Large-Scale Nyström Kernel Matrix Approximation Using Randomized SVD

Mu Li, Wei Bi, James T. Kwok, and Bao-Liang Lu, *Senior Member, IEEE*

***Abstract*— The Nyström method is an efficient technique for the eigenvalue decomposition of large kernel matrices. However, to ensure an accurate approximation, a sufficient number of columns have to be sampled. On very large data sets, the singular value decomposition (SVD) step on the resultant data submatrix can quickly dominate the computations and become prohibitive. In this paper, we propose an accurate and scalable Nyström scheme that first samples a large column subset from the input matrix, but then only performs an approximate SVD on the inner submatrix using the recent randomized low-rank matrix approximation algorithms. Theoretical analysis shows that the proposed algorithm is as accurate as the standard Nyström method that directly performs a large SVD on the inner submatrix. On the other hand, its time complexity is only as low as performing a small SVD. Encouraging results are obtained on a number of large-scale data sets for low-rank approximation. Moreover, as the most computational expensive steps can be easily distributed and there is minimal data transfer among the processors, significant speedup can be further obtained with the use of multiprocessor and multi-GPU systems.**

***Index Terms*— Distributed computing, graphics processor, large-scale learning, low-rank matrix approximation, Nyström method, randomized SVD.**

I. Introduction

IN RECENT YEARS, kernel methods have been successfully applied in various real-world problems with highly complex and nonlinear structures. Well-known examples include the support vector machine, kernel Fisher discriminant analysis, and kernel principal component analysis [1]–[3]. In kernel methods, the kernel matrix plays the central role of describing the similarities among data samples. For applications in manifold learning and dimensionality reduction, the eigenvectors of this kernel matrix can also be used to reveal intrinsic clustering structures and low-dimensional data manifolds [4]–[6].

Given a set of n samples, the kernel matrix K is of size $n \times n$. This quadratic space complexity, together with the often-involved cubic time complexity, can be demanding in modern big data applications. A useful approach to reduce these computational burdens is to utilize the decaying spectra of kernel matrices and perform low-rank approximation. In other words, K is approximated by GG', for some $G \in \mathbb{R}^{n\times k}$. With $k \ll n$, the complexities associated in the handling of G is much lower than those with K. The optimal rank-k approximation (with respect to the spectral or Frobenius norm) can be obtained by a direct eigenvalue decomposition of K, and then construct G from the top k eigenvalues and eigenvectors.

However, standard eigenvalue decomposition algorithms take $O(n^3)$ time, which can again be prohibitive. Alternatively, one may perform partial singular value decomposition (SVD) using the Krylov subspace methods, such as the Arnoldi method [7]. However, time reduction is significant only when K is sparse and matrix–vector multiplications can be computed efficiently [8]. Moreover, it depends heavily on the singular spectrum structure of K. If there has no sharp jump or gap, convergence of the Krylov method can be slow and the obtained decomposition inaccurate. In addition, as the procedure is iterative, the matrix–vector multiplications involved cannot be performed in parallel. In addition, if k eigenvalues/eigenvectors are to be obtained, at least k passes over K will be required, which can be prohibitive for huge matrices [9].

A more general alternative is to use the Nyström method [8], [10]–[12]. It selects a subset of $m \ll n$ columns from K, and then uses the correlations between the sampled columns and the remaining columns to form a low-rank approximation of the full matrix. Since only a portion of K is sampled for computation, the time complexity can be reduced significantly. The space complexity is also smaller as only the sampled columns need to stored, while all the other matrices involved in the computation can be efficiently computed from them. This makes the Nyström method highly scalable. Fowlkes *et al.* [12] successfully applied it to spectral clustering for image segmentation. It has also been popularly used for tasks, such as Gaussian processes [8] and manifold learning [13]. A number of efficient, classical multidimensional scaling (MDS) algorithms, including the FastMap [14], MetricMap [15], and Landmark MDS [16], have all been

Manuscript received October 21, 2013; revised September 4, 2014; accepted September 7, 2014. Date of publication October 8, 2014; date of current version December 16, 2014. The work of M. Li, W. Bi, and J. T. Kwok was supported by the Research Grants Council, Hong Kong, under Grant 614012. The work of M. Li and B. L. Lu was supported in part by the National Natural Science Foundation of China under Grant 61272248, in part by the National Basic Research Program of China under Grant 2013CB329401, in part by the Science and Technology Commission of Shanghai Municipality under Grant 13511500200, and in part by the European Union Seventh Framework Programme under Grant 247619.

M. Li is with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA, and also with the Institute of Deep Learning at Baidu, Beijjng 100085, China (e-mail: muli@cs.cmu.edu).

W. Bi and J. T. Kwok are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: weibi@cse.ust.hk; jamesk@cse.ust.hk).

B.-L. Lu is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: bllu@sjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2014.2359798

shown to be variations of the Nyström algorithm [17]. More recently, it is further extended for semisupervised learning [18] and scaling up of nonlinear SVMs [19]–[21].

Computationally, the Nyström method only has to decompose a much smaller $m \times m$ matrix, which consists of the intersection of the selected columns and their corresponding rows. Obviously, the more columns are sampled, the more accurate is the resultant approximation. However, there is a tradeoff between accuracy and efficiency. When the data set is very large, even that small matrix may no longer be small in practice. For example, when there are several millions examples, sampling only 1% of the columns will lead to an intersection matrix that is larger than $10\,000 \times 10\,000$.

Instead of using only one Nyström approximation, Kumar *et al.* [22] recently proposed the use of an ensemble of n_e Nyström approximators (or *experts*). Empirically, this leads to a more accurate approximation, as the total number of columns sampled is larger than that of a single expert in standard Nyström. Moreover, its computational cost is (roughly) only n_e times larger.

Recently, a class of randomized algorithms is proposed for constructing approximate, low-rank matrix decompositions [9]. It also extends the Monte Carlo algorithms in [23], on which the analysis of the Nyström method in [11] is based. Unlike the standard Nyström method which simply samples a column subset for approximation, it first constructs a low-dimensional subspace that captures the action of the input matrix, and then a standard factorization is performed on the matrix restricted to that subspace. Althoughit is a randomized algorithm, it is shown that this can yield an accurate approximation with very high probability. On the other hand, the algorithm needs to have at least one pass over the whole input matrix, and is thus more expensive than the Nyström method, which only accesses a column subset. On very large data sets, this performance difference can be significant.

In this paper, we combine the merits of the standard Nyström method and the randomized SVD algorithm. The standard Nyström is highly efficient but requires a large enough number of columns to be sampled, while the randomized SVD algorithm is highly accurate but less efficient. Motivated by the observation that the ensemble Nyström algorithm is essentially using a block-diagonal matrix approximation, we will adopt a large column subset and then speed up the inner SVD step by randomized SVD. Both theoretical analysis and experimental results confirm that the error in the randomized SVD step is more than compensated for by the ability to use a large column subset, leading to an efficient and accurate eigenvalue decomposition even for very large input matrices.

Moreover, the proposed method can be easily extended for distributed computing that allows the use of more CPUs and GPUs. However, distributed computing also has some overheads. First, the network bandwidth and delay are often 1000 times worse than memory-to-CPU transfer, and so data transmission becomes more expensive. Second, as the nodes may have different processing speeds, the system needs to wait for the slowest node (straggler) to complete its update before the next iteration can proceed. In other words, the system can only move forward only at the pace of the slowest node. However, the proposed combination of Nyström and randomized SVD algorithms is highly suitable for distributed implementation because:

1) most parts of the Nyström method (except for the SVD part) can be naturally parallelized;
2) in practice, the randomized SVD algorithm only requires a few iterations, which significantly reduces the network traffic and synchronization cost.

The rest of this paper is organized as follows. Section II first reviews the Nyström and randomized SVD algorithms. Section III then describes the proposed algorithm, and its theoretical analysis. Section IV describes the extension to distributed environments. Experimental results on single machines, multiprocessor, and multi-GPU systems are presented in Section V. Finally, the last section gives some concluding remarks.

Some preliminary results have been reported in [24]. In addition to providing a more thorough literature review, this paper contributes the following.

1) Adds a new section on the implementations of the proposed algorithms on distributed environments.
2) Provides more intuitions and explanations.
3) While the error analysis in [24] holds only for uniform sampling of the column subset, this is extended to the more sophisticated non-uniform sampling scheme proposed in [11]. Moreover, technical proofs are included.
4) Provides stronger experimental evidence to demonstrate the merits of the proposed algorithms, particularly for its use on large data sets with multiprocessor and multi-GPU systems.

Notations: The transpose of vector/matrix is denoted by the superscript ^{T}. Moreover, $\mathrm{Tr}(A)$ denotes the trace of matrix $A = [A_{ij}]$, A^+ is its pseudoinverse, $\mathrm{ran}(A)$ its range, $A^{(i)}$ is the ith column of A, $\|A\|_2 = \max\{\sqrt{\lambda} : \lambda \text{ is an eigenvalue of } A^T A\}$ is its spectral norm, $\|A\|_F = \sqrt{\mathrm{Tr}(A^T A)}$ is its Frobenius norm, and $\sigma_i(A)$ denotes the ith largest singular value of A.

II. Related Work

A. Nyström Method

The Nyström method originates from the field of integral equations [10]. Given a symmetric positive semidefinite (psd) matrix $K \in \mathbb{R}^{n \times n}$, it obtains a low-rank approximation using a set C of $m \ll n$ columns sampled from K. Without loss of generality, we can reorder the columns and rows so that C and K is written as

$$C = \begin{bmatrix} W \\ E \end{bmatrix} \quad \text{and} \quad K = \begin{bmatrix} W & E^T \\ E & F \end{bmatrix} \tag{1}$$

where $W \in \mathbb{R}^{m \times m}$ is the matrix containing the intersection of C and the corresponding m rows of K, and $E \in \mathbb{R}^{(n-m) \times m}$, $F \in \mathbb{R}^{(n-m) \times (n-m)}$. For a given $k \leq m$, the Nyström method generates a rank-k approximation $\tilde{K}_k$ of K as

$$\tilde{K}_k = C W_k^+ C^T \tag{2}$$

where W_k is the best rank-k approximation of W. Let the SVD of W be $U\Lambda U^T$, where U is an orthonormal matrix and $\Lambda = \text{diag}(\sigma_1, \ldots, \sigma_m)$ is the diagonal matrix containing the singular values of W in nonincreasing order. Then, $W_k^+ = \sum_{i=1}^{k} \sigma_i^{-1} U^{(i)} U^{(i)^T}$. Recently, the Nyström method is further generalized to the inductive setting with the incorporation of side information [25].

Performing SVD on W takes $O(m^3)$ time, while the other matrix multiplications involved above take $O(nmk)$ time. Thus, the total time complexity is $O(m^3+nmk)$. Since $m \ll n$, this is much lower than the $O(n^3)$ complexity required by a direct SVD on K.

In using the Nyström method, an important issue is how to obtain the sampled columns. First, we introduce some notations in [26], which will also be used in our proofs. The sampling matrix is denoted $S \in \mathbb{R}^{n\times m}$, where $S_{ij} = 1$ if the ith column of K is chosen in the jth trial; and $S_{ij} = 0$ otherwise. The matrix

$$C = KSD \tag{3}$$

then consists of the sampled columns of G, possibly scaled by a diagonal matrix $D \in \mathbb{R}^{m\times m}$.

The simplest and most common sampling scheme is uniform sampling, with replacement [8] or without replacement [27]. A variety of more sophisticated sampling schemes have also been studied. For example, Drineas and Mahoney [11] proposed a nonuniform sampling (with replacement) scheme, in which column i of K is sampled based on the diagonal element in that column, with probability

$$p_i = \frac{K_{ii}}{\text{Tr}(K)}. \tag{4}$$

The jth selected column, denoted s_j, is then scaled as $(1/mp_{s_j})^{1/2}$, that is

$$D = \text{diag}\left(\frac{1}{\sqrt{mp_{s_1}}}, \ldots, \frac{1}{\sqrt{mp_{s_m}}}\right). \tag{5}$$

Ouimet and Bengio [28] studied a greedy sampling scheme based on the feature space distance between the candidate column and the span of previously chosen columns. Farahat *et al.* [29] proposed to select the column that constructs the best rank-1 Nyström approximation of the current residual matrix. Zhang *et al.* [21], [30] used k-means clustering to sample the columns.

Recently, the leverage score has also been used to define the sampling probabilities [27], [31]–[35]. Specifically, Drineas *et al.* [31] defined $p_i = l_i/k$, where $l_i = \|(U_k)_{(i)}\|^2$ is the leverage score, U_k is the matrix containing the k leading eigenvectors of K, and $(U_k)_{(i)}$ is the ith row of U_k. However, this requires eigen-decomposition, an operation that the Nyström algorithm aims to avoid in the first place. Faster schemes have been proposed to approximate the leverage scores [32], [33], though they still take $O(n^2 \log n)$ time, which is much more expensive than uniform sampling and the scheme in (4). Moreover, the leverage scores for kernel matrices of the linear and RBF kernels are relatively uniform, and empirically simple uniform sampling performs well [33]. Talwalkar and Rostamizadeh [35] studied the connection between matrix coherence and the performance of the Nyström method, and explained why nonuniform sampling may not outperform uniform sampling. An empirical comparison of some of these sampling schemes can be found in [27]. In addition, Wang and Zhang [36] proposed an adaptive sampling scheme and analyzed the sampling error. Furthermore, Yang *et al.* [37] discussed the difference of sampling strategy between the Nyström method and the random Fourier features.

B. Ensemble Nyström Algorithm

Instead of using one column subset, the ensemble Nyström method [22] repeats the column sampling procedure $n_e > 1$ times, producing a total of n_e Nyström approximations which are then combined together. Specifically, let m_e be the number of columns sampled by each base learner (Nyström approximator), and $C = [C^{(1)}, \ldots, C^{(n_e)}] \in \mathbb{R}^{n\times m_e n_e}$, with each $C^{(i)} \in \mathbb{R}^{n\times m_e}$, be the matrix that contains all the sampled columns. For each $C^{(i)}$, we obtain its rank-k approximation $\tilde{K}_k^{(i)}$ by the Nyström method. Finally, the obtained approximations are weighted to form an ensemble approximation as

$$\tilde{K}^{ens} = \sum_{i=1}^{n_e} \mu_i \tilde{K}_k^{(i)} \tag{6}$$

where μ_i's are the mixture weights. A number of choices for μ_i's has been investigated [22]. Empirically, the best choice is based on ridge regression, which optimizes a regression objective involving the μ_i's over a validation set of columns. The total time complexity of the ensemble Nyström method is

$$O(nn_e m_e k + n_e m_e^3 + C_{\boldsymbol{\mu}}) \tag{7}$$

where $C_{\boldsymbol{\mu}}$ is the cost of computing the mixture weights.

The above assumes that the ensemble Nyström method is implemented on one single machine. Typically, it is more computationally attractive when used in a parallel computing environment, with each base approximator being run independently on a separate machine. The n_e factor in (7) can then be removed, and the complexity reduces to $O(nm_e k + m_e^3 + C_\mu)$.

C. Randomized Low-Rank Approximation

Recently, Halko *et al.* [9] presented a class of simple but highly efficient randomized algorithms for computing low-rank matrix approximations. Given a matrix W, these randomized algorithms operate in two stages. First, using random sampling, it constructs a low-dimensional subspace to approximate the range of W. In the second stage, W is restricted to the obtained subspace and a standard decomposition (e.g., QR and SVD) of the reduced matrix is computed.

In the following, we elaborate on the procedure to obtain the rank-k SVD of a real[1] symmetric matrix $W \in \mathbb{R}^{m\times m}$. Specifically, we first generate an $m \times (k + p)$ standard Gaussian random matrix $\boldsymbol{\Omega}$ (i.e., each entry of $\boldsymbol{\Omega}$ is an independent Gaussian random variable with mean zero and variance one). We then form the matrix $Y = W\boldsymbol{\Omega}$ and construct a matrix Q whose columns form an orthogonal basis for

[1] In general, W can be a complex-valued rectangular matrix.

Algorithm 1 Randomized SVD [9]

Input: symmetric matrix $W \in \mathbb{R}^{m\times m}$, rank k, over-sampling parameter p, power parameter q.
Output: U, Λ.
1: $\Omega \leftarrow$ a $m \times (k+p)$ standard Gaussian random matrix;
2: $Z \leftarrow W\Omega$, $Y \leftarrow W^{q-1}Z$;
3: find an orthonormal matrix Q (e.g., by QR decomposition) such that $Y = QQ^TY$;
4: solve $B(Q^T\Omega) = Q^TZ$;
5: perform SVD on B to obtain $V\Lambda V^T = B$;
6: $U \leftarrow QV$.

the range of Y (e.g., by QR decomposition). The number of columns in Ω is often set to be slightly higher than the required rank k by an *over-sampling parameter* p. Typically, p is a small number such as 5 or 10, and enables $Y = W\Omega$ to have a better chance to span the k-dimensional subspace of W. In some applications, the spectrum of W may decay slowly, and the above scheme will produce a poor basis. To address this problem, one can left-multiply $W\Omega$ by W^{q-1} (i.e., $Y = W^{q-1}Z$, where $Z = W\Omega$), where q is the number of steps of the power iteration (typically a small integer, such as 1 or 2). Y then has the same singular vectors as W, but its singular values decay much faster as $\sigma_i(Y) = \sigma_i(W)^q$.

In the second stage, W is restricted to the obtained subspace from Y, leading to the reduced matrix $B = Q^TWQ$. A standard SVD is computed on B to obtain $V\Lambda V^T$. The SVD of W can then be approximated as

$$W \simeq QBQ^T = (QV)\Lambda(QV)^T. \tag{8}$$

The whole randomized SVD procedure is shown in Algorithm 1. As can be seen, it is easy to implement and can be applied on large-scale problems. On the theoretical side, it also has bounds guaranteeing its approximation errors. However, since it needs to have at least one pass over the whole input matrix, it is computationally more expensive than Nyström-based methods that only access a column subset. Specifically, it takes $O(m^2k)$ time[2] to compute Z and Y, $O(mk)$ time for the QR decomposition, $O(mk^2)$ time to obtain B, and $O(k^3)$ time for the SVD. Hence, the total time complexity is $O(m^2k + k^3)$, which is quadratic in m.

III. Combining Nyström With Randomized SVD

Obviously, the more columns are sampled, the more accurate is the Nyström approximation. Hence, the ensemble Nyström method samples a total of mn_e columns, instead of m columns for a single Nyström approximation. However, there is a tradeoff between accuracy and efficiency. If the standard Nyström method were used, this would have taken $O(m^3n_e^3)$ time for the SVD of the $mn_e \times mn_e$ W matrix.[3] The ensemble Nyström method alleviates this problem by replacing the expensive SVD by n_e SVDs on smaller $m \times m$ matrices.

[2]Here, we compute Y by multiplying W to a sequence of $m \times (k+p)$ matrices, as $WZ, W(WZ), \ldots, W(W^{q-2}Z)$.

[3]This matrix will be denoted $W_{(mn_e)}$ in the sequel.

Algorithm 2 Proposed Algorithm

Input: Psd matrix $K \in \mathbb{R}^{n\times n}$, sampling matrix $S \in \mathbb{R}^{n\times m}$, scaling matrix $D = \text{diag}\{d_1, \ldots, d_m\} \in \mathbb{R}^{m\times m}$, rank k, over-sampling parameter p, power parameter q.
Output: U and Λ.
1: $C \leftarrow KSD$; // sampled, scaled columns of K
2: $W \leftarrow DS^TC$; // $m \times m$ scaled submatrix
3: $[\tilde{U}, \Lambda] \leftarrow \text{randsvd}(W, k, p, q)$ using Algorithm 1;
4: $U \leftarrow C\tilde{U}\Lambda^+$.

Our key observation is that, using (2), the ensemble Nyström approximation in (6) can be rewritten as

$$\tilde{K}^{\text{ens}} = C\,\text{diag}\left(\mu_1\big(W_k^{(1)}\big)^+, \ldots, \mu_{n_e}\big(W_k^{(n_e)}\big)^+\right)C^T \tag{9}$$

where $W_k^{(i)} \in \mathbb{R}^{m\times m}$ is the W matrix in (1) corresponding to $\tilde{K}_k^{(i)}$, and $\text{diag}(\mu_1(W_k^{(1)})^+, \ldots, \mu_{n_e}(W_k^{(n_e)})^+)$ is the block-diagonal matrix

$$\begin{bmatrix} \mu_1\big(W_k^{(1)}\big)^+ & & \\ & \ddots & \\ & & \mu_{n_e}\big(W_k^{(n_e)}\big)^+ \end{bmatrix}.$$

In other words, the ensemble Nyström algorithm can be equivalently viewed as approximating $W_{(mn_e)}^+$ by the block-diagonal matrix $\text{diag}(\mu_1(W_k^{(1)})^+, \ldots, \mu_{n_e}(W_k^{(n_e)})^+)$. Despite the simplicity in (9), which allows the computations of $(W_k^{(i)})^+$'s to be distributed over parallel machines, the inverse of a block-diagonal matrix is another block-diagonal matrix. Hence, no matter how sophisticated the mixture weights [μ_i's in (9)] are estimated, this block-diagonality approximation is rarely valid unless $W_{(mn_e)}$ is block diagonal. This, however, is highly unlikely in typical applications of the Nyström method.

As pointed out in [27], one can replace the base algorithm in the ensemble (which is a standard Nyström algorithm) with better Nyström variants. In the following, we propose a novel approach that can obtain better performance in an efficient manner. As inspired by the ensemble Nyström method, the proposed method will also sample more columns, or, equivalently, use a $m \gg k$ that is much larger than the one typically used in the standard Nyström method. However, instead of using the block-diagonality simplification to decompose and distribute the smaller SVD problems over parallel machines, we will use a more accurate procedure to solve the large SVD problem directly on a serial machine. In particular, we will adopt the randomized low-rank matrix approximation technique reviewed in Section II-C.

The proposed algorithm is shown in Algorithm 2. Essentially, it combines the high efficiency of the Nyström method, which, however, requires a large enough column subset for accurate approximation, with the ability of the randomized algorithm to produce a very accurate SVD but still relatively efficient approximation. Using the approximate SVD $W \simeq \tilde{U}\Lambda\tilde{U}^T$ obtained in Step 3, K can be approximated as

$$K \simeq \hat{K} = C\tilde{U}\Lambda^+\tilde{U}^TC^T = (C\tilde{U}\Lambda^+)\Lambda(C\tilde{U}\Lambda^+)^T. \tag{10}$$

TABLE I
TIME COMPLEXITIES FOR THE VARIOUS METHODS TO OBTAIN A RANK-k NYSTRÖM APPROXIMATION OF AN $n \times n$ MATRIX, WITH $n \gg m$

method	time complexity
Nyström	$O(nmk + m^3)$
randomized SVD	$O(n^2k + k^3)$
proposed method	$O(nmk + m^2k + k^3)$

Instead of using the randomized SVD algorithm for the inner SVD, one might want to apply other approximations, such as using the standard Nyström method again. However, the Nyström method is not good at approximating the trailing eigenvalues, which are important in computing the inverse of W. Preliminary experiments show that in order for the resultant approximation on K to be accurate, the inner Nyström needs to sample close to m columns, which, however, will lead to little speedup over a standard SVD.

A. Time Complexity

The proposed algorithm is much more efficient than the standard Nyström method. Recall that the time complexity of the standard Nyström is $O(nmk + m^3)$. The $O(m^3)$ term resulting from the SVD step is now replaced by $O(m^2k + k^3)$ for the randomized SVD step in the proposed algorithm. Hence, its time complexity is $O(nmk + m^2k + k^3)$, which is much more efficient than the standard Nyström method as $m \geq k$.

A summary of the time complexities of the various methods is shown in Table I. Recall that $n \gg m$. All the methods scale linearly with n, except for randomized SVD which scales quadratically with n.

Moreover, if the *total* number of columns sampled in the ensemble Nyström method is the same as those in the other methods (i.e., $n_e m_e = m$), its complexity can be rewritten as $O(nmk + mm_e^2 + C_\mu)$ for the serial implementation. If we further assume that $m_e \simeq k$, this further reduces to $O(nmk + mk^2 + C_\mu)$, which is of the same order as the proposed algorithm. Recall, however, that the ensemble Nyström relies on the block-diagonality simplification, its approximation error might be larger than the proposed algorithm, unless the block-diagonality assumption is satisfied.

B. Error Analysis

In this section, we show that the proposed algorithm, which only performs an approximate SVD on the W matrix, is as accurate as the standard Nyström method, which performs an exact, but much more expensive SVD.

Recall from (3) that $C = KSD$ contains the sampled columns of K scaled by D. Assume that $K = X^T X$. Then, $H = XSD$ contains the corresponding sampled columns of X scaled by D. Moreover, $C = X^T H$, $W = H^T H$.

1) Approximation Error With Respect to the Spectral Norm

Theorem 1: For the $\hat{K}$ in (10), $\mathbb{E}\,\|K - \hat{K}\|_2 \leq \zeta^{1/q}\|K - K_k\|_2 + (1 + \zeta^{1/q})\,\mathbb{E}\,\|XX^T - HH^T\|_2$, where K_k is the best rank-k approximation of K, and $\zeta = 1 + \sqrt{k/p - 1} + e\sqrt{k + p}/p\sqrt{m - k}$.

Proof is in Appendix A. Note that this upper bounds the approximation error irrespective of the column sampling scheme, which is determined by $\mathbb{E}\,\|XX^T - HH^T\|_2$. The following two corollaries apply this theorem to the sampling schemes of [8] and [11], respectively.

Corollary 1: If we choose the columns of K uniformly at random without replacement and $D = \sqrt{n/m}I$ (where I is the identity matrix), then

$$\mathbb{E}\,\|K - \hat{K}\|_2 \leq \zeta^{1/q}\|K - K_k\|_2 + (1 + \zeta^{1/q})\frac{n}{\sqrt{m}}K_{ii}^* \quad (11)$$

where $K_{ii}^* = \max_i K_{ii}$.

Proof: Result follows on substituting in Corollary 5. ■

Remark 1: As shown in [9], the power iteration drives $\zeta^{1/q}$ toward 1 exponentially fast as q increases, and so the error in (11) decreases with the number of sampled columns m. In particular, if we replace $\zeta^{1/q}$ by 1, then the right-hand side of (11) becomes $\|K - K_k\|_2 + 2n/\sqrt{m}K_{ii}^*$, which is the same as that for the standard Nyström method using m columns (this can be obtained by combining (6) and (10) in [22]). In other words, Algorithm 2 is as accurate as performing a large SVD in the standard Nyström method.

Corollary 2: If we choose the columns of K according to probabilities defined in (4) and the scaling matrix D in (5), then

$$\mathbb{E}\,\|K - \hat{K}\|_2 \leq \zeta^{1/q}\|K - K_k\|_2 + (1 + \zeta^{1/q})\frac{1}{\sqrt{m}}\,\mathrm{Tr}(K).$$

Proof: Result follows on substituting in Corollary 6. ■

2) Approximation Error With Respect to the Frobenius Norm: A similar bound can be obtained for the approximation error in terms of the Frobenius norm. However, since there is no analogous theory for power iteration with respect to the Frobenius norm [9, Remark 10.1], the analysis here is restricted to $q = 1$ and the resultant bound is quite loose. However, as will be observed in Section V, empirically the approximation with just $q = 2$ is already very good.

Theorem 2: For the $\hat{K}$ in (10)

$$\mathbb{E}\,\|K - \hat{K}\|_F \leq \sum_{i > k+p}\sigma_i(K) + 2\zeta_F\|K - K_k\|_F + 4\zeta_F\,\mathbb{E}\,\|XX^T - HH^T\|_F \quad (12)$$

where K_k is the best rank-k approximation of K, and $\zeta_F = k + p/\sqrt{p - 1}$.

The proof is in Appendix B. As in Section III-B, the upper bound in Theorem 2 can be applied to various column sampling schemes.

Lemma 1: $\sum_{i > k+p}\sigma_i(K) \leq nK_{ii}^*$.

Corollary 3: If we choose the columns of K uniformly at random without replacement and $D = \sqrt{n/m}I$, then $\mathbb{E}\,\|K - \hat{K}\|_F \leq 2\zeta_F\|K - K_k\|_F + (1 + 4\zeta_F/\sqrt{m})nK_{ii}^*$, where $K_{ii}^* = \max_i K_{ii}$.

Proof: Result follows on substituting in the results in Corollary 5 and Lemma 1. ■

Corollary 4: If we choose the columns of K according to probabilities defined in (4) and the scaling matrix D in (5), then $\mathbb{E}\,\|K - \hat{K}\|_F \leq 2\zeta_F\|K - K_k\|_F + (1 + 4\zeta_F/\sqrt{m})\,\mathrm{Tr}(K)$.

Algorithm 3 Template for Distributing the Nyström Algorithm on N Nodes

1: divide all data points into N parts $\{X_1, \ldots, X_N\}$;
2: **for** node $i = 1, 2, \ldots, N$ **do** // in parallel
3: get the ith part of the data X_i;
4: get the m sampled points S, and the rank-k approximation of the corresponding $m \times m$ kernel matrix $W \approx \tilde{U}\Lambda\tilde{U}^T$;
5: compute kernel submatrix C_i between X_i and S;
6: compute $U_i \leftarrow C_i\tilde{U}\Lambda^+$;
7: **end for**
8: aggregate $U \leftarrow [U_1, \ldots, U_N]$.

Proof: Result follows on substituting in the results in Corollary 6 and Lemma 1. ■

IV. Distributed Implementation

In this section, we discuss the parallelization of the Nyström method in a distributed environment. This may be a multiprocessor system, in which each processing node has one or a few CPUs sharing the same main memory and stores different subsets of the data. Another popular setup is a multi-GPU system. As is well known, GPUs are most suitable for compute intensive, memory intensive, and highly parallel computations. The single precision (SP) and double precision (DP) floating point performance (FLOPS) of GPUs, as well as its memory bandwidth, typically far exceed those of state-of-the-art CPUs. Taking the NVIDIA Tesla C1060 GPU card as an example. Each card contains a T10 GPU, which is equipped with 240 streaming processor cores and can achieve 933 SP GFLOPS and 78 DP GFLOPS (peak), together with 102 GB/s memory bandwidth. In contrast, the Intel Core i7-980X CPU has 6 cores with a 3.3-GHz clock rate, and thus only 158.4 SP GFLOPS and 79.2 DP GFLOPS (peak), and 25.6 GB/s maximum memory bandwidth. A modern machine often holds up to 4 GPUs.

To fully demonstrate the advantages of distributed computing, the algorithm must achieve two criteria. First, data transfer between the different nodes should be minimal. Second, the computations should be easily parallelized.

A. Template Distributed Nyström Algorithm

Algorithm 3 shows a template for the distributed Nyström method. Note from the standard version (Algorithm 2) that C, the matrix containing the sampled columns, is of size $n \times m$, and thus may not fit into the node's memory (where the node can be a processing node in a multiprocessor system or a GPU card in a multi-GPU system) when n is very large. Hence, in Algorithm 3, the sampled data points (denoted S), but not the sampled columns of the kernel matrix, are stored in each node. The kernel evaluations between each sampled point and the rest of the data samples are computed on-the-fly. Specifically, the input data set is divided among the N nodes, as $\{X_1, \ldots, X_N\}$. Matrix U in Step 4 of Algorithm 2 can then be obtained as

$$U = C\tilde{U}\Lambda^+ = [C_1\tilde{U}\Lambda^+; C_2\tilde{U}\Lambda^+; \ldots; C_N\tilde{U}\Lambda^+] \quad (13)$$

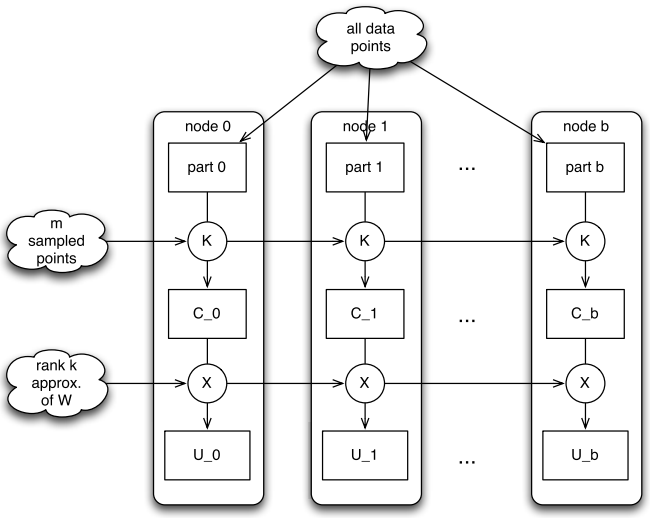


Fig. 1. Splitting the data into blocks in a distributed Nyström implementation.

where C_i is the kernel submatrix between X_i and S. A graphical representation is shown in Fig. 1. Note that Step 4 also requires a procedure to compute the rank-k approximation of the sampled $m \times m$ matrix W, which will be discussed in Section IV-B.

For very large data sets, node i's memory may not be sufficient to store both X_i and U_i. In this case, one can store X_i in some slower storage (such as hard-disk for a processing node or main memory for a GPU card). Iteratively, a row block of X_i is loaded, the corresponding submatrix of C_i computed, and subsequently the corresponding row block of U_i is computed and transferred back to the main memory, disk, or a distributed storage, such as Hadoop Distributed File System (HDFS). Finally, all the row blocks of U_i are combined to form U. To hide the communication cost, one can also perform prefetch and transfer in the background.

The most costly steps in Algorithm 3 are on computing the kernel submatrices C_is and obtaining U_i. Since most kernel functions (such as the linear and Gaussian kernels) involve simple vector inner products or distances, computation of C_i can be easily parallelized when GPUs are used. For the computation of U_i, it involves only matrix–matrix multiplication, which can again be efficiently handled[4] by GPUs.

B. Obtaining the Rank-k Approximation of W

The rank-k approximation in Step 4 of Algorithm 3 can be computed in a single node (say, node 0), and then broadcast the result to all the nodes. However, W may be too large to be efficiently processed by one single machine, and requiring all the other nodes to wait for the completion of W may be wasteful. This can be alleviated using the distributed randomized SVD procedure shown in Algorithm 4. Note that during the gather operation (Steps 7 and 11), a synchronization

[4]Empirically, matrix–matrix multiplications can achieve 60% of peak SP FLOPS and 97% of peak DP FLOPS on NVIDIA GPUs [38].

Algorithm 4 Distributed Randomized SVD on N Nodes

1: divide m sampled points into N parts, with m_i points in part i;
2: **for** node $i = 1, 2, \ldots, N$ **do** // in parallel
3: get all m sampled points S;
4: compute the $m_i \times m$ kernel submatrix W_i between S_i and S;
5: draw a $m \times (k + p)$ standard Gaussian random matrix Ω with the same random seed shared by all nodes;
6: $Z_i \leftarrow W_i \Omega$;
7: gather $Z \leftarrow [Z_1; \ldots, Z_N]$;
8: $Y \leftarrow Z$
9: **for** $j = 1, \ldots, q - 1$ **do**
10: $Y_i = W_i Y$;
11: gather $Y \leftarrow [Y_1; \ldots, Y_N]$;
12: **end for**
13: **end for**
14: obtain U by performing Steps 4–6 of Algorithm 1 in node 1.

TABLE II
TIME COMPLEXITIES FOR THE VARIOUS NYSTRÖM-BASED METHODS WHEN IMPLEMENTED ON N NODES, WHERE $T(D)$ IS THE COMMUNICATION COST FOR TRANSFERRING D AMOUNT OF DATA

method	time complexity	communication
standard Nyström	$O(\frac{m^3}{N} + \frac{nmk}{N})$	$T(mk + \frac{nk}{N})$
ensemble of standard Nyström	$O(\frac{m^3}{n_e^2 N} + \frac{nmk}{N})$	$T(mk + \frac{nk}{N})$
proposed Nyström	$O(\frac{m^2 k}{N} + k^3 + \frac{nmk}{N})$	$T(mk + \frac{nk}{N})$
ensemble of proposed Nyström	$O(\frac{m^2 k}{n_e N} + k^3 + \frac{nmk}{N})$	$T(mk + \frac{nk}{N})$

barrier is imposed on all the nodes [39] and a single node collects the partial results from each node. On the other hand, distributed SVD algorithms [40] can also be used. However, the iterative nature of these solvers may lead to significant synchronization overhead.

C. Time Complexity

Table II shows the time complexities for the various Nyström-based methods when implemented on N nodes, where $T(D)$ is the communication cost for transferring D amount of data. For the ensemble Nyström algorithm, its total number of columns sampled is the same as that in the nonensemble version, i.e., $n_e m_e = m$. This ensures the time for computing the U matrix (which dominates the running time, as will be observed in Fig. 5 and Fig. 7) is the same [i.e., $O(nmk/N)$] across all the methods.

As can be seen, the time complexity is again reduced with the use of the proposed method. Moreover, note that all methods have the same communication cost. The dominating term might be $T(mk)$, as all the nodes have to wait. For the second term, nodes can do data transmission in the background without affecting the computation.

TABLE III
DATA SETS USED

data	number of samples	dimensionality
satimage	4,435	36
RCV1	23,149	47,236
MNIST	60,000	784
covtype	581,012	54
MNIST-8M	8,100,000	784

V. EXPERIMENTS

A. Experiments on Single Machine

In this section, we study the efficiency of the proposed method in solving the large dense eigen-systems in low-rank approximation. All the implementations are in MATLAB. Experiments are run on a PC with an Intel Core i7-3770 3.4-GHz CPU and 32-G memory.

We use a number of data sets from the LIBSVM archive[5] (Table III). The linear kernel is used for the RCV1 text data set, and the Gaussian kernel for the others. The following methods are compared.

1) Standard Nyström method [8] (denoted nys).
2) The proposed method in Algorithm 2 (denoted ours): we fix the over-sampling p to 5, and the power parameter q to 2.
3) Nyström method using the Arnoldi method [7] [denoted nys(krylov)].
4) Randomized SVD (denoted r-svd) [9]: similar to the proposed method, we also use $p = 5$ and $q = 2$.

For the Nyström-based methods (i.e., the first three methods above), the columns are sampled uniformly without replacement. This has been shown to perform well for kernel matrices of the linear and Gaussian kernels [33]. Due to randomness in the sampling process, we perform 10 repetitions and report the averaged result. As for the randomized SVD algorithm, there is no sampling and the whole input matrix is always used. The best rank-k approximation could have been obtained by a direct SVD on the whole input matrix. However, this is computationally expensive, even on medium-sized data sets, and so is not compared here.

1) Different Numbers of Columns: In the first experiment, we fix $k = 600$ and gradually increase the number of sampled columns (m). Fig. 2 shows the relative approximation error $\|K - \hat{K}\|_F / \|K\|_F$ with time. As can be seen, the randomized SVD algorithm is often the most accurate, albeit also the most expensive. For the Nyström variants, their approximation errors decrease with the number of sampled columns m. The Nyström method can be as accurate as randomized SVD when m is large enough. However, since Nyström takes $O(m^3)$ time for the SVD step, it also quickly becomes computationally infeasible. The proposed method is almost as accurate as standard Nyström, but is faster.

2) Different Ranks: In the second experiment, we study the approximation performance on the MNIST data set when the desired rank (k) varies. As can be observed from Fig. 3, when k increases, the approximation error decreases, while the

[5] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

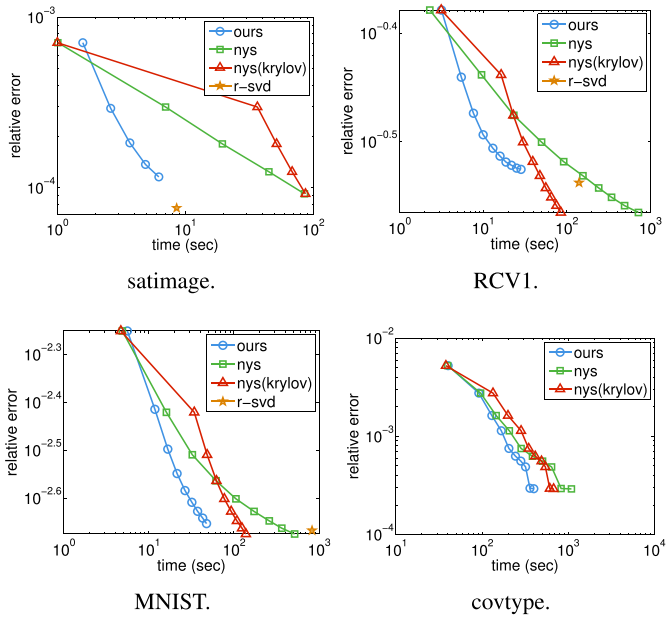


Fig. 2. Relative approximation errors of the various methods versus time. Each point on the curve corresponds to a fixed number of sampled columns. The randomized SVD algorithm cannot be run on the covtype data set because it is too large.

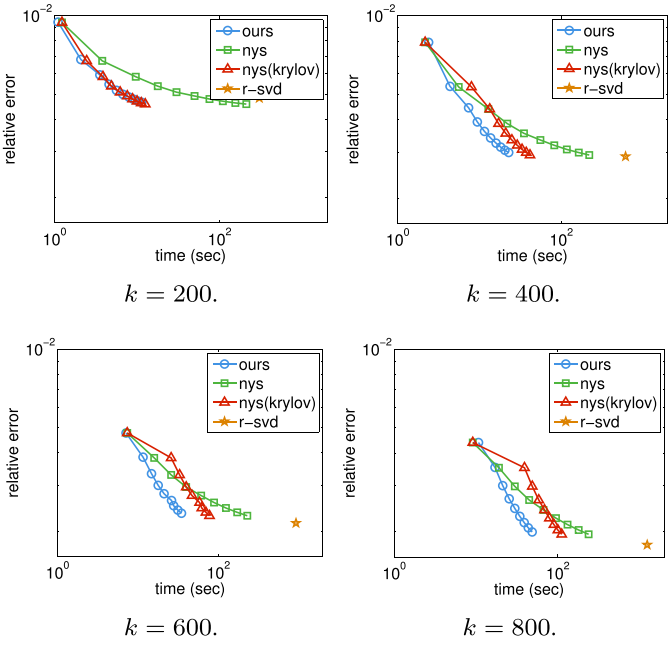


Fig. 3. Relative approximation errors with time on the MNIST data set. Each point on the curve corresponds to a fixed number of sampled columns.

time increases across all methods. Hence, there is a tradeoff between accuracy and efficiency. Nevertheless, the relative performance comparison among the various methods is still the same as in Section V-A1.

B. Speedup With GPUs

In this section, we demonstrate the speedup that can be achieved on GPUs. Experiments are performed on a machine with two Intel Xeon X5560 2.8-GHz CPUs, 32-G RAM, and four NVIDIA Tesla C1060 GPU cards. We implement our GPU-based algorithm with MATLAB2013a's parallel computing toolbox, which enables standard MATLAB code to be run on NVIDIA CUDA-supported GPUs.

TABLE IV
TIME (IN SECONDS) USED BY THE PROPOSED METHOD ON THE MNIST-8 M DATA SET WITH VARYING NUMBER OF SAMPLES n

n	CPU	1 GPU	2 GPUs	3 GPUs	4 GPUs
8×10^4	27.6	12.0	7.2	6.1	5.3
4×10^5	115.0	35.5	21.3	16.4	12.7
8×10^5	226.5	53.9	39.1	27.9	22.4
4×10^6	1,066.7	283.5	178.1	125.5	94.2
8.1×10^6	2,647.8	473.5	281.9	224.6	81.7

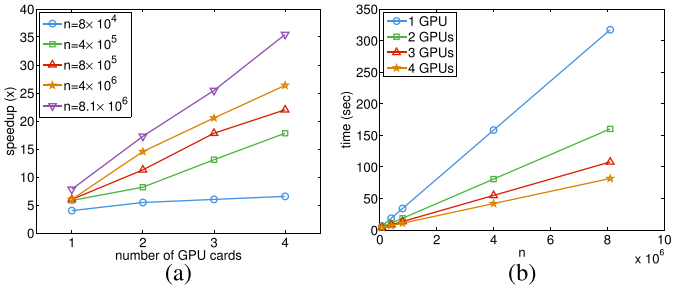


Fig. 4. Speed of the proposed method on the MNIST-8 M data set. (a) GPU speedup versus number of GPUs. (b) Time versus number of samples.

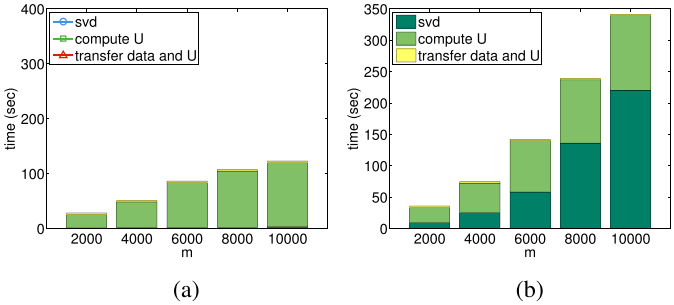


Fig. 5. Breakdown of the time consumption by the proposed method and the Nyström method on the GPU, with varying number of sampled columns. (a) Proposed method. (b) Standard Nyström.

We use the MNIST-8 M data set, and set $k = 600$, $m = 6000$. Table IV compares the time used by the proposed method when it is run on the CPU and GPU, respectively, with the number of samples varied from 80 K to 8.1 M. As can be seen, running on the GPU is much faster that running on the CPU. Fig. 4(a) shows the corresponding speedup factors, and Fig. 4(b) shows the variation of time versus n. As can be seen, the speedup scales linearly with the number of GPUs used, and the time increases linearly with n.

Fig. 5(a) shows a breakdown of the GPU time when m is varied from 2000 to 10 000 (with $k = 600$). Recall that we only need to: 1) transfer the m sampled points from the CPU to the GPU and 2) transfer the obtained U from GPU back to CPU. Given the high data transfer bandwidth[6] between

[6]According to http://wiki.accelereyes.com/wiki/index.php/GPU_Memory_Transfer, the CPU-to-GPU data transfer rate is 2.5 GB/s, while that for GPU-to-CPU is 3.9 GB/s. They are close to the rates of our machine.

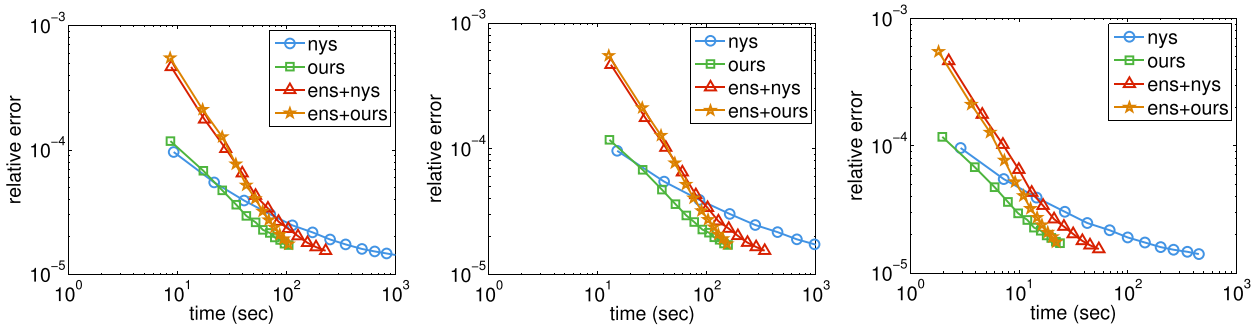


Fig. 6. Relative approximation errors on the MNIST-8.1 M in various multi-GPU and multiprocessor environments. Each point on the curve corresponds to a fixed number of sampled columns. (a) 2CPU–4GPU. (b) 64CPU. (c) 64CPU–16GPU.

TABLE V
CPU AND GPU COMPARISON ON THE MNIST-8 M DATA SET, WITH VARYING NUMBER OF SAMPLED COLUMNS m

m	time (sec) run on CPU	time (sec) run on 4 GPUs	(speedup)	relative approx. error
2,000	908.1	24.5	(37.1x)	5.9×10^{-4}
4,000	1,789.4	47.2	(37.9x)	5.5×10^{-4}
6,000	2,647.8	81.7	(32.4x)	5.3×10^{-5}
8,000	3,556.5	104.8	(33.9x)	5.4×10^{-5}
10,000	4,426.4	119.5	(37.0x)	3.0×10^{-5}
20,000	8,988.2	253.8	(35.4x)	1.1×10^{-5}

CPU and GPU, the time spent on data transfer here takes just about 1.5 s and is thus negligible. As can be seen, most of the time is spent on computing U, which grows linearly with m. Consequently, the total time also scales linearly with m. The standard Nyström algorithm can also benefit by running on the GPU, and results are shown in Fig. 5(b). In contrast to the proposed method [Fig. 5(a)], the time is now dominated by the SVD decomposition step. This again demonstrates the advantage in using the randomized SVD algorithm to approximate the inverse of W.

As the C1060 GPU has 4-GB memory, it can handle a maximum m of around 20 000 without causing out-of-memory problems (a 20 000 $\times$ 20 000 kernel submatrix occupies 3.2-G memory). Table V compares the speedups on the MNIST-8 M full set when m is varied from 2000 to 20 000. As can be seen, running on GPUs is again much faster at all values of m, and the speedups are from 32$\times$ to 38$\times$. As discussed above, the time for the proposed method scales linearly with m. Recall from Table I that the time for the standard Nyström also scales linearly with m. Hence, as shown in Table V, the speedup over standard Nyström is close to constant with respect to m. Moreover, sampling more columns leads to better approximation error, which is consistent with the observation in Fig. 2.

C. Multi-CPU and Multi-GPU Environments

In this section, experiments are performed on the following environments.

1) 2CPU–4GPU configuration: A single machine with two Intel Xeon X5560 2.8-GHz CPUs, 32-G RAM, and four NVIDIA Tesla C1060 GPU cards.
2) 64 CPU configuration: A cluster of 16 machines. Each one has four AMD Opteron 6272 2.1-GHz CPUs and 128-G RAM. The machines are connected via a 1 G network.
3) 64CPU–16GPU configuration: Same machine as above, but with each one having a NVIDIA Tesla K20C Kepler GPU.

The following methods are compared.

1) nys: Standard Nyström method. The distributed Lanczos algorithm [40] is used to solve the SVD. Its implementation is essentially the same as Algorithm 4, but involves more synchronization.
2) ours: The proposed distributed Nyström method (Algorithm 3), with the distributed randomized SVD procedure in Algorithm 4.
3) ens + nys: Ensemble Nyström [22] with nys as the base learner. As in [22], we use an additional $s = 20$ columns for training the mixture weights by ridge regression, and another $s' = 20$ columns for choosing the regularization parameters. The number of base learners n_e is fixed at 4,[7] and the set of CPUs/GPUs are divided evenly among the base learners.
4) ens + ours: Ensemble Nyström as above, but with the proposed Nyström algorithm as base learner.

All methods are in C++. Matrix computations are implemented with the Eigen 3 package[8] (which supports multithreading) on CPUs, and with CUDA[9] on GPUs. The asynchronous distributed framework parameter server [41], [42] is used to scale these algorithms for distributed implementation.

Experiments are performed on the MNIST-8 M data set. We set $k = 500$, and vary the number of sampled columns (m) in the range $\{2 \times 10^3, 4 \times 10^3, 6 \times 10^3, \ldots, 24 \times 10^3\}$.

Results on the relative approximation error are shown in Fig. 6. As can be seen, the approximation errors can all be reduced to a very low level with a sufficient number of sampled columns. In particular, ours outperforms nys, and ens + ours outperforms ens + nys. This agrees with the obser-

[7]Preliminary experiments using $n_e = 16$ on the 64CPU–16GPU configuration lead to much higher approximation errors.
[8]http://eigen.tuxfamily.org/
[9]https://developer.nvidia.com/about-cuda

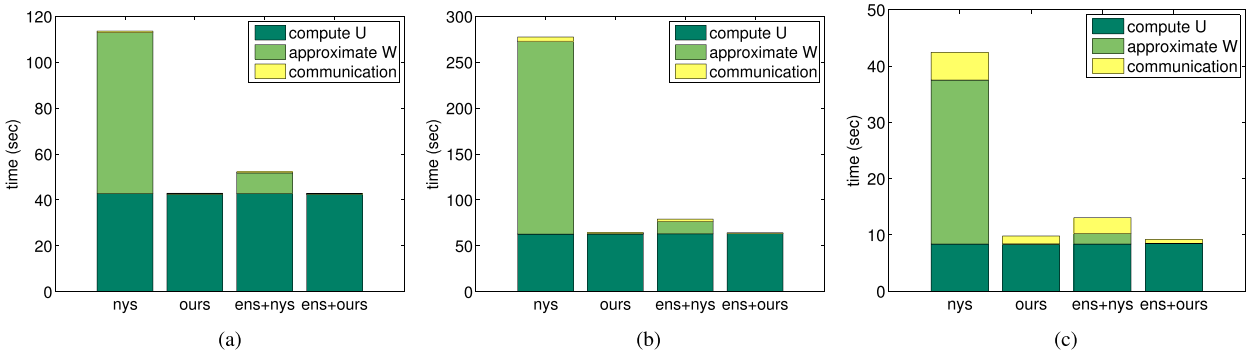


Fig. 7. Breakdown for the timing results in Fig. 6 (with $m = 10\,000$). (a) 2CPU–4GPU. (b) 64CPU. (c) 64CPU–16GPU.

vations on the single-machine implementations in Section V-A. Moreover, ours performs better than ens + ours (note that both are distributed algorithms), with the performance gap gradually reduces when m is large. As discussed in Section III, this is because of the block-diagonality assumption used by ensemble Nyström. In addition, as expected, the use of GPUs leads to a much faster decrease of the approximation error [Fig. 6(b) versus (c)]. In addition, the 64CPU configuration runs (slightly) slower than the 2CPU–4GPU configuration [Fig. 6(a) versus (b)]. One of the reasons is that data in the multiprocessor system have to be communicated over a network, which is far slower than CPU–GPU communication.

A more detailed breakdown of the timing results (for $m = 10 \times 10^3$) is shown in Fig. 7. As can be seen, the proposed distributed randomized SVD algorithm (used in ours and ens + ours) is much more efficient than the distributed Lanczos algorithm (used in nys and ens + nys) in approximating the W matrix. Moreover, though the use of more GPUs allows all algorithms to run faster, the problem of straggler, namely, that the system needs to wait for the slowest node to complete its update before the next iteration can proceed, becomes more severe. In other words, the system can only move forward only at the pace of the slowest node, and the communication overhead increases [Fig. 7(c)].

VI. Conclusion

In this paper, we proposed an accurate and scalable Nyström approximation scheme for very large data sets. It first samples a large column subset from the input matrix, and then performs an approximate SVD on the inner submatrix using the recent randomized low-rank matrix approximation algorithms. Both theory and experiments demonstrate that the proposed algorithm is as accurate as the standard Nyström method that directly performs a large SVD on the inner submatrix, but with a much lower time complexity. Moreover, this can be efficiently distributed for use in environments with multiple CPUs and multiple GPUs.

Appendix

The error analysis will depend on a number of results in [9], [27] and [43]. For the readers' convenience, we also list these in this section. First, the following result is used in the proof of [9, Proposition 8.6].

Proposition 1 [9]: Suppose that P is an orthogonal projector, D is a nonnegative diagonal matrix, and integer $q \geq 1$. Then, $\|PDP\|_2^q \leq \|PD^qP\|_2$.

Theorem 3 [9, Th. 10.6]: Suppose that $A \in \mathbb{R}^{m\times n}$ with singular values $\sigma_1 \geq \sigma_2 \geq \cdots$. Choose a target rank k and an oversampling parameter $p \geq 2$, where $k + p \leq \min\{m, n\}$. Draw an $n \times (k + p)$ standard Gaussian matrix Ω, and construct the sample matrix $Y = A\Omega$. Then, $\mathbb{E}\,\|(I - P_Y)A\|_2 \leq (1 + (k/p - 1)^{1/2})\sigma_{k+1} + e(k + p)^{1/2}/p(\sum_{j>k}\sigma_j^2)^{1/2}$, and $\mathbb{E}\,\|(I - P_Y)A\|_F \leq (1 + k/p - 1)^{1/2}(\sum_{i>k}\sigma_i^2)^{1/2}$.

Proposition 2 [27, Corollary 2]: Suppose $A \in \mathbb{R}^{m\times n}$. Choose a set S of size m uniformly at random without replacement from $\{1, \ldots, n\}$, and let C be the columns of A corresponding to indices in S scaled by $(n/m)^{1/2}$. Then, $\mathbb{E}_C\,\|AA^T - CC^T\|_F \leq n/m^{1/2}(\max_i \|A^{(i)}\|)^2$, where $A^{(i)}$ is the ith column of A.

Corollary 5: Let $K_{ii}^* = \max_i K_{ii}$. Then, $\mathbb{E}\,\|XX^T - HH^T\|_2 \leq n/\sqrt{m}K_{ii}^*$.

The following is a well-known result from matrix perturbation theory.

Proposition 3 [43]: Given matrices $A \in \mathbb{R}^{n\times n}$ and $E \in \mathbb{R}^{n\times n}$, then $\max_i |\sigma_i(A + E) - \sigma_i(A)| \leq \|E\|_2$, and $\sum_{i=1}^n(\sigma_i(A + E) - \sigma_i(A))^2 \leq \|E\|_F^2$.

Proposition 4 [11, Th. 1]: Suppose $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{Z}^+$ such that $1 \leq c \leq n$ and $\{p_i\}_{i=1}^n$ are such that $p_k = \|A^{(k)}\|^2/\|A\|_F^2$. Construct C with the Basic Matrix Multiplication algorithm of [26], and let CC^T be an approximation to AA^T. Then, $\mathbb{E}\,\|AA^T - CC^T\|_F \leq 1/\sqrt{c}\|A\|_F^2$.

Corollary 6: $\mathbb{E}\,\|XX^T - HH^T\|_2 \leq (1/\sqrt{m})\,\mathrm{Tr}(K)$.

A. Proof of Theorem 1

First, we introduce the following lemmas.

Lemma 2: Let P be an orthogonal projector and S be a psd matrix. For integer $q \geq 1$, $\|PS\|_2 \leq \|PS^q\|_2^{1/q}$.

Lemma 3: Let U be an orthonormal basis of the range of matrix $R \in \mathbb{R}^{n\times k}$. Then, for any $X \in \mathbb{R}^{n\times n}$, $\|X^TX - X^TUU^TX\|_2 \leq \|XX^T - RR^T\|_2$.

Proposition 5: In Algorithm 1, the expectation of $\|W - QQ^TW\|_2$ (with respect to the randomness in the

Gaussian random matrix) is

$$\mathbb{E}_Q \|W - QQ^T W\|_2 \leq \zeta^{1/q} \sigma_{k+1}(W) \tag{14}$$

where $\zeta = 1 + (k/p - 1)^{1/2} + e(k+p)^{1/2}/p(m-k)^{1/2}$.

Proof: Recall that Q in Algorithm 2 is an orthonormal basis of Y, thus, $P_Y = QQ^T$. First, consider $q = 1$. Using Theorem 3 and that $\sum_{i=k+1}^{m} \sigma_i^2(W) \leq (m-k)\sigma_{k+1}^2(W)$, we obtain (14). Now, for integer $q > 1$, $\mathbb{E}\|(I - P_Y)W\|_2 \leq \left(\mathbb{E}\|(I - P_Y)W\|_2^q\right)^{1/q}$, on using Hölder's inequality. Note that $I - P_Y$ is also an orthogonal projector. Hence, on using Lemma 2, we have $\mathbb{E}\|(I - P_Y)W\|_2 \leq \left(\mathbb{E}\|(I - P_Y)B\|_2\right)^{1/q}$, where $B = W^q$. Using (14) with $q = 1$ (which has just been proved), we obtain $\mathbb{E}\|(I - P_Y)W\|_2 \leq (\zeta\sigma_{k+1}(B))^{1/q} = \zeta^{1/q}\sigma_{k+1}(W)$. ∎

Proof (of Theorem 1): From Algorithm 1 and (8), $\tilde{U}\Lambda\tilde{U}^T = QBQ^T = Q(Q^T W Q)Q^T$. Hence

$$\hat{K} = C\tilde{U}\Lambda^{+}\tilde{U}^T C^T = CQ(Q^T W Q)^{+} Q^T C. \tag{15}$$

Let $R = HQ$, where Q is as defined in Algorithm 1. Let U_R be an orthonormal basis of ran(R). From (15)

$$\begin{aligned}\hat{K} &= CQ(Q^T W Q)^{+} Q^T C^T \\ &= X^T HQ(Q^T H^T HQ)^{+} Q^T H^T X \\ &= X^T P_{HQ} X = X^T U_R U_R^T X. \end{aligned} \tag{16}$$

Using Lemma 3, we have $\|K - \hat{K}\|_2 = \|X^T X - X^T U_R U_R^T X\|_2 \leq \|XX^T - RR^T\|_2 \leq \|XX^T - HH^T\|_2 + \|HH^T - RR^T\|_2$. Then, $\|HH^T - RR^T\|_2 = \|H(I - QQ^T)H^T\|_2 \leq \|(I - QQ^T)H^T H\|_2 = \|W - QQ^T W\|_2$. Assume H is fixed, then the randomness in $\|HH^T - RR^T\|_2$ is due to the Gaussian random matrix (i.e., random variable Q). By Proposition 5, $\mathbb{E}\|HH^T - RR^T\|_2 \leq \mathbb{E}\|W - QQ^T W\|_2 \leq \zeta^{1/q}\sigma_{k+1}(W) = \zeta^{1/q}\sigma_{k+1}(HH^T) \leq \zeta^{1/q}\sigma_{k+1}(XX^T) + \zeta^{1/q}\|XX^T - HH^T\|_2$, where the last step is due to Proposition 3. Now, we take expectation over both Q and H (i.e., also including the randomness in selecting the columns) and putting all these together, we have $\mathbb{E}\|K - \hat{K}\|_2 \leq \mathbb{E}_H\|XX^T - HH^T\|_2 + \mathbb{E}_H(\mathbb{E}_{Q|H}\|HH^T - RR^T\|_2) \leq \zeta^{1/q}\sigma_{k+1}(XX^T) + (1+\zeta^{1/q})\,\mathbb{E}_H\|XX^T - HH^T\|_2 \leq \zeta^{1/q}\|K - K_k\|_2 + (1+\zeta^{1/q})\,\mathbb{E}\|XX^T - HH^T\|_2$, where the last step uses $\|K - K_k\|_2 = \sigma_{k+1}(K) = \sigma_{k+1}(XX^T)$.

B. Proof of Theorem 2

As shown in Section III-B, we first consider the error in approximating W from Algorithm 1, which is a direct application of Theorem 3.

Corollary 7: In Algorithm 1, the expectation of $\|W - QQ^T W\|_F$ with respect to the randomness in the Gaussian random matrix is $\mathbb{E}_Q\|W - QQ^T W\|_F \leq (1 + k/p - 1)^{1/2}(\sum_{i>k}\sigma_i^2(W))^{1/2}$.

Next, we introduce the following lemmas.

Lemma 4: Given matrices $A \in \mathbb{R}^{n\times t}$ and $B \in \mathbb{R}^{n\times s}$, with $n \geq \max\{s,t\}$. Then, for any $k \leq \min\{s,t\}$, $\sum_{i=1}^{k}(\sigma_i^2(A) - \sigma_i^2(B)) \leq \sqrt{k}\|AA^T - BB^T\|_F$.

Lemma 5: For matrices $A \in \mathbb{R}^{n\times k}$ and $B \in \mathbb{R}^{n\times n}$, with $n \geq k$. Let U be an orthonormal basis of ran(A). Then, $\sum_{i=1}^{k}\sigma_i^2(A) - \|U^T B\|_F^2 \leq \sqrt{k}\|AA^T - BB^T\|_F$.

Lemma 6: Given $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n}$, and $k \leq n$, we have $|(\sum_{i>k}\sigma_i^2(A))^{1/2} - (\sum_{i>k}\sigma_i^2(B))^{1/2}| \leq \|A - B\|_F$.

Proof of Theorem 2: Let $R = HQ$ and U_R be an orthonormal basis of ran(R). From (16), $\|K - \hat{K}\|_F = \|X^T X - X^T U_R U_R^T X\|_F$. Since $I - U_R U_R^T$ is an orthogonal projector, it is psd. Thus, for any vector u, $u^T X^T(I - U_R U_R^T)Xu = (Xu)^T(I - U_R U_R^T)(Xu) \geq 0$, and so $X^T X - X^T U_R U_R^T X$ is also psd. Moreover

$$\|K - \hat{K}\|_F \leq \mathrm{Tr}\left(X^T X - X^T U_R U_R^T X\right) = \|X\|_F^2 - \left\|U_R^T X\right\|_F^2. \tag{17}$$

Using Lemmas 4 and 5

$$\begin{aligned}&\|X\|_F^2 - \left\|U_R^T X\right\|_F^2 \\ &= \left(\|X\|_F^2 - \sum_{i=1}^{k+p}\sigma_i^2(X)\right) + \left(\sum_{i=1}^{k+p}\sigma_i^2(X) - \sum_{i=1}^{k+p}\sigma_i^2(H)\right) \\ &\quad + \left(\sum_{i=1}^{k+p}\sigma_i^2(H) - \sum_{i=1}^{k+p}\sigma_i^2(R)\right) \\ &\quad + \left(\sum_{i=1}^{k+p}\sigma_i^2(R) - \left\|U_R^T X\right\|_F^2\right) \\ &\leq \sum_{i>k+p}\sigma_i^2(X) + \sqrt{k+p}\|XX^T - HH^T\|_F \\ &\quad + \sqrt{k+p}\|HH^T - RR^T\|_F \\ &\quad + \sqrt{k+p}\|XX^T - RR^T\|_F \\ &\leq \sum_{i>k+p}\sigma_i^2(X) + 2\sqrt{k+p}\|XX^T - HH^T\|_F \\ &\quad + 2\sqrt{k+p}\|HH^T - RR^T\|_F. \end{aligned} \tag{18}$$

Let $P = I - QQ^T$, which is an orthogonal projector. Then

$$\begin{aligned}\|HH^T - RR^T\|_F^2 &= \|HH^T - HQQ^T H^T\|_F^2 \\ &= \mathrm{Tr}(HPH^T HPH^T) = \|PH^T HP\|_F \leq \|PH^T H\|_F \\ &= \|H^T H - QQ^T H^T H\|_F^2 = \|W - QQ^T W\|_F^2. \end{aligned}$$

Using Corollary 7, and let $\alpha = (1 + k/p - 1)^{1/2}$, then on assuming that H is fixed

$$\begin{aligned}&\mathbb{E}\|HH^T - RR^T\|_F \\ &\quad = \mathbb{E}\|W - QQ^T W\|_F \\ &\quad \leq \alpha\left[\sum_{i>k}\sigma_i^2(W)\right]^{1/2} = \alpha\left[\sum_{i>k}\sigma_i^2(HH^T)\right]^{1/2} \\ &\quad \leq \alpha\left[\sum_{i>k}\sigma_i^2(XX^T)\right]^{1/2} + \alpha\|XX^T - HH^T\|_F \\ &\quad = \alpha\|K - K_k\|_F + \alpha\|XX^T - HH^T\|_F \end{aligned} \tag{19}$$

on using Lemma 6. Now, taking the randomness of H into account, and on combining (17)–(19), we obtain

$$\begin{aligned}&\mathbb{E}\|K - \hat{K}\|_F \\ &\quad \leq \sum_{i>k+p}\sigma_i(K) + 2\sqrt{k+p}\,\mathbb{E}_H\|XX^T - HH^T\|_F\end{aligned}$$

$$
\begin{aligned}
&+2\sqrt{k+p}\,\mathbb{E}_H\left(\mathbb{E}_{Q|H}\|HH^T-RR^T\|\right)\\
&\le \sum_{i>k+p}\sigma_i(K)+2\alpha\sqrt{k+p}\|K-K_k\|_F\\
&\quad+2(1+\alpha)\sqrt{k+p}\,\mathbb{E}\|XX^T-HH^T\|_F\\
&\le \sum_{i>k+p}\sigma_i(K)+\frac{2(k+p)}{\sqrt{p-1}}\|K-K_k\|_F\\
&\quad+\frac{4(k+p)}{\sqrt{p-1}}\,\mathbb{E}\|XX^T-HH^T\|_F
\end{aligned}
$$

where the last inequality is due to $\alpha \ge 1$ and $\alpha(k+p)^{1/2} = ((k+p)(k+p-1)/p-1)^{1/2} \le k+p/(p-1)^{1/2}$.

REFERENCES

[1] B. Schölkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA, USA: MIT Press, 2002.

[2] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Comput.*, vol. 10, no. 5, pp. 1299–1319, 1998.

[3] S. Mika, G. Rätsch, J. Weston, B. Schoölkopf, and K. Müller, "Fisher discriminant analysis with kernels," in *Proc. IEEE 9th Signal Process. Soc. Workshop Neural Netw. Signal Process.*, Aug. 1999, pp. 41–48.

[4] U. von Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA, USA: MIT Press, 2002.

[6] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[7] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Philadelphia, PA, USA: SIAM, 1998.

[8] C. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Advances in Neural Information Processing Systems 13*, T. Leen, T. Dietterich, and V. Tresp, Eds. Cambridge, MA, USA: MIT Press, 2001.

[9] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM Rev.*, vol. 53, no. 2, pp. 217–288, 2011.

[10] C. T. H. Baker, *The Numerical Treatment of Integral Equations*. Oxford, U.K.: Clarendon, 1977.

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

[12] 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, Feb. 2004.

[13] A. Talwalkar, S. Kumar, and H. Rowley, "Large-scale manifold learning," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, Anchorage, AK, USA, Jun. 2008, pp. 1–8.

[14] C. Faloutsos and K.-I. Lin, "FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, San Jose, CA, USA, May 1995, pp. 163–174.

[15] J. T.-L. Wang, X. Wang, K.-I. Lin, D. Shasha, B. A. Shapiro, and K. Zhang, "Evaluating a class of distance-mapping algorithms for data mining and clustering," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Diego, CA, USA, Aug. 1999, pp. 307–311.

[16] V. de Silva and J. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA, USA: MIT Press, 2003, pp. 721–728.

[17] J. C. Platt, "FastMap, MetricMap, and landmark MDS are all Nyström algorithms," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, Jan. 2005, pp. 1–15.

[18] K. Sinha and M. Belkin, "Semi-supervised learning using sparse eigenfunction bases," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds. Red Hook, NY, USA: Curran Associates, 2010.

[19] K. Kobayashi and F. Komaki, "Information criteria for support vector machines," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 571–577, May 2006.

[20] S. Lee and S. J. Wright, "ASSET: Approximate stochastic subgradient estimation training for support vector machines," in *Proc. Int. Conf. Pattern Recognit. Appl. Methods*, Algarve, Portugal, Feb. 2012, pp. 223–228.

[21] K. Zhang and J. T. Kwok, "Clustered Nyström method for large scale manifold learning and dimension reduction," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1576–1587, Oct. 2010.

[22] S. Kumar, M. Mohri, and A. Talwalkar, "Ensemble Nyström method," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds. Red Hook, NY, USA: Curran Associates, 2010.

[23] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix," *SIAM J. Comput.*, vol. 36, no. 1, pp. 158–183, 2006.

[24] M. Li, J. T. Kwok, and B.-L. Lu, "Making large-scale Nyström approximation possible," in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, Jun. 2010, pp. 631–638.

[25] K. Zhang, L. Lan, J. Liu, A. Rauber, and F. Moerchen, "Inductive kernel low-rank decomposition with priors: A generalized Nyström method," in *Proc. 29th Int. Conf. Mach. Learn.*, Edinburgh, U.K., Jun. 2012, pp. 305–312.

[26] P. Drineas, R. Kannan, and M. Mahoney, "Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication," *SIAM J. Comput.*, vol. 36, no. 1, pp. 132–157, 2007.

[27] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the Nyström method," *J. Mach. Learn. Res.*, vol. 13, pp. 981–1006, Apr. 2012.

[28] M. Ouimet and Y. Bengio, "Greedy spectral embedding," in *Proc. 10th Int. Workshop Artif. Intell. Statist.*, Jan. 2005, pp. 253–260.

[29] A. K. Farahat, A. Ghodsi, and M. S. Kamel, "A novel greedy algorithm for Nyström approximation," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, Apr. 2011, pp. 1–9.

[30] K. Zhang, I. Tsang, and J. Kwok, "Improved Nyström low-rank approximation and error analysis," in *Proc. 25th Int. Conf. Mach. Learn.*, Helsinki, Finland, Jul. 2008, pp. 1232–1239.

[31] P. Drineas, M. W. Mahoney, and S. Muthukrishnan, "Relative-error CUR matrix decompositions," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 844–881, 2008.

[32] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 3475–3506, 2012.

[33] A. Gittens and M. W. Mahoney, "Revisiting the Nyström method for improved large-scale machine learning," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 567–575.

[34] M. Mohri and A. Talwalkar, "Can matrix coherence be efficiently and accurately estimated?" in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, 2011, pp. 534–542.

[35] A. Talwalkar and A. Rostamizadeh, "Matrix coherence and the Nyström method," in *Proc. 26th Int. Conf. Uncertainty Artif. Intell.*, Santa Catalina Island, CA, USA, 2010, pp. 572–579.

[36] S. Wang and Z. Zhang. (2014). "Efficient algorithms and error analysis for the modified Nyström method." [Online]. Available: http://arxiv.org/abs/1404.0138

[37] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou, "Nyström method vs random Fourier features: A theoretical and empirical comparison," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 476–484.

[38] V. Volkov and J. W. Demmel, "Benchmarking GPUs to tune dense linear algebra," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2008, pp. 1–11.

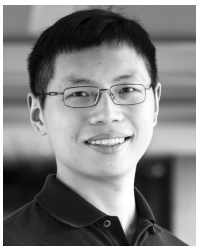
[39] J. Albrecht, C. Tuttle, A. C. Snoeren, and A. Vahdat, "Loose synchronization for large-scale networked systems," in *Proc. USENIX Annu. Tech. Conf.*, 2006, pp. 301–314.

[40] E. J. Kontoghiorghes, *Handbook of Parallel Computing and Statistics*. Boca Raton, FL, USA: CRC Press, 2010.

[41] M. Li, L. Zhou, A. Li, F. Xia, D. G. Andersen, and A. Smola, "Parameter server for distributed machine learning," in *Proc. NIPS Workshop Big Learn.*, 2013.

[42] M. Li *et al.*, "Scaling distributed machine learning with the parameter server," in *Proc. OSDI*, 2014, pp. 583–598.

[43] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*. Boston, MA, USA: Academic, 1990.



Mu Li received the bachelor's and master's degrees from Shanghai Jiao Tong University, Shanghai, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

His research focuses on the codesign of distributed computing systems and large-scale machine learning algorithms. He led the project for building the world's largest machine learning system for Baidu's computational advertising, which is one of the core businesses of Baidu. He has a wide range of publications in the top conferences in computer science, with topics spanning theory (FOCS), machine learning (NIPS and ICML), applications (CVPR and KDD), and operation systems (OSDI).

Wei Bi received the bachelor's degree in computer science from Sun Yat-sen University, Guangzhou, China, in 2010. She is currently pursuing the Ph.D. degree in computer science with the Hong Kong University of Science and Technology, Hong Kong.

Her current research interests include machine learning, data mining, application problems on computer vision, and other problems in artificial intelligence.

Dr. Bi was a recipient of the Google Ph.D. Fellowship in Machine Learning in 2013 and the Google Anita Borg Scholarship in 2014.

James T. Kwok received the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Hong Kong, in 1996.

He is currently a Professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His current research interests include machine learning, kernel methods, pattern recognition, and artificial neural networks.

Dr. Kwok was a recipient of the IEEE Outstanding Paper Award in 2004, and the Second Class Award in Natural Sciences from the Ministry of Education, China, in 2008. He has been the Program Co-Chair for a number of international conferences, and served as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2006 to 2012. He is currently an Associate Editor of the *Neurocomputing* journal, and a Board Member of the Asia Pacific Neural Network Assembly.

Bao-Liang Lu (M'94–SM'01) received the B.S. degree in instrument and control engineering from the Qingdao University of Science and Technology, Qingdao, China, in 1982, the M.S. degree in computer science and technology from Northwestern Polytechnical University, Xi'an, China, in 1989, and the Dr. Eng. degree in electrical engineering from Kyoto University, Kyoto, Japan, in 1994.

He was with the Qingdao University of Science and Technology from 1982 to 1986. From 1994 to 1999, he was a Frontier Researcher with the Bio-Mimetic Control Research Center, Institute of Physical and Chemical Research (RIKEN), Nagoya, Japan, and a Research Scientist with the RIKEN Brain Science Institute, Wako, Japan, from 1999 to 2002. Since 2002, he has been a Full Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, where he has been an Adjunct Professor with the Laboratory for Computational Biology, Shanghai Center for Systems Biomedicine, since 2005. His current research interests include brain-like computing, neural network, machine learning, computer vision, bioinformatics, brain–computer interface, and affective computing.

Prof. Lu was the President of the Asia Pacific Neural Network Assembly (APNNA) and the General Chair of the 18th International Conference on Neural Information Processing in 2011. He is currently an Associate Editor of *Neural Networks* and a Board Member of APNNA.