited to the spectral clustering problem.

5.1 Clever initialization of the Lloyd-Max algorithm

Recall that the k-means objective on X is to find the k centroids $\mathsf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ that minimize the following cost function:

$$f(\mathsf{C};\mathsf{X}) = \sum_{\mathbf{x}\in\mathsf{X}} \min_{\mathbf{c}\in\mathsf{C}} \|\mathbf{x}-\mathbf{c}\|_2^2. \tag{23}$$

and that $\mathsf{C}^* = \arg\min_{\mathsf{C}} f(\mathsf{C};\mathsf{X})$ is the optimal solution attaining cost $f^* = f(\mathsf{C}^*;\mathsf{X})$. Recall also that the Lloyd-Max algorithm (see Algorithm 2) converges to a local minimum of f, that we will denote by C_{lm}, for which the cost function equals $f_{\text{lm}} = f(\mathsf{C}_{\text{lm}};\mathsf{X})$. It is crucial to note that the initialization of centroids C_{ini} in the first step of the Lloyd-Max algorithm, which usually is done by randomly selecting k points in X, is what determines the distance $|f^* - f_{\text{lm}}|$ to the optimal. As such, significant efforts have been devoted to smartly selecting C_{ini} by various sampling schemes.

As usual, we also face here the usual trade-off between sampling *effectively* and *efficiently*. The fastest sampling method is of course uniformly at random, but it does not come with any guarantee on the quality of the local minimum C_{lm} it leads to. An alternative sampling scheme, called k-means++ initialization, is based on the following more general $\mathbf{D}^2$-`sampling` algorithm.

Algorithm 6: $\mathbf{D}^2$-`sampling`.
Input. X, m the number of required samples

1. Initialize B with any $\mathbf{x}$ chosen uniformly at random from X.
2. Iterate the following steps until B contains m elements:
   a. Compute $d_i = \min_{b\in\mathsf{B}} \|\mathbf{x}_i - b\|_2^2$.
   b. Define the probability of sampling $\mathbf{x}_i$ as $d_i / \sum_i d_i$.
   c. Sample $\mathbf{x}_{\text{new}}$ from this probability distribution and add it to B.

Output: B a sample set of size m.

k-means++ initialization boils down to running Alg. 6 with $m = k$ to obtain a set of k initial centroids. Importantly, when the Lloyd-Max heuristic is run with this initialization, the following guarantee holds:

Theorem 7 ([6]). *For any set of data points, the cost f_{lm} obtained after Lloyd-Max initialized with k-means++ is controlled in expectation:* $\mathbb{E}(f_{lm}) \leq 8(\log k + 2) f^*$.

In terms of computation cost, $\mathbf{D}^2$-`sampling` with $m = k$ runs in $\mathscr{O}(nkd)$, that is, $\mathscr{O}(nk^2)$ in our setting of a spectral embedding X in dimension k. This work inspired other initialization techniques that come with similar guarantees and are in some cases faster [12, 10]. The interested reader is referred to the review [27] for further analyses on the initialization of k-means.

5.2 Graph agnostic sampling methods: coresets

The rest of Sec. 5, considers sampling methods that fall in the following framework: (i) sample a subset S of X, (ii) solve k-means on S, (iii) lift the result back on the whole dataset X. Sec. 5.2, focuses on coresets: general sampling methods designed for any arbitrary k-means problem; whereas in Sec. 5.3, we will take into account the specific nature of the spectral features encountered in spectral clustering algorithms.

5.2.1 Definition

Let $\mathsf{S} \subset \mathsf{X}$ be a subset of X of size m. To each element $s \in \mathsf{S}$ associate a weight $\omega(s) \in \mathbb{R}^+$. Define the estimated k-means cost associated to the weighted set S as:

$$\tilde{f}(\mathsf{C};\mathsf{S}) = \sum_{s\in\mathsf{S}} \omega(s) \min_{\mathbf{c}\in\mathsf{C}} \|s-\mathbf{c}\|_2^2. \tag{24}$$

Definition 2 (Coreset). Let $\varepsilon \in (0, \frac{1}{2})$. The weighted subset S is a ε-coreset for f on X if, *for every set* C, the estimated cost is equal to the exact cost up to a relative error:

$$\forall \mathsf{C} \qquad \left| \frac{\tilde{f}(\mathsf{C};\mathsf{S})}{f(\mathsf{C};\mathsf{X})} - 1 \right| \leq \varepsilon. \tag{25}$$

This is the so-called "strong" coreset definition[19], as the ε-approximation is required for all C. The great interest of finding a coreset S comes from the following fact. Writing $\tilde{\mathsf{C}}^*$ the set minimizing $\tilde{f}$, the following inequalities hold

$$(1-\varepsilon) f(\mathsf{C}^*;\mathsf{X}) \leq (1-\varepsilon) f(\tilde{\mathsf{C}}^*;\mathsf{X}) \leq \tilde{f}(\tilde{\mathsf{C}}^*;\mathsf{S}) \leq \tilde{f}(\mathsf{C}^*;\mathsf{S}) \leq (1+\varepsilon) f(\mathsf{C}^*;\mathsf{X}).$$

The first inequality comes from the fact that C^* is optimal for f, the second and last inequality are justified by the coreset property of S, and the third inequality comes from the optimality of $\tilde{\mathsf{C}}^*$ for $\tilde{f}$. This has two consequences:

1. First of all, since $\varepsilon < \frac{1}{2}$:

$$f(\mathsf{C}^*;\mathsf{X}) \leq f(\tilde{\mathsf{C}}^*;\mathsf{X}) \leq (1+4\varepsilon) f(\mathsf{C}^*;\mathsf{X}),$$

   meaning that $\tilde{\mathsf{C}}^*$ is a well controlled approximation of C^* with a multiplicative error on the cost.
2. Estimating $\tilde{\mathsf{C}}^*$ can be done using the Lloyd-Max algorithm on the weighted subset[20] S, thus reducing the computation time from $\mathscr{O}(nk^2)$ to $\mathscr{O}(mk^2)$.

[19] A weaker version of this definition exists in the literature where the ε-approximation is only required for C^*.

[20] Generalizing Algorithm 2 to a weighted set is straightforward: in step 2b, instead of computing the center of each cluster, compute the weighted barycenter.

Coreset methods for k-means thus follow the general procedure:

Algorithm 7. Coresets to avoid k-means on X.
Input. X, sampling set size m, and number of clusters $k \leq m$.

1. Compute a weighted coreset S of size m using a `coreset-sampling` algorithm.
2. Run the Lloyd-Max algorithm on the weighted set S to obtain the set of k centroids $\tilde{\mathsf{C}}$.
3. "Closest-centroid lifting": classify the whole dataset X based on the Voronoi cells of $\tilde{\mathsf{C}}$.

Output: A set of k centroids $\mathsf{C} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$.

Coreset methods compete with one another on essentially two levels: the coreset size m should be as small as possible in order to decrease the time of Lloyd-Max on S, and the coreset itself should be sampled efficiently (at least faster than running k-means on the whole dataset!), which turns out in fact to be a strong requirement. The reader interested in an overview of coreset construction techniques is referred to the recent review [99], as well as chapter 2 of this book.

5.2.2 An instance of coreset-sampling algorithm

We focus on a particular coreset algorithm proposed in [11] that builds upon results developed in [79, 45]: it is not state-of-the-art in terms of coreset size, but has the advantage of being easy to implement and fast enough to compute. It reads:

Algorithm 8: a coreset sampling algorithm [11].
Input. X, m the number of required samples, t an iteration number

1. Repeat t times: draw a set of size k using $\mathbf{D}^2$`-sampling`. Out of the t sets obtained, keep the set B that minimizes $f(\mathsf{B};\mathsf{X})$.
2. $\alpha \leftarrow 16(\log k + 2)$
3. For each $b_\ell \in \mathsf{B}$, define B_ℓ the set of points in X in the Voronoi cell of b_ℓ
4. Set $\phi = \frac{1}{n} f(\mathsf{B};\mathsf{X})$.
5. For each $b_\ell \in \mathsf{B}$ and each $\mathbf{x} \in \mathsf{B}_\ell$, define

$$s(\mathbf{x}) = \frac{\alpha}{\phi}\|\mathbf{x} - b_\ell\|_2^2 + \frac{2\alpha}{\phi|\mathsf{B}_\ell|}\sum_{\mathbf{x}' \in \mathsf{B}_\ell}\|\mathbf{x}' - b_\ell\|_2^2 + \frac{4n}{|\mathsf{B}_\ell|}$$

6. Define the probability of sampling $\mathbf{x}_i$ as $\mathbf{p}_i = s(\mathbf{x}_i)/\sum_{\mathbf{x}} s(\mathbf{x})$
7. $\mathsf{S} \leftarrow$ sample m nodes i.i.d. with replacement from $\mathbf{p}$ and associate to each sample s the weight $\omega_s = \frac{1}{m\mathbf{p}_s}$.

Output: A weighted set S of size m.

Theorem 2.5 of [11] states:

Theorem 8. *Let $\varepsilon \in (0,1/4)$ and $\delta \in (0,1)$. Let S be the output of Alg. 8 with $t = \mathscr{O}(\log 1/\delta)$. Then, with probability at least $1-\delta$, S is a ε-coreset provided that:*

$$m = \Omega\left(\frac{k^4 \log k + k^2 \log 1/\delta}{\varepsilon^2}\right). \tag{26}$$

The computation cost of running this coreset sampling algorithm, running Lloyd-Max on the weighted coreset, and lifting the result back to X is dominated, when[21] $n \gg k$, by step 1 of Alg. 6 and thus sums up to $\mathscr{O}(nk^2 \log 1/\delta)$.

Remark 1. The coreset sampling strategy underlying this algorithm relies on the concept of sensitivity [79]. Many other constructions of coresets for k-means are possible [99] with better theoretical bounds then (26). Nevertheless, as the coreset line of research has been essentially theoretical, practical implementations of coreset-sampling algorithms are scarce. A notable exception is for instance the work in [49] that proposes a scalable hybrid coreset-inspired algorithm for k-means. Other exceptions are the sampling algorithms based on the farthest-first procedure, a variant of $\mathbf{D}^2$-`sampling` that chooses each new sample to be $\arg\max_i d_i$ instead of drawing it according to a probability proportional to d_i. Once S of size m is drawn, then $\forall s \in \mathsf{S}$, each weight ω_s is set to be the cardinal of the Voronoi cell associated to s. Authors in [113] show that such weighted sets computed by different variants of the farthest-first algorithm are ε-coresets, but for values of ε that can be very large. For a fixed ε, the number of samples necessary to have a ε-coreset with this type of algorithm is unknown (see also chapter 3 of this book).

5.3 Graph-based sampling methods

The methods discussed so far in this section are graph agnostic both for the sampling procedure *and* the lifting: they do not take into account that, in spectral clustering, X are in fact spectral features of a known graph.

A recent line of work [133, 95, 52, 53] based on Graph Signal Processing (GSP) [122, 118] leverages this additional knowledge for accelerating both the sampling and the lifting steps. For the purpose of the following discussion, define by $\mathbf{z}_\ell \in \mathbb{R}^n$ the ground truth indicator vector of cluster ℓ, i.e., $\mathbf{z}_\ell(i) = 1$ if node v_i is in cluster ℓ, and 0 otherwise. The goal of spectral clustering is, of course, to recover $\{\mathbf{z}_\ell\}_{\ell=1,\dots,k}$.

[21] To be precise, the statement holds if $n \geq \mathscr{O}\left(\frac{k^4}{\varepsilon^2}\frac{\log k}{\log 1/\delta}\right)$

Broadly, GSP-based methods can be summarized in the following general methodology [133]:

Algorithm 9. Graph-based sampling strategies to avoid k-means on X.
Input. X, m the number of required samples, k the number of desired clusters

1. Choose the random sampling strategy. Either:
   a. **uniform (i.i.d.)** Draw m i.i.d. samples uniformly.
   b. **leverage score (i.i.d.)** Compute $\forall \mathbf{x}_i, p_i^* = \|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2^2/k$. Draw m i.i.d. samples from $\mathbf{p}^*$. (optional:) set the weight of each sample s to $1/p_s^*$.
   c. **DPP** Sample a few times independently from a DPP with kernel $\mathbf{K}_k = \mathbf{U}_k\mathbf{U}_k^\top$. (optional:) set the weight of each sample s to $1/\pi_s$.
2. Run the Lloyd-Max algorithm on the (possibly weighted) set S to obtain the k reduced cluster indicator vectors $\mathbf{z}_\ell^r \in \mathbb{R}^m$.
3. Lift each reduced indicator vector $\{\mathbf{z}_\ell^r\}_{\ell=1,\dots,k}$ to the full graph either with
   a. **Least-square** Solve (33) with $\mathbf{y} \leftarrow \mathbf{z}_\ell^r$.
   b. **Tikhonov** Solve (34) with $\mathbf{y} \leftarrow \mathbf{z}_\ell^r$.
   In both cases, $\mathbf{P}_\mathsf{S}$ should be set to $\frac{1}{N}\mathbf{I}_m$ if uniform sampling was chosen, to $\mathrm{diag}(p_{s_1}^*, \dots p_{s_m}^*)$ if leverage score sampling was chosen, and to $\mathrm{diag}(\pi_{s_1}, \dots \pi_{s_m})$ if DPP sampling was chosen.
4. Assign each node j to the cluster ℓ for which $\hat{\mathbf{z}}_\ell(j)/\|\hat{\mathbf{z}}_\ell\|_2$ is maximal.

Output: A partition of X in k clusters

To aid understanding, let us start by a high-level description of Algorithm 9. The indicator vectors $\mathbf{z}_\ell$ are interpreted as graph signals that are (approximately) bandlimited on the similarity graph G (see Sec. 5.3.1 for a precise definition). As such, there is no need to measure these indicator vectors everywhere: one can take advantage of generalized Shannon-type sampling theorems to select the set S of m nodes to measure (step 1). Then k-means is performed on S to obtain the indicator vectors $\mathbf{z}_\ell^r \in \mathbb{R}^m$ on the sample set S (step 2). These reduced indicator vectors are interpreted as noisy measurements of the global cluster indicator vectors $\mathbf{z}_\ell$ on S. The solutions $\mathbf{z}_\ell^r$ are lifted back to X as $\hat{\mathbf{z}}_\ell$ via solving an inverse problem taking into account the bandlimitedness assumption or via label-propagation on the graph structure reminiscent of semi-supervised learning techniques (step 3). As the lifted solutions $\hat{\mathbf{z}}_\ell$ do not have a binary structure as true indicator vectors should have, an additional assignment step is necessary: assign each node j to the class ℓ for which $\frac{\hat{\mathbf{z}}_\ell(j)}{\|\hat{\mathbf{z}}_\ell\|_2}$ is maximal (step 4).

The rest of this section is devoted to the discussion of the three sampling schemes as well as the two lifting procedures considered in this framework. To this end, we will first introduce a few graph signal processing (GSP) concepts in Sec. 5.3.1 before discussing in Sec. 5.3.2 several examples of graph sampling theorems appropriate to the spectral clustering context.

5.3.1 A brief introduction to graph signal processing (GSP)

Denote by $\mathbf{U} = (\mathbf{u}_1|\dots|\mathbf{u}_n) \in \mathbb{R}^{n\times n}$ the matrix of orthonormal eigenvectors of the Laplacian matrix $\mathbf{L}$, with the columns ordered according to their associated sorted eigenvalues: $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$. In the GSP literature [122, 118], these eigenvectors are interpreted as graph Fourier modes for two main reasons:

- By analogy to the ring graph, whose Laplacian matrix is exactly the (symmetric) double derivative discrete operator, and is thus diagonal in the basis formed by the classical 1D discrete Fourier modes.
- A variational argument stemming from the Dirichlet form can be exploited to express eigenvectors $\mathbf{u}_i$ of $\mathbf{L}$ as the basis of minimal variation $\mathbf{x}^\top\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{ij}\mathbf{W}_{ij}[\mathbf{x}(i)-\mathbf{x}(j)]^2$ on G and eigenvalues λ_i as a sum of local variations of $\mathbf{u}_i$, i.e., a generalized graph frequency.

A *graph signal* $\mathbf{z} \in \mathbb{R}^n$ is a signal that is defined on the nodes of a graph: its i-th element is associated to node v_i. Given the previous discussion, the graph Fourier transform of $\mathbf{z}$, denoted by $\tilde{\mathbf{z}}$, is its projection on the graph Fourier modes: $\tilde{\mathbf{z}} = \mathbf{U}^\top\mathbf{z} \in \mathbb{R}^n$. The notion of graph filtering naturally follows as a multiplication in the Fourier domain. More precisely, define a real-valued filter function $h(\lambda)$ defined on $[0,\lambda_n]$. The signal $\mathbf{x}$ filtered by h reads $\mathbf{U}h(\Lambda)\mathbf{U}^\top\mathbf{x}$, where we use the convention $h(\Lambda) = \mathrm{diag}(h(\lambda_1),h(\lambda_2),\dots,h(\lambda_n))$. In the following, we will use the following notation for graph filter operators:

$$h(\mathbf{L}) = \mathbf{U}h(\Lambda)\mathbf{U}^\top. \tag{27}$$

For more details on the graph Fourier transform and filtering, their various definitions and interpretations, we refer the reader to [131].

Of interest for the discussion in this chapter, one may define bandlimited graph signals as linear combinations of the first few low-frequency Fourier modes. Writing $\mathbf{U}_k = (\mathbf{u}_1|\dots|\mathbf{u}_k) \in \mathbb{R}^{n\times k}$, we have the formal definition:

Definition 3 (k-bandlimited graph signal). A graph signal $\mathbf{z} \in \mathbb{R}^n$ is k-bandlimited if $\mathbf{z} \in \mathrm{span}(\mathbf{U}_k)$, i.e., $\exists\ \alpha \in \mathbb{R}^k$ such that $\mathbf{z} = \mathbf{U}_k\alpha$.

To grasp why the notion of k-bandlimitedness lends itself natually to the approximation of spectral clustering, consider momentarily a graph with k disconnected components and $\mathbf{z}_\ell \in \mathbb{R}^n$ the indicator vector of cluster ℓ. It is a well known property of the (combinatorial) Laplacian that $\{\mathbf{z}_\ell\}_{\ell=1,\dots,k}$ form a set of orthogonal eigenvectors of $\mathbf{L}$ associated to eigenvalue 0: that is, the set of indicator vectors $\{\mathbf{z}_\ell\}_{\ell=1,\dots,k}$ form a basis of $\mathrm{span}(\mathbf{U}_k)$. Understanding arbitrary graphs with block structure as a perturbation of the ideal disconnected component case, the indicator vectors $\{\mathbf{z}_\ell\}_{\ell=1,\dots,k}$ of the blocks should live close to $\mathrm{span}(\mathbf{U}_k)$ (in the sense that the difference between any $\mathbf{z}_\ell$ and its orthogonal projection onto $\mathrm{span}(\mathbf{U}_k)$ is small). This in turn implies that every $\mathbf{z}_\ell$ should be approximately k-bandlimited.

As we will see next, the bandlimitedness assumption is very useful because it enables us to make use of generalized versions of Nyquist-Shannon sampling theorems, taking into account the graph.

5.3.2 Graph sampling theorems

The periodic sampling paradigm of the Shannon theorem for classical bandlimited signals does not apply to graphs without specific regular structure. In fact, a number of sampling schemes have been recently developed with the purpose of generalizing sampling theorems to graph signals [117, 29, 134, 109] (see [88] for a review of existing schemes).

Let us introduce some notations. Sampling entails selecting a set $\mathsf{S} = (s_1, \ldots, s_m)$ of m nodes of the graph. To each possible sampling set, we associate a measurement matrix $\mathbf{M} = (\delta_{s_1}|\delta_{s_2}|\ldots|\delta_{s_m})^\top \in \mathbb{R}^{m\times n}$ where $\delta_{s_i}(j) = 1$ if $j = s_i$, and 0 otherwise. Now, consider a k-bandlimited signal $\mathbf{z} \in \text{span}(\mathbf{U}_k)$. The measurement of $\mathbf{z}$ on S reads:

$$\mathbf{y} = \mathbf{M}\mathbf{z} + \mathbf{n} \in \mathbb{R}^m, \tag{28}$$

where $\mathbf{n}$ models measurement noise. The sampling question boils down to: how should we sample S such that one can recover any bandlimited $\mathbf{z}$ given its measurement $\mathbf{y}$? There are three important components to this question: (i) how many samples m do we allow ourselves ($m = k$ being the strict theoretical minimum)? (ii) how much does it cost to sample? (iii) how do we in practice recover $\mathbf{z}$ from $\mathbf{y}$ and how much does that inversion cost?

There are a series of works that propose greedy algorithms to find the "best" set S of minimal size $m = k$ that embed all k-bandlimited signals (see for instance [129] and references therein). These algorithms cost $\mathscr{O}(nk^4)$ and are thus not competitive in our setting[22]. Moreover, in our case, we don't really need to be that strict on the number of samples and can allow more than k samples. A better choice is to use random graph sampling techniques. In the following we consider two types of independent sampling (uniform and leverage-score sampling) as well as a more involved method based on determinantal point processes.

Independent sampling. In the i.i.d. setting, one defines a discrete probability distribution $\mathbf{p} \in \mathbb{R}^n$ over the node set V. The sampling set S is then generated by drawing m nodes independently with replacement from $\mathbf{p}$. At each draw, the probability to sample node v_i is denoted by p_i. We have $\sum_i p_i = 1$ and write $\mathbf{P} = \text{diag}(\mathbf{p})$. Under this sampling scheme, the following Restricted Isometry Property holds for the associated measurement matrix $\mathbf{M}$ [109].

Theorem 9. *For any $\delta, \varepsilon \in (0,1)$, with probability at least $1-\delta$:*

$$(1-\varepsilon)\|\mathbf{z}\|_2^2 \leq \frac{1}{m}\|\mathbf{M}\mathbf{P}^{-1/2}\mathbf{z}\|_2^2 \leq (1+\varepsilon)\|\mathbf{z}\|_2^2 \tag{29}$$

for all $\mathbf{z} \in span(\mathbf{U}_k)$ provided that

$$m \geq \frac{3}{\varepsilon^2}(\nu_{\mathbf{p}}^k)^2 \log\frac{2k}{\delta} \tag{30}$$

[22] It takes longer to find a good sample than to run k-means on the whole dataset!

where $\nu_{\mathbf{p}}^k$ is the so-called graph weighted coherence:

$$\nu_{\mathbf{p}}^k = \max_i \left\{ p_i^{-1/2} \|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2 \right\}. \tag{31}$$

This property is important as it says, in a nutshell, that any two different bandlimited signals will be identifiable post-sampling provided the number of samples is large enough. The concept of large enough depends on $(\nu_{\mathbf{p}}^k)^2$: a measure of the interplay between the probability distribution and the norms of the rows of $\mathbf{U}_k$. In the uniform i.i.d. case since $p_i = 1/n$, one has $(\nu_{\mathbf{p}}^k)^2 = n \max_i \|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2^2$, which stays under control only for very regular graphs, but can be close to n in irregular graphs such as the star graph. The good news is that there exists an optimal sampling distribution (in the sense that it minimizes the right-hand side of inequality (30)) that adapts to the graph at hand:

$$p_i^* = \frac{\|\mathbf{U}_k^\top \boldsymbol{\delta}_i\|_2^2}{k} \tag{32}$$

In fact, in this case, $(\nu_{\mathbf{p}^*}^k)^2$ matches its lower bound k and the necessary number of samples m to embed all bandlimited signals drops to $\mathscr{O}(k \log k)$. The distribution $\mathbf{p}^*$ is also referred to by the name "leverage scores" in parts of the literature (see discussion in Sec. 3.1) [43]. As such, i.i.d. sampling under $\mathbf{p}^*$ will be referred to as leverage score sampling.

Now, for lifting, there are several options.

- If one uses the unbiased decoder

$$\hat{\mathbf{z}} = \operatorname*{arg\,min}_{\mathbf{w} \in \mathrm{span}(\mathbf{U}_k)} \|\mathbf{P}_{\mathsf{S}}^{-1/2}(\mathbf{M}\mathbf{w} - \mathbf{y})\|_2^2 \tag{33}$$

  where $\mathbf{P}_{\mathsf{S}}^{-1/2} = \mathbf{M}\mathbf{P}^{-1/2}\mathbf{M}^\top$, then the following reconstruction result holds [109]:

  Theorem 10. *Let S be the i.i.d. nodes sampled with distribution $\mathbf{p}$ and $\mathbf{M}$ be the associated sampling matrix. Let $\varepsilon, \delta \in (0,1)$ and suppose that m satisfies* (30)*. With probability at least $1-\delta$, for all $\mathbf{z} \in span(\mathbf{U}_k)$ and $\mathbf{n} \in \mathbb{R}^m$, the solution $\hat{\mathbf{z}}$ of* (33) *verifies:*

$$\|\hat{\mathbf{z}} - \mathbf{z}\|_2 \leq \frac{2}{\sqrt{m(1-\varepsilon)}} \|\mathbf{P}_{\mathsf{S}}^{-1/2}\mathbf{n}\|_2.$$

  This means that a noiseless measurement of a k-bandlimited signal yields a perfect reconstruction. Also, this quantifies how increasing m reduces the error of reconstruction due to a noisy measurement. Note that this error may be large if there is a significant measurement noise on a node that has a low probability of being sampled. However, by definition, this is not likely to happen.
- One can also use a label-propagation decoder reminiscent to semi-supervised learning techniques [15, 28]:

$$\hat{\mathbf{z}} = \underset{\mathbf{w}\in\mathbb{R}^n}{\arg\min} \|\mathbf{P}_{\mathsf{S}}^{-1/2}(\mathbf{M}\mathbf{w}-\mathbf{y})\|_2^2 + \gamma\, \mathbf{w}^\top g(\mathbf{L})\mathbf{w}, \tag{34}$$

where γ is a regularization parameter, $g(\mathbf{L})$ a graph filter operator as in (27) with $g(\lambda)$ a non-decreasing function. As g is non-decreasing, the regularization term of (34) penalizes high frequency solutions, that is, solutions that are not smooth along paths of the graph. Theorems controlling the error of reconstruction are more involved and we refer the reader to Section 3.3 of [109] for details.

- Other decoders [14, 106] are in principle possible, replacing for instance the ℓ_2 Laplacian-based regularization $\mathbf{w}^\top g(\mathbf{L})\mathbf{w}$ by ℓ_1-regularizers $\|\nabla\mathbf{w}\|_1$, but they come with an increased computation cost, lesser guarantees, and have not been used for spectral clustering: we will thus not detail them further.

Let us discuss the computation costs of the previous sampling and lifting techniques. In terms of sampling time, uniform sampling is obviously the most efficient and runs in $\mathscr{O}(k)$. Leverage score sampling is dominated by the computation of the optimal sampling distribution $\mathbf{p}^*$ of (32), which takes $\mathscr{O}(nk)$ time[23]. In terms of lifting time, solving the decoder of (33) costs $\mathscr{O}(nk+mk^2)$. Solving the decoder of (34) costs $\mathscr{O}(et)$ via the conjugate gradient method, where t is the iteration number of the gradient solver (usually around 10 or 20 iterations suffice to obtain good accuracy when $g(\mathbf{L}) = \mathbf{L}$).

This discussion calls for a few remarks. First of all, these theorems are valid if we suppose that $\mathbf{z}$ is exactly k-bandlimited, which is in fact only an approximation if we consider $\mathbf{z}$ to be the ground truth indicator vectors of the k clusters to detect in the spectral clustering context. In this case, we can always decompose $\mathbf{z}$ as the sum of its orthogonal projection onto $\mathrm{span}(\mathbf{U}_k)$ and its complement β: $\mathbf{z} = \mathbf{U}_k\mathbf{U}_k^\top\mathbf{z} + \beta$. (28) becomes $\mathbf{y} = \mathbf{M}\mathbf{U}_k\mathbf{U}_k^\top\mathbf{z} + \mathbf{n}$ where $\mathbf{n}$ now represents the sum of a measurement noise and the distance-to-model term $\mathbf{M}\beta$. The aforementioned theorems can then be applied to $\mathbf{U}_k\mathbf{U}_k^\top\mathbf{z}$. Moreover, note that the decoder of (34) is not only faster than the other ones in general, it also does not constrain the solution $\hat{\mathbf{z}}$ to be exactly in $\mathrm{span}(\mathbf{U}_k)$, which is in fact desirable in the spectral clustering context: we thus advocate for the decoder of (34).

DPP sampling. Determinantal Point Processes are a class of correlated random sampling strategies that strive to increase "diversity" in the samples, based on a kernel $\mathbf{K}$ expliciting the similarity between variables. DPP sampling has been used successfully in a number of applications in machine learning (see for instance [74]).

Denote by $[n]$ the set of all subsets of $\{1,2,\ldots,n\}$. An element of $[n]$ could be the empty set, all elements of $\{1,2,\ldots,n\}$ or anything in between. DPPs are defined as follows:

Definition 4 (Determinantal Point Process [74]). Consider a point process, i.e., a process that randomly draws an element $\mathsf{S} \in [n]$. It is determinantal if, $\forall\, \mathsf{A} \subseteq \mathsf{S}$,

[23] Note that the complexity is different from the leverage score computation of the Nyström techniques of Sections 3.1 and 4.1 because, here, we suppose $\mathbf{U}_k$ known whereas $\mathbf{U}_k$ was not known in the previous sections. With $\mathbf{U}_k$ known, computing the leverage scores only entails computing the normalized energy of each line of $\mathbf{U}_k$.

$$\mathbb{P}(\mathsf{A} \subseteq \mathsf{S}) = \det(\mathbf{K}_\mathsf{A}),$$

where $\mathbf{K} \in \mathbb{R}^{n \times n}$, a semi-definite positive matrix $0 \preceq \mathbf{K} \preceq 1$, is called the marginal kernel; and $\mathbf{K}_\mathsf{A}$ is the restriction of $\mathbf{K}$ to the rows and columns indexed by the elements of A.

The marginal probability π_i of sampling an element i is thus $\mathbf{K}_{ii}$. Consider the following projective kernel:

$$\mathbf{K}_k = \mathbf{U}_k \mathbf{U}_k^\top. \tag{35}$$

One can show that DPP samples from such projective kernels are necessarily of size k. After measuring the k-bandlimited signal $\mathbf{z}$ on a DPP sample S, one has the choice between the same decoders as before (see Eqs. (33) and (34)). For instance:

Theorem 11. *For all $\mathbf{z} \in span(\mathbf{U}_k)$, let $\mathbf{y} = \mathbf{M}\mathbf{z} + \mathbf{n} \in \mathbb{R}^k$ be a noisy measurement of $\mathbf{z}$ on a DPP sample obtained from kernel $\mathbf{K}_k$. The decoder of* (33) *with $\mathbf{P} = diag(\pi_1, \ldots, \pi_n)$ necessarily enables perfect reconstruction up to the noise level. Indeed, one obtains:*

$$\|\hat{\mathbf{z}} - \mathbf{z}\|_2 \leq \frac{1}{\sqrt{\lambda_{min}\left(\mathbf{U}_k^\top \mathbf{M}^\top \mathbf{P}_\mathsf{S}^{-1} \mathbf{M} \mathbf{U}_k\right)}} \|\mathbf{P}_\mathsf{S}^{-1/2} \mathbf{n}\|_2. \tag{36}$$

Proof. The proof is only partly in [129] and we complete it here. Let us write $\mathbf{z} = \mathbf{U}_k \alpha$. Solving (33) entails computing $\hat{\alpha} \in \mathbb{R}^k$ s.t. $\|\mathbf{P}_\mathsf{S}^{-1/2}(\mathbf{M}\mathbf{U}_k\hat{\alpha} - \mathbf{y})\|_2^2$ is minimal. Setting the derivative w.r.t. $\hat{\alpha}$ to 0, and replacing $\mathbf{y}$ by $\mathbf{M}\mathbf{U}_k\alpha + \mathbf{n}$, yields:

$$\mathbf{U}_k^\top \mathbf{M}^\top \mathbf{P}_\mathsf{S}^{-1} \mathbf{M} \mathbf{U}_k \hat{\alpha} = \mathbf{U}_k^\top \mathbf{M}^\top \mathbf{P}_\mathsf{S}^{-1} \mathbf{M} \mathbf{U}_k \alpha + \mathbf{U}_k^\top \mathbf{M}^\top \mathbf{P}_\mathsf{S}^{-1} \mathbf{n}.$$

Recall that S is a sample from a DPP with kernel $\mathbf{K}_k$: $\det(\mathbf{M}\mathbf{U}_k\mathbf{U}_k^\top\mathbf{M}^\top)$ is thus strictly superior to 0, which implies that $\mathbf{M}\mathbf{U}_k$ is invertible, which in turn implies that $\hat{\alpha} = \alpha + (\mathbf{M}\mathbf{U}_k)^{-1}\mathbf{n}$. One thus has $\|\hat{\mathbf{z}} - \mathbf{z}\|_2 = \|\hat{\alpha} - \alpha\|_2 = \left\|(\mathbf{M}\mathbf{U}_k)^{-1}\mathbf{n}\right\|_2 = \left\|(\mathbf{P}_\mathsf{S}^{-1/2}\mathbf{M}\mathbf{U}_k)^{-1}\mathbf{P}_\mathsf{S}^{-1/2}\mathbf{n}\right\|_2$. Using the matrix 2-norm to bound this error yields

$$\|\hat{\mathbf{z}} - \mathbf{z}\|_2 \leq \sqrt{\lambda_{\max}\left[\left(\mathbf{U}_k^\top \mathbf{M}^\top \mathbf{P}_\mathsf{S}^{-1} \mathbf{M} \mathbf{U}_k\right)^{-1}\right]} \|\mathbf{P}_\mathsf{S}^{-1/2}\mathbf{n}\|_2,$$

as claimed.

Several comments are in order:

- The particular choice of kernel $\mathbf{K}_k = \mathbf{U}_k\mathbf{U}_k^\top$ implies that the marginal probability of sampling node v_i, $\pi_i = \|\mathbf{U}_k^\top \delta_i\|_2^2$, is proportional to the leverage scores p_i^*. The major difference between the i.i.d. leverage score approach and the DPP approach comes from the negative correlations induced by the DPP. In fact, the probability of jointly sampling nodes v_i and v_j in the DPP case is $\pi_i\pi_j - \mathbf{K}_{ij}^2 = \pi_i\pi_j - (\delta_i^\top \mathbf{U}_k \mathbf{U}_k^\top \delta_j)^2$. The interaction term $(\delta_i^\top \mathbf{U}_k \mathbf{U}_k^\top \delta_j)^2$ will

be typically large if v_i and v_j are in the same cluster, and small if not. In other words, different from the i.i.d. leverage score case where each new sample is drawn regardless of the past, the DPP procedure avoids to sample nodes containing redundant information.

- Whereas the leverage score approach only guarantees a RIP with high probability after $\mathcal{O}(k \log k)$ samples, the DPP approach has a stronger deterministic guarantee: it enables perfect invertibility (up to the noise level) after precisely $m = k$ samples. The reconstruction guarantee of (36) is nevertheless not satisfactory: even corrected by the marginal probabilities $\mathbf{P}_{\mathsf{S}}$, the matrix $\mathbf{U}_k^\top \mathbf{M}^\top \mathbf{P}_{\mathsf{S}}^{-1} \mathbf{M} \mathbf{U}_k$ can still have a very small $\lambda_{\min}$, such that reconstruction may be quite sensitive to noise. Improving this control is still an open problem. In practice, sampling independently 2 or 3 times from a DPP with kernel $\mathbf{K}_k$, creates a set S of size $2k$ or $3k$ that is naturally more robust to noise.
- Whereas independent sampling is straightforward, sampling from a DPP with arbitrary kernel costs in general $\mathcal{O}(n^3)$ (see Alg. 1 of [74] due to [62]). Thankfully, in the case of a projective kernel such as $\mathbf{K}_k$, one can sample a set in $\mathcal{O}(nk^2)$ based on Alg. 3 of [130].

6 Perspectives

Almost two decades have passed since spectral clustering was first introduced. Since then, a large body of work has attempted to accelerate its computation. *So, has the problem been satisfactorily addressed?* – or, despite all these works, is there still room for improvement and further research?

To answer, we must first define what "satisfactorily addressed" would entail. As we have seen, the prototypical spectral clustering algorithm can be divided in three sub-problems: the similarity graph computation runs in $\mathcal{O}(dn^2)$; the spectral embedding computation runs in $\mathcal{O}(t(ek + nk^2))$ using an Arnoldi algorithm with t implicit restarts and assuming that e is the number of edges; and the k-means step runs in $\mathcal{O}(tnk^2)$, with t now being a bound on the number of iterations of the Loyd-Max algorithm. Our criteria for evaluating an approximation algorithm aiming to accelerate one (or more) of these sub-problems are two-fold:

- We ask that the approximation algorithm's computation cost is effectively lighter than the cost of the sub-problem(s) it is supposed to accelerate! The ultimate achievement is an order-of-magnitude improvement w.r.t. n (or e), d and/or k, especially when the complexity has no hidden constants (i.e., the algorithm is practically implementable). When such a gain is not possible, a gain on the constants of the theoretical cost is also considered worthwhile.
- The algorithm should come with convincing guarantees in terms of the quality of the found solution. Heuristics or partially motivated methods do not cut it. We require that, under *mild assumptions*, the proposed solution is *provably close to* the exact solution. Let us clarify two aspects of this statement further:

- It is difficult to concretely classify assumptions as mild, but a useful rule of thump is checking whether the theoretical results are meaningful for the significant majority of cases where spectral clustering would be used.
- The control of the approximation error comes in different flavors, that we detail here from the tightest to the loosest. The best possible error control in our context is a control over the clustering solution itself, via error measures such as the misclustering rate. This is unfortunately unrealistic in many cases. An excellent alternative is the multiplicative error –considered as the gold standard in approximation theory– over the k-means cost[24], ensuring that the cost of the approximation is not larger than $1+\varepsilon$ times the cost of the exact solution. Next comes the additive error over the cost: ensuring that the cost difference between approximated and exact solutions is not larger than ε. All these error controls are referred to as end-to-end controls, and represent the limit of what we will consider a *satisfactory* error control.

Reviewing the literature, we were surprised to discover that there are rarely any algorithms meeting fully the proposed criteria: a faster algorithm with end-to-end control over the approximation error under mild assumptions. Let us revisit one by one the different approaches presented in Sections 3, 4 and 5 examining them in light of our criteria for success. In each category of approximation algorithms, we order the methods according to the power of their error control.

Sampling methods in the original feature space [Sec. 3].

- *Representative points methods* as described in [147, 63] allow for an end-to-end control on the miss-clustering rate ρ, which is unfortunately quite loose. The constants involved in Theorem 3 are in fact undefined –thus potentially large–, which is problematic knowing that ρ is by definition between 0 an 1. Also, the theorem's assumptions include independence of the ε_i, which is hard to justify in practice. On the other hand, the computation gain of such methods is very appealing.
- *Feature projection methods*, where the dimension d of the original feature space is reduced to a dimension $d' \leq d$ based on Johnson-Lindenstrauss arguments, come with a multiplicative error control on the pairwise distances in the original feature space, thus providing a control on the obtained kernel matrix. The impact of this initial approximation on the final clustering result has not been studied.
- *Nyström-inspired methods* [48, 85, 19, 97] can be very efficient in practice especially because they do not need to build the graph. However, precisely because they do not build the graph, these methods cannot exactly perform two key parts of the prototypical spectral clustering algorithm: the k-NN sparsification and the exact degree computation. The partial knowledge and sparsity of the kernel matrix also makes sampling difficult, as using leverage scores sampling is not possible anymore, whereas most other sampling schemes do not work very well with

[24] A control in terms of the k-means cost is usually considered as k-means is the last step of spectral clustering. Nevertheless, recalling the minimum cut perspective of Sec. 2.2, the control should arguably be in terms of `rcut` or `ncut` costs.

sparse matrices and come with weak guarantees. To the extent of our knowledge, there is also no convincing mathematical argument proving that using these methods will yield a clustering that is of similar quality to that produced by the exact spectral clustering algorithm.

- *Sketching methods* such as the Random Fourier Features [110] is yet another way of obtaining a pointwise multiplicative $(1+\varepsilon)$ error on the Gaussian kernel computation. RFF enable to compute a provably good low-rank approximation of the kernel. They nevertheless suffer from the same problems as Nyström-based techniques: without building the graph, sparsification and degree-normalization are uncontrolled. In addition, the guarantees on the low-rank approximation of the kernel do not transfer easily to guarantees of approximation of the spectral embedding $\mathbf{U}_k$.
- *Approximate nearest neighbours methods* are numerous and varied, and come with different levels of guarantees. Practical implementations of algorithms, however, often set aside theoretical guarantees to gain on efficiency and performances; and comparisons are usually done on benchmarks rather than on theoretical performances. In the best of cases, there is a control on how close the obtained nearest neighbour similarity graph is to the exact one; but with no end-to-end control.

Spectral embedding approximation methods [Sec. 4].

- *Random eigenspace projection* is a very fast method and has been rigorously analyzed [95, 133, 105, 111]. It is true that a successful application depends on obtaining a good estimate of the k-th eigenvalue, which is very hard when the k-th eigenvalue gap is relatively small. Nevertheless, our current understanding of spectral clustering suggests that it only works well when the gap is (at least) moderately large. As such, though there are definitely situations in which random eigenspace projection will fail to provide an acceleration, these correspond to cases where one should not be using spectral clustering in the first place. The same argumentation can also be used in defense of all methods that come with mild gap assumptions (see coarsening, and spectral sparsification).
- *Simple coarsening methods*, such as the heavy-edge matching heuristic [69], have nearly-linear complexity, seem to work well in practice, and are accompanied by end-to-end additive error control [92]. Nevertheless, the current analysis of these heuristics only accounts for very moderate reductions ($m \geq n/2$) and thus does not fully prove their success: in real implementations coarsening is used in a multi-level fashion resulting to a drastic decrease in the graph size ($m = \mathcal{O}(n/2^c)$ for c levels), whereas the end-to-end control only works for a single level.
- *Advanced coarsening methods*, such as local variation methods [90], come with much stronger guarantees that allow for drastic size reduction and acceleration. Yet, thus far, all evidence suggests that finding a good enough coarsening is computationally as hard as solving the spectral clustering problem itself. As a consequence, it is at this point unclear whether these methods can be used to accelerate spectral clustering.

- *Spectral sparsification techniques* come with excellent guarantees in theory: one may prove that a spectral sparsifier can be computed in nearly-linear time and, moreover, the latter's spectrum will be provably close to the original one. Yet, we have reasons to doubt their practicality. Indeed, current algorithms are very complex, feature impractically large constants, and are only relevant for dense graphs. In addition, spectral sparsifiers, by definition, approximate the entire spectrum of a graph Laplacian matrix. However, spectral clustering only needs an approximation of a tiny fraction of the spectrum. From that perspective, it is reasonable to conclude that without modification current approaches will not yield the best possible approximation.
- *Nyström-approximation* applied directly to the Laplacian matrix is a good option, especially when combined with leverage score sampling. Nevertheless, an end-to-end error control has only been partially derived and is not yet satisfactory.

Sampling to accelerate the k-means step [Sec. 5].

- *Exact methods to accelerate the Lloyd-Max algorithm*, may they be via avoiding unnecessary distance calculations or via a careful initialization are always useful and should be taken into account.
- *Coresets* come with the strongest guarantees: the minimum number of samples to guarantee a $(1+\varepsilon)$ multiplicative error on the cost function has been well studied. Nevertheless, practical coreset sampling methods are scarce; and in the best cases, the sampling cost is of the same order of the Lloyd-Max running cost itself.
- *Graph-based sampling* comes with strong guarantees, but not over the k-means cost: on the reconstruction error based on a k-bandlimited model that is only an approximation in practice. Moreover, we interpret the reduced indicator vectors $\mathbf{z}_\ell^r$ obtained by running Lloyd-Max on the sampled set S as (possibly noisy) measurements of $\mathbf{z}_\ell$ on S. This interpretation currently lacks solid theoretical ground and impedes an end-to-end control of this approximation method. Nevertheless, the leverage-score based sampling allows for a reduction in order of magnitude of the Lloyd-Max running cost.
- *Other methods* to accelerate k-means are not always appropriate to the spectral clustering context. Spectral feature dimension reduction is unnecessary in our context where $d = k$, sketching methods appropriate to distributed cases where n is very large are not appropriate neither as the spectral features need centralized data to be computed in any case.

In practice. The attentive reader will have remarked that, unsurprisingly, the tighter the error control, the more expensive the computation, and vice versa. Also, although we have put here an emphasis on the approximation error controls, it should not undermine the fact that methods from the whole spectrum are in practice useful, depending on the situation at hand, and specifically on the range of values of n, d and k. In very large d situations, a first step of random projection (or feature selection if some features are suspected to be too noisy) should be considered. Then, in situations where the exact computation of the proximity graph is too expensive, one may resort either to sketching methods or to Nyström-type methods to decrease the

cost from quadratic to linear in n, and directly obtain an approximation of the spectral embedding without any explicit graph construction. These methods, however, do not take into account a sparsity constraint on the proximity graph and are usually rough on the degree correction they make.

The role of the sparsity constraint is not well understood theoretically, but seems to be important in some practical cases [138]. In such instances, a better option is to use approximate nearest neighbours methods to create a sparse similarity graph, and work from there. In extremely large data, say $n \geq 10^8$, the only workable methods are the representative-based, with, if possible, a first k-means (or compressive k-means [70]) to reduce n to m ; or, in last resort, a uniform random sampling strategy.

In situations where one has to deal with such a large similarity graph that Arnoldi iterations are too expensive to compute the spectral embedding (either a graph created via approximate nearest neighbours, or if the original data *is* a graph), projection methods such as in [133, 21], coarsening methods such as in [90], or Nyström-based methods are different possibilities.

Sampling methods to accelerate the last k-means step may seem to be a theoretical endeavour given that the Lloyd-Max algorithm is already very efficient. Due to the quadratic term in k, it is nevertheless in practice useful when k grows large. In this situation, hierarchical k-means [103] is a nice option. Coresets, because they are so stringent on the error control, have a hard time actually accelerating k-means, unless hybrid coreset-inspired methods are envisioned [49]. Finally, graph-based methods, because they take into account that spectral features are in fact derived from the graph itself, enable significant acceleration and are well-suited to the spectral clustering context.

Future research. Different directions of research could be envisioned to improve the state-of-the-art:

- For Nyström-inspired methods in the context of Sec. 3 (directly applied on the original data) as well as the other methods based on computing a low-rank approximation of the kernel matrix $\mathbf{K}$, further work is needed to control both the sparsification and the degree correction, in order to bridge the gap between a provably good low-rank approximation of $\mathbf{K}$ to a provably good low-rank approximation of $\mathbf{R}$.
- For Nyström methods in the context of Sec. 4 (applied on a known or well-approximated similarity graph), it would be interesting to extend Theorem 2 (for instance) to a control over $\|\mathbf{A}_k - \tilde{\mathbf{A}}_k\|$ instead of $\|\mathbf{A} - \tilde{\mathbf{A}}\|$. This would enable a tighter use of Davis-Kahan's perturbation theorem in the discussion of Sec. 4.1 and, *in fine*, a better end-to-end guarantee.
- Projection-based methods of Sec. 4.3.2 currently necessitate to compute a value λ_* known to be in the interval $[\lambda_k, \lambda_{k+1})$. The algorithm used to do so is based on eigencount techniques that turn out to require as much computation time as the Lanczos iterations needed to compute $\mathbf{U}_k$ exactly. One should relax this constraint to obtain end-to-end guarantees as a function of the distance between a coarsely estimated λ_* and the target interval.

- The derivation and analysis of randomized multi-level coarsening schemes with end-to-end guarantees is very much an open problem. We suspect that, by utilizing spectrum-dependent sampling-schemes akin to leverage-scores one should be able to achieve results superior to heavy edge-matching in nearly linear time.
- There is an interesting similarity between coreset techniques and the graph-based sampling strategies discussed in Sec. 5.3 and it would be interesting to investigate this link theoretically, maybe paving the way to coresets for spectral clustering?

Finally, accelerating the prototypical spectral algorithm depicted in Algorithm 1 should not be the sole objective of researchers in this field. Indeed, taking the graph cut point-of-view of Sec. 2.2, Algorithm 1 makes three insufficiently motivated choices: (i) To begin with, the sparsification step in Algorithm 1 is not well understood. Apart from the fact that it is always computationally more convenient to work with a sparse similarity graph then a dense one, the precise effect of sparsification on the clustering performance has not been analyzed. (ii) As mentioned in Sec. 2.2.3, the relaxation employed by spectral relaxation is not unique. Why should we focus our attention on this one versus another? See for instance [24, 112] for recent alternative options. (iii) Finally, the use of k-means on the spectral features is not yet fully justified. Most of the end-to-end guarantees presented here compare the k-means cost of the exact solution to the k-means cost of the approximate solution. Given that the very use of k-means is not theoretically grounded, this choice of guarantee is debatable. Other options, such as a control over the `rcut` or `ncut` objectives are possible (as in [96]) and should be further investigated.

Acknowledgments

This work was partially supported by the CNRS PEPS I3A (Project RW4SPEC) and the Swiss National Science Foundation (Project DLEG, grant PZ00P2 179981).

References

1. D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.
2. H. T. Ali and R. Couillet. Improved spectral community detection in large heterogeneous networks. page 49.
3. J. Altschuler, A. Bhaskara, G. Fu, V. Mirrokni, A. Rostamizadeh, and M. Zadimoghaddam. Greedy Column Subset Selection: New Bounds and Distributed Algorithms. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2539–2548, New York, New York, USA, June 2016. PMLR.
4. E. Anagnostopoulos, I. Z. Emiris, and I. Psarros. Low-Quality Dimension Reduction and High-Dimensional Approximate Nearest Neighbor. In L. Arge and J. Pach, editors, *31st International Symposium on Computational Geometry (SoCG 2015)*, volume 34 of *Leib-*

niz International Proceedings in Informatics (LIPIcs), pages 436–450, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
5. A. Andoni and P. Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, Oct. 2006.
6. D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.
7. S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *J. ACM*, 45(6):891–923, Nov. 1998.
8. M. Aumüller, E. Bernhardsson, and A. Faithfull. ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms. In C. Beecks, F. Borutta, P. Kröger, and T. Seidl, editors, *Similarity Search and Applications*, pages 34–49, Cham, 2017. Springer International Publishing.
9. Y. Avrithis, I. Z. Emiris, and G. Samaras. High-dimensional approximate nearest neighbor: k-d Generalized Randomized Forests. *arXiv:1603.09596 [cs]*, Mar. 2016. arXiv: 1603.09596.
10. O. Bachem, M. Lucic, H. Hassani, and A. Krause. Fast and Provably Good Seedings for k-Means. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 55–63. Curran Associates, Inc., 2016.
11. O. Bachem, M. Lucic, and A. Krause. Practical Coreset Constructions for Machine Learning. *arXiv:1703.06476 [stat]*, Mar. 2017. arXiv: 1703.06476.
12. B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable K-means++. *Proc. VLDB Endow.*, 5(7):622–633, Mar. 2012.
13. Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*. SIAM, 2000.
14. P. C. Bellec, J. Salmon, and S. Vaiter. A sharp oracle inequality for graph-slope. *Electron. J. Statist.*, 11(2):4851–4870, 2017.
15. Y. Bengio, O. Delalleau, and N. L. Roux. Label Propagation and Quadratic Criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.
16. R. Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 1997.
17. T. D. Bie and N. Cristianini. Fast sdp relaxations of graph cut clustering, transduction, and other combinatorial problems. *Journal of Machine Learning Research*, 7(Jul):1409–1436, 2006.
18. C. Bordenave, M. Lelarge, and L. Massoulié. Non-backtracking Spectrum of Random Graphs: Community Detection and Non-regular Ramanujan Graphs. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1347–1357, Oct. 2015.
19. D. Bouneffouf and I. Birol. Sampling with minimum sum of squared similarities for nystrom-based large scale spectral clustering. In *IJCAI*, pages 2313–2319, 2015.
20. C. Boutsidis, P. Drineas, and M. W. Mahoney. Unsupervised Feature Selection for the k-means Clustering Problem. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 153–161. Curran Associates, Inc., 2009.
21. C. Boutsidis, A. Gittens, and P. Kambadur. Spectral clustering via the power method-provably. In *International Conference on Machine Learning (ICML)*, 2015.
22. C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. Randomized Dimensionality Reduction for k -Means Clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, Feb. 2015.
23. M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In *AISTATS*, 2003.
24. X. Bresson, T. Laurent, D. Uminsky, and J. von Brecht. Multiclass Total Variation Clustering. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1421–1429. Curran Associates, Inc., 2013.

25. D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, page 333, Washington, DC, USA, 2010. ACM Press.
26. D. Calvetti, L. Reichel, and D. C. Sorensen. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2(1):21, 1994.
27. M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200 – 210, 2013.
28. O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.
29. S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic. Discrete Signal Processing on Graphs: Sampling Theory. *CoRR*, abs/1503.05432, 2015.
30. X. Chen and D. Cai. Large Scale Spectral Clustering with Landmark-Based Representation. In *AAAI Conference on Artificial Intelligence*, 2011.
31. R. Chitta, R. Jin, and A. K. Jain. Efficient Kernel Clustering Using Random Fourier Features. In *2012 IEEE 12th International Conference on Data Mining*, pages 161–170, Dec. 2012.
32. F. R. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.
33. M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 163–172, New York, NY, USA, 2015. ACM.
34. L. Dall'Amico, R. Couillet, and N. Tremblay. Optimized Deformed Laplacian for Spectrum-based Community Detection in Sparse Heterogeneous Graphs. *arXiv:1901.09715*, Jan. 2019. arXiv: 1901.09715.
35. M. Dash and P. W. Koot. Feature Selection for Clustering. In L. LIU and M. T. ÖZSU, editors, *Encyclopedia of Database Systems*, pages 1119–1125. Springer US, Boston, MA, 2009.
36. C. Davis. The rotation of eigenvectors by a perturbation. *Journal of Mathematical Analysis and Applications*, 6(2):159–173, 1963.
37. A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1117–1126. Society for Industrial and Applied Mathematics, 2006.
38. I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11), 2007.
39. E. Di Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 2016.
40. W. Dong, C. Moses, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web - WWW '11*, page 577, Hyderabad, India, 2011. ACM Press.
41. P. Drineas, A. M. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in Large Graphs and Matrices. In *SODA*, volume 99, pages 291–299. Citeseer, 1999.
42. P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183, 2006.
43. P. Drineas and M. W. Mahoney. Lectures on randomized numerical linear algebra. *The Mathematics of Data*, 25:1, 2018.
44. R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM (JACM)*, 61(1):1, 2014.
45. D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011.

46. M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.
47. S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.
48. 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, Feb. 2004.
49. G. Frahling and C. Sohler. A fast k-means implementations using coresets. *International Journal of Computational Geometry & Applications*, 18(06):605–625, 2008.
50. A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.
51. C. Fu, C. Xiang, C. Wang, and D. Cai. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *arXiv:1707.00143 [cs]*, July 2017. arXiv: 1707.00143.
52. A. Gadde, A. Anis, and A. Ortega. Active Semi-supervised Learning Using Sampling Theory for Graph Signals. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 492–501, New York, New York, USA, 2014. ACM.
53. A. Gadde, E. E. Gad, S. Avestimehr, and A. Ortega. Active learning for community detection in stochastic block models. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1889–1893, July 2016.
54. A. Gittens and M. W. Mahoney. Revisiting the nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 17(1):3977–4041, 2016.
55. R. Gribonval, G. Blanchard, N. Keriven, and Y. Traonmilin. Compressive Statistical Learning with Random Feature Moments. *arXiv:1706.07180 [cs, math, stat]*, June 2017. arXiv: 1706.07180.
56. S. Guattery and G. L. Miller. On the performance of spectral graph partitioning methods. In *SODA*, volume 95, pages 233–242, 1995.
57. S. Guattery and G. L. Miller. On the quality of spectral separators. *SIAM Journal on Matrix Analysis and Applications*, 19(3):701–719, 1998.
58. N. Halko, P. Martinsson, and J. Tropp. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2):217–288, 2011.
59. G. Hamerly and J. Drake. *Accelerating Lloyd's Algorithm for k-Means Clustering*, pages 41–78. Springer International Publishing, Cham, 2015.
60. X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in neural information processing systems*, pages 507–514, 2006.
61. B. Hendrickson and R. W. Leland. A multi-level algorithm for partitioning graphs. *SC*, 95(28):1–14, 1995.
62. J. B. Hough, M. Krishnapur, Y. Peres, and B. Virág. Determinantal Processes and Independence. *Probability Surveys*, 3(0):206–229, 2006.
63. L. Huang, D. Yan, N. Taft, and M. I. Jordan. Spectral clustering with perturbed data. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 705–712. Curran Associates, Inc., 2009.
64. B. Hunter, T. Strohmer, T. E. Simos, G. Psihoyios, and C. Tsitouras. Compressive Spectral Clustering. pages 1720–1722, Rhodes (Greece), 2010.
65. E. Isufi, A. Loukas, A. Simonetto, and G. Leus. Autoregressive moving average graph filtering. *IEEE Transactions on Signal Processing*, 65(2):274–288, 2017.
66. Y. Kalantidis and Y. Avrithis. Locally Optimized Product Quantization for Approximate Nearest Neighbor Search. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.
67. T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An Efficient k-Means Clustering Algorithm: Analysis and Implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 24:881–892, 2002.
68. D. R. Karger. Minimum cuts in near-linear time. *Journal of the ACM (JACM)*, 47(1):46–76, 2000.

69. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.
70. N. Keriven, N. Tremblay, Y. Traonmilin, and R. Gribonval. Compressive K-means. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373, Mar. 2017.
71. P. Kolev and K. Mehlhorn. A note on spectral clustering. *arXiv preprint arXiv:1509.09188*, 2015.
72. I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving sdd linear systems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 235–244, Oct 2010.
73. F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborová, and P. Zhang. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences*, 110(52):20935–20940, 2013.
74. A. Kulesza and B. Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.
75. A. Kumar and R. Kannan. Clustering with spectral norm and the k-means algorithm. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 299–308. IEEE, 2010.
76. A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 454–462. IEEE, 2004.
77. S. Kumar, M. Mohri, and A. Talwalkar. Sampling techniques for the nystrom method. In *Artificial Intelligence and Statistics*, pages 304–311, 2009.
78. S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the nyström method. *Journal of Machine Learning Research*, 13(Apr):981–1006, 2012.
79. M. Langberg and L. J. Schulman. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 598–607. SIAM, 2010.
80. B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.
81. Q. Le, T. Sarlós, and A. Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, volume 85, 2013.
82. H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press, 2007.
83. J. R. Lee, S. O. Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.
84. J. Lei and A. Rinaldo. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1):215–237, 2015.
85. M. Li, X.-C. Lian, J. Kwok, and B.-L. Lu. Time and space efficient spectral clustering via column sampling. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2297–2304. IEEE Computer Society, 2011.
86. Q. Li, W. Liu, L. Li, and R. Wang. Towards Large Scale Spectral Problems via Diffusion Process. In *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7, Nov. 2017.
87. S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar. 1982.
88. P. Lorenzo, S. Barbarossa, and P. Banelli. Chapter 9 - sampling and recovery of graph signals. In P. M. Djurić and C. Richard, editors, *Cooperative and Graph Signal Processing*, pages 261 – 282. Academic Press, 2018.
89. A. Loukas. How close are the eigenvectors of the sample and actual covariance matrices? In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2228–2237, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

90. A. Loukas. Graph reduction with spectral and cut guarantees. *arXiv preprint arXiv:1808.10650*, 2018.
91. A. Loukas, A. Simonetto, and G. Leus. Distributed autoregressive moving average graph filters. *IEEE Signal Processing Letters*, 22(11):1931–1935, Nov 2015.
92. A. Loukas and P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *Interenational Conference on Machine Learning (ICML)*, 2018.
93. K. Makarychev, Y. Makarychev, and I. Razenshteyn. Performance of Johnson-Lindenstrauss Transform for k-Means and k-Medians Clustering. *arXiv:1811.03195 [cs]*, Nov. 2018. arXiv: 1811.03195.
94. Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *arXiv:1603.09320 [cs]*, page 13, 2016. arXiv: 1603.09320.
95. L. Martin, A. Loukas, and P. Vandergheynst. Fast approximate spectral clustering for dynamic networks. 2018.
96. M. Mohan and C. Monteleoni. Beyond the nystrom approximation: Speeding up spectral clustering using uniform sampling and weighted kernel k-means. In *IJCAI*, 2017.
97. M. Mohan and C. Monteleoni. Exploiting sparsity to improve the accuracy of nyström-based large-scale spectral clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.
98. M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.
99. A. Munteanu and C. Schwiegelshohn. Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms. *KI - Künstliche Intelligenz*, Dec. 2017.
100. C. Musco and C. Musco. Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, pages 3833–3845, 2017.
101. J. Newling and F. Fleuret. Fast k-means with accurate bounds. In *International Conference on Machine Learning*, pages 936–944, 2016.
102. A. Ng, M. Jordan, Y. Weiss, and others. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.
103. D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168, June 2006.
104. R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The Effectiveness of Lloyd-Type Methods for the k-Means Problem. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 165–176, Oct. 2006.
105. J. Paratte and L. Martin. Fast eigenspace approximation using random signals. *arXiv preprint arXiv:1611.00938*, 2016.
106. R. Pena, X. Bresson, and P. Vandergheynst. Source localization on graphs via ℓ_1 recovery and spectral graph theory. In *2016 IEEE 12th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–5, July 2016.
107. R. Peng, H. Sun, and L. Zanetti. Partitioning well-clustered graphs: Spectral clustering works! In *Conference on Learning Theory*, pages 1423–1455, 2015.
108. J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.
109. G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst. Random sampling of bandlimited signals on graphs. *Applied and Computational Harmonic Analysis*, 2016.
110. A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.
111. D. Ramasamy and U. Madhow. Compressive spectral embedding: sidestepping the SVD. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 550–558. Curran Associates, Inc., 2015.

112. S. S. Rangapuram, P. K. Mudrakarta, and M. Hein. Tight Continuous Relaxation of the Balanced k-Cut Problem. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3131–3139. Curran Associates, Inc., 2014.
113. F. Ros and S. Guillaume. ProTraS: A probabilistic traversing sampling algorithm. *Expert Systems with Applications*, 105:65–76, Sept. 2018.
114. A. Saade, F. Krzakala, and L. Zdeborová. Spectral Clustering of graphs with the Bethe Hessian. page 9.
115. I. Safro, P. Sanders, and C. Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:2–2, 2015.
116. T. Sakai and A. Imiya. Fast Spectral Clustering with Random Projection and Sampling. In P. Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 5632 of *Lecture Notes in Computer Science*, pages 372–384. Springer Berlin Heidelberg, 2009.
117. A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega. Eigendecomposition-Free Sampling Set Selection for Graph Signals. *arXiv:1809.01827 [eess]*, Sept. 2018. arXiv: 1809.01827.
118. A. Sandryhaila and J. Moura. 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, Sept. 2014.
119. D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web - WWW '10*, page 1177, Raleigh, North Carolina, USA, 2010. ACM Press.
120. D. Sedley. An Introduction to Plato's Theory of Forms. *Royal Institute of Philosophy Supplement*, 78:3–22, July 2016.
121. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug. 2000.
122. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, May 2013.
123. D. I. Shuman, P. Vandergheynst, and P. Frossard. Chebyshev polynomial approximation for distributed signal processing. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8. IEEE, 2011.
124. D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.
125. D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011.
126. M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.
127. D. J. Sutherland and J. Schneider. On the Error of Random Fourier Features. *arXiv:1506.02785 [cs, stat]*, June 2015. arXiv: 1506.02785.
128. A. Tarsitano. A computational study of several relocation methods for k-means algorithms. *Pattern Recognition*, 36(12):2955 – 2966, 2003.
129. N. Tremblay, P. O. Amblard, and S. Barthelmé. Graph sampling with determinantal processes. In *2017 25th European Signal Processing Conference (EUSIPCO)*, pages 1674–1678, Aug. 2017.
130. N. Tremblay, S. Barthelme, and P.-O. Amblard. Optimized Algorithms to Sample Determinantal Point Processes. *arXiv:1802.08471 [cs, stat]*, Feb. 2018. arXiv: 1802.08471.
131. N. Tremblay, P. Gonccalves, and P. Borgnat. Chapter 11 - design of graph filters and filterbanks. In P. M. Djurić and C. Richard, editors, *Cooperative and Graph Signal Processing*, pages 299 – 324. Academic Press, 2018.
132. N. Tremblay, G. Puy, P. Borgnat, R. Gribonval, and P. Vandergheynst. Accelerated Spectral Clustering Using Graph Filtering Of Random Signals. In *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*, 2016. accepted.
133. N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst. Compressive Spectral Clustering. In *33rd International Conference on Machine Learning*, New York, United States, June 2016.

134. M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo. Signals on Graphs: Uncertainty Principle and Sampling. *IEEE Transactions on Signal Processing*, 64(18):4845–4860, Sept. 2016.
135. O. B. Ulrike von Luxburg, Mikhail Belkin. Consistency of Spectral Clustering. *The Annals of Statistics*, 36(2):555–586, 2008.
136. N. K. Vishnoi et al. Lx= b. *Foundations and Trends® in Theoretical Computer Science*, 8(1–2):1–141, 2013.
137. M. Vladymyrov and M. A. Carreira-Perpinan. Fast, accurate spectral clustering using locally linear landmarks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3870–3879, May 2017.
138. U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.
139. D. Wagner and F. Wagner. Between Min Cut and Graph Bisection. In A. M. Borzyszkowski and S. Soko\lowski, editors, *Mathematical Foundations of Computer Science 1993*, pages 744–750, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
140. J. Wang, J. Wang, Q. Ke, G. Zeng, and S. Li. Fast approximate k-means via cluster closures. In *Multimedia data mining and analytics*, pages 373–395. Springer, 2015.
141. L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek. Approximate Spectral Clustering. In T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5476 of *Lecture Notes in Computer Science*, pages 134–146. Springer Berlin Heidelberg, 2009.
142. Y. Wang and Z. Feng. Towards scalable spectral clustering via spectrum-preserving sparsification. *arXiv preprint arXiv:1710.04584*, 2017.
143. Y.-C. Wei and C.-K. Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *1989 IEEE International Conference on Computer-Aided Design(ICCAD)*, volume 00, pages 298–301.
144. S. T. Wierzchoń and M. A. K\lopotek. Spectral Clustering. In *Modern Algorithms of Cluster Analysis*, pages 181–259. Springer International Publishing, Cham, 2018.
145. D. P. Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.
146. L. Wu, P.-Y. Chen, I. E.-H. Yen, F. Xu, Y. Xia, and C. Aggarwal. Scalable Spectral Clustering Using Random Binning Features. *arXiv:1805.11048 [cs, stat]*, May 2018. arXiv: 1805.11048.
147. D. Yan, L. Huang, and M. I. Jordan. Fast Approximate Spectral Clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 907–916, Paris, France, 2009. ACM.
148. K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.
149. Z. Zhao and H. Liu. Spectral feature selection for supervised and unsupervised learning. In *Proceedings of the 24th international conference on Machine learning - ICML '07*, pages 1151–1157, Corvalis, Oregon, 2007. ACM Press.