2015 IEEE International Conference on Data Mining

Multi-level Approximate Spectral Clustering

Lijun Wang, Ming Dong, Alexander Kotov
Department of Computer Science
Wayne State University
Detroit, USA
Email: ljwang@wayne.edu, mdong@wayne.edu, kotov@wayne.edu

***Abstract*—Clustering is a task of finding natural groups in datasets based on measured or perceived similarity between data points. Spectral clustering is a well-known graph-theoretic approach, which is capable of capturing non-convex geometries of datasets. However, it generally becomes infeasible for analyzing large datasets due to relatively high time and space complexity. In this paper, we propose Multi-level Approximate Spectral (MAS) clustering to enable efficient analysis of large datasets. By integrating a series of low-rank matrix approximations (i.e., approximations to the affinity matrix and its subspace, as well as those for the Laplacian matrix and the Laplacian subspace), MAS achieves great computational and spacial efficiency. MAS provides a general framework for fast and accurate spectral clustering, which works with any kernels, various fast sampling strategies and different low-rank approximation algorithms. In addition, it can be easily extended for distributed computing. From a theoretical perspective, we provide rigorous analysis of its approximation error in addition to its correctness and computational complexity. Through extensive experiments we demonstrate superior performance of the proposed method relative to several well-known approximate spectral clustering algorithms.**

I. Introduction

Cluster analysis has been extensively applied to information discovery, text mining, Web analytics, and bioinformatics [1], [2]. Spectral clustering is a well-known graph-theoretic approach [3], which has gained popularity for its flexibility and ability to capture non-convex geometries of datasets. Despite these desirable properties, spectral clustering is prohibitively complex for analyzing large volumes of data, since it relies on similarity matrix and eigen-decomposition of an $n \times n$ Laplacian matrix, where n is the number of data points. This entire process generally requires $\mathcal{O}(n^2)$ space and $\mathcal{O}(n^3)$ time. In many cases, only a few top eigenvectors are needed. Thus, approaches such as subspace iteration, Krylov based methods, and randomized SVD [4] have been proposed to gain a lower time complexity. In the case that n is $100,000$, there will be ten billion elements in the similarity matrix, requiring about 37GB memory for storage. Computing of the Laplacian matrix and its eigen-decompsition are also on the same order of space complexity. The heavy computation costs and high memory demand make the traditional spectral clustering algorithm unsuitable for running on a single workstation. Distributed and cloud computing environments can be an alternative, however deployment of parallel spectral clustering [5] is also prohibitive due to nonlinear scalability with $\mathcal{O}(n^2)$ space complexity.

Many schemes have been previously proposed to reduce computational and space complexity of spectral clustering to enable its application to analysis of large datasets. One scheme uses quantization and downsampling during data preprocessing to reduce the size of a dataset. For example, the fast approximate spectral clustering by Yan *et al.* [6] applies k-means algorithm to reduce the dimensionality of a dataset. Dhillon *et al.* [7] proposed weighted graph cuts to address the scalability issue by generating multi-level shrunk graphs. Although it has been shown that such preprocessing minimizes the effect of dimensionality reduction on spectral clustering, including the preprocessing step results in heavy computational and storage overhead. The other scheme involves using low-rank approximation and allows to save memory. The methods proposed by Nyström [8] are based on numerical integration theory [9] and use randomly selected samples to approximate the affinity matrix, as well as the eigenvectors of the Laplacian. It has been shown that Nyström-based spectral clustering is empirically efficient for image segmentation [8] and clustering large datasets [6]. However, with random sampling, one cannot obtain the explicit connection between the reduction of data size and approximation accuracy. Kai *et al.* [10] addressed this issue by proposing an improved Nyström spectral clustering, in which the quantization error of sampling is minimized by choosing k-means cluster centers as representative points. More importantly, for very large datasets, all of the aforementioned algorithms can only choose a limited number of data samples, usually determined by the available memory. For example, for a dataset with four million data points and ten features, at most a few dozens of samples could be selected on a PC with 8GB RAM. This number is further reduced when the dataset has a large number of features. With an insufficient sampling, the algorithms often fail to achieve accurate matrix approximation, and consequently produce poor clustering results. How to efficiently select representative samples and seamlessly integrate various sampling strategies to approximate spectral clustering is the key factor for large data analytics and has not been well studied in the literature. Methods addressing this issue will

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.38

also be useful when one wants to adapt spectral clustering algorithms to distributed computing environment.

In this paper, we propose Multi-level Approximate Spectral clustering (MAS) along with various fast sampling strategies. The main idea of the proposed method is that an approximate embedding space can be obtained and used for spectral clustering by sequentially computing a series of low-rank matrix approximations (i.e., approximations to the affinity subspace), then to the affinity matrix, followed by an approximation to the graph's Laplacian matrix, and finally to the eigenvectors. Although several methods for approximate spectral clustering with low-rank matrix approximation have been previously proposed [11], [12], MAS differs from them in several important ways.

First, MAS provides a general framework for fast spectral clustering that works with any kernels and various low-rank approximation algorithms [13]. Second, instead of uniform sampling, MAS selects data samples based on a data-dependent nonuniform probability distribution, known as optimal sampling probabilities. According to [13], such sampling preferentially chooses data points that are more informative and more representative of the data, in a sense that they tend to be well correlated with more data points. The proposed fast sampling strategies make MAS highly applicable to data intensive applications. Third, MAS splits large volume of data into blocks and combines the approximation with each block, while [11] took an incremental approach. Therefore, MAS is more suitable for distributed computing. Fourth, from a theoretical perspective, we mathematically derive the approximation error of MAS in addition to proving its correctness and linear computational complexity. Through extensive experiments performed on real-world datasets, we demonstrate the superior performance of MAS for clustering large-scale data relative to several well-known approximate spectral clustering algorithms.

The rest of the paper is organized as follows. The MAS model formulation and algorithm description are presented in Section II. Theoretical analysis of the approximation error and algorithm computational complexity is provided in Section III. Experimental results are reported in Section IV. Finally, we conclude in Section V.

II. Multi-level Approximate Spectral Clustering

Table I lists all major symbols. Our model is based on the theories in [14]. The key idea of MAS is, the top k eigenvectors of the Laplacian matrix L can be approximated by its judiciously-chosen subspace. Specifically, we first compute a representative subspace and low-rank approximation to the affinity matrix S, then we construct the corresponding approximate subspace of L, and finally an approximation to its top k eigenvectors is computed. With the proposed four-level sequential computation of low-rank matrix approximations, MAS is able to sample sufficient data points to achieve accurate approximation of a large dataset, while selection of many data samples is generally prohibitive using other methods due to limited memory. In our method, the affinity matrix is used for clustering, which is constructed by a standard kernel. Unlike the approach based on Gaussian kernel and uniform sampling proposed in [11], MAS provides a more general framework with fast, data-dependent sampling strategies that can work with any kernel. In addition, MAS splits large volumes of data into blocks and thus can be easily extended to distributed computing, unlike [11] which follows an incremental strategy.

Table I
Symbol Definitions.

Symbol	Definition
$A_{d\times n}$	data matrix with the size of $d \times n$
$S_{n\times n}$	the affinity matrix with the size of $n \times n$
$L_{n\times n}$	the Laplacian matrix with the size of $n \times n$
L_{sub}	the representative column matrix of L with the size of $n \times c$
X_{ij}	the entry(i,j) of matrix X
$X(i,:)$	the i^{th} row of X
$X(:,j)$	the j^{th} column of X
X^T	the transpose of X
c	the number of the selected columns (the size of subspace)
k	the number of the clusters
$C_{n\times c}$	the representative column matrix of S with the size of $n \times c$
$D_{n\times n}$	diagonal matrix with each degree on its diagonal
$U_{n\times k}$	the top k eigenvectors estimated by the proposed method
$\tilde{X}$	the approximation to a matrix X
I_C	the index set of the selected samples
$\|.\|_F$	the Frobenius norm

A. Computing the subspace and low-rank approximation of the affinity matrix

In MAS, the affinity matrix is used for clustering, which is constructed by a standard kernel. Given affinity matrix S, we need to compute the low-rank approximation $\tilde{S}$, as well as its corresponding subspace C. To generate the low-rank approximation to the affinity matrix in the MAS method, we employ the Nyström-based approximation algorithm [13], which uses numerical integration theory to generate the Nyström extension to the eigenvectors, and derive an approximation to the Gram matrix from the Nyström extension. Notice that an approximation to the affinity matrix may result in negative values. However, the non-negativity of an affinity approximation, in general, is not a prerequisite to approximate-based spectral clustering algorithms [8], including ours.

Specifically, according to the Nyström-based approximation, the affinity subspace C is first constructed by a set of columns (denote the index set as I_C) selected from a sampling distribution, $\{p_i\}_{i=1}^n$, such that $\sum_{i=1}^n p_i = 1$. Then, C is rescaled by,

$$C'(:,i) = C(:,i)/\sqrt{c \cdot p_i}; \tag{1}$$

where c is the number of samples. Next, a $c \times c$ matrix

$$W(i,:) = C'(I_C(i),:)/\sqrt{c \cdot p_i}, \tag{2}$$

is computed by the scaled intersection between the selected columns and the corresponding rows. Let W_k denote the best rank-k approximation to W, and W_k^{-1} the Moore-Penrose generalized inverse of W_k, based on the Nyström algorithm, we get an approximation to the affinity matrix as,

$$\tilde{S} = C' W_k^{-1} C'^T. \tag{3}$$

To achieve an accurate approximation, sampling is an important step. MAS provides a general framework, in which various sampling strategies can be utilized, subject to $\sum_{i=1}^{n} p_i = 1$ and $p_i \geq 0$. The simplest one is to use uniform sampling, according to which the samples are selected with equal probability ($p_i = \frac{1}{n}$) [15]. In addition, [13] proposes data-dependent nonuniform probability distribution, i.e.,

$$p_i = \frac{\|S(:,i)\|^2}{\sum_{j=1}^{n} \|S(:,j)\|^2}, \tag{4}$$

as the *optimal sampling probabilities* with respect to approximating S. That is, from the perspective of approximation precision, Equation (4) minimizes the approximation error, and thus can be used to generate an approximation with maximum accuracy. As stated in [13], this sampling preferentially chooses data points that are more informative and more representative of the data, in the sense that they tend to be well correlated with more data points. In addition, a general data-dependent nonuniform distribution is defined in [14] as,

$$p_i \geq \beta \frac{\|S(:,i)\|^2}{\sum_{j=1}^{n} \|S(:,j)\|^2}, \quad s.t. \sum_{i=1}^{n} p_i = 1, \beta \leq 1, \tag{5}$$

and is optimal if $\beta = 1$.

From Equation (5), the sampling probabilities in the affinity domain are computed based on S, which is usually dense even though the original matrix A is sparse. When the dataset is large, storing the whole affinity matrix becomes infeasible in practice. In the following, we provide two sampling strategies that can overcome this problem. In particular, the first one samples the data in the affinity space, and thus can be used for any non-negative kernels. The second one is designed specifically for a linear kernel, and computes an approximation to the data matrix, and then constructs an approximation to the affinity matrix and its subspace.

To sample the data in the affinity matrix, we propose an iterative procedure to compute a nearly-optimal sampling distribution based on the approximation to the affinity matrix. Specifically, we first initialize the sampling probabilities with a uniform distribution, i.e., $P(0) = \{p_i\}_{i=1}^{n}$, where $p_i = 1/n$. In the t^{th} iteration, we compute the sampling probabilities by averaging the results in previous iterations, i.e., $p_i = \frac{1}{t}\sum_{l=0}^{t-1} P(l)(i)$, from which the subspace C is selected, and then $\tilde{S}$ is computed using Equation (3). Next, we compute $P(t)$ by

$$P(t)(i) = \frac{(\sum_{j=1}^{n} \tilde{S}_{ji}^{(t)})^2}{\sum_{l=1}^{n} (\sum_{j=1}^{n} \tilde{S}_{lj}^{(t)})^2}, \quad i \in [1,n], \tag{6}$$

where

$$\sum_{j=1}^{n} \tilde{S}_{ji}^{(t)} = (\tilde{S}^{(t)} \cdot \vec{\mathbf{1}})_i = (C'(W_k^{-1}(C'^T \vec{\mathbf{1}})))_i \tag{7}$$

and $\vec{\mathbf{1}}$ is a n-dimensional column vector of ones. Now we state,

Proposition 1. *Equation (6) provides a set of nearly-optimal probabilities for sampling the columns of S.*

Proof: Since S is the affinity matrix, all entries are non-negative. Hence,

$$\begin{aligned}
p_i &= \frac{(\sum_{j=1}^{n} \tilde{S}_{ji})^2}{\sum_{l=1}^{n} (\sum_{j=1}^{n} \tilde{S}_{lj})^2} \\
&= \frac{\sum_{j=1}^{n} \tilde{S}_{ji}^2 + 2\sum_{r,l\in[1,n]} (\tilde{S}_{ri}\tilde{S}_{li})}{\sum_{l=1}^{n}\sum_{j=1}^{n} \tilde{S}_{lj}^2 + 2\sum_{j,r,l\in[1,n]} (\tilde{S}_{rj}\tilde{S}_{lj})} \\
&\geq \frac{\sum_{j=1}^{n} \tilde{S}_{ji}^2}{\sum_{l=1}^{n}\sum_{j=1}^{n} \tilde{S}_{lj}^2 + 2\sum_{j,r,l\in[1,n]} (\tilde{S}_{rj}\tilde{S}_{lj})} \\
&= \beta \frac{\sum_{j=1}^{n} \tilde{S}_{ji}^2}{\sum_{l=1}^{n}\sum_{j=1}^{n} \tilde{S}_{lj}^2} \\
&\approx \beta \frac{\|S(:,i)\|^2}{\sum_{j=1}^{n} \|S(:,j)\|^2},
\end{aligned} \tag{8}$$

where

$$\beta = \frac{\sum_{l=1}^{n}\sum_{j=1}^{n} \tilde{S}_{lj}^2}{\sum_{l=1}^{n}\sum_{j=1}^{n} \tilde{S}_{lj}^2 + 2\sum_{j,r,l\in[1,n]} (\tilde{S}_{rj}\tilde{S}_{lj})} \leq 1. \tag{9}$$

In addition, we have $\sum_{i=1}^{n} p_i = 1$. Hence, $\{p_i\}_{i=1}^{n}$ are nearly-optimal probabilities for sampling the columns of S. The proof is completed. ■

In Proposition 1, we prove that our sampling strategy is nearly-optimal given the current approximation. In Algorithm 1, we present a heuristic, iterative method, in which the averaged distribution is used in each step to sample S, such that we can achieve a good balance between covering the entire sample space and selecting the most representative samples.

The second sampling method is based on the original matrix A, when the affinity matrix is computed by a linear kernel, i.e., $S = A^T A$. The linear kernel may have negative values, when A contains negative entries. To make a linear kernel non-negative, we need to normalize S. Specifically, the dot product of $A(:,i)$ and $A(:,j)$ represents the cosine value between the i^{th} and j^{th} data points, so the range of values in S is [-1,1], and the linear kernel can be rewritten as $S = A^T A + E$ to remove the negative entries, where E is

Algorithm 1 COMPUTEPROB

INPUT: the data matrix A, and the maximum iteration nMaxIter

OUTPUT: the sampling probabilities $\{p_i\}_{i=1}^n$

METHOD:

1) Initialize $P(0) = \{p_i\}_{i=1}^n$ with $p_i = 1/n$, and t = 1;
2) Do
   a) Sampling the subspace C under $p_i = \frac{1}{\mathrm{t}} \sum_{l=0}^{\mathrm{t}-1} P(l)(i)$ and scale it by Equation (1);
   b) Compute $\tilde{S}$ by Equation (3);
   c) Compute sampling probabilities by Equation (6) to get P(t);
   d) t = t + 1;

   until t == nMaxIter;
3) return $\{p_i\}_{i=1}^n$, where $p_i = \frac{1}{\mathrm{nMaxIter}} \sum_{l=0}^{\mathrm{nMaxIter-1}} P(l)(i)$.

a $n \times n$ matrix of ones. To generate $\tilde{S}$ and its representative subspace, we first compute the low-rank approximation and subspace of the original matrix A as,

$$\tilde{A} = T_0 M_0 R_0, \tag{10}$$

where T_0 is regarded as a subspace of A. Then, the approximate subspace of the affinity matrix can be obtained by computing the similarities between data samples and the whole dataset as,

$$\tilde{C} = A^T T_0 + E, \tag{11}$$

where, again, E is an $n \times c$ matrix of ones. Correspondingly, the approximation to the affinity matrix can be computed by $\tilde{A}$ with the linear kernel as,

$$\tilde{S} = R_0^T M_0^T T_0^T T_0 M_0 R_0 + E. \tag{12}$$

To efficiently select the representative subspace and generate the compact approximation of data points, we employ the Colibri method [16], which samples data points with a sampling distribution optimal in the feature space. We state,

Proposition 2. *Define*

$$p_i = \frac{\|A(:,i)\|^2}{\sum_{j=1}^n \|A(:,j)\|^2}, \quad i \in [1,n], \tag{13}$$

where $A(:,i)$ is the i^{th} column of A, then $\{p_i\}_{i=1}^n$ are nearly-optimal for sampling the columns of S, when S is computed by the linear kernel, i.e., $S(i,j) = A(:,i)^T A(:,j) + 1$, s.t. $\|A(:,i)\| = \|A(:,j)\| = 1$ (the proof is omitted for abbreviation).

When n is extremely large, the subspace $C_{(n\times c)}$ is limited to having a very small volume. Generally, approximation errors increase with the decrease of the subspace size. Hence, using a small subspace to approximate a large affinity matrix is not reliable. To solve this issue, we use $\tilde{C}$, a low-rank approximation of C, as the nearly representative subspace to compute $\tilde{S}$. As a general approach to solve the scalability problem, $\tilde{C}$ can be applied to any sampling.

Suppose we have selected a large volume of samples $C_{(n\times c)}$, which cannot be loaded completely in RAM. To utilize $C_{(n\times c)}$ and save both time and space, we compute its low-rank matrix approximation. We first split $C_{(n\times c)}$ into l sub-column blocks $\{C_{i(n\times m)}\}_{i=1}^l$ ($c = l \times m$), and each is fit into RAM. Consequently, C is obtained as $C = [C_1, C_2, ..., C_l]$. Next, we modify the Colibri method [16] to sequentially process block data and compute $\tilde{C}_{(n\times c)}$ with the form,

$$\tilde{C} = T_1 M_1 R_1, \tag{14}$$

where T_1 contains r real columns in C that are linearly-independent; M_1 is called "core matrix" with the size of $r \times r$, which is updated with each sample selected and finally equals to $(T_1^T T_1)^{-1}$; and R_1 is of the size $r \times c$ and computed by

$$R_1 = T_1^T C = [T_1^T C_1, T_1^T C_2, ..., T_1^T C_l].$$

Finally, we obtain the corresponding low-rank approximation to S as

$$\tilde{S} = \tilde{C}' W_k^{-1} \tilde{C}'^T, \tag{15}$$

where $\tilde{C}'$ is the result after scaling $\tilde{C}$ by Equation (1).

Regarding the spatial requirements, $\tilde{C}$ needs $(n \times r + r \times r + r \times c)$ units, compared with C for $(n \times c)$ units. Due to $(r \ll c)$ and $(r, c \ll n)$, the use of $\tilde{C}$ instead of C achieves great spatial savings for large-scale datasets. From the running time perspective, using $\tilde{C}$ to compute $\tilde{S}$ by Equation (15) results in great computational efficiency as well.

Here, we proposed several sampling strategies that can be used in different situations. If we know in advance that the data has low variance, uniform sampling can be adopted. Otherwise, optimal sampling strategy should be used. In the case that the size of data is large and the similarity matrix is not available, we can approximate the optimal probabilities by sampling either from the data matrix by Equation (13) or from the approximated similarity matrix by Equation (6). Finally, we also proposed a block method to help obtain sufficient data samples.

B. Compute the subspace of the Laplacian matrix

Given the subspace of the affinity matrix, we next compute the corresponding subspace of L. First, we compute the approximation of the degrees as,

$$\vec{\tilde{\mathbf{d}}} = \tilde{S}\vec{\mathbf{1}} \tag{16}$$

Then, the degree matrix $\tilde{D}$ is constructed as a diagonal matrix with each item of $\vec{\tilde{\mathbf{d}}}$ on its diagonal, and $\tilde{D}^{-\frac{1}{2}}$ with $1/\sqrt{\tilde{d}_i}$ on its diagonal. We also compute the subspace of

$\tilde{D}^{-\frac{1}{2}}$ as,

$$\tilde{D}_C^{-\frac{1}{2}} = \tilde{D}^{-\frac{1}{2}}(I_C, I_C), \tag{17}$$

where I_C is the index set of selected samples. Consequently, the approximation to L is

$$\tilde{L} = \tilde{D}^{-\frac{1}{2}} S \tilde{D}^{-\frac{1}{2}} \approx \tilde{D}^{-\frac{1}{2}} \tilde{S} \tilde{D}^{-\frac{1}{2}}, \tag{18}$$

and the subspace of $\tilde{L}$ is

$$\tilde{L}_{sub} = \tilde{D}^{-\frac{1}{2}} C \tilde{D}_C^{-\frac{1}{2}} \approx \tilde{D}^{-\frac{1}{2}} \tilde{C} \tilde{D}_C^{-\frac{1}{2}}. \tag{19}$$

We claim,

Proposition 3. *If C is a subspace of S, then $\tilde{L}_{sub}$ (Equation (19)) is an approximate subspace of L (the proof is straightforward and omitted for abbreviation).*

Consequently, we conclude that if $\tilde{C}$ is an approximation to the subspace of S, then $\tilde{L}_{sub}$ is an approximation to the subspace of L.

According to the LINEARTIMESVD algorithm in [14], $\tilde{L}_{sub}$ needs to be selected based on nearly-optimal probabilities (refer to Equation (5)) in the Laplacian domain so that the top eigenvectors can be approximated with high accuracy. In our model, we generate $\tilde{L}_{sub}$ according to the subspace of S. We claim that,

Proposition 4. *Given the optimal probabilities $\{p_i\}_{i=1}^n$ for sampling the columns of the affinity matrix S, where*

$$p_i = \frac{\|S(:,i)\|^2}{\sum_{j=1}^n \|S(:,j)\|^2}, \tag{20}$$

then, $\{p_i\}_{i=1}^n$ are nearly-optimal probabilities for constructing $\tilde{L}_{sub}$ in the Laplacian domain. Further, the nearly optimal probabilities computed by Equations (5, 6, 13) are nearly-optimal for sampling the columns of the Laplacian matrix (the proof is omitted for abbreviation).

C. Compute the approximation to the top eigenvectors of L

In the LINEARTIMESVD algorithm [14], it is shown that the top singular values and the corresponding singular vectors of a matrix could be approximated from its subspace matrix, selected under the nearly-optimal probabilities. Based on the theories in [14], we state that

Proposition 5. *By using MAS, the approximation to the top eigenvectors (singular vectors) of L could be computed from $\tilde{L}_{sub}$ (the proof is omitted for abbreviation).*

Given $\tilde{L}_{sub}$, which consists of those c columns of $\tilde{L}$ that are selected based on the nearly-optimal probabilities, we first scale it by

$$\tilde{L}'_{sub} = \tilde{L}_{sub}(:,i)/\sqrt{cp_i}, \quad i \in [1,n], \tag{21}$$

where p_i is the sampling probability used in approximating the affinity matrix. Then, the singular values and left singular vectors of L can be approximated by those of $\tilde{L}'_{sub}$. Based on linear algebra, these can be calculated by first performing an SVD of ${\tilde{L}'}_{sub}^T \tilde{L}'_{sub}$ to compute the right singular vectors of $\tilde{L}'_{sub}$ as

$${\tilde{L}'}_{sub}^T \tilde{L}'_{sub} = V \Sigma V^T, \tag{22}$$

where V is the right singular vectors of $\tilde{L}'_{sub}$, and Σ is the singular values. Thus, the left singular vectors of $\tilde{L}'_{sub}$ can be calculated as

$$U = \tilde{L}'_{sub} * V(:, 1:k)\Sigma^{-\frac{1}{2}}, \tag{23}$$

which will be approximations to the left singular vectors (eigenvectors) of L. Algorithm 2 shows the entire process of the MAS method.

Algorithm 2 MAS

INPUT: the data matrix A, the maximum iteration nMaxIter, and the cluster number k
OUTPUT: Cluster labels
METHOD:

1) Compute C (or $\tilde{C}$) and $\tilde{S}$:
   a) Compute the sampling probabilities $\{p_i\}_{i=1}^n$ by Equation (5), COMPUTEPROB(A, nMaxIter), or Equation (13);
   b) Construct the subspace C or $\tilde{C}$ under $\{p_i\}_{i=1}^n$;
   c) Scale the subspace (Equation (1)) and compute $\tilde{S}$ (Equation (3) or (15)), or achieve $\tilde{S}$ by Equation (12) if sampling in the data space;
2) Compute the degree matrix by Equations (16)-(17);
3) Compute $\tilde{L}_{sub}$ by Equation (19);
4) Compute the approximations to the top k eigenvectors of L by Equations (21) - (23);
5) Normalize the row vectors: $U(t,:) \leftarrow U(t,:)./\|(U(t,:))\|_F, \quad t \in [1,n]$.
6) Regarding each row of U as a point, do clustering via k-means, and return the cluster labels.

III. Theoretical Analysis

A. Analysis of Approximation error

To examine the approximation error of the computed eigenvectors, we study the distance between the optimal (by SVD) and the approximated embedding spaces (by MAS) by projecting the Laplacian matrix to the space spanned by the top singular vectors.

We claim

Proposition 6. *Let H_k represent the top k eigenvectors of L by SVD and U_k be the ones from $\tilde{L}$ by MAS, then*

$$\begin{aligned} &\|H_k H_k^T L - U_k U_k^T \tilde{L}\|_F^2 \\ \leq &\|L - L_k\|_F^2 + \|L - \tilde{L}\|_F^2 + \\ &\|\tilde{L} - \tilde{L}_k\|_F^2 + 2\sqrt{k}\|\tilde{L}\tilde{L}^T - \tilde{L}_{sub}\tilde{L}_{sub}^T\|_F. \end{aligned} \tag{24}$$

where L_k is the best rank-k approximation to L, and $\tilde{L}_k$ is the best rank-k approximation to $\tilde{L}$ (the proof is omitted).

In Proposition 6, we notice $\|L-L_k\|_F^2$ is a constant, which equals to $\sum_{t\geq k}\sigma_t^2(L)$ (the sum of the squares of small singular values of L). $\|L-\tilde{L}\|_F^2$ provides the approximation error to the Laplacian matrix,

$$\begin{aligned}\|L-\tilde{L}\|_F^2 &= \sum_{i,j\in[1,n]}(\frac{S_{ij}}{\sqrt{d_i d_j}}-\frac{S_{ij}}{\sqrt{\tilde{d}_i\tilde{d}_j}})^2 \\ &\leq \varepsilon\|S\|_F^2, \end{aligned} \tag{25}$$

where $\varepsilon = \max_{i,j\in[1,n]}(\frac{1}{\sqrt{d_i d_j}}-\frac{1}{\sqrt{\tilde{d}_i\tilde{d}_j}})^2$. As seen, the approximation error of the Laplacian matrix is related to that of the degrees. From Equation (16), the degree vector is computed with $\tilde{S}$, so the approximation error of the degrees is,

$$\|\vec{\tilde{\mathbf{d}}}-\vec{\mathbf{d}}\|_F = \|\tilde{S}\vec{\mathbf{1}}-S\vec{\mathbf{1}}\|_F = \|(\tilde{S}-S)\vec{\mathbf{1}}\|_F. \tag{26}$$

Since

$$\|(S-\tilde{S})E\|_F = \sqrt{n}\|(S-\tilde{S})\vec{\mathbf{1}}\|_F \leq \|S-\tilde{S}\|_F\|E\|_F, \tag{27}$$

where E is a $n\times n$ matrix with all ones, we get

$$\|\vec{\tilde{\mathbf{d}}}-\vec{\mathbf{d}}\|_F \leq \sqrt{n}\|S-\tilde{S}\|_F. \tag{28}$$

Hence, the approximation error to the degrees is related to that of the affinity matrix. In our model, we use the Nyström-based approximation [13] to compute the affinity matrix, which has the following error bound (THEOREM 3 in [13]),

$$\|S-\tilde{S}_k\| \leq \|S-S_k\|_F+\epsilon\sum_{i=1}^{n}S_{ii}^2, \quad s.t., \epsilon>0, \quad c\geq\frac{64k\eta^2}{\epsilon^4}, \tag{29}$$

where $\tilde{S}_k$ is the rank-k approximation to S, and S_k is the best rank-k approximation. Clearly, the approximation error in the affinity matrix depends on ϵ, which is expected to be small in accurate approximations. Since ϵ is related with c by $c\geq\frac{64k\eta^2}{\epsilon^4}$, c, the number of samples, should be large.

For $\|\tilde{L}-\tilde{L}_k\|_F^2$, it equals $\sum_{t\geq k}\sigma_t^2(\tilde{L})$, and $\|\tilde{L}\tilde{L}^T-\tilde{L}_{sub}\tilde{L}_{sub}^T\|_F$ has the error bound as follows (THEOREM 1 in [17]),

$$\|\tilde{L}\tilde{L}^T-\tilde{L}_{sub}\tilde{L}_{sub}^T\|_F \leq (\frac{\eta^2}{\beta c})^{\frac{1}{2}}\|\tilde{L}\|_F^2, \tag{30}$$

Thus, with large c, $\|\tilde{L}\tilde{L}^T-\tilde{L}_{sub}\tilde{L}_{sub}^T\|_F$ is also small.

The above analysis shows that when c is large, the approximation error of the computed eigenvectors is small. That is, sampling sufficient data points in practice is necessary to achieve accurate approximation. This is also the main advantage of MAS over existing methods.

B. Efficiency analysis

From Algorithm 2, the first step of MAS is to compute the sampling probabilities, which use $\mathcal{O}(n)$ time. Then, generating $\tilde{S}$ requires $\mathcal{O}(nc+c^3)$ time by the method in [13], and computing $\tilde{C}$ needs $\mathcal{O}(nc+c^3)$ time by the Colibri method [16]. Hence, the total time for computing $\tilde{C}$ and $\tilde{S}$ is $\mathcal{O}(n)$. Next, computing the degree matrix requires $\mathcal{O}(2cn+c^2)$ time, $\tilde{L'}_{sub}^T\tilde{L'}_{sub}$ for $\mathcal{O}(n+nc+nc^2+2c^3)$, its SVD for $\mathcal{O}(c^3)$, and U_k for $\mathcal{O}(k(ck+c^2+nc+n))$. Note that because c is a very small number, we choose the standard SVD which requires $\mathcal{O}(c^3)$ time (Equations 22 and 23). This time complexity can be further reduced using, for example, randomized SVD. Finally, k-means is used to cluster U_k with each row as a point, which needs $\mathcal{O}(lnk)$ time, where l is the number of iterations. Since $c\ll n$ and $k\ll c$, the total time of MAS is $\mathcal{O}(nc^2+lnk)$.

IV. Experiments and results

In this section, we evaluate the clustering performance of MAS on several real-world datasets, by comparing it with the state-of-the-art approximate spectral clustering methods. The similarity matrix is computed using the standard linear kernel. All the algorithms were implemented using MATLAB 7. The experiments were performed on a machine with a 3.0GHz Intel Xeon CPU, 3.0GB RAM and the Windows operating system. All the reported results have been averaged over 10 runs.

A. Evaluation methods

In our experiments, we implemented several algorithms under the MAS framework by applying different sampling strategies. The first one is C-MAS, in which the sample distribution is computed by Algorithm 1; the second one is D-MAS, where sampling probabilities are computed based on the data matrix (Equation (13)); the third one is S-MAS, where optimal sampling probabilities for the affinity matrix S are computed (Equation (4)); the fourth one is L-MAS, where optimal sampling probabilities for the Laplacian matrix L are computed (Equation (4) with L instead); and the fifth one is U-MAS, where uniform sampling ($p_i=1/n$) without replacement is employed (same as in [11]). In order to evaluate the MAS approach, we compare it with three state-of-the-art approximate spectral clustering algorithms, including Nyström-based spectral clustering, i.e., Nyström with random sampling [8], Nyström with k-means as a preprocessor [10], and a quantization-based fast spectral clustering, KASP [6]. In our experiments, various subspace sizes c are selected to test the performance of MAS and Nyström, which also serve as the numbers of representative points in the k-means step of Nyström and KASP. To execute k-means, we use random initialization and 500 iterations.

All our comparisons are conducted by using the following evaluation metrics:

1) Normalized mutual information (NMI) [18]: The NMI value is computed from the confusion matrix based on the true and predicted cluster labels, ranging in

$[0, 1]$. A high NMI value indicates that the predicted clustering matches the ground truth well.

2) Approximation Error: To evaluate the approximation accuracy of eigenvectors, we compute the sum-square-error (SSE) by

$$SSE = \frac{\|L - U_k U_k^T L\|_F}{\|L\|_F}, \tag{31}$$

where L is the Laplacian matrix, and U_k is a matrix with the computed eigenvectors.

Table II
REAL-WORLD DATASETS.

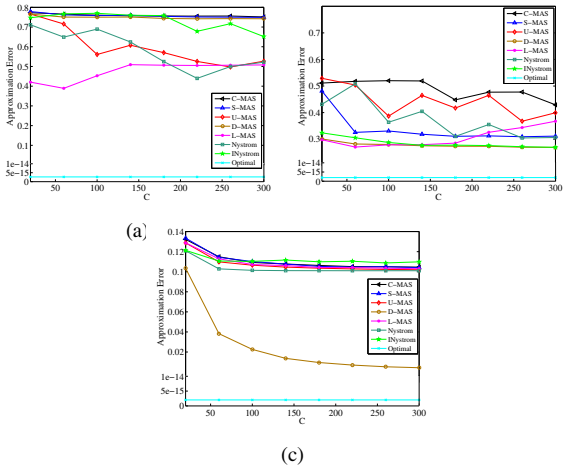
Medium-size datasets			
Name	*Category*	*No. of clusters*	*Size of data* ($d \times n$)
WebKB	*web pages*	*6*	$1,938 \times 4,991$
Reuters	*texts*	*10*	$19,418 \times 2,301$
Letter	*images*	*26*	$16 \times 15,000$
Rcv	*texts*	*52*	$47,236 \times 15,564$
Newsgroup	*texts*	*20*	$6,163 \times 18,846$
Large-size datasets			
Name	*Category*	*No. of clusters*	*Size of data* ($d \times n$)
Real-Sim	*texts*	*2*	$2,011 \times 72,120$
MNIST	*images*	*10*	$784 \times 76,054$
Acoustic	*time series*	*3*	$50 \times 78,823$
Webspam	*web pages*	*2*	$128 \times 154,123$

B. Data Description

To test the proposed method, we collect a few real-world large-scale datasets for various applications, including text mining, web pages grouping, images clustering, and time series analysis. Specifically, the first real-world dataset we used is WebKB [1], a WWW-pages collection from computer science departments of various universities. We process each web page as a document and mainly use $4,991$ web pages with six categories in the experiment. Reuters-21578, Distribution 1.0 [2] is a collection of documents from Reuters newswire, which contains $21,578$ documents from 135 topics. In our experiments, we use a subset of this collection with 10 clusters. Another text corpus we used is 20 $Newsgroups$ [3], a collection of approximately $20,000$ messages from UseNet news, partitioned (nearly) evenly across 20 different newsgroups. In addition, we collect six datasets from the website of LIBSVM [4] with clusters ranging from 2 to 52. Among them, Letter and Rcv are medium-size datasets with less than $20,000$ data points, and Real-Sim, MNIST, Acoustic, and Webspam are large-size datasets with over $70,000$ samples. For all the text collections, the common words are removed, and the meaningful words are stemmed using Porter's suffix-stripping algorithm. Table II gives the detailed description of all the datasets.

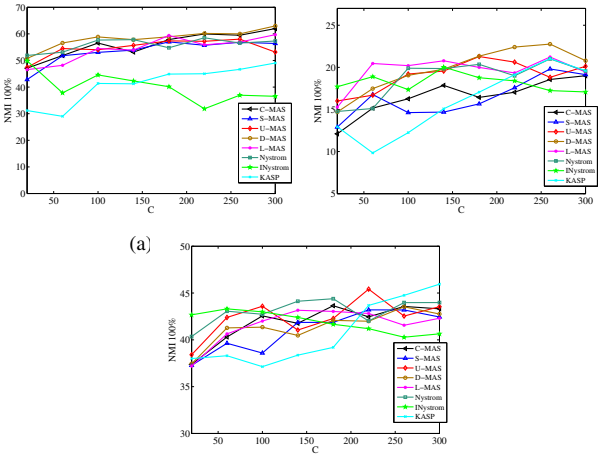
C. Experimental results

[1]http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/
[2]http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html
[3]http://people.csail.mit.edu/jrennie/20Newsgroups/
[4]http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/



(a) (c)

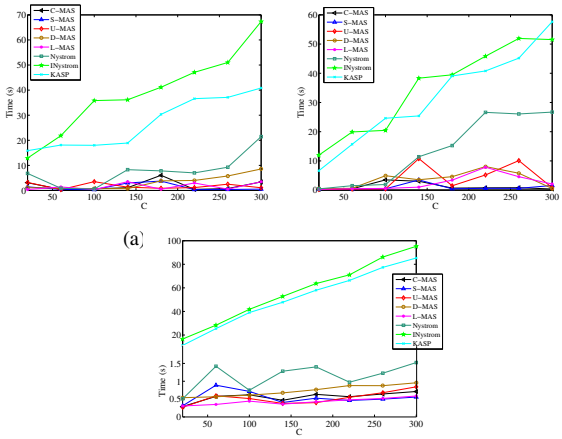
Figure 1. Comparison of approximation error with various c: (a) the results on Reuters; (b) the results on WebKB; and (c) the results on MNIST.



(a) (c)

Figure 2. Comparison of NMI with various c: (a) the results on Reuters; (b) the results on WebKB; and (c) the results on MINIST.

1) Results on c: Since sampling is an important step in the proposed method, we first study how subspaces selected by various sampling strategies affect the clustering performance. Among the five algorithms based on the MAS framework (see section IV-A), S-MAS computes the optimal sampling probabilities based on the affinity matrix S, and L-MAS with the Laplacian matrix L. Small datasets are necessary in these two cases to compute both S and L. In our experiment, we construct three small real datasets, by randomly selecting 500 data points from Reuters, WebKB, and MNIST, respectively, and evaluate the eight methods, namely C-MAS, D-MAS, S-MAS, L-MAS, U-MAS, Nyström, INyström, and KASP with respect to approximation error, clustering accuracy, and running time. In the following, we report the clustering results for different algorithms with c ranging from 10 to 300.



(a)

(c)

Figure 3. Comparison of running time with various c: (a) the results on Reuters; (b) the results on WebKB; and (c) the results on MINIST.

First, we compare the approximation error (Equation 31) with various subspaces. Since KASP doesn't provide eigenvectors, we only compare the MAS and Nyström algorithms. Besides, we use Normalized cut to generate the true eigenvectors to the Laplacian matrix, and compute its optimal rank-k approximation and the corresponding minimum SSE. In Figure 1, we plot the average SSE against c with each curve for an algorithm, and plot the minimum SSE as a flat dotted line, on Reuters, WebKB, and MINIST, respectively. As observed, seven algorithms achieve comparable results with different subspaces on the three datasets. In particular, L-MAS, U-MAS and Nyström gain more accurate approximation on Reuters, D-MAS and INyström on WebKB, and D-MAS on MNIST.

Next, we study the effects of subspace selection on clustering accuracy. In Figure 2, we plot the average NMI against c for eight algorithms. From this figure, we observe all algorithms achieve comparable clustering accuracy with various subspaces, except for INyström and KASP, which generally perform poorly on the datasets. As shown, the eight algorithms have a similar tendency with the increase of c: starting with a small and incomplete subspace, the performance is poor; when more exemplars are added, the subspace will include more bases of the data, so the NMI values increase, indicating the performance becomes better; until c increases over a certain point, e.g., $c = 100$ on Reuters, redundant bases will be included in the subspace, resulting in a stable performance. Also notice that when c is large, the clustering accuracy may be lower due to the inclusion of noisy samples, e.g., U-MAS with $c = 300$ on Reuters. We notice, with a complete subspace ($c = 100$ on Reuters, $c = 200$ on WebKB, and $c = 250$ on MNIST), D-MAS and C-MAS performs better than the other methods on Reuters, D-MAS on WebKB, and KASP does on MNIST.

In Figure 3, we plot the running time with varying values of c. The MAS algorithms generally run faster than the Nyström-based methods and KASP. Notice that Nyström can run as quickly as the MAS methods when c is small, however, it requires much more time as c increases. For example, Nyström requires more than double time of MAS when c is equal to 300. That is because the Nyström method needs to solve for the orthogonalized approximate eigenvectors, in which matrix multiplication takes much longer computational time when the subspace is large (refer to Section 3.3 in [8]). KASP runs slowly. Its running time on all the datasets is about 100 times longer than that of the MAS methods when c is greater than 100. INyström runs even slower than KASP. The slow computational speed of KASP and INyström is mainly due to the running of the preprocessing step.

In summary, 1) MAS runs more efficiently than the existing algorithms, especially when processing a large subspace, and thus highly applicable in real-world applications; 2) The proposed sampling schemes perform well under the MAS framework. They can achieve comparable approximation quality as the optimal sampling strategy. As shown next, the optimal sampling strategy is not applicable for large datasets.

2) Results on Medium-size Datasets: In the following experiments, on the medium-size datasets, we skip S-MAS and L-MAS due to their expensive computation of S and L, and only evaluate the other three MAS algorithms, i.e., C-MAS, D-MAS, and U-MAS. From a theoretical point of view, in order to achieve a high approximation accuracy, one should select as many data samples as possible. However, as shown in the previous experiment, sampling too many data points may add noises in the basis of the affinity and Laplacian subspaces, resulting in a lower clustering accuracy. In addition, using a large volume of the subspaces can add heavy loads on both storage and computation. In our experiments, we empirically set the maximum subspace size at 500 for all the datasets, and use three different volumes of subspace, i.e., $c = 100$, 300, and 500, to evaluate the performance of MAS.

In Table III, we present the clustering results of the six algorithms on the medium-size datasets with less than $20,000$ samples. Clearly, among all the methods, the D-MAS gains the highest clustering accuracy on most datasets in all subspaces, while C-MAS and U-MAS fall slightly behind D-MAS. Nyström based methods gain comparable accuracy with MAS. On Newsgroup20, INyström performs the best among the six. Compared with the other five methods, KASP generally performs poorly on these datasets.

The actual running time for each algorithm is also reported in Table III. The MAS algorithms generally run fast on all datasets. Nyström can run as quick as MAS on some data, e.g., Letter and Newsgroup, while needs more time on WebKB and Rcv. Due to the slow preprocessing step, INyström and KASP require a running time tens of times longer than the other three methods. Thus, they are not

Table III
EXPERIMENTAL RESULTS ON MEDIAN REAL-WORLD DATASETS AT THE SUBSPACE SIZES 100, 300, AND 500, INCLUDING NMI AND RUNNING TIME IN PARENTHESIS. BOLD INDICATES THE BEST PERFORMANCE (HIGHEST NMI OR SHORTEST TIME) FOR EACH DATASET IN EACH COLUMN.

Datasets	*Method*	*Result: $NMI \times 100\%$ (Running Time (s))*		
		$c = 100$	$c = 300$	$c = 500$
Reuters	*D-MAS*	*55.10 (1.31)*	***57.04*** *(2.49)*	***56.90*** *(4.86)*
	C-MAS	***55.21*** *(1.57)*	*56.47 (3.11)*	*56.85 (5.68)*
	U-MAS	*54.43 (**1.02**)*	*50.48 (**1.94**)*	*52.02 (**4.83**)*
	Nyström	*54.86 (1.07)*	*53.91 (9.49)*	*49.91 (27.67)*
	INyström	*53.98 (49.23)*	*48.06 (118.03)*	*49.48 (185.30)*
	KASP	*35.20 (46.94)*	*36.08 (113.65)*	*43.68 (170.54)*
WebKB	*D-MAS*	***21.39*** *(**1.65**)*	***21.34*** *(**2.68**)*	***21.83*** *(**3.57**)*
	C-MAS	*20.02 (1.81)*	*20.33 (3.84)*	*20.36 (8.52)*
	U-MAS	*17.06 (1.74)*	*17.75 (3.20)*	*17.88 (7.99)*
	Nyström	*15.57 (1.67)*	*17.79 (16.52)*	*18.02 (42.49)*
	INyström	*14.97 (128.21)*	*17.66 (312.18)*	*16.97 (440.86)*
	KASP	*15.84 (125.76)*	*16.28 (306.57)*	*20.27 (433.06)*
Letter	*D-MAS*	***39.46*** *(20.00)*	***43.89*** *(28.15)*	***43.74****(38.58)*
	C-MAS	*37.29 (**15.01**)*	*37.42 (**18.54**)*	*37.87 (**31.41**)*
	U-MAS	*34.53 (25.66)*	*35.96 (28.04)*	*34.99 (37.30)*
	Nyström	*30.28 (21.66)*	*32.51 (29.02)*	*39.11 (42.36)*
	INyström	*39.26 (193.99)*	*39.49 (566.71)*	*39.44 (951.68)*
	KASP	*33.24 (174.85)*	*31.93 (545.14)*	*34.28 (936.68)*
Rcv	*D-MAS*	*42.82 (**34.57**)*	*49.66 (**37.37**)*	***52.55*** *(**47.61**)*
	C-MAS	*38.15 (42.33)*	*45.48 (50.93)*	*48.44 (66.12)*
	U-MAS	*41.03 (39.93)*	*48.56 (55.14)*	*50.07 (62.57)*
	Nyström	*42.77 (37.66)*	*50.19 (52.22)*	*50.53 (85.97)*
	INyström	*49.05 (475.50)*	***51.57*** *(1061.80)*	*52.45 (1554.10)*
	KASP	***50.65*** *(444.60)*	*45.92 (1020.30)*	*45.51 (1522.3)*
Newsgroup	*D-MAS*	*27.75 (**16.22**)*	*34.25 (20.80)*	*37.56 (29.99)*
	C-MAS	*22.42 (16.30)*	*35.19 (25.38)*	*32.18 (31.78)*
	U-MAS	*21.93 (16.97)*	*32.32 (**20.15**)*	*34.30 (**25.46**)*
	Nyström	*26.83 (16.98)*	*33.96 (18.20)*	*38.02 (26.66)*
	INyström	***39.11*** *(409.50)*	***44.79*** *(977.60)*	***46.59*** *(1457.90)*
	KASP	*30.65 (393.90)*	*27.94 (953.50)*	*29.86 (1426.4)*

suitable to handle large, high-dimensional datasets.

3) Results on Large-size Datasets: We have also evaluated the MAS methods on four large-size datasets, each containing over $72,000$ data points. In our experiment, Nyström can select at most 100 samples, limited by the 3G memory in our PC. Since INyström and KASP require extremely large working space when processing these data, we skip these two algorithms in the following experiment. Besides, we also skip S-MAS and L-MAS due to expensive computation of S and L on the large-scale datasets.

In Figure 4(a), we show the comparison of clustering accuracy among the four methods with different subspace sizes (note that $c = 300$ and 500 are only applicable to the MAS methods) on the Real-Sim dataset. As seen, with the same subspace size, i.e., $c = 100$, MAS gains higher clustering accuracy than Nyström. When using larger subspace $c = 300$ and 500, the MAS methods perform better. This indicates that the subspace with $c = 100$ is incomplete, and not sufficient to achieve accurate clustering. Figure 4 (b) shows the comparison of the NMI values on the MNIST dataset. We notice, with $c = 100$, U-MAS performs best among all the four methods, and D-MAS achieves a comparable clustering accuracy with Nyström, while C-MAS falls slightly behind. However, with c increasing, all MAS algorithms gain better clustering performance. In Figure 4 (c), we show the clustering results on Acoustic. Notice when $c = 500$, the clustering accuracy for the MAS algorithms decreases slightly compared with that at $c = 300$. This indicates that the subspace with $c = 300$ contains enough bases for clustering. Further increasing the subspace size will include more redundant, noisy data samples, potentially leading to worse clustering performance. Finally, we present clustering accuracy on Webspam in Figure 4 (d). Clearly, the MAS methods perform much better than Nyström. As c increases, the MAS methods gain no noticeable improvement on clustering performance, indicating the subspace with $c = 100$ is sufficient for clustering.

Table IV
COMPARISONS OF RUNNING TIME (S) ON FOUR LARGE-SIZE DATASETS. BOLD INDICATES THE SHORTEST RUNNING TIME.)

Real-Sim				*MINIST*		
c	*C-MAS*	*D-MAS*	*U-MAS*	*C-MAS*	*D-MAS*	*U-MAS*
$c = 100$	*17.22*	***15.97***	*12.92*	*41.38*	***21.81***	*34.84*
$c = 300$	*50.22*	*46.53*	*58.90*	*82.05*	*70.61*	*61.68*
$c = 500$	*108.98*	*97.89*	*89.23*	*172.07*	*162.75*	*183.65*
Nyström	*16.56*			*37.18*		
Acoustic				*Webspam*		
c	*C-MAS*	*D-MAS*	*U-MAS*	*C-MAS*	*D-MAS*	*U-MAS*
$c = 100$	*22.08*	*21.29*	*24.05*	*48.11*	*43.50*	*62.90*
$c = 300$	*36.93*	*31.39*	*34.99*	*191.03*	*178.93*	*142.30*
$c = 500$	*136.95*	*136.91*	*119.68*	*248.45*	*235.19*	*235.79*
Nyström	***20.00***			***40.17***		

In Table IV, we report the actual running time for the three algorithms. The middle three lines show the time for the MAS methods with $c = 100$, 300, and 500, respectively, and the last line shows the running time required by Nyström with $c = 100$. As seen, D-MAS and Nyström request the lowest time at $c = 100$, while C-MAS and U-MAS need a little more time. At $c = 300$ and 500, we use the approximation of the subspace in the computation, so that more samples can be included. At $c = 300$, the computation of the low-rank approximation to the affinity subspace can be performed in the memory on all the large-size datasets except for Webspam, so all MAS algorithms require not much more time than that at $c = 100$. However, at $c = 500$, MAS uses extra space in the disk to compute $\tilde{C}$, resulting in frequent data swaps between the memory and the disk. Consequently, more computational time is required.

V. CONCLUSION AND FUTURE WORK

In this paper, we present a novel method for fast spectral clustering with multi-level approximations. By integrating a

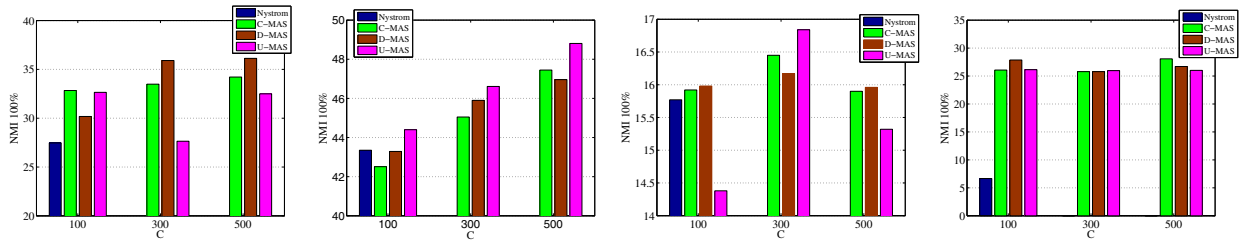


Figure 4. Comparison of clustering accuracy among Nyström, C-MAS, D-MAS, and U-MAS on the large-size datasets with $c = 100, 200, 500$, respectively: (a) the results on Real-Sim; (b) the results on MNIST; (c) the results on Acoustic; and (d) the results on Webspam.

series of low-rank matrix approximations, MAS gains efficiency in both computational time and space, which makes it possible to sample sufficient data points and filter out noisy samples, leading to accurate computation of eigenvectors, and consequently, clustering results. MAS can process large-scale datasets with millions of data points and dimensions on a single workstation. The major computation involved in the algorithm is the matrix multiplication and eigen-decomposition, both of which have been redesigned with a MapReduce mechanism and Apache Spark to run on a distributed platform [19]. In the future, we plan to implement MAS in a distributed environment.

REFERENCES

[1] J. Yin and J. Wang, "A dirichlet multinomial mixture model-based approach for short text clustering," in *KDD '14*, 2014, pp. 233–242.

[2] T. A. Bjørklund, M. Götz, J. Gehrke, and N. Grimsmo, "Workload-aware indexing for keyword search in social networks," in *CIKM '11*, 2011, pp. 535–544.

[3] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[4] E. Drinea, P. Drineas, and P. S. Huggins, "A randomized singular value decomposition algorithm for image processing applications," in *In Proceedings of the 8th panhellenic conference on informatics*, 2001, pp. 278–288.

[5] Y. Song, W.-Y. Chen, H. Bai, C.-J. Lin, and E. Y. Chang, "Parallel spectral clustering," in *ECML PKDD '08: Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, 2008, pp. 374–389.

[6] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *KDD '09*, 2009, pp. 907–916.

[7] I. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, 2004.

[9] C. T. H. Baker, *The numerical treatment of integral equations*. Oxford, New York, USA: Oxford: Clarendon Press, 1997.

[10] K. Zhang, I. W. Tsang, and J. T. Kwok, "Improved nyström low-rank approximation and error analysis," in *ICML '08*. New York, NY, USA: ACM, 2008, pp. 1232–1239.

[11] L. Wang and M. Dong, "Multi-level low-rank approximation-based spectral clustering for image segmentation," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2206 – 2215, 2012.

[12] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek, "Approximate spectral clustering," in *Advances in Knowledge Discovery and Data Mining*, vol. 5476, 2009, pp. 134–146.

[13] P. Drineas and M. W. Mahoney, "On the nyström method for approximating a gram matrix for improved kernel-based learning," *J. Mach. Learn. Res.*, vol. 6, pp. 2153–2175, 2005.

[14] P. Drineas, R. Kannan, and M. Mahoney, "Fast monte-carlo algorithms for matrices ii: Computing low-rank approximations to a matrix," *SIAM Journal on Computing*, vol. 36, pp. 158–183, 2006.

[15] P. Drineas, E. Drinea, and P. Huggins, "An experimental evaluation of a monte-carlo algorithm for singular value decomposition," Tech. Rep., 2003.

[16] H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos, "Colibri: fast mining of large static and dynamic graphs," in *KDD '08*, 2008, pp. 686–694.

[17] P. Drineas, R. Kannan, and M. Mahoney, "Fast monte carlo algorithms for matrices i: approximating matrix multiplication," *SIAM Journal on Computing*, vol. 36, pp. 132–157, 2006.

[18] A. Strehl, J. Ghosh, and C. Cardie, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[19] C. tao Chu, S. K. Kim, Y. an Lin, Y. Yu, G. Bradski, K. Olukotun, and A. Y. Ng, "Map-reduce for machine learning on multicore," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hoffman, Eds. MIT Press, 2007, pp. 281–288. [Online]. Available: http://papers.nips.cc/paper/3150-map-reduce-for-machine-learning-on-multicore.pdf