Time and Space Efficient Spectral Clustering via Column Sampling

Mu Li[1] Xiao-Chen Lian[1] James T. Kwok[2] Bao-Liang Lu[1,3]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[2]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong
[3]MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong University, Shanghai 200240, China
{limu.cn, lianxiaochen}@gmail.com, jamesk@cse.ust.hk, bllu@sjtu.edu.cn

Abstract

Spectral clustering is an elegant and powerful approach for clustering. However, the underlying eigen-decomposition takes cubic time and quadratic space w.r.t. the data set size. These can be reduced by the Nyström method which samples only a subset of columns from the matrix. However, the manipulation and storage of these sampled columns can still be expensive when the data set is large. In this paper, we propose a time- and space-efficient spectral clustering algorithm which can scale to very large data sets. A general procedure to orthogonalize the approximated eigenvectors is also proposed. Extensive spectral clustering experiments on a number of data sets, ranging in size from a few thousands to several millions, demonstrate the accuracy and scalability of the proposed approach. We further apply it to the task of image segmentation. For images with more than 10 millions pixels, this algorithm can obtain the eigenvectors in 1 minute on a single machine.

1. Introduction

Spectral clustering has been widely used in diverse areas such as data mining [10, 5], signal processing [1] and computer vision [11, 12]. Given n data points, it is based on the eigen-decomposition of an $n \times n$ matrix that is derived from the pairwise similarities of these points. For example, the normalized cut algorithm [12], which is the focus of this paper, computes the eigenvectors corresponding to the k smallest eigenvalues of the normalized Laplacian matrix. Using the k respective components of the data points in these eigenvectors, another clustering algorithm (e.g., the k-means) is then invoked to form the final clusters. However, the requirement of computing and storing the affinity matrix, and the $\mathcal{O}(n^3)$ complexity of decomposing the Laplacian matrix, severely restrict the scalability of spectral clustering to large data sets [12].

As only several eigenvectors are required in the procedure, a general approach to alleviate this problem is by using low-rank matrix approximations, among which the Nyström method [4, 17] is the most popular. It samples $m \ll n$ columns from the original $n \times n$ matrix, and then forms a low-rank approximation of the full matrix by using the correlations between the sampled columns and the remaining $n - m$ columns. As only a portion of the full matrix is computed and stored, the Nyström method can reduce the time and space complexities significantly. Fowlkes *et al*. successfully applied this to spectral clustering for image segmentation [6]. Besides this, the Nyström method has also been popularly used for tasks such as Gaussian processes [17] and manifold learning [15].

However, when the data set is huge, even the Nyström method can suffer from storage problems. Specifically, since the sampled columns have to be frequently used in the computational procedure, typically they have to be stored in the main memory. For data sets containing millions of samples, even storing only 1% of the columns (in double precision) takes about 80GB.

Another problem with the Nyström method is that the eigenvectors obtained are not orthogonal in general. As has been shown in [8], the lack of orthogonality can sometimes adversely affect the approximation quality. Thus, an additional orthogonalization step may have to be performed. However, in image segmentation, this is further complicated by that the affinity matrix used is typically sparsified and not positive semidefinite (psd). The corresponding orthogonalization procedure proposed in [6] then takes cubic time, making it impractical to implement in practice.

In this paper, we propose a time- and space-efficient spectral clustering algorithm based on the Nyström method. However, in contrast to the approach in [6], we directly compute a rank-k approximation of $M = I - L$ (where I is the identity matrix and L is the normalized Laplacian matrix) using the sampled columns of the affinity matrix. With just one single pass on the sampled columns, both the approximated eigenvectors of M and the affinity degrees can

be computed simultaneously. Thus, these sampled columns no longer have to be stored, alleviating the storage problem when n is very large. We further propose a general algorithm to orthogonalize the approximated eigenvectors, irrespective of whether the underlying matrix is psd or not.

Our contributions are twofold. First, we reduce the time and space complexities of the existing Nyström-based spectral clustering algorithm [6]. Theoretical analysis shows that the proposed algorithm can exactly recover the degrees of the sampled points. Experimentally, the algorithm can scale to much larger data sets (with millions of samples) than the existing algorithms. Second, we apply the proposed algorithm to the segmentation of images with very sparse affinity matrices. We demonstrate its scalability on images with more than 10 million pixels. To our best knowledge, this is the first attempt to scale spectral clustering to such a large image segmentation task.

The rest of this paper is organized as follows. Section 2 briefly reviews spectral clustering and the Nyström based approach. Section 3 then describes the proposed algorithm. Experimental results are presented in Section 4, and Section 5 gives some concluding remarks.

Notations. For a symmetric matrix $A \in \mathbb{R}^{n\times n}$, we denote by $A_k = U\Lambda U$ its rank-k approximation, where Λ contains its k largest eigenvalues and U contains its k leading eigenvectors. Moreover, $\mathrm{diag}(v)$ is the matrix with the elements of v on the main diagonal.

2. Related Works

2.1. Spectral Clustering

Spectral clustering is based on analyzing the affinity matrix $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ defined on a set of n data points $X = \{x_1, \ldots, x_n\}$, where $a_{ij} \geq 0$ is the similarity / affinity between x_i and x_j. In general, there are several ways to group X into clusters [16]. In this paper, we focus on the popular method of normalized cut [12]. It first computes the normalized Laplacian $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ of A, where $D = \mathrm{diag}(A1)$ is the degree matrix and 1 is the vector of all ones. Next, the k trailing eigenvectors of L are obtained, where k is usually not less than the number of clusters. Finally, k-means clustering is performed on these eigenvectors to obtain the clusters.

2.2. Nyström Method

The Nyström method [4, 17] has been commonly used to construct low-rank matrix approximations. Given a symmetric matrix $A \in \mathbb{R}^{n\times n}$, it first samples $m \ll n$ columns from A. Without loss of generality, we can assume these to be the first m columns of A, and thus A can be rewritten as

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} \text{ and } A_{:1} = \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix}, \tag{1}$$

where $A_{:1} \in \mathbb{R}^{n\times m}$ is the submatrix containing the selected columns, and $A_{11} \in \mathbb{R}^{m\times m}$ is the submatrix containing the intersection of the selected rows and columns. The Nyström approximation of A is then given by

$$\hat{A}_{\text{nys}} = A_{:1}A_{11}^{-1}A_{:1}^T. \tag{2}$$

2.3. Using Nyström in Spectral Clustering

The normalized cut requires the computation of the k trailing eigenvectors of the normalized Laplacian $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, which are the same as the k leading eigenvectors of the $n \times n$ matrix

$$M = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}. \tag{3}$$

The direct eigen-decomposition of M takes $\mathcal{O}(n^3)$ time, and can be expensive when n is large. In [6], this problem is alleviated by using the Nyström method to approximate the eigenvectors as follows. First, to avoid using the whole A, the degree matrix D is approximated as

$$\hat{D} = \mathrm{diag}(\hat{A}_{\text{nys}}1) = \mathrm{diag}(A_{:1}A_{11}^{-1}A_{:1}^T1), \tag{4}$$

by using the Nyström approximation of A in (2). Let $D_{:1} = \mathrm{diag}(A_{:1}^T1)$. The submatrices in M corresponding to A_{11} and $A_{:1}$ are

$$M_{11} = D_{:1}^{-\frac{1}{2}}A_{11}D_{:1}^{-\frac{1}{2}}, \quad M_{:1} = D^{-\frac{1}{2}}A_{:1}D_{:1}^{-\frac{1}{2}}. \tag{5}$$

Using $\hat{D}$, $M_{:1}$ can then be approximated as

$$\hat{M}_{:1} \simeq \hat{D}^{-\frac{1}{2}}A_{:1}\hat{D}_{:1}^{-\frac{1}{2}} = [M_{11}\hat{M}_{21}^T]^T,$$

and M as

$$\hat{M} = \hat{M}_{:1}M_{11}^{-1}\hat{M}_{:1}^T, \tag{6}$$

where $\hat{M}_{:1}$ now contains the approximate eigenvectors of M.

In general, $\hat{M}_{:1}$ is not orthogonal. To obtain better clustering results, Fowlkes *et al.* [6] proposed the following orthogonalization procedure when A is psd. First, construct

$$S = M_{11} + M_{11}^{-\frac{1}{2}}\hat{M}_{21}^T\hat{M}_{21}M_{11}^{-\frac{1}{2}}. \tag{7}$$

Then, perform singular value decomposition (SVD) on S to obtain $U\Lambda T^T$. The orthogonal decomposition of $\hat{M}$ can be obtained as $\hat{M} = V\Lambda V^T$ and $V^TV = I$, where

$$V = \hat{M}_{:1}M_{11}^{-\frac{1}{2}}U\Lambda^{-\frac{1}{2}}. \tag{8}$$

The whole algorithm is summarized in Algorithm 1. When A is not psd, a more complicated two-step orthogonalization step has to be used [6].

The time complexity of Algorithm 1 is $\mathcal{O}(nm^2 + m^3)$. As for space, since $A_{:1}$ and $\hat{M}_{:1}$ need to be accessed more than once (e.g., in (4), (5), (6), (7), (8)), typically they have

Algorithm 1 Orthogonal Nyström approximation for a psd A [6].

1: $\hat{D} \leftarrow \text{diag}(A_{:1}A_{11}^{-1}A_{:1}^T\mathbf{1})$ and $D_{:1} \leftarrow \text{diag}(A_{:1}\mathbf{1})$.
2: $\hat{M}_{:1} = [M_{11}\hat{M}_{21}^T]^T \leftarrow \hat{D}^{-\frac{1}{2}}A_{:1}D_{11}^{-\frac{1}{2}}$.
3: $S \leftarrow M_{11} + M_{11}^{-\frac{1}{2}}\hat{M}_{21}^T\hat{M}_{21}M_{11}^{-\frac{1}{2}}$.
4: SVD: $S = U\Lambda T$.
5: $V \leftarrow \hat{M}_{:1}M_{11}^{-\frac{1}{2}}U\Lambda^{-\frac{1}{2}}$.

to be stored in main memory so as to avoid duplicate computations. However, this leads to $\mathcal{O}(nm)$ space complexity. In large-scale applications, even the sampling of a small subset of columns can result in the matrices A_1 and $\hat{M}_{:1}$ being too large to fit into main memory. Moreover, when A is not psd, the two-step orthogonalization procedure, which has a time complexity $\mathcal{O}(n^3)$, is often impractical [6].

3. Proposed Approach

Recall that spectral clustering only needs the k trailing eigenvectors of the normalized Laplacian L, or, equivalently, the k leading eigenvectors of $M = I - L$. Therefore, our goal is to find a rank-k approximation M_k of M. Let the eigen-decomposition of M_k be $U\Lambda U^T$, where Λ contains the k largest eigenvalues of M, and U contains the corresponding eigenvectors. As $M_k = MM_k^{-1}M$, so on using (3), we have

$$\begin{aligned} M_k &= D^{-1/2}AD^{-\frac{1}{2}}M_k^{-1}D^{-\frac{1}{2}}A^TD^{-1/2} \\ &= D^{-1/2}AS_AA^TD^{-1/2}, \end{aligned} \tag{9}$$

where

$$S_A = D^{-\frac{1}{2}}M_k^{-1}D^{-\frac{1}{2}} \tag{10}$$

is a rank-k matrix. Note that although this M_k is the best rank-k approximation of M, (9) requires the use of the whole affinity matrix A. This can be too expensive for large data sets.

3.1. A Good Approximate Degree Matrix

Note that AS_AA^T in (9) plays the role of the affinity matrix A in (3). Since the handling of A is expensive for large data sets, we consider in the following an approximation which only involves the m sampled columns $A_{:1}$ as in the Nyström method. As will be seen, this resultant approximation is in the form of AS_AA^T, and captures most of the information available in A. A comparison with some other possible approximations will be discussed in Section 3.4.

Recall that the definition of S_A in (10) relies on M_k, which in turn is the rank-k approximation of M in (9). Hence, we first define an analogous matrix $M_* \in \mathbb{R}^{m\times m}$ which is based on the submatrix A_{11}:

$$M_* = D_*^{-\frac{1}{2}}A_{11}D_*^{-\frac{1}{2}}, \tag{11}$$

where $D_* = \text{diag}(A_{11}1)$. Analogous to the definition of S_A in (10), we then define the following matrix $S_* \in \mathbb{R}^{m\times m}$ based on the rank-k approximation $M_{*,k}$ of M_*:

$$S_* = D_*^{-\frac{1}{2}}M_{*,k}^{-1}D_*^{-\frac{1}{2}}. \tag{12}$$

Obviously, $S_* = S_A$ when $m = n$. Finally, define, in a manner analogous to (9),

$$\hat{A} = A_{:1}S_*A_{:1}^T. \tag{13}$$

The following proposition shows that $\hat{A}$ can exactly recover the degrees of the m sampled columns. The proof is in Appendix A.

Proposition 1. *Assume that graph G has no more than k connected components. Then $A_{:1}1 = \hat{A}_{:1}1$.*

In other words, this approximation $\hat{A}$ contains all the degree information about the selected columns that is in A. The assumption of Proposition 1 is reasonable since k is typically chosen not less than the number of connected components; otherwise we obtain k eigenvectors according to the 0 eigenvalue of L and are not unique [16]. Moreover, $\hat{A}$ also has a close relationship with the Nyström approximation $\hat{A}_{\text{nys}}$ of A in (2). Specifically, when $k = m$, we have $\hat{A} = A_{:1}D_*^{-\frac{1}{2}}M_*^{-1}D_*^{-\frac{1}{2}}A_{:1}^T = A_{:1}A_{11}^{-1}A_{:1}^T = \hat{A}_{\text{nys}}$ on using (11). Thus, $\hat{A}$ is similar to $\hat{A}_{\text{nys}}$, but with a further rank-k approximation on M_*.

3.2. Rank-k Approximation of M

Let the eigen-decomposition of $M_{*,k}$ be $V\Lambda V^T$, where $V \in \mathbb{R}^{m\times k}$ and $\Lambda \in \mathbb{R}^{k\times k}$. On combining this with (12) and (13), we obtain $\hat{A} = Q\Lambda Q^T$, where

$$Q = A_{:1}D_*^{-\frac{1}{2}}V\Lambda^{-1}. \tag{14}$$

Since this $\hat{A}$ contains all the degree information about the selected columns, we can approximate the degree matrix as

$$\hat{D} = \text{diag}(\hat{A}1) = \text{diag}(Q\Lambda Q^T1).$$

Subsequently, a rank-k approximation of M can be obtained as

$$\hat{M}_k = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}} = \hat{D}^{-\frac{1}{2}}Q\Lambda Q^T\hat{D}^{-\frac{1}{2}} = U\Lambda U^T, \tag{15}$$

where $U = \hat{D}^{-\frac{1}{2}}Q$ contains its eigenvectors. Note that although the derivation of U above involves $\hat{A}$, it can be directly obtained from Q without first forming $\hat{A}$.

When the data set is large, storage of the affinity matrix A, or even its submatrix $A_{:1}$, can become infeasible. Fortunately, note that, here, $A_{:1}$ is only needed once in the computation of Q in (14). Thus, instead of storing the whole $A_{:1}$ in memory, we can obtain Q row by row. For its ith

row, we first compute the ith row of $A_{:1}$ (which contains the affinities between pattern i and the sampled patterns in $A_{:1}$), and then multiply it by $D_*^{-\frac{1}{2}} V \Lambda^{-1}$. Only the $n \times k$ matrix Q needs to be stored. The whole algorithm is shown in Algorithm 2.

Algorithm 2 The proposed algorithm.

Input: Data points $X = \{x_1, \ldots, x_n\}$ and m sampled points $Z = \{z_1, \ldots, z_m\}$ from X. Desired rank k.

Output: The approximated M_k: $\hat{M}_k = U \Lambda U^T$.

1: Form the affinity matrix $A_{11} \in \mathbb{R}^{m \times m}$ using Z.
2: $D_* \leftarrow \mathrm{diag}(A_{11} 1)$.
3: $M_* \leftarrow D_*^{-\frac{1}{2}} A_{11} D_*^{-\frac{1}{2}}$.
4: Perform partial eigen-decomposition of M_*. Keep the k largest eigenvalues in Λ and the k leading eigenvectors in V.
5: $B \leftarrow D_*^{-\frac{1}{2}} V \Lambda^{-1}$.
6: **for** $i = 1$ to n **do**
7: Form the affinity vector $a \in \mathbb{R}^{1 \times m}$ between x_i and points in Z.
8: Form the ith row of Q as aB.
9: **end for**
10: $\hat{D} \leftarrow \mathrm{diag}(Q \Lambda Q^T 1)$.
11: $U \leftarrow \hat{D}^{-\frac{1}{2}} Q$.

3.3. Time and Space Complexities

Since only partial eigen-decomposition is used in Algorithm 2, it takes $\mathcal{O}(nmk)$ time to compute k approximate eigenvectors. In contrast, as the procedure in [6] needs to compute A_{11}^{-1}, it will take $\mathcal{O}(nmk + m^3)$ time instead.

As for the space complexity, when the matrix $A_{:1}$ is small, it can be stored in main memory and so takes $\mathcal{O}(nm)$ space. However, when n is large, $A_{:1}$ may not be able to fit into memory. As discussed in the previous section, since the proposed algorithm only needs to access $A_{:1}$ once, we can compute it row by row. The space complexity is then reduced to $\mathcal{O}(nk)$. On the other hand, Algorithm 1 needs to use $A_{:1}$ more than once. To avoid computing $A_{:1}$ repeatedly, we will have to store it in memory, leading to $\mathcal{O}(nm)$ space. A comparison of the time and space complexities between [6] and the proposed algorithm is shown in Table 1.

	un-orthogonalized		orthogonalized	
	Fowlkes *et al.* [6]	ours	Fowlkes *et al.* [6]	ours
time	$\mathcal{O}(nmk + m^3)$	$\mathcal{O}(nmk)$	$\mathcal{O}(nm^2 + m^3)$	$\mathcal{O}(nmk)$
space	$\mathcal{O}(nm)$	$\mathcal{O}(nk)$	$\mathcal{O}(nm)$	$\mathcal{O}(nk)$

Table 1. Time and space complexities for computing k approximate eigenvectors.

3.4. Comparison with Other Approximations

Other ways to obtain a rank-k approximation of M are possible. A straightforward approximation can be obtained by first forming the rank-k Nyström approximation $\hat{A}^{(1)} = A_{:1} A_{11,k}^{-1} A_{:1}^T$ of A. Then, using (3), form a rank-k approximation of M as $\hat{M}_k^{(1)} = (\hat{D}^{(1)})^{\frac{1}{2}} \hat{A}^{(1)} (\hat{D}^{(1)})^{\frac{1}{2}}$, where $\hat{D}^{(1)} = \mathrm{diag}(\hat{A}^{(1)} 1)$ is the approximate degree matrix. However, note that even when we sample all the n columns from A to form $A_{:1}$, $\hat{M}_k^{(1)} \neq M_k$ in general and hence this cannot be expected to be a good approximation.

Another approximation, motivated from (10), is to use

$$S^{(2)} = D_{:1}^{-\frac{1}{2}} M_{11,k}^{-1} D_{:1}^{-\frac{1}{2}}, \tag{16}$$

instead of S_* in (12). Here, $D_{:1} = \mathrm{diag}(A_{:1} 1)$. It can be shown that the corresponding approximation of M becomes $\hat{M}_k^{(2)} = \hat{M}_{:1} M_{11,k}^{-1} M_{:1}^T$, which is similar to the Nyström approximation of M in (6) except for using a rank-k approximation on M_{11}. However, with this approximation, we will then need to access $A_{:1}$ twice, once to form $D_{:1}$ and another to calculate Q in (14). Subsequently, to avoid computing $A_{:1}$ twice, we will need $A_{:1}$ to be stored, which takes $\mathcal{O}(nm)$ space and can be infeasible when n is large.

Moreover, unlike S_* proposed in Section 3.1, these two approximations may not recover the degrees of $A_{:1}$ (as in Proposition 1). Empirical results in Section 4.2 confirm that these two alternatives are less accurate.

3.5. Orthogonalizing the Approximate Eigenvectors

Note that the approximate eigenvectors U obtained in (15) may not be orthogonal. In this section, we propose an inexpensive orthogonalization procedure which can transform any decomposition of the form $T = U \Lambda U^T$ to $T = \tilde{U} \tilde{\Lambda} \tilde{U}^T$, such that $\tilde{U}$ is orthogonal (i.e., $\tilde{U}^T \tilde{U} = I$). Our procedure is more general than Algorithm 1 in that it does not require T or Λ to be psd.

The orthogonalization procedure is shown in Algorithm 3. Note that $P = U^T U$ is always psd and so it is safe to use $\Sigma^{\frac{1}{2}}$ and $\Sigma^{-\frac{1}{2}}$. The following proposition shows the correctness of Algorithm 3.

Proposition 2. *In Algorithm 3,* $U \Lambda U^T = \tilde{U} \tilde{\Lambda} \tilde{U}^T$ *and* $\tilde{V}^T \tilde{V} = I$.

It can be easily seen that the time complexity of Algorithm 3 is $\mathcal{O}(nk^2)$ and its space complexity is $\mathcal{O}(nk)$. Thus, unlike the orthogonalization procedure in [6], this does not increase the time complexity of the proposed algorithm (Table 1).

Algorithm 3 Procedure to orthogonalize U, where $T = U\Lambda U^T$.

Input: $U \in \mathbb{R}^{n\times k}$, $\Lambda \in \mathbb{R}^{k\times k}$.
Output: orthogonalized $\tilde{U}$ and $\tilde{\Lambda}$.

1: $P \leftarrow U^T U$.
2: eigen-decomposition: $P = V\Sigma V^T$.
3: $B \leftarrow \Sigma^{\frac{1}{2}} V^T \Lambda V \Sigma^{\frac{1}{2}}$.
4: eigen-decomposition: $B = \tilde{V}\tilde{\Lambda}\tilde{V}^T$. Reorder the eigenvalues in $\tilde{\Lambda}$ and according eigenvalues if necessary.
5: $\tilde{U} \leftarrow UV\Sigma^{-\frac{1}{2}}\tilde{V}$.

4. Experiments

4.1. Spectral Clustering

In this section, we perform spectral clustering experiments on a number of data sets with various sizes (Table 2). The USPS, MNIST and MNIST8M[1] data sets contain handwritten digit images, with the raw pixels as features. We select the digits 0–4 for MNIST and MNIST8M. The RCV1[2] is an extensively used text data set. It contains categorized news from Reuters, and each news article is represented by a sparse log-transformed TF-IDF feature vector [9]. After removing news categories with fewer than $1,000$ articles, we obtain 55 categories that are used in the experiment. The ImageNet[3] is an image database containing images organized according to the WordNet hierarchy. Each image is represented as a bag of words based on the SIFT features. We select the 100 largest categories from the training set. Each of these contains an average of 2,000 images.

For the ImageNet data set, we use the histogram intersection kernel [14], which has been commonly used on SIFT features, to construct the affinity matrix. For the other data sets, we use the Gaussian kernel $\exp(-\mu\|x-y\|^2)$, where μ is set to the inverse of the average distance among all data pairs (*i.e.* $\mu^{-1} = \frac{1}{n^2}\sum_{ij}\|x_i - x_j\|^2$).

size	data	#samples	dim	#clusters
small	USPS	9,298	256	10
	MNIST	35,735	784	5
medium	RCV1	160,633	47,236	55
	image-net	191,050	1,000	100
large	MNIST8M	4,130,460	784	5

Table 2. Data sets used in the spectral clustering experiments.

4.1.1 Setup

The following spectral clustering methods will be compared:

1. Normalized cut algorithm [12] (denoted ncut).
2. k-means-based approximate spectral clustering [18] (denoted kasp).
3. Standard Nyström-based spectral clustering [6] (denoted psc).
4. The proposed algorithm (denoted ours).

The kasp is implemented[4] in R, while the others are in Matlab. The implementation for psc is from the highly optimized code used in [2][5]. The kasp and psc implementations can directly output the clustering results. For ncut and ours, after dropping the leading eigenvector of M, we perform k-means clustering using the remaining c leading eigenvectors, and with each data row normalized to unit norm. As the $c+1$ leading eigenvectors are required, we set $k = c+1$ for ours.

Recall that both psc and ours are based on column sampling. The number of sampled columns is fixed at $m = 1,000$, and these columns are selected by uniform sampling without replacement.

The following measures are used for performance evaluation: (1) accuracy; (2) normalized mutual information (NMI) [13], whose value is always between zero and one, and the higher the better; and (3) time. Moreover, we follow the common practice of permuting the obtained clusters to best match the ground truth. To reduce statistical variability, results for all the methods are averaged over 10 repetitions. All experiments, including the image segmentation in Sec. 4.2, are performed on a machine with Intel Xeon X5540 CPU and 16GB memory.

4.1.2 Results on Small and Medium-Sized Data Sets

Table 3 shows the results on the small data sets. The variances of the time measurements are very small and so are not reported. Moreover, only the orthogonalized versions of psc and ours are shown. Those for the un-orthogonalized versions are similar. As can be seen, the three methods sc, psc and ours, yield very similar accuracies and NMI values, while kasp is much inferior. This may be due to that a fixed kernel parameter (μ) has been used in our experiment, while the kasp experiments in [18] search μ over a range of possible values. As for the time, ours is faster than the others. Recall that the kasp implementation is in R while the others are in Matlab. Hence, the time comparison for kasp is for reference only.

On the medium-sized data sets, ncut fails to run because of insufficient memory. As for kasp, its implementation does not allow use of the sparse data format and histogram intersection kernel. Hence, it is also dropped from the comparison. Results comparing the orthogonalized and

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[2] http://alumni.cs.ucsb.edu/~wychen/sc
[3] http://www.image-net.org/challenges/LSVRC/2010/download-public
[4] http://www.stat.berkeley.edu/~dhyan/fasp.html.
[5] http://alumni.cs.ucsb.edu/~wychen/sc.html