Spectral Clustering on a Budget

Ohad Shamir
Microsoft Research
Cambridge, MA 02142
USA

Naftali Tishby
School of Computer Science and Engineering
The Hebrew University
Jerusalem 91904, Israel

Abstract

Spectral clustering is a modern and well known method for performing data clustering. However, it depends on the availability of a similarity matrix, which in many applications can be non-trivial to obtain. In this paper, we focus on the problem of performing spectral clustering under a budget constraint, where there is a limit on the number of entries which can be queried from the similarity matrix. We propose two algorithms for this problem, and study them theoretically and experimentally. These algorithms allow a tradeoff between computational efficiency and actual performance, and are also relevant for the problem of speeding up standard spectral clustering.

1 Introduction

Over the past decade, spectral clustering methods have gained popularity as a method to perform data clustering - one of the most basic tasks of machine learning. These methods enjoy some important advantages, such as the ability to cluster non-vectorial data, and often yield superior empirical performance. Moreover, they are well-studied and supported theoretically.

To apply spectral clustering, one needs to obtain a *similarity matrix*, which quantifies the similarities between any pair of data objects. However, in many cases, obtaining the similarity between all data objects is highly non-trivial. For example, in bioinformatics, clustering often needs to be performed on large-scale molecular interaction networks (such as protein-protein or protein-DNA interactions, see for instance [22]). Due to the sheer number of possible interactions, measuring all of them reliably is usually infeasible. In fields ranging from social network analysis to sensor networks, obtaining full data on the network and the relation between its members is often hard or impossible, and network survey and reconstruction techniques have received significant attention [5, 11]. Similar problems are encountered in any setting where the similarity data has to be elicited from human subjects, who are unable or unwilling to provide full information.

In this paper, we focus on the problem of performing spectral clustering under a *budget* constraint. Namely, a situation where we can only query a limited number of entries from the similarity matrix, but still wish to cluster comparably well as if we had the entire matrix at hand. We propose and study, theoretically and empirically, two algorithms for this task. The first algorithm is a simple and efficient randomized procedure, with formal performance guarantees. The theoretical analysis indicates that its performance improves as the data is more easily clustered. In particular, for well-clustered data and an $n \times n$ similarity matrix, a budget of $\tilde{O}(n)$ (i.e., linear up to logarithmic factors) will suffice. The second algorithm is adaptive, and has better empirical performance. On the flip side, it is much more computationally demanding, and without better theoretical guarantees.

While this is not the main focus of our paper, we note that our algorithms are also relevant for speeding up spectral clustering. In particular, using the first algorithm and for well-clustered data, one can perform spectral clustering in $\tilde{O}(n)$ time, as opposed to $O(n^2)$ with standard methods.

1.1 Related Work

There is a huge body of work on spectral clustering and algorithms to implement it (notable references are [23, 20, 15, 25]). While most of this work assumes that the similarity matrix is known, there have been some lines of work relevant to our setting.

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

The problem of scaling or speeding up spectral clustering (other than simply coming up with efficient eigensolvers for the task) has received attention, and some of the methods used are also implicitly relevant here. For example, one common approach is to sample and cluster a small subset of the data objects, with the clustering result being extended somehow to the original dataset. However, most work in this direction is heuristic and does not provide any performance guarantees. More importantly, the choice of the subset is often based on knowledge of the entire data matrix, an unrealistic assumption in our setting. A good example is [27], where the subset is chosen by partitioning the data space using k-means or random projection trees. Note that this algorithm is not relevant when we do not have a-priori access to the entire dataset. Moreover, due to the partitioning step, the algorithm needs to assume that the data objects are vectors in $\mathbb{R}^n$, while our approach only requires the existence of a similarity matrix, which can also apply to non-vectorial data. In addition, while [27] is noteworthy for providing performance guarantees, these crucially depend on the outcome of the partitioning step, which is hard to formalize without making strong assumptions.

Another related line of work concerns efficient low-rank matrix approximation and matrix reconstruction/completion. In particular, methods which are based on matrix sampling are potentially useful for our setting, if they allow us to reconstruct the entire similarity matrix. One particularly well-known approach involves sampling entire rows or columns from the matrix, and completing the original matrix via techniques such as the Nyström method [12, 7, 6, 18]. For applications of this method to clustering, see [10, 21]. In terms of other approaches, [1] discuss low-rank matrix approximation by uniformly sampling entries i.i.d. with some small probability p. More recently, there has been work on applying notions from compressed sensing, to prove that sampling entries from an approximately low-rank matrix allows approximate reconstruction [4, 16]. However, all these methods are problematic for our setting. First of all, these methods assume that the underlying matrix is approximately low-rank. In the context of spectral clustering, this might not hold even for "well-behaved" data (see Sec. 4). Many of the proposed algorithms use a non-uniform sampling, where the distribution must be determined in advance by examining the entire similarity matrix - infeasible when the matrix is initially unknown. For algorithms based on the Nyström method, we note that being able to sample entire rows/columns of the similarity matrix might be unrealistic. Also, sampling entries i.i.d. is not suitable for a hard budget constraint, since the number of sampled entries becomes a random quantity in itself, with relatively large variance. Finally, the guarantees provided for all these algorithms focus on matrix reconstruction, which is different than the goal we have in mind here. [10] and [21], which do discuss these methods in the context of clustering, do not provide formal guarantees.

2 Setting and Notation

We use upper-case letters to denote matrices (e.g., W), and the corresponding lower-case letters with indices to denote entries in the matrix (e.g., $w_{i,j}$). The $\|\cdot\|$ notation denotes spectral norm for matrices, and Euclidean norm for vectors.

We use A to denote the similarity matrix: it is a symmetric matrix of size $n \times n$, composed of nonnegative entries, which we will assume w.l.o.g. to be bounded in $[0, 1]$. $a_{i,j}$ denotes the similarity of object i to object j, and is larger the more similar they are. We assume for simplicity that $a_{i,i}$, the "self-similarity" of each object i, equals 1 for all $i = 1, \ldots, n$. Rather than having access to the entire matrix, it is assumed we can only query at most b entries, where in general $b \ll n^2$.

This paper deals with spectral clustering, a full survey of which is beyond our scope (for the interested reader, an excellent tutorial is provided in [25]). Spectral clustering is based on the spectral properties of the *Laplacian* of the similarity matrix A. Formally speaking, we define the Laplacian L of A as $L = D - A$, where D is a diagonal matrix with $d_{i,i} = \sum_{j=1}^{n} a_{i,j}$ (we note that other definitions of a Laplacian also exist). We denote the eigenvalues of L, in increasing order, by $\lambda_1 \leq \lambda_2 \leq \ldots$, with associated eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots$. For Laplacians, it always holds that $\lambda_1 = 0$, with an associated eigenvector $\mathbf{v}_1 = \frac{1}{\sqrt{n}}\mathbf{1}$. To prevent ambiguity, we will assume that $\mathbf{v}_1$ always takes this form. In this paper, we focus on the case of bipartitioning, e.g., where the data is to be divided into two clusters. Spectral bipartitioning algorithms are common, and more clusters may be obtained by recursing. We note however that our algorithms and some of the theoretical results are readily extendable to performing k-way clustering directly - a fuller study of which is left to future work. With bipartitioning, the common approach is to split the data based on thresholding the values of the 2nd eigenvector of L (e.g., if $\mathbf{v}_2$ is the eigenvector, then one cluster is $\{j : v_{2,j} \leq a\}$, and the other is $\{j : v_{2,j} > a\}$ for some a). The idea is that for well-clustered data, the entries in $\mathbf{v}_2$ will be roughly divided into two level sets, indicating the two clusters in the data [25]. In our budgeted setting, we cannot compute L or its eigenvectors precisely, so a major goal will be to approximate $\mathbf{v}_2$ as well as possible.

3 Results

In this section, we propose and discuss two algorithms for spectral clustering under a budget constraint. In both cases, we will focus on the issue of obtaining an approximation $\tilde{\mathbf{v}}_2$ of the unknown 2nd eigenvector $\mathbf{v}_2$. Once we have such an approximation, we can use thresholding to derive the actual data partition. The bounds we obtain are in terms of $\|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\|$: It is well-known (e.g., [15, 14]) that if $\|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\| \ll 1$, then under appropriate conditions[1], the clusterings induced by $\tilde{\mathbf{v}}_2$ and $\mathbf{v}_2$ are very similar. When this happens, then the output of our algorithm allows us to perform clustering nearly as well as if we had the entire similarity matrix at hand.

3.1 Fast Randomized Algorithm

Our first algorithm is a simple and efficient procedure, where we randomly choose and query only b entries uniformly at random, setting all other unknown entries to zero, and computing the 2nd eigenvector of the corresponding Laplacian.

Algorithm 1 Randomized Algorithm

Input: budget b
Sample b entries of A uniformly at random (without replacement) from $\{a_{i,j} : i < j\}$
Construct a matrix $\tilde{A}$, where
$\tilde{a}_{i,i} = 2b/n(n-1)$ for all $i \in \{1, \ldots, n\}$
$\tilde{a}_{i,j} = \tilde{a}_{j,i} = a_{i,j}$ if $a_{i,j}$ was queried
$\tilde{a}_{i,j} = \tilde{a}_{j,i} = 0$ if $a_{i,j}$ was not queried
Return the 2nd eigenvector $\tilde{\mathbf{v}}_2$ of $\tilde{A}$'s Laplacian

In terms of theoretical guarantees, the following theorem bounds the distance between the algorithm's output, $\tilde{\mathbf{v}}_2$, and $\mathbf{v}_2$, which is the 2nd eigenvalue of the Laplacian of the "real" similarity matrix A. As mentioned earlier, a small bound on this distance implies, under appropriate conditions, that both vectors induce a similar clustering [14].

Theorem 1. *Suppose that* $b \leq \frac{n(n-1)}{4}$. *Let* $\mathbf{v}_2$ *denote the* 2*nd eigenvector of* $L = D - A$. *Then with probability at least* $1-\delta$ *over the randomness of the algorithm, it holds that* $\min\{\|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\|, \|(-\tilde{\mathbf{v}}_2) - \mathbf{v}_2\|\}$ *is at most*

$$\frac{4}{\lambda_3 - \lambda_2}\left(\sqrt[3]{\frac{n^5}{b^2}}\log\left(\frac{4n}{\delta}\right) + \sqrt{\frac{n^3}{b}\log\left(\frac{4n}{\delta}\right)}\right),$$

where λ_2, λ_3 *are the* 2*nd and* 3*rd smallest eigenvalues of* L. *In particular, if* $b \geq sn\log^{3/2}(4n/\delta)$ *for some constant* s, *then the bound takes the form*

$$\frac{4n}{\lambda_3 - \lambda_2}\left(\frac{1}{s^{2/3}} + \frac{1}{\sqrt{s\log(4n/\delta)}}\right).$$

The choice for the sign of $\tilde{\mathbf{v}}_2$ depends on the eigensolver used, and is inconsequential for the purposes of spectral clustering. Also, the condition on b is merely to simplify the bound - see Eq. (9) in the supplementary material for a version without this condition.

The theorem implies that the quality of the algorithm's output improves with the size of the budget b, as well as the eigengap[2] $\lambda_3 - \lambda_2$. In fact, this eigengap is well known to be a measure of how clusterable the data is (see [25],[20],[14]). Thus, our algorithm has the interesting property that it works better when the data is "easy" to cluster. In particular, when the data is well-clustered, $\lambda_3 - \lambda_2$ can be expected to be on the order of $\Omega(n)$. In that case, $\tilde{\mathbf{v}}_2$ will be an excellent approximation to $\mathbf{v}_2$, as long as the budget b is on the order of $n\log^{3/2}(n)$. Note that this is close to optimal, since in order to have any hope in estimating $\mathbf{v}_2$, we must sample at least one entry from most rows/columns, implying $b \geq \Omega(n)$.

Although this is not our main goal, we note that Algorithm 1 can also be used as a simple way to speed up standard spectral clustering. This is an important issue, since the runtime of straightforward implementations of spectral clustering scale cubically with n, thus precluding their use for large datasets. Indeed, given a known similarity matrix A, the algorithm's analysis implies that spectral clustering can be performed on a sparsified version of A, without hurting the results too much. In particular, if the data is well-clustered, we can sparsify A to have only $O(n\log^{3/2}(n))$ non-zero entries[1]. The computationally dominant part of spectral clustering is extracting the first few eigenvectors of A's Laplacian, and this can be done very efficiently if the matrix is sparse. In particular, for a matrix with $\tilde{O}(n)$ non-zero entries as above, which approximates a matrix with eigengap $\lambda_3 - \lambda_2 = \Omega(n)$, we can extract the eigenvectors (say, by the power method or Lanczos iterations) and perform spectral clustering in $\tilde{O}(n)$ time.

We now turn to discuss why Thm. 1 holds. The key idea is that Algorithm 1 essentially constructs an unbiased estimate of the actual matrix A (up to scaling). The analysis is based on the premise that the spectral properties of $\tilde{A}$ reflect those of A well, even if most of the entries remain unsampled. This premise has been used in the past to analyze sampling-based spectral

[1] See [15, 14] for a precise formulation. Roughly speaking, we need the data to be reasonably well-clustered, an assumption we will rely upon anyway later on.

[2] If desired, the eigengap can also be estimated from $\tilde{A}$ - see the supplementary material for details.

algorithms (such as [1, 2]), and there exists a substantial literature on large deviation bounds for matrices. However, these works are based on assumptions which *do not* hold here - in particular, that the entries of the random matrix are independent. Indeed, since we sample a fixed number of entries according to our budget, the event that a certain entry was sampled does influence the probability that other entries were sampled. Thus, new techniques are required here. The key observation is that while the matrix entries are dependent, they have a particular statistical structure, known as *negative dependence*, which allows us to extend relevant large deviation results and apply them to our setting. Intuitively, negatively dependent random variables are such that if one subset of the variables is "large", then a disjoint subset of the variables must be "small". Although not independent, such random variables enjoy similar concentration of measure behavior as independent random variables.

More formally, we say that $x_1, \ldots, x_m$ are negatively dependent, if for all disjoint subsets $I, J \subseteq \{1, \ldots, m\}$ and all non-decreasing functions f, g,

$$\mathbb{E}[f(x_i, i \in I)g(x_j, j \in J)] \leq \mathbb{E}[f(x_i, i \in I)]\mathbb{E}[g(x_j, j \in J)].$$

We will need the following two useful facts:

Lemma 1. *If $x_1, \ldots, x_m$ are negatively dependent:*

1. *$f_1(x_1), \ldots, f_m(x_m)$ are negatively dependent for any non-decreasing functions $f_1, \ldots, f_m$.*
2. *$\mathbb{E}\left[\prod_{i=1}^m x_i\right] \leq \prod_{i=1}^m \mathbb{E}[x_i]$.*

These facts appear, for instance, in section 3.1 of [9]. Also, we will need the following lemma, whose proof is a straightforward exercise (see problem 3.1 in [8]):

Lemma 2. *Suppose that $x_1, \ldots, x_m$ are random variables taking values in $\{0, 1\}$, such that $x_1 + \ldots + x_m$ is a fixed constant with probability 1. Then $x_1, \ldots, x_m$ are negatively dependent.*

The rough outline of the proof of Thm. 1 is as follows. First, in Lemma 3, we establish that the entries of $\tilde{A}$ are negatively dependent. Second, in Lemma 4, we extend a concentration bound on the spectral norm, due to [26], from independent entries to negatively dependent entries. This allows us to bound the deviation of A from its empirical estimate. Third, in Lemma 5, we note that Bernstein's large deviation inequality can be applied to negatively dependent random variables (a simpler inequality such as Hoeffding's is not strong enough for our purposes here). This allows us to bound the deviation of D from its empirical estimate. Combining the deviations of A and D provides us with a spectral bound on our estimate of L. Fourth, in Lemma 6, we use results from matrix perturbation theory in order to reduce that spectral bound to a bound on $\|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\|$. Some of the proofs are only sketched, with the full proof appearing in the supplementary material.

Lemma 3. *The set of entries $\{\tilde{a}_{i,j} : i \leq j\}$ are negatively dependent.*

Proof. Each entry $\tilde{a}_{i,j}$ equals $a_{i,j}\tilde{x}_{i,j}$, where $\tilde{x}_{i,j}$ is an indicator of whether entry $a_{i,j}$ was sampled. Thus, by Lemma 1, we only need to show that $\{\tilde{x}_{i,j} : i \leq j\}$ are negatively dependent. This follows from Lemma 2. □

Lemma 4. *Let $\tilde{W}$ be a zero-mean symmetric random matrix. Suppose that the matrix entries $\{\tilde{w}_{i,j} : i \leq j\}$ are negatively dependent, with each entry having a variance of at most σ^2 and absolute value at most[3] K with probability 1. Also, assume that each $\tilde{w}_{i,j}$ takes only two possible values $r_{i,j}, s_{i,j}$ such that $r_{i,j} < s_{i,j}, |r_{i,j}| < |s_{i,j}|$. Then for any $t > 0$,*

$$\Pr\left(\|\tilde{W}\| \geq 2\sigma\sqrt{n} + tn^{1/3}\log(n)\right) \leq 2n^{1-t/3(\sigma^2 K)^{1/3}}.$$

Proof. The proof is based on the method described in section 2 of [26], to which we refer the reader for a full exposition. In a nutshell, it implies that if one can upper bound $\mathbb{E}[\mathrm{Trace}(\tilde{W}^k)]$ by $2n(2\sigma\sqrt{n})^k$ for some positive even number k (and in particular, when we take $k = (\sigma/K)^{1/3}n^{1/6}$), then the lemma holds.

In [26], this was proved by first observing that $\mathbb{E}[\mathrm{Trace}(\tilde{W}^k)]$ can be upper bounded by

$$\sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} \mathbb{E}[\tilde{w}_{i_1,i_2}\tilde{w}_{i_2,i_3} \cdots \tilde{w}_{i_{k-1},i_k}\tilde{w}_{i_k,i_1}].$$

The crucial (and only) use of the independence assumption in the proof came here: it was argued that products containing a particular entry $\tilde{w}_{i,j}$ only once can be discarded, since the entries are independent and $\mathbb{E}[\tilde{w}_{i,j}] = 0$. A bit more formally, independence was used to argue that for any positive integers $q, n_1, \ldots, n_q$ and any $q+1$ distinct entries $\tilde{w}_{i_0,j_0}, \ldots \tilde{w}_{i_q,j_q}$ from the matrix, it holds that

$$\mathbb{E}\left[\tilde{w}_{i_0,j_0}\prod_{k=1}^{q}\tilde{w}_{i_k,j_k}^{n_k}\right] \leq 0. \quad (1)$$

When the entries are independent, the l.h.s. equals $\mathbb{E}[\tilde{w}_{i_0,j_0}]\prod_{k=1}^{q}\mathbb{E}[\tilde{w}_{i_k,j_k}^{n_k}]$, which is 0 since $\mathbb{E}[\tilde{w}_{i_0,j_0}] = 0$.

[3] Strictly speaking, for the proof we need to assume that $(\sigma/K)^{1/3}n^{1/6}$ is an even number. This can always be achieved by making σ or K slightly larger. For simplicity, we will ignore this issue.

However, Eq. (1) also holds when the random variables are negatively dependent, under the conditions of the lemma. To see why, note that for any n, $\tilde{w}_{i,j}^n$ is a non-decreasing function of $\tilde{w}_{i,j}$ on its domain $\{r_{i,j}, s_{i,j}\}$, under the conditions on $r_{i,j}, s_{i,j}$. Using Lemma 1, we get that

$$\mathbb{E}\left[\tilde{w}_{i_0,j_0}\prod_{k=1}^{q}\tilde{w}_{i_k,j_k}^{n_k}\right] \leq \mathbb{E}\left[\tilde{w}_{i_0,j_0}\right]\prod_{k=1}^{q}\mathbb{E}\left[\tilde{w}_{i_k,j_k}^{n_k}\right] = 0.$$

Thus, the proof from [26] carries through in our setting as well. □

Lemma 5 (Bernstein's bound for negatively dependent random variables)**.** *Let $x_1, \ldots, x_n$ be a set of n zero-mean, negatively dependent random variables, whose variance is at most σ^2 and whose absolute value is at most 1 with probability 1. Then for any $\epsilon > 0$,*

$$\Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n}x_i\right| > \epsilon\right) \leq 2\exp\left(-\frac{n\epsilon^2}{2(\sigma^2+\epsilon/3)}\right). \quad (2)$$

Proof. The standard proof of Bernstein's inequality (e.g., [3]) is based on applying Markov's inequality to get that $\Pr(\frac{1}{n}\sum_i x_i > \epsilon)$ is upper bounded for any $t > 0$ by $\mathbb{E}[e^{t\sum_i x_i}]/e^{tn\epsilon}$. The key (and only) use of independence is in rewriting $\mathbb{E}[e^{t\sum_i x_i}]$ as $\prod_i \mathbb{E}[e^{tx_i}]$. However, even if $x_1, \ldots, x_n$ are (negatively) dependent, then Lemma 1 allows us to upper bound $\mathbb{E}[e^{t\sum_i x_i}]$ by $\prod_i \mathbb{E}[e^{tx_i}]$. The rest of the proof follows verbatim the standard proof of Bernstein's inequality. □

Lemma 6. *Let $L, \tilde{L}$ be the Laplacians of two symmetric matrices $A, \tilde{A}$, whose 2nd eigenvectors are $\mathbf{v}_2, \tilde{\mathbf{v}}_2$ respectively. Suppose that the smallest eigenvalue of L is simple (i.e., has multiplicity 1). Then choosing the sign of $\tilde{\mathbf{v}}_2$ so that $\|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\|$ is minimized, it holds that*

$$\|\tilde{\mathbf{v}}_2 - \mathbf{v}_2\| \leq \sqrt{2}\ \min\left\{\frac{\|\tilde{L}-L\|}{\lambda_3-\lambda_2}, 1\right\},$$

where λ_2, λ_3 are L's 2nd and 3rd smallest eigenvalues.

Proof Sketch. Denote $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ to be the eigenvalues and eigenvectors of L, and let $0 = \tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq \cdots \leq \tilde{\lambda}_n$ and $\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, \ldots, \tilde{\mathbf{v}}_n$ to be the eigenvalues and eigenvectors of $\tilde{L}$ (we recall that a Laplacian is always positive semidefinite and has a 0 eigenvalue, see [25]).

By applying a classical eigenvector perturbation theorem due to Davis and Kahan (see section V in [24]), we have that if $[\mathbf{v}_1, \mathbf{v}_2]$ is the subspace spanned by the first two eigenvectors of L, and $\tilde{V}$ is the subspace spanned by the eigenvectors of $\tilde{L}$ whose eigenvalue is at most λ_2, then

$$\left\|\sin\left(\Theta\left([\mathbf{v}_1, \mathbf{v}_2], \tilde{V}\right)\right)\right\| \leq \frac{\|\tilde{L}-L\|}{\lambda_3-\lambda_2},$$

where $\sin\left(\Theta\left([\mathbf{v}_1, \mathbf{v}_2], \tilde{V}\right)\right)$ is the diagonal matrix with the sines of the canonical angles between the subspaces $[\mathbf{v}_1, \mathbf{v}_2], \tilde{V}$ along the main diagonal. The lemma follows after several manipulations. □

With these lemmas in hand, we can now sketch the proof of Thm. 1.

Proof sketch of Thm. 1. Let $c = \frac{n(n-1)}{2b}$, and let $\tilde{D}, \tilde{L}$ be the analogues of D, L w.r.t. the matrix $\tilde{A}$. By the triangle inequality,

$$\|c\tilde{L} - L\| \leq \|c\tilde{D} - D\| + \|c\tilde{A} - A\|. \quad (3)$$

We treat each term separately. By Lemma 3, it can be shown that $c\tilde{A} - A$ is a matrix with zero-mean, negatively dependent entries. By applying Lemma 4, it can be shown that with probability at least $1-\delta$,

$$\|c\tilde{A} - A\| \leq 2\sqrt{cn} + 3\sqrt[3]{c^2 n}\log(2n/\delta). \quad (4)$$

Turning to analyze $\|c\tilde{D} - D\|$, one can show via Lemma 5 that with probability at least $1-\delta$, if δ is not too small,

$$\|c\tilde{D} - D\| \leq \sqrt{3cn\log(2n/\delta)}.$$

The theorem follows by plugging the above and Eq. (4) into Eq. (3) with a union bound, applying Lemma 6, and performing a few more technical steps. □

3.2 Adaptive Algorithm

While Algorithm 1 is simple and provably effective, it is a non-adaptive algorithm, which samples entries uniformly at random. Intuitively, a more budget-efficient strategy might be to query entries one by one, using already-known entries to adaptively pick the next entry to query. This is related to the idea of active learning, where the learner actively picks which instances to query for a label, rather than passively receiving labeled examples. Of course, unlike active learning, we cannot hope for a dramatic improvement in general, since for well-clustered data, the budget required for Algorithm 1 is already optimal up to logarithmic factors (see the discussion following Thm. 1). Nevertheless, one might wonder whether a more adaptive algorithm can lead to improved empirical performance.

Below, we present one such algorithm, which empirically seems to perform even better than Algorithm 1.

The idea of the algorithm is as follows: suppose we already queried a subset of entries, inducing a partial matrix $\hat{A}$ whose Laplacian $\hat{L}$'s 2nd eigenvector is $\hat{\mathbf{v}}_2$. Now, we need to decide which entry to query next. For some entries, if we query them and use their value to update $\hat{A}$, then $\hat{\mathbf{v}}_2$ might change a lot. For other entries, querying them might leave $\hat{\mathbf{v}}_2$ almost unchanged. Moreover, as we query more entries, $\hat{\mathbf{v}}_2$ will tend to move closer and closer[4] to the 2nd eigenvector of the "real" Laplacian L. So intuitively, we would like to query entries which will cause $\hat{\mathbf{v}}_2$ to change a lot, hopefully moving substantially closer to $\mathbf{v}_2$. Unfortunately, we don't know the values of the entries beforehand, so we cannot know in advance which entry will cause $\hat{\mathbf{v}}_2$ to change substantially. However, we can still do a *sensitivity* check: if we slightly perturb a certain unsampled entry $\hat{a}_{i,j}$ (and simultaneously $\hat{a}_{j,i}$ in the same way, since both entries will be updated following a query), by how much does it change $\hat{\mathbf{v}}_2$?

Therefore, at each time step, we can compute the *derivative* of $\hat{\mathbf{v}}_2$ w.r.t. perturbing each entry, and query the entry for which the norm of the derivative is largest. One more twist that we add to the algorithm is that instead of picking the entry to query according to the derivative at each step, we alternate between doing that and picking an entry uniformly at random from among the remaining unqueried entries. This is for theoretical as well as practical reasons: Theoretically, it allows us to prove that the algorithm will never be much worse than Algorithm 1, in terms of the required budget. Practically, it enhances the stability of the algorithm, preventing it from getting 'stuck' in querying a group of unprofitable entries, and ensuring that $\hat{A}$ does not get overly distorted compared to the actual similarity matrix A. We return to these issues in Sec. 4.

The pseudo-code of our algorithm is presented as Algorithm 2. We note that in each even iteration, the algorithm indeed computes the squared norm of the derivative of the Laplacian's 2nd eigenvector. The full technical derivation is presented in the supplementary material, and is based on a combination of techniques used in [17] and [19].

Compared to Algorithm 1, it should be emphasized that Algorithm 2 is much more computationally demanding, since a full eigendecomposition is required every other iteration (requiring $O(n^3)$ flops in general). On the other hand, it tends to have better empirical performance, as discussed in Sec. 4.

In terms of implementation, we note that one can directly compute a matrix whose (i,j)-th entry equals the squared derivative norm w.r.t. perturbation in entry (i,j). Depending on the programming language and libraries used, this can be more efficient than looping through all i,j and computing the derivative for each one. This derivative matrix equals

$$(R - R^\top) \circ (R - R^\top) \circ \left(S + S^\top - 2UU^\top\right),$$

where $\circ$ is entry-wise product, $R = \hat{\mathbf{v}}_k \mathbf{1}^\top$, $S = \left(\sum_{l=1}^n \hat{\mathbf{u}}_l \circ \hat{\mathbf{u}}_l\right) \mathbf{1}^\top$, and U is the square matrix whose l-th column is $\hat{\mathbf{u}}_l$. Deriving this expression is a straightforward technical exercise.

Other than that, there are a few other ways to make the implementation of Algorithm 2 somewhat more efficient. In particular, note that at each iteration, we perform an eigendecomposition of the Laplacian from scratch. This is potentially wasteful, since the Laplacian changes only by a few entries at each round, so we should be able to use the eigendecomposition from the previous iteration to compute the updated eigendecomposition quickly. Indeed, there are several relevant algorithms for solving updated eigendecomposition problems (see [13]). However, to the best of our knowledge, all these algorithms still require $O(n^3)$ flops, so the saving is only in the constant of the $O(\cdot)$ notation.

In terms of theoretical guarantees, analyzing Algorithm 2 is considerably more complex than Algorithm 1. While leaving a fuller theoretical study of Algorithm 2 to future work, we can still show the following "sanity check" theorem. Its proof is sketched below, and presented in the supplementary material.

Theorem 2. *For any budget size b, the guarantees of Thm. 1 for Algorithm 1, with a budget of size b, also hold for Algorithm 2, with a budget of size $2b$.*

This theorem should be intuitively plausible, since Al-

Algorithm 2 Adaptive Algorithm

Input: budget b.
Initialize $\hat{A} = \frac{2b}{n(n-1)} I_n$ (I_n is the identity matrix)
Initialize $S = \{(i,j) \in \{1,\ldots,n\}^2 : i < j\}$
For $t = 1, 2, \ldots, b$
  If t is odd, pick (i,j) uniformly at random from S
  If t is even:
    - Compute eigenvalues and eigenvectors $\{\hat{\lambda}_1,\ldots,\hat{\lambda}_n\}, \{\hat{\mathbf{v}}_1,\ldots,\hat{\mathbf{v}}_n\}$ of $\hat{A}$'s Laplacian
    - For $l = 1,\ldots,n$, let $\hat{\mathbf{u}}_l = \hat{\mathbf{v}}_l/(\lambda_k - \lambda_l)$ if $\lambda_k \neq \lambda_l$, and $\mathbf{0}$ otherwise
    - $(i,j) := \operatorname{argmax}_{(i,j)\in S} (\hat{v}_{k,i} - \hat{v}_{k,j})^2 \sum_{l=1}^n \left(\hat{u}_{l,i} - \hat{u}_{l,j}\right)^2$
  Let $\hat{a}_{i,j} = a_{i,j}$, $\hat{a}_{j,i} = a_{j,i}$, $S = S \setminus (i,j)$
Return 2nd eigenvector of the Laplacian of $\hat{A}$.

[4]The distance does not necessarily decrease monotonically, but it is possible to show that $\|\hat{L} - L\|$, which can be used to upper bound $\|\hat{\mathbf{v}}_2 - \mathbf{v}_2\|$, decreases monotonically.

gorithm 2 does the same things as Algorithm 1 at 1/2 of the iterations (i.e., uniform sampling). However, the proof is not completely straightforward, since the random sampling steps are interleaved and influenced by the derivative steps.

Proof Sketch. A key component of the proof is the following observation: Let $\hat{L}$ be the Laplacian of the matrix $\hat{A}$ during some point in the run of algorithm 2, let $\hat{L}'$ be the Laplacian of the matrix obtained from $\hat{A}$ by setting some entry pairs $\hat{a}_{i,j}, \hat{a}_{j,i}$ to 0 in an arbitrary manner. Then it holds that $\|\hat{L} - L\| \leq \|\hat{L}' - L\|$.

To see why this is true, it is enough to consider the case where a single entry pair $\hat{a}_{i,j}, \hat{a}_{j,i}$ is set to 0, and then repeat the argument. In this case, it is easy to verify that for any vector $\mathbf{v}$,

$$\mathbf{v}^\top(\hat{L} - L)\mathbf{v} - \mathbf{v}^\top(\hat{L}' - L)\mathbf{v} = \hat{a}_{i,j}(v_i - v_j)^2.$$

In particular, if we pick $\mathbf{v}$ to be the maximal eigenvector of $(\hat{L} - L)$, and $\mathbf{v}'$ to be the maximal eigenvector of $(\hat{L}' - L)$, then

$$\|\hat{L}' - L\| - \|\hat{L} - L\| = \mathbf{v}'^\top(\hat{L} - L)\mathbf{v}' - \mathbf{v}^\top(\hat{L}' - L)\mathbf{v}$$
$$\geq \mathbf{v}^\top(\hat{L} - L)\mathbf{v} - \mathbf{v}^\top(\hat{L}' - L)\mathbf{v} = \hat{a}_{i,j}(\hat{v}_i - \hat{v}_j)^2 \geq 0.$$

With this observation at hand, the basic idea of the proof of Thm. 2 is that after running Algorithm 2 with a budget size $2b$, the matrix $\hat{A}$ always includes b revealed entries which "look" as if they were sampled uniformly without replacement. Notice that these entries might not be the entries queried in the even iterations of Algorithm 2, since these were performed on a matrix with some entries already revealed in a non-random manner. A careful formalization of this intuition, combined with the observation above, leads to the theorem statement. □

4 Experiments

In this section, we present a preliminary empirical study of our algorithms. We followed [14] in choosing and constructing the datasets, since that paper also studied spectral bi-partitioning under uncertainty (although in their case, it dealt with data perturbation). The datasets consist of several artificial datasets (see Fig. 1), and 5 real datasets taken from the UCI repository (see Fig. 2). Compared to [14], we made two changes in choosing the datasets: one is that we created several variants of the 2D Gaussian dataset, with increasing distances between the centers, to better study how the algorithms degrade with the hardness of the clustering task. The other change is in dropping the UCI breast dataset, since even with the full similarity matrix, we were unable to spectral cluster it in a satisfactory manner. Clearly, if clustering is too hard with the full data, one cannot hope to do well with only partial data. For the real datasets, we chose either 300 random examples or the entire dataset if it was smaller. The similarity matrix was constructed with a Gaussian kernel.

For each dataset, we studied 4 algorithms. Two of them are Algorithm 1 and Algorithm 2 from this paper. For comparison, we also applied the Nyström method, as described in [10]. This method is based on uniformly sampling entire rows of the matrix, and thus might be potentially useful for our setting, in cases where querying for entire rows is not unrealistic. Finally, we also studied a "Derivatives Only" variant of Algorithm 2, where entries were always picked according to the derivative criterion, rather than alternating between that and picking an entry randomly. Each algorithm was ran on increasing budget sizes (i.e. it was allowed to incrementally query additional entries). After obtaining the 2nd eigenvector estimate, the cluster partitioning was done by thresholding the values of the 2nd eigenvector with respect to the mean value. The performance of each algorithm was measured in terms of misclustering rate. In other words, we measured the proportion of data objects which were clustered differently, compared to spectral clustering of the full similarity matrix. Each algorithm was ran 5 times over each dataset, and we report the averaged results.

From the results plotted in Fig. 1 and Fig. 2, one can clearly see that both Algorithm 1 and 2 perform quite well, with satisfactory results obtained based on seeing just a small portion of the matrix entries. Moreover, Algorithm 2, which adaptively chooses which entries to query, tends to perform better than Algorithm 1. The Nyström-based algorithm sometimes performs rather well, but in other cases performs very badly. This should not be surprising. The reason is that the Nyström method is based on a low-rank assumption for the similarity matrix. Specifically, it assumes that the small number of rows given as input essentially span all other rows. This works well when the similarity matrix is, say, approximately block diagonal, and the data is separated into two tight clusters in Euclidean space. However, when the data structure is more complex (e.g., in the two concentric circles dataset, and some of the other datasets), the method should not be expected to perform well. Finally, the variant of Algorithm 2, which did not choose entries randomly at all, tends to be less stable and often worse than Algorithm 2.

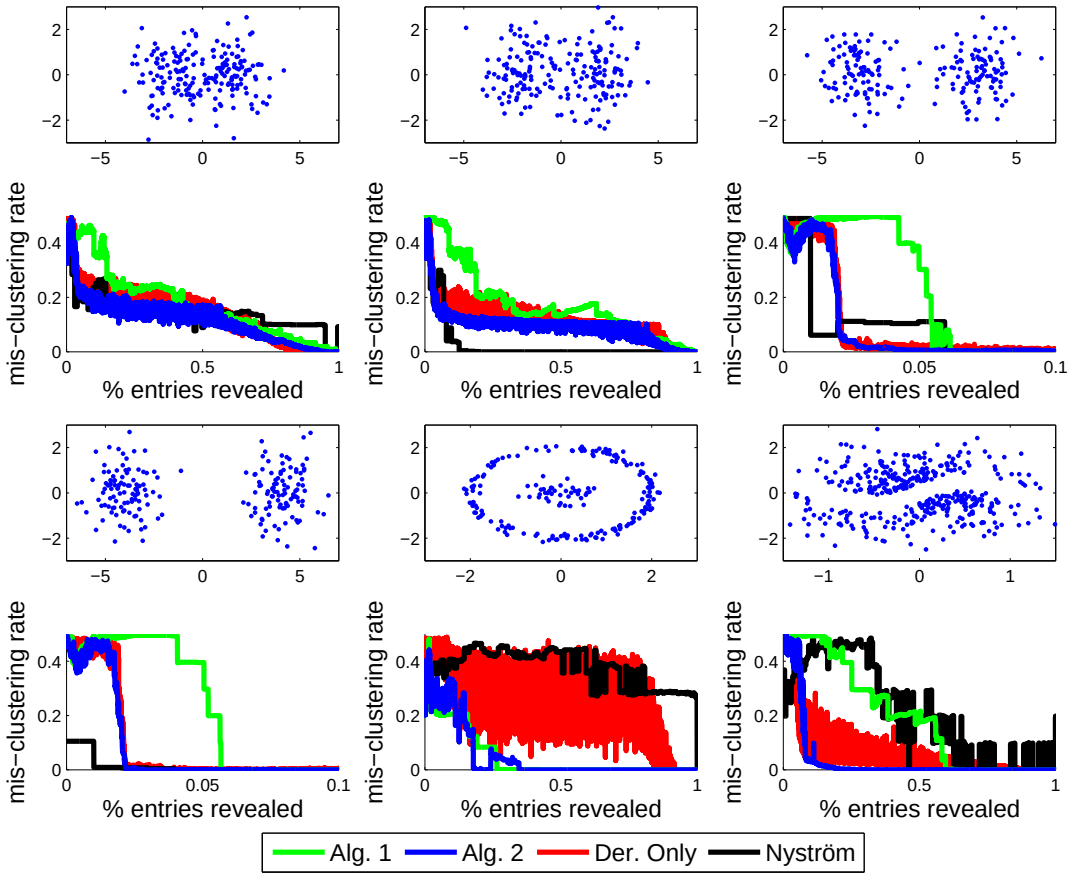

Figure 1: Results for artificial datasets. Above each graph is a representation of the dataset used. Note that the X-axis is not uniform across the plots.

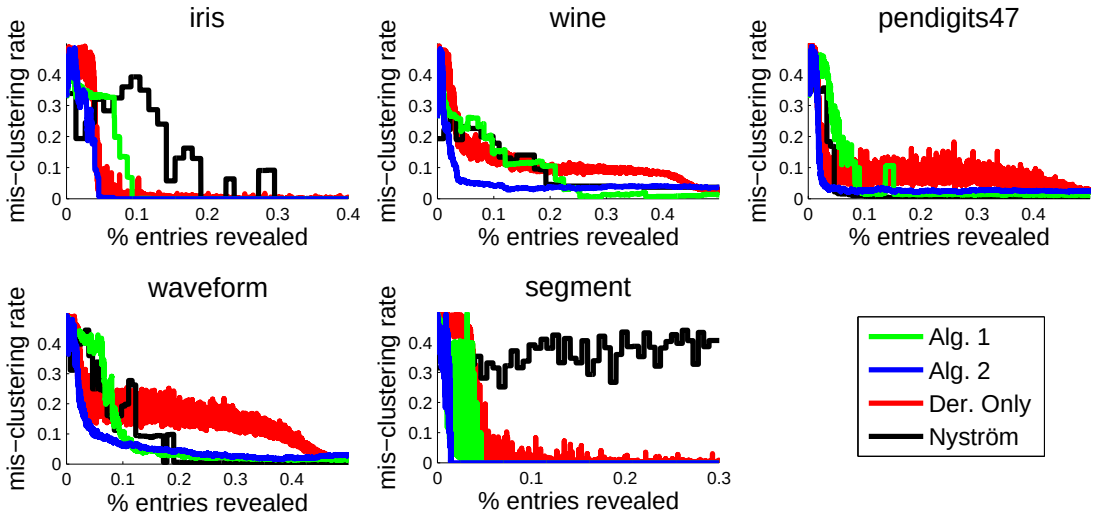

Figure 2: Results for datasets from the UCI repository. Note that the X-axis is not uniform across the plots.

References

[1] D. Achlioptas and F. McSherry. Fast computation of low-rank matrix approximations. *J. ACM*, 54(2), 2007.

[2] Yossi Azar, Amos Fiat, Anna R. Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of STOC*, pages 619–626, 2001.

[3] S. Boucheron, G. Lugosi, and O. Bousquet. Concentration inequalities. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 208–240. Springer, 2003.

[4] E. Candès and Y. Plan. Accurate low-rank matrix recovery from a small number of linear measurements. *CoRR*, abs/0910.0413, 2009.

[5] P. Drineas, A. Javed, M. Magdon-Ismail, G. Pandurangan, R. Virrankoski, and A. Savvides. Distance matrix reconstruction from incomplete distance information for sensor network localization. In *Proceedings of IEEE SECON*, pages 536–544, 2006.

[6] P. Drineas, R. Kannan, and M. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM J. Comput.*, 36(1):158–183, 2006.

[7] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[8] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[9] D. Dubhashi and D. Ranjan. Balls and bins: A study in negative dependence. Technical Report RS-96-25, BRICS, U. of Aarhus, July 1996.

[10] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, 2004.

[11] O. Frank. Network sampling and model fitting. In *Models and Methods in Social Network Analysis*. Cambridge University Press, 2005.

[12] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.

[13] M. Gu and S. Eisenstat. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15(4):1266–1276, October 1994.

[14] L. Huang, D. Yan, M. Jordan, and N. Taft. Spectral clustering with perturbed data. In *Proceedings of NIPS*, pages 705–712, 2008.

[15] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004.

[16] R. Keshavan, S. Oh, and A. Montanari. Matrix completion from a few entries. *CoRR*, abs/0901.3150, 2009.

[17] T. Kollo and H. Neudecker. Asymptotics of eigenvalues and unit-length eigenvectors of sample variance and correlation matrices. *J. of Multivariate Analysis*, 47:283–300, 1993.

[18] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. On sampling-based approximate spectral decomposition. In *ICML*, 2009.

[19] J. Magnus and H. Neudecker. *Matrix Differential Calculus: with Applications to Statistics and Econometrics*. Wiley, 3 edition, 1999.

[20] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proceedings of NIPS*, pages 849–856, 2001.

[21] T. Sakai and A. Imiya. Fast spectral clustering with random projection and sampling. In *Proceedings of MLDM*, pages 372–384, 2009.

[22] T. Sen, A. Kloczkowski, and R. Jernigan. Functional clustering of yeast proteins from the protein-protein interaction network. *BMC Bioinformatics*, July 2006.

[23] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[24] G. Stewart and J. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

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

[26] V. Vu. Spectral norm of random matrices. *Combinatorica*, 27(6):721–736, November 2007.

[27] D. Yan, L. Huang, and M. Jordan. Fast approximate spectral clustering. In *Proceedings of SIGKDD*, pages 907–916, 2009.