Machine Learning (2022) 111:2037–2091
https://doi.org/10.1007/s10994-022-06140-9

Randomized approximate class-specific kernel spectral regression analysis for large-scale face verification

Ke Li[1] · **Gang Wu**[1]

Received: 14 April 2021 / Revised: 7 November 2021 / Accepted: 9 February 2022 /
Published online: 21 March 2022

Abstract
Kernel methods are known to be effective to analyse complex objects by implicitly embedding them into some feature space. The approximate class-specific kernel spectral regression (ACS-KSR) method is a powerful tool for face verification. This method consists of two steps: an eigenanalysis step and a kernel regression step, however, it may suffer from heavily computational overhead in practice, especially for large-sample data sets. In this paper, we propose two randomized algorithms based on the ACS-KSR method. The main contribution of our work is four-fold. First, we point out that the formula utilized in the eigenanalysis step of the ACS-KSR method is *mathematically incomplete*, and we give a correction to it. Moreover, we consider how to efficiently solve the ratio-trace problem and the trace-ratio problem involved in this method. Second, it is well known that kernel matrix is approximately low-rank, however, to the best of our knowledge, there are few theoretical results that can provide simple and feasible strategies to determine the numerical rank of a kernel matrix without forming it explicitly. To fill-in this gap, we focus on the commonly used Gaussian kernel and provide a practical strategy for determining numerical rank of the kernel matrix. Third, based on numerically low-rank property of the kernel matrix, we propose a modified Nyström method with fixed-rank for the kernel regression step, and establish a probabilistic error bound on the approximation. Fourth, although the proposed Nyström method can reduce the computational cost of the original method, it is required to form and store the reduced kernel matrix explicitly. This is unfavorable to extremely large-sample data sets. To settle this problem, we propose a randomized block Kaczmarz method for kernel regression problem with multiple right-hand sides, in which there is no need to compute and store the reduced kernel matrix explicitly. The convergence of this method is established. Comprehensive numerical experiments on real-world data sets are performed to show the effectiveness of our theoretical results and the efficiency of the proposed methods.

Keywords Face verification · Approximate class-specific kernel spectral regression (ACS-KSR) · Kernel matrix · Nyström method · Block Kaczmarz method

Editor: Paolo Frasconi.

✉ Gang Wu
gangwu@cumt.edu.cn; gangwu76@126.com

Extended author information available on the last page of the article

1 Introduction

The face verification problem has attracted great attention for more than two decades due to its application demands including security, human–computer interaction, human behaviour analysis for assisted living, and so on (Barr et al., 2007; Lei et al., 2012; Li et al., 2011; Tefas & Pitas, 2011). In essence, face verification is conceptually different from the famous face recognition problem (Cao et al., 2018; Iosifidis & Gabbouj, 2016a, 2016b, 2017; Iosifidis et al., 2015). On one hand, face recognition is a multi-class problem, which concentrates on recognizing the identity of a person from a pool of some known person identities. On the other hand, face verification is a binary problem where it focuses on verifying whether a facial image depicts a person of interest (Iosifidis & Gabbouj, 2016b; Wu et al., 2019).

Subspace learning method is done by projecting images into a lower dimensional space, and after that recognition is performed by measuring the distances between known images and the image to be recognized. Two representative subspace learning methods are principal component analysis (PCA) (Duda et al., 2000) and linear discriminant analysis (LDA) (Li et al., 2011; Lei et al., 2012; Duda et al., 2000). PCA is an unsupervised method in which the discriminative information encoded in the labels of training data is not exploited. Hence, its discrimination power is often limited (Zhou et al., 2013). On the other hand, the maximal dimensionality of the learnt subspace in LDA is restricted by the number of classes s, which limits the application of LDA in face verification problem. Indeed, since the rank of the between-class scatter matrix is at most $s - 1$, the subspace learned by the LDA method has only one dimension in binary (two-class) problems, which might not be the optimal choice for discrimination problems (Iosifidis & Gabbouj, 2016; Zhou et al., 2013).

To remedy these limitations, class-specific approaches are investigated in Zafeiriou et al. (2012), Kittler et al. (2000), Goudelis et al. (2007), Iosifidis et al. (2015), Arashloo and Kittler (2014). In class-specific subspace learning techniques, an optimal subspace that highlights the discrimination of one class (noted as *client class* hereafter) from all other possibilities (i.e. data not belonging to the client class, forming the so-called *impostor class*) is determined (Iosifidis and Gabbouj, 2016). Meanwhile, to achieve nonlinear data projections which have been found to outperform linear ones with a large extend in face recognition and verification problems, class-specific subspace learning methods can be extended to their nonlinear counterparts by exploiting the well-known kernel trick (Baudat & Anouar, 2000; Hofmann et al., 2008; Lu et al., 2003; Müller et al., 2001).

Kernel methods are well known to be effective in dealing with nonlinear machine learning problems in general, and are often required for machine learning tasks on complex data sets (Hofmann et al., 2008; Müller et al., 2001). The main idea behind kernel machines is to map the data from the input space to a higher dimension feature space via a nonlinear map, where the mapped data can be then analysed by linear models. Kernel learning techniques aim at constructing kernel matrices whose structure is well aligned with the learning target, which improves the generalization performance of kernel methods (Lan et al., 2017; Tran et al., 2020). However, most kernel learning approaches are computationally very expensive. It is well known that kernel versions of subspace learning techniques require to compute the kernel matrix $K \in \mathbb{R}^{n \times n}$ explicitly, where n is the training set cardinality. However, as the computational complexities and the storage requirements are $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ (Iosifidis et al., 2015; Tavernier et al., 2019), respectively, both approaches above will become computationally intractable as n is large.

Class-specific kernel discriminant analysis (CS-KDA) (Goudelis et al., 2007; Iosifidis et al., 2015) and class-specific kernel spectral regression (CS-KSR) (Arashloo & Kittler, 2014; Iosifidis et al., 2015) are commonly used class-specific kernel approaches. Recently, Iosifidis and Gabbouj (2016) put forward an approximate class-specific kernel spectral regression (ACS-KSR) method that employs the reduced kernel matrix $\widetilde{K} \in \mathbb{R}^{r\times n}(r < n)$ to take place of the kernel matrix, which speedups the computation. Roughly speaking, this method is composed of two steps: the eigenanalysis step for computing the eigenvector matrix T, and a kernel regression step for the reconstruction weights matrix A. Unfortunately, we find that the *widely used* eigenanalysis step in the ACS-KSR method (Iosifidis & Gabbouj, 2016) and other class-specific kernel discriminant analysis methods (Cao et al., 2018; Iosifidis & Gabbouj, , 2017; Iosifidis et al., 2015) is *incomplete*. Moreover, the explicit computation of the cross-product matrix $\widetilde{K}\widetilde{K}^T$ in the kernel regression step is computationally impracticable, in addition to some useful information may be lost (Golub & Van Loan, 2014). Therefore, it is necessary to revisit the ACS-KSR method and improve the numerical performance of this type of methods.

With the development of science and technology, the ability to generate data at the scale of millions and even billions has increased rapidly, posing great computational challenges to scientific computation problems involving large-scale kernel matrices (Hofmann et al., 2008). Low-rank approximations are popular techniques to reduce the highly computational cost of algorithms involving large-scale kernel matrices (Halko et al., 2011; Hofmann et al., 2008; Wang et al., 2018; Wathen & Zhu, 2015). Indeed, the success of these low-rank approximation algorithms hinges on a large spectrum gap or a fast decay of the spectrum of the kernel matrix (Halko et al., 2011; Pan et al., 2011). This motivates the analysis on the numerical rank of kernel matrix. In recent years, the low-rank property and low-rank approximation of kernel matrix have attracted great attention (Cambier & Darve, 2019; Iske et al., 2017; Wang et al., 2018; Wathen & Zhu, 2015; Xing & Chow, 2020). For example, the low-rank property of kernel matrix is investigated in Wang et al. (2018), Wathen and Zhu (2015), an interpolation method is used to construct the approximation of kernel matrix (Cambier & Darve, 2019; Xing & Chow, 2020), and a low-rank approximation is constructed in Iske et al. (2017), with the help of hierarchical low-rank property of kernel matrix. Although there has been a lot of research on the low-rank approximation of the kernel matrix, the estimation of the numerical rank of the kernel matrix is still in the theoretical stage. To the best of our knowledge, most existing results usually require the key information of kernel matrices, and there are few theoretical results could provide simple and feasible strategies for determining the numerical rank of a kernel matrix without forming the matrix explicitly. Moreover, estimations to the upper bound of the numerical rank are often too large to be used in practice.

To fill in this gap and to tackle the computational challenges mentioned above, we aim to improve the approximate class-specific kernel spectral regression method in this work. The main contribution is four-fold. First, we give a correction to the eigenanalysis step used in the ACS-KSR method, and consider how to solve the *ratio-trace* problem and the *trace-ratio* problem by exploiting the structure of the intra-class and out-of-class scatter matrices. Second, we consider low-rank property of the Gaussian kernel matrix, and provide a practical strategy for determining numerical rank of kernel matrix without forming it beforehand. Third, based on the numerically low-rank property of Gaussian kernel matrix, we provide a modified Nyström method with fixed-rank for the kernel regression step, and establish a probabilistic error bound on the approximation. Although the proposed Nyström method can reduce the computational cost of the original method, it is required to form and store the reduced kernel matrix $\widetilde{K}$ explicitly. In the era of big data, however,

Table 1 Some notations used in this paper

Notations	Description
m, n	The data dimension and the number of training samples
n_1, n_2	The number of client class, and the number of impostor class, with $n = n_1 + n_2$
d, r	Discriminant space dimensionality, and reference vector set cardinality
$\mathbf{g}, [n]$	The binary label vector, and the set $\{1, 2, \ldots, n\}$
k, l	The target rank and the number of sampling, with $l > k$
$[n] \setminus [r]$	The set whose elements belong to $[n]$ but not $[r]$, i.e., the set subtraction
$\mathbf{0}, I_i$	Zero matrix or vector, and identity matrix of dimension i
$\dim(\mathscr{W})$	Dimension of the subspace $\mathscr{W}$
$\mathrm{span}\{A\}$	Subspace spanned by the columns of a matrix A
$A^T, A^\dagger$	Transpose and Moore–Penrose inverse of a matrix A
A_i, A_{ii}	The i-th column and the i-th diagonal element of A
$\mathrm{rank}(A), \mathrm{tr}(A)$	Rank and trace of a matrix A
$[\![A]\!]_k$	The best rank-k approximation to a matrix A, with $k \leq \mathrm{rank}(A)$
$\mathscr{N}(A), \mathscr{R}(A)$	Null space and range of a matrix A
$\|\cdot\|_2, \ \|\cdot\|_F$	2-norm and Frobenius norm of a vector or matrix
$\mathscr{N}(A) \setminus \mathscr{N}(B)$	The subspace in $\mathscr{N}(A)$ but not in $\mathscr{N}(B)$
$\sigma_k(A), \lambda_k(A)$	The k-th largest singular value and eigenvalue of A
$\mathbb{E}(\cdot), \mathbb{P}(\cdot)$	Expectation and probability

the reduced kernel matrix may be so huge that it can not be stored in main memory, and the proposed Nyström method can still be time-consuming. To deal with this problem, the fourth contribution of this paper is to propose a randomized block Kaczmarz method for kernel regression problem with multiple right-hand sides. The convergence of this method is established.

The structure of this paper is as follows. In Sect. 2, we briefly introduce the face verification problem, the class-specific kernel discriminant analysis method and its two variations. The eigenanalysis step involved in the CS-KSR and ACS-KSR methods is corrected in Sect. 3, moreover, we consider how to solve the trace-ratio or the ratio-trace problems involved in this step. In Sect. 4, we focus on the numerically low-rank property of the Gaussian kernel matrix, and propose a modified and fixed-rank Nyström method. To further reduce the computational overhead, in Sect. 5, we propose a randomized block Kaczmarz method for regression with multiple right-hand sides. In Sect. 6, we perform numerical experiments on some real-world data sets, to show the numerical behavior of the proposed algorithms as well as the effectiveness of our theoretical results. Some conclusions are drawn in Sect. 7. MATLAB notations are utilized in our algorithms whenever necessary, and some notations used in this paper are listed in Table 1.

2 Class-specific kernel discriminant analysis and its variants

Denote by $\mathscr{U}$ a training set consists of n facial images. Assume that all facial images in $\mathscr{U}$ have been preprocessed and represented by facial vectors $\mathbf{x}_i \in \mathbb{R}^m, i = 1, 2, \ldots, n$, which are followed by binary labels $\mathbf{g}_i \in \{+1, -1\}$ denoting whether the facial vector

$\mathbf{x}_i$ belongs to the client (+1) or the impostor (−1) class. Suppose that there are n_1 facial images belong to the client class (i.e., the person of interest), while the remaining n_2 facial images belong to the impostor class.

The kernel approaches map the input space $\mathbb{R}^m$ to the kernel space $\mathscr{F}$ by using a nonlinear function $\phi(\cdot)$, and then determine a linear projection W in the kernel space $\mathscr{F}$, such that

$$\mathbf{y}_i = W^T \phi(\mathbf{x}_i), \tag{1}$$

where $W \in \mathbb{R}^{|\mathscr{F}| \times d}$. However, the data representation $\phi(\mathbf{x}_i)$ in $\mathscr{F}$ cannot be computed directly in practice, and the kernel trick is used instead (Lu et al., 2003; Müller et al., 2001; Zheng et al., 2013). Indeed, the multiplication in (1) is inherently computed by using dot products in $\mathscr{F}$. More precisely, one exploits the kernel function $\kappa(\cdot, \cdot)$ to express dot products $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ between training data in the kernel space $\mathscr{F}$. The dot products between all the training vectors in the kernel space $\mathscr{F}$ are stored in the kernel matrix $K \in \mathbb{R}^{n \times n}$ whose i-th column is

$$\mathbf{k}_i = [\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_1), \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_n)]^T, \quad i = 1, 2, \ldots, n.$$

Denote by $\Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_n)] \in \mathbb{R}^{|\mathscr{F}| \times n}$, the kernel matrix K can be written as $K = \Phi^T \Phi$, and the projection matrix W can be represented as

$$W = \sum_{i=1}^{n} \phi(\mathbf{x}_i) \mathbf{a}_i^T = \Phi A,$$

where $A \in \mathbb{R}^{n \times d}$ is a reconstruction weights matrix. A combination of (1) and (2) yields

$$\mathbf{y}_i = A^T \Phi^T \phi(\mathbf{x}_i) = A^T \mathbf{k}_i. \tag{3}$$

In Class-Specific Kernel Discriminant Analysis (CS-KDA) (Goudelis et al., 2007), we denote by $\mathbf{m} = \frac{1}{n_1} \sum_{i, \mathbf{g}_i = 1} \mathbf{y}_i$ the client class mean vector, and define D_I, D_C the distances of the impostor vectors and the client vectors from the client class mean vector $\mathbf{m}$, respectively. From (1), we have $\mathbf{m} = W^T \mathbf{m}_\phi$, where $\mathbf{m}_\phi = \frac{1}{n_1} \sum_{i, \mathbf{g}_i = 1} \phi(\mathbf{x}_i)$ is the client class mean expressed in $\mathscr{F}$. Hence,

$$D_I = \sum_{i, \mathbf{g}_i = -1} \|W^T \phi(\mathbf{x}_i) - W^T \mathbf{m}_\phi\|_2^2 \quad \text{and} \quad D_C = \sum_{i, \mathbf{g}_i = 1} \|W^T \phi(\mathbf{x}_i) - W^T \mathbf{m}_\phi\|_2^2.$$

The objective of CS-KDA is to determine data representations $\mathbf{y}_i \in \mathbb{R}^d$ in a feature space, such that the client class is as compact as possible, while the impostor class is spread far away as much as possible from the client class. Mathematically, we aim to seek a matrix W^*, such that

$$W^* = \underset{\substack{W \in \text{span}\{\Phi\} \\ W \in \mathbb{R}^{|\mathscr{F}| \times d}}}{\text{argmax}} \frac{D_I}{D_C}. \tag{4}$$

That is,

$$W^* = \underset{\substack{W \in \text{span}\{\Phi\} \\ W \in \mathbb{R}^{|\mathscr{F}| \times d}}}{\text{argmax}} \frac{\text{tr}(W^T S_I W)}{\text{tr}(W^T S_C W)}, \tag{5}$$

where $\text{tr}(\cdot)$ is the trace of a matrix, and

$$S_I = \sum_{i, \mathbf{g}_i = -1} \big(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\big)\big(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\big)^T \in \mathbb{R}^{|\mathscr{F}| \times |\mathscr{F}|} \tag{6}$$

and

$$S_C = \sum_{i, \mathbf{g}_i = 1} \big(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\big)\big(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\big)^T \in \mathbb{R}^{|\mathscr{F}| \times |\mathscr{F}|} \tag{7}$$

are the out-of-class and the in-class scatter matrices in $\mathscr{F}$, respectively.

However, solving (5) directly is impractical since $|\mathscr{F}|$ is very large or even infinite in practice. Fortunately, by substituting (2) in (6) and (7), the Eq. (5) can be equivalently expressed as the following *trace-ratio problem* on A (Iosifidis et al., 2015)

$$\mathscr{T}(A) = \max_{A \in \mathbb{R}^{n \times d}} \frac{\text{tr}(A^T M_I A)}{\text{tr}(A^T M_C A)}, \tag{8}$$

where

$$M_I = K_I K_I^T - \frac{1}{n_1} K_I \mathbf{1}_I \mathbf{1}_C^T K_C^T - \frac{1}{n_1} K_C \mathbf{1}_C \mathbf{1}_I^T K_I^T + \frac{n_2}{n_1^2} K_C \mathbf{1}_C \mathbf{1}_C^T K_C^T \in \mathbb{R}^{n \times n}, \tag{9}$$

and

$$M_C = K_C K_C^T - \frac{1}{n_1} K_C \mathbf{1}_C \mathbf{1}_C^T K_C^T \in \mathbb{R}^{n \times n}, \tag{10}$$

and $\mathbf{1}_I \in \mathbb{R}^{n_2}$ and $\mathbf{1}_C \in \mathbb{R}^{n_1}$ are vectors of all ones, $K_I \in \mathbb{R}^{n \times n_2}$ and $K_C \in \mathbb{R}^{n \times n_1}$ are matrices formed by the columns of K corresponding to the impostor and client class data, respectively. However, the trace-ratio problem (8) is difficult to solve (Jia et al., 2009; Wang et al., 2007), and one often solves the following *ratio-trace problem* instead

$$\hat{\mathscr{T}}(A) = \max_{A \in \mathbb{R}^{n \times d}} \text{tr}\Big(\big(A^T M_C A\big)^{-1}\big(A^T M_I A\big)\Big), \tag{11}$$

which reduces to a generalized eigenproblem $M_I \mathbf{a} = \lambda M_C \mathbf{a}$. By (9) and (10), the rank of M_I and M_C are at most $n_2 + 3$ and $n_1 - 1$, respectively, and both of the two matrices are rank-deficient. Thus, the generalized eigenvalue problem can be non-regular (Golub & Van Loan, 2014). Notice that (8) and (11) are not mathematically equivalent in general (Shi & Wu, 2021).

Recently, a spectral regression-based method for (5) was proposed in Arashloo and Kittler (2014), Iosifidis et al. (2015). Let $(\lambda, \mathbf{w})$ be an eigenpair satisfying $S_I \mathbf{w} = \lambda S_C \mathbf{w}$. From (2), we have $\mathbf{w} = \Phi \mathbf{a}$, moreover, if we set $K\mathbf{a} = \mathbf{t}$, where $K = \Phi^T \Phi$ is the kernel matrix, then this eigenproblem reduces to Arashloo and Kittler (2014), Iosifidis et al. (2015)

$$P_I \mathbf{t} = \lambda P_C \mathbf{t}, \tag{12}$$

where

$$P_I = \mathbf{e}_I \mathbf{e}_I^T - \frac{1}{n_1}\mathbf{e}_I \mathbf{e}_C^T - \frac{1}{n_1}\mathbf{e}_C \mathbf{e}_I^T + \frac{1}{n_1^2}\mathbf{e}_C \mathbf{e}_C^T, \tag{13}$$

and

$$P_C = (1 - \frac{2}{n_1} + \frac{1}{n_1^2})\mathbf{e}_C \mathbf{e}_C^T, \tag{14}$$

and $\mathbf{e}_C \in \mathbb{R}^n$ is a vector whose elements $\mathbf{e}_{C,i} = 1$ if $\mathbf{g}_i = 1$ and $\mathbf{e}_{C,i} = 0$ if $\mathbf{g}_i = -1$, moreover, $\mathbf{e}_I \in \mathbb{R}^n$ is a vector whose elements $\mathbf{e}_{I,i} = 1$ if $\mathbf{g}_i = -1$ and $\mathbf{e}_{I,i} = 0$ if $\mathbf{g}_i = 1$. However, both P_I and P_C are singular, and (12) is non-regular. Usually, some regularization techniques are used, and the following eigenproblem is solved instead

$$(P_I + \alpha I_n)\mathbf{t} = \lambda P_C \mathbf{t}, \tag{15}$$

where α is a regularization parameter. In the Class-Specific Kernel Spectral Regression (CS-KSR) method, the reconstruction weights matrix A is computed as follows

- Eigenanalysis Step: Compute $T = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_d]$ from solving the large-scale eigenproblem (15), where $\mathbf{t}_i$ is the eigenvector corresponding to the i-th largest eigenvalue and d is the dimension of the discriminant space.
- Kernel Regression Step: Solving $K\mathbf{a}_i = \mathbf{t}_i, i = 1, \dots, d$, for the reconstruction weights matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_d]$.

In the aforementioned CS-KDA and CS-KSR approaches, we have to calculate and store the kernel matrix $K \in \mathbb{R}^{n \times n}$, whose computational cost is $\mathcal{O}(n^3)$ flops, and the space complexity is $\mathcal{O}(n^2)$. Here n is the number of training samples. Therefore, forming and storing the full kernel matrix K explicitly are very time consuming, especially for large classification problems (Tavernier et al., 2019).

To partially overcome this difficulty and speedup the kernel regression step, an approximate class-specific kernel spectral regression (ACS-KSR) method is proposed (Iosifidis & Gabbouj, 2016), in which an approximate kernel space is exploited. More precisely, in terms of the structure of the intra-class and out-of-class scatter matrices, Iosifidis and Gabbouj (2016) show that the eigenanalysis step can be solved by applying a much simpler and faster process involving only the class labels of the training data; see Algorithm 1. Furthermore, recall that CS-KSR resorts to the following kernel regression problem

$$W^* = \underset{\substack{W \in \text{span}\{\Phi\} \\ W \in \mathbb{R}^{|\mathcal{F}| \times d}}}{\text{argmin}} \quad \|W^T \Phi - T^T\|_F^2, \tag{16}$$

and the matrix A is obtained from expressing W as a linear combination of all training data representations in the kernel space, i.e., $W = \Phi A$.

The key idea of Iosifidis and Gabbouj (2016) is that the matrix W is expressed as a linear combination of r reference vectors, i.e., $W = \Psi A$, where $\Psi \in \mathbb{R}^{|\mathcal{F}| \times r}$ with $r < n$. In this case, the kernel regression problem (16) can be written as

$$A^* = \underset{A\in\mathbb{R}^{r\times d}}{\operatorname{argmin}} \|A^T\Psi^T\Phi - T^T\|_F^2 = \underset{A\in\mathbb{R}^{r\times d}}{\operatorname{argmin}} \|\widetilde{K}^T A - T\|_F^2, \tag{17}$$

where $\widetilde{K} = \Psi^T\Phi \in \mathbb{R}^{r\times n}$ is a reduced kernel matrix expressing the training data representations in a kernel space defined on the reference data Ψ. As a result, we have from (17) that Golub and Van Loan (2014)

$$A^* = (\widetilde{K}\widetilde{K}^T)^{-1}\widetilde{K}T. \tag{18}$$

The ACS-KSR method is presented in Algorithm 2.

Algorithm 1 Eigenanalysis Step: Calculation of the matrix T [23]

Input: Training labels $\mathbf{g}$ and the discriminant space dimensionality d;
Output: The target matrix T.
1. Computing the number of training data: $n = length(\mathbf{g})$;
2. Let $T = rand(2, d+1)$ and $Z = zeros(n, d+1)$;
3. Compute the client $(+1)$ index and impostor (-1) index:
$\mathbf{f}_1 = find(\mathbf{g} == 1)$; $\mathbf{f}_2 = find(\mathbf{g} == -1)$;
4. Let the rows in Z corresponding to the index $\mathbf{f}_1$ be the first row vector in T:
$Z(\mathbf{f}_1, :) = repmat(T(1,:), length(\mathbf{f}_1), 1)$;
5. Let the rows in Z corresponding to the index $\mathbf{f}_2$ be the second row vector in T:
$Z(\mathbf{f}_2, :) = repmat(T(2,:), length(\mathbf{f}_2), 1)$;
6. Let $Z(:,1) = ones(n,1)/\sqrt{n}$;
7. Compute the economized QR decomposition of Z: $[Q, \sim] = qr(Z, 0)$;
8. Let $Q(:,1) = []$; $T = Q$.

Algorithm 2 The Approximate Class-Specific Kernel Spectral Regression Method (ACS-KSR) [23]

Input: Reduced kernel matrix $\widetilde{K}$, training labels $\mathbf{g}$ and the discriminant space dimensionality d;
Output: The reconstruction weights matrix A.
1. Compute the target matrix T by using Algorithm 1;
2. Compute the Cholesky factorization: $[R, p_c] = chol(\widetilde{K}\widetilde{K}^T)$;
3. $A = R \setminus (R^T \setminus \widetilde{K}) * T$.

Remark 1 Some remarks are in order. The adoption of such approximate kernel regression scheme leads to an important reduction on memory requirements, which allows one to apply ACS-KSR method to large-scale verification problems. Unfortunately, we find that the formulas (13) and (14) for computing T, which are widely used in Iosifidis et al. (2015), Iosifidis and Gabbouj (2016), Iosifidis and Gabbouj (2016), Iosifidis and Gabbouj (2017), Cao et al. (2018), are incomplete. On the other hand, an explicit computation of $\widetilde{K}\widetilde{K}^T$ will cost us $\mathcal{O}(nr^2)$ flops, and some useful information in $\widetilde{K}$ may be lost when forming the cross-product matrix (Golub & Van Loan, 2014). Furthermore, both the CS-KDA method and the ACS-KSR method focus on ratio-trace problems, rather than the original trace-ratio

problem (5). Thus, it is necessary to give new insight into the ACS-KSR method, and improve this method substantially.

3 On the ratio-trace and the trace-ratio problems for the eigenanalysis step

In this section, we first show that (13) and (14) are *incomplete* for solving the ratio-trace problem (11) corresponding to (8). Some corrections to the two matrices P_I and P_C are given. Second, we consider how to solve the *trace-ratio* problem (8) and the corresponding *ratio-trace* problem (11) efficiently.

Theorem 3.1 *Let $E_I \in \mathbb{R}^{n\times n_2}$ and $E_C \in \mathbb{R}^{n\times n_1}$ be matrices constituted by some columns of the identity matrix $I_n \in \mathbb{R}^{n\times n}$, corresponding to the impostor and client class index, respectively. Then under the above notations,* (8) *is equivalent to the following trace-ratio problem*

$$\max_{T\in\mathbb{R}^{n\times d}} \frac{\mathrm{tr}(T^T P_I T)}{\mathrm{tr}(T^T P_C T)}, \tag{19}$$

where

$$P_I = E_I E_I^T - \frac{1}{n_1}\mathbf{e}_I\mathbf{e}_C^T - \frac{1}{n_1}\mathbf{e}_C\mathbf{e}_I^T + \frac{n_2}{n_1^2}\mathbf{e}_C\mathbf{e}_C^T, \tag{20}$$

and

$$P_C = E_C E_C^T - \frac{1}{n_1}\mathbf{e}_C\mathbf{e}_C^T, \tag{21}$$

where $\mathbf{e}_C \in \mathbb{R}^n$ is a vector with elements $\mathbf{e}_{C,i} = 1$ if $\mathbf{g}_i = 1$, and $\mathbf{e}_{C,i} = 0$ if $\mathbf{g}_i = -1$, and $\mathbf{e}_I \in \mathbb{R}^n$ is a vector with elements $\mathbf{e}_{I,i} = 1$ if $\mathbf{g}_i = -1$ and $\mathbf{e}_{I,i} = 0$ if $\mathbf{g}_i = 1$.

Proof Notice that $KE_I = K_I$, $KE_C = K_C$, and $E_I\mathbf{1}_I = \mathbf{e}_I$, $E_C\mathbf{1}_C = \mathbf{e}_C$. It follows from (9) that

$$\begin{aligned} M_I =& K_I K_I^T - \frac{1}{n_1}K_I\mathbf{1}_I\mathbf{1}_C^T K_C^T - \frac{1}{n_1}K_C\mathbf{1}_C\mathbf{1}_I^T K_I^T + \frac{n_2}{n_1^2}K_C\mathbf{1}_C\mathbf{1}_C^T K_C^T \\ =& K\Big(E_I E_I^T - \frac{1}{n_1}E_I\mathbf{1}_I\mathbf{1}_C^T E_C^T - \frac{1}{n_1}E_C\mathbf{1}_C\mathbf{1}_I^T E_I^T + \frac{n_2}{n_1^2}E_C\mathbf{1}_C\mathbf{1}_C^T E_C^T\Big)K \\ =& K\Big(E_I E_I^T - \frac{1}{n_1}\mathbf{e}_I\mathbf{e}_C^T - \frac{1}{n_1}\mathbf{e}_C\mathbf{e}_I^T + \frac{n_2}{n_1^2}\mathbf{e}_C\mathbf{e}_C^T\Big)K \\ \equiv& KP_I K, \end{aligned} \tag{22}$$

where $P_I = E_I E_I^T - \frac{1}{n_1}\mathbf{e}_I\mathbf{e}_C^T - \frac{1}{n_1}\mathbf{e}_C\mathbf{e}_I^T + \frac{n_2}{n_1^2}\mathbf{e}_C\mathbf{e}_C^T$.

Similarly, we obtain from (10) that

$$\begin{aligned} M_C =& K_C K_C^T - \frac{1}{n_1} K_C \mathbf{1}_C \mathbf{1}_C^T K_C^T \\ =& K\big(E_C E_C^T - \frac{1}{n_1} E_C \mathbf{1}_C \mathbf{1}_C^T E_C^T\big) K \\ =& K\big(E_C E_C^T - \frac{1}{n_1} \mathbf{e}_C \mathbf{e}_C^T\big) K \\ \equiv& K P_C K, \end{aligned} \tag{23}$$

where $P_C = E_C E_C^T - \frac{1}{n_1}\mathbf{e}_C \mathbf{e}_C^T$. Substitute (22) and (23) into (8), and note that $KA = T$, the trace-ratio problem (8) can be equivalently rewritten as (19). □

Remark 2 Theorem 3.1 indicates that (13) and (14) are mathematically incomplete, and (20), (21) give some corrections to them. Unlike (13) and (14), it is seen that the two new matrices are not rank-2 and rank-1 matrices any more.

With (20) and (21) at hand, we consider how to solve the optimization problem (19) efficiently. So as to get structured intra-class and out-of-class scatter matrices, we first reorder the elements in the binary label vector $\mathbf{g}$. More precisely, suppose that the training binary label vector $\mathbf{g}$ is permuted to the binary label $\widetilde{\mathbf{g}}$, in which all the client (+1) classes in $\widetilde{\mathbf{g}}$ are sorted before all the impostor (−1) classes. Mathematically speaking, there exists a permutation matrix $P \in \mathbb{R}^{n\times n}$ such that $P\mathbf{g} = \widetilde{\mathbf{g}}$. Corresponding to $\mathbf{e}_I$, $\mathbf{e}_C$, E_I and E_C, we define the four variables $\widetilde{\mathbf{e}}_I$, $\widetilde{\mathbf{e}}_C$, $\widetilde{E}_I$ and $\widetilde{E}_C$ with respect to the new binary label $\widetilde{\mathbf{g}}$. Moreover, we have that $P\mathbf{e}_I = \widetilde{\mathbf{e}}_I$, $P\mathbf{e}_C = \widetilde{\mathbf{e}}_C$, $PE_I = \widetilde{E}_I$ and $PE_C = \widetilde{E}_C$, and

$$\widetilde{E}_I^T \widetilde{E}_C = \mathbf{0}, \quad \widetilde{E}_C \mathbf{1}_C = \widetilde{\mathbf{e}}_C, \quad \widetilde{\mathbf{e}}_I^T \widetilde{E}_C = \mathbf{0}. \tag{24}$$

Denote by $\widetilde{P}_I = PP_I P^T$, then it follows from (20) that

$$\begin{aligned} \widetilde{P}_I =& PP_I P^T = P\Big(E_I E_I^T - \frac{1}{n_1}\mathbf{e}_I \mathbf{e}_C^T - \frac{1}{n_1}\mathbf{e}_C \mathbf{e}_I^T + \frac{n_2}{n_1^2}\mathbf{e}_C \mathbf{e}_C^T\Big) P^T \\ =& \widetilde{E}_I \widetilde{E}_I^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_I \widetilde{\mathbf{e}}_C^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_C \widetilde{\mathbf{e}}_I^T + \frac{n_2}{n_1^2}\widetilde{\mathbf{e}}_C \widetilde{\mathbf{e}}_C^T. \end{aligned} \tag{25}$$

Similarly, denote by $\widetilde{P}_C = PP_C P^T$, we have from (21) that

$$\widetilde{P}_C = PP_C P^T = P\Big(E_C E_C^T - \frac{1}{n_1}\mathbf{e}_C \mathbf{e}_C^T\Big) P^T = \widetilde{E}_C \widetilde{E}_C^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_C \widetilde{\mathbf{e}}_C^T. \tag{26}$$

Therefore, combining (25) and (26), the Eq. (19) can be rewritten as

$$\frac{\mathrm{tr}(T^T P_I T)}{\mathrm{tr}(T^T P_C T)} = \frac{\mathrm{tr}\big(T^T P^T (PP_I P^T) PT\big)}{\mathrm{tr}\big(T^T P^T (PP_C P^T) PT\big)} \equiv \frac{\mathrm{tr}(\widetilde{T}^T \widetilde{P}_I \widetilde{T})}{\mathrm{tr}(\widetilde{T}^T \widetilde{P}_C \widetilde{T})},$$

where $\widetilde{T} = PT$, and we make use of the property $P^T P = I_n$, as P is a permutation matrix.

In summary, we solve the target matrix T in the following two steps:

- Solving the following trace-ratio problem

$$\widehat{T}_{tr} = \underset{\widetilde{T}\in\mathbb{R}^{n\times d}}{\operatorname{argmax}} \frac{\operatorname{tr}(\widetilde{T}^T\widetilde{P}_I\widetilde{T})}{\operatorname{tr}(\widetilde{T}^T\widetilde{P}_C\widetilde{T})}, \tag{27}$$

or the ratio-trace problem

$$\widehat{T}_{rt} = \underset{\widetilde{T}\in\mathbb{R}^{n\times d}}{\operatorname{argmax}} \operatorname{tr}\left[(\widetilde{T}^T\widetilde{P}_C\widetilde{T})^{-1}(\widetilde{T}^T\widetilde{P}_I\widetilde{T})\right] \tag{28}$$

for the matrix $\widehat{T}$, where $\widetilde{P}_I$ and $\widetilde{P}_C$ are defined in (25) and (26), respectively.

- Let $T = P^T\widehat{T}$.

Remark 3 An advantage of the problem (27) over the original one (19) is that, one can take full advantage of the special structure of matrices $\widetilde{P}_I$ and $\widetilde{P}_C$. Keep in mind that there is no need to form and store the perturbation matrix P explicitly in the two methods.

Next, we propose two methods for solving the ratio-trace problem (28) and the trace-ratio problem (27), respectively.

3.1 Solution of the ratio-trace problem for the eigenanalysis step

It is well known that the *trace-ratio problem* (27) is difficult to solve (Jia et al., 2009; Wang et al., 2007). As an alternative, one often solves the relatively easier *ratio-trace problem* (28). Note that it is different from the one given in (12) which is widely used in Iosifidis et al., (2015), Iosifidis and Gabbouj (2016), Iosifidis and Gabbouj (2016), Iosifidis and Gabbouj (2017), Cao et al., (2018). First, we show that both $\widetilde{P}_I$ and $\widetilde{P}_C$ are positive semidefinite matrices. Recall that we have obtained four structured variables $\widetilde{\mathbf{e}}_I$, $\widetilde{\mathbf{e}}_C$, $\widetilde{E}_I$ and $\widetilde{E}_C$ with respect to the new binary label $\widetilde{\mathbf{g}}$. In fact, due to the characteristics of the new binary label $\widetilde{\mathbf{g}}$, we see that $\widetilde{E}_C$ and $\widetilde{E}_I$ are the first n_1 columns and the last n_2 columns of the identity matrix $I_n \in \mathbb{R}^{n\times n}$, respectively. In addition, the first n_1 elements of $\widetilde{\mathbf{e}}_C$ are all 1 and the rest are all 0, and the last n_2 elements of $\widetilde{\mathbf{e}}_I$ are all 1 and the rest are all 0. Thus, by (25) and (26), the two matrices $\widetilde{P}_I$ and $\widetilde{P}_C$ are block matrices of the following form, i.e.,

$$\widetilde{P}_I = \begin{pmatrix} \frac{n_2}{n_1^2}\mathbf{1}_C\mathbf{1}_C^T & -\frac{1}{n_1}\mathbf{1}_C\mathbf{1}_I^T \\ -\frac{1}{n_1}\mathbf{1}_I\mathbf{1}_C^T & I_{n_2} \end{pmatrix}, \tag{29}$$

and

$$\widetilde{P}_C = \begin{pmatrix} I_{n_1} - \frac{1}{n_1}\mathbf{1}_C\mathbf{1}_C^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{30}$$

On one hand, since the eigenvalues of $I_{n_1} - \frac{1}{n_1}\mathbf{1}_C\mathbf{1}_C^T$ are either 1 or 0, $\widetilde{P}_C$ is positive semidefinite. On the other hand, we consider the following positive semidefinite matrix

$$B = \begin{pmatrix} \frac{\sqrt{n_2}}{n_1}\mathbf{1}_C \\ -\frac{1}{\sqrt{n_2}}\mathbf{1}_I \end{pmatrix}\begin{pmatrix} \frac{\sqrt{n_2}}{n_1}\mathbf{1}_C^T & -\frac{1}{\sqrt{n_2}}\mathbf{1}_I^T \end{pmatrix} = \begin{pmatrix} \frac{n_2}{n_1^2}\mathbf{1}_C\mathbf{1}_C^T & -\frac{1}{n_1}\mathbf{1}_C\mathbf{1}_I^T \\ -\frac{1}{n_1}\mathbf{1}_I\mathbf{1}_C^T & \frac{1}{n_2}\mathbf{1}_I\mathbf{1}_I^T \end{pmatrix}.$$

Notice that

$$\widetilde{P}_I - B = \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{n_2} - \frac{1}{n_2}\mathbf{1}_I\mathbf{1}_I^T \end{pmatrix}$$

is a positive semidefinite matrix. Thus, $\widetilde{P}_I$ is also positive semidefinite.

Indeed, the solution of the ratio-trace problem (28) can be reduced to the following generalized eigenvalue problem (Duda et al., 2000)

$$\widetilde{P}_I\widetilde{\mathbf{t}} = \widetilde{\lambda}\widetilde{P}_C\widetilde{\mathbf{t}}.$$

However, both $\widetilde{P}_I$ and $\widetilde{P}_C$ may be singular, and this generalized eigenvalue problem can be non-regular in practice (Golub and Van Loan, 2014). One remedy is to use the regularized technique

$$(\widetilde{P}_I + \alpha I_n)\widehat{\mathbf{t}} = \widehat{\lambda}\widetilde{P}_C\widehat{\mathbf{t}}, \tag{31}$$

where $\alpha > 0$ is a user-described regularization parameter.

Denote by $\widetilde{P} = (\widetilde{P}_I + \alpha I_n)^{-1}\widetilde{P}_C$, we are interested in the eigenvectors corresponding to the smallest d eigenvalues of the matrix $\widetilde{P}$. As

$$\begin{aligned}\widetilde{P}_I &= \widetilde{E}_I\widetilde{E}_I^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_I\widetilde{\mathbf{e}}_C^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_C\widetilde{\mathbf{e}}_I^T + \frac{n_2}{n_1^2}\widetilde{\mathbf{e}}_C\widetilde{\mathbf{e}}_C^T \\ &= \begin{pmatrix} \widetilde{E}_I & \widetilde{\mathbf{e}}_I & \widetilde{\mathbf{e}}_C \end{pmatrix} \begin{pmatrix} I_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{n_1} \\ \mathbf{0} & -\frac{1}{n_1} & \frac{n_2}{n_1^2} \end{pmatrix} \begin{pmatrix} \widetilde{E}_I^T \\ \widetilde{\mathbf{e}}_I^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix} \in \mathbb{R}^{n\times n},\end{aligned}$$

it follows from the Sherman–Morrison–Woodbury formula (Golub & Van Loan, 2014) that

$$\begin{aligned}(\widetilde{P}_I + \alpha I_n)^{-1} &= \left[\alpha I_n + \begin{pmatrix} \widetilde{E}_I & \widetilde{\mathbf{e}}_I & \widetilde{\mathbf{e}}_C \end{pmatrix} \begin{pmatrix} I_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{n_1} \\ \mathbf{0} & -\frac{1}{n_1} & \frac{n_2}{n_1^2} \end{pmatrix} \begin{pmatrix} \widetilde{E}_I^T \\ \widetilde{\mathbf{e}}_I^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix}\right]^{-1} \\ &= \frac{1}{\alpha}I_n - \frac{1}{\alpha^2}\begin{pmatrix} \widetilde{E}_I & \widetilde{\mathbf{e}}_I & \widetilde{\mathbf{e}}_C \end{pmatrix}\Theta^{-1} \begin{pmatrix} I_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{n_1} \\ \mathbf{0} & -\frac{1}{n_1} & \frac{n_2}{n_1^2} \end{pmatrix} \begin{pmatrix} \widetilde{E}_I^T \\ \widetilde{\mathbf{e}}_I^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix},\end{aligned}$$

where

$$\Theta = I_{n_2+2} + \frac{1}{\alpha}\begin{pmatrix} I_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{n_1} \\ \mathbf{0} & -\frac{1}{n_1} & \frac{n_2}{n_1^2} \end{pmatrix} \begin{pmatrix} \widetilde{E}_I^T \\ \widetilde{\mathbf{e}}_I^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix} \begin{pmatrix} \widetilde{E}_I & \widetilde{\mathbf{e}}_I & \widetilde{\mathbf{e}}_C \end{pmatrix} \in \mathbb{R}^{(n_2+2)\times(n_2+2)}.$$

Thus,

$$\begin{aligned}\widetilde{P} &= (\widetilde{P}_I + \alpha I_n)^{-1}\widetilde{P}_C \\ &= \frac{1}{\alpha}\widetilde{P}_C - \frac{1}{\alpha^2}\left(\widetilde{E}_I\ \widetilde{\mathbf{e}}_I\ \widetilde{\mathbf{e}}_C\right)\Theta^{-1}\begin{pmatrix} I_{n_2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\frac{1}{n_1} \\ \mathbf{0} & -\frac{1}{n_1} & \frac{n_2}{n_1^2} \end{pmatrix}\begin{pmatrix} \widetilde{E}_I^T \\ \widetilde{\mathbf{e}}_I^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix}\widetilde{P}_C. \end{aligned} \tag{32}$$

Next, we will prove that

$$\begin{pmatrix} \widetilde{E}_I^T \\ \widetilde{\mathbf{e}}_I^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix}\widetilde{P}_C = \begin{pmatrix} \widetilde{E}_I^T\widetilde{P}_C \\ \widetilde{\mathbf{e}}_I^T\widetilde{P}_C \\ \widetilde{\mathbf{e}}_C^T\widetilde{P}_C \end{pmatrix} = \mathbf{0}. \tag{33}$$

On one hand, we obtain from (26) that

$$\widetilde{P}_C = \widetilde{E}_C\widetilde{E}_C^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_C\widetilde{\mathbf{e}}_C^T = \left(\widetilde{E}_C\ \widetilde{\mathbf{e}}_C\right)\begin{pmatrix} I_{n_1} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{n_1} \end{pmatrix}\begin{pmatrix} \widetilde{E}_C^T \\ \widetilde{\mathbf{e}}_C^T \end{pmatrix}.$$

From $\widetilde{E}_C\mathbf{1}_C = \widetilde{\mathbf{e}}_C$, we have $\text{span}\{\widetilde{P}_C\} \subseteq \text{span}\{\widetilde{E}_C\}$. A combination of the above equation with (24) yields

$$\widetilde{E}_I^T\widetilde{P}_C = \mathbf{0} \ \text{ and } \ \widetilde{\mathbf{e}}_I^T\widetilde{P}_C = \mathbf{0}. \tag{34}$$

On the other hand, we have from (26) that

$$\widetilde{\mathbf{e}}_C^T\widetilde{P}_C = \widetilde{\mathbf{e}}_C^T(\widetilde{E}_C\widetilde{E}_C^T - \frac{1}{n_1}\widetilde{\mathbf{e}}_C\widetilde{\mathbf{e}}_C^T) = \mathbf{1}_C^T\widetilde{E}_C^T - \frac{n_1}{n_1}\widetilde{\mathbf{e}}_C^T = \widetilde{\mathbf{e}}_C^T - \widetilde{\mathbf{e}}_C^T = \mathbf{0}. \tag{35}$$

So we get (33) from combining (34) and (35). In conclusion, we have from (30), (32) and (33) that

$$\widetilde{P} = \frac{1}{\alpha}\widetilde{P}_C = \frac{1}{\alpha}\begin{pmatrix} I_{n_1} - \frac{1}{n_1}\mathbf{1}_C\mathbf{1}_C^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}. \tag{36}$$

Thus, $\widetilde{P}$ has $n_2 + 1$ eigenvalues 0 and $n_1 - 1$ eigenvalues which equal to $\frac{1}{\alpha}$. In practice, we often have $d \leq n_1 - 1$ and $n_2 \geq n_1$ (Iosifidis and Gabbouj 2016), and (28) can be reduced to the problem of finding d vectors in the null space of $\widetilde{P}$.

Hence, it is only necessary to consider the null space of $\widetilde{P}$. Assume that $\mathbf{x} \in \mathcal{N}(\widetilde{P})$, and let $\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T]^T \in \mathbb{R}^n$, with $\mathbf{x}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{n_2}$. Then we have from (36) that

$$\begin{pmatrix} I_{n_1} - \frac{1}{n_1}\mathbf{1}_C\mathbf{1}_C^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix},$$

which can be equivalently rewritten as

$$\begin{cases} (I_{n_1} - \frac{1}{n_1}\mathbf{1}_C\mathbf{1}_C^T)\mathbf{x}_1 = \mathbf{0}, \\ \qquad\qquad \forall\, \mathbf{x}_2 \in \mathbb{R}^{n_2}. \end{cases} \tag{37}$$

As a result, the solution of (28) has the following form

$$\widehat{T}_{rt} = \begin{pmatrix} \mathbf{1}_C & \mathbf{1}_C & \cdots & \mathbf{1}_C \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_d \end{pmatrix} \in \mathbb{R}^{n\times d}, \tag{38}$$

where $\mathbf{x}_i \in \mathbb{R}^{n_2}, i = 1, 2, \ldots, d$, are arbitrary such that the columns of $\widehat{T}_{rt}$ are linear independent. In summary, we have Algorithm 3 for the eigenanalysis step.

Algorithm 3 Eigenanalysis Step: Calculation of T based on the *ratio-trace* model (3.10)

Input: Training labels $\mathbf{g}$, the discriminant space dimensionality d, the number of training samples n and the number of client class n_1;
Output: The target matrix T.
1. $T = rand(n - n_1, d)$;
2. $T = [ones(n_1, d); T]$;
3. $[T, \sim] = qr(T, 0)$;
4. $[\sim, \mathbf{c}] = sort(\mathbf{g}, \text{'descend'})$;
5. $[\sim, \mathbf{c}] = sort(\mathbf{c})$;
6. $T = T(\mathbf{c}, :)$.

3.2 Solution of the trace-ratio problem for the eigenanalysis step

In the previous subsection, we solve the ratio-trace problem (28) for the eigenanalysis step. However, the ratio-trace model and the trace-ratio model (27) are not mathematically equivalent (Park and Park, 2008; Shi and Wu, 2021). The trace-ratio problem has regained great concerns in recent years. The reason is that the trace-ratio model can yield markedly improved recognition results compared with the ratio-trace model (Jia et al., 2009; Ngo et al., 2012; Shi & Wu, 2021; Wang et al., 2007).

In this subsection, we focus on the *trace-ratio* problem (3.9). It has been long believed that there is *no closed-form solution* for the trace-ratio problem, and some commonly used techniques are inner-outer iterative methods (Jia et al., 2009; Ngo et al., 2012; Wang et al., 2007; Zhao et al., 2013). Recently, Shi and Wu point out that the trace-ratio problem has a *close-form solution* when the dimension of data points is greater than or equal to the number of training samples (Shi & Wu, 2021), as the following theorem indicates.

Theorem 3.2 *Shi and Wu* (2021) *Let* $\widetilde{P}_T = \widetilde{P}_I + \widetilde{P}_C$, *then the subspace* $\mathcal{M}(\widetilde{P}_C) \setminus \mathcal{M}(\widetilde{P}_T)$, *i.e., the subspace in* $\mathcal{M}(\widetilde{P}_C)$ *but not in* $\mathcal{M}(\widetilde{P}_T)$, *is the solution space of the trace-ratio problem* (27). *Let d be the reducing dimension, if* $\dim(\mathcal{M}(\widetilde{P}_C) \setminus \mathcal{M}(\widetilde{P}_T)) \geq d$, *then any orthonormal basis of a d-dimensional subspace of* $\mathcal{M}(\widetilde{P}_C) \setminus \mathcal{M}(\widetilde{P}_T)$, *is a solution to* (27).

Based on Theorem 3.2 and the structure of the three matrices $\widetilde{P}_I$, $\widetilde{P}_C$ and $\widetilde{P}_T$, we consider how to solve trace-ratio problem (27) efficiently. First, we obtain from (36) that

$$\mathcal{M}(\widetilde{P}_C) = \mathcal{M}(\widetilde{P}) = \text{span}\{\widehat{T}_{rt}\}. \tag{39}$$

Second, it follows from (29) and (30) that

$$\begin{aligned}\widetilde{P}_T =& \widetilde{P}_I + \widetilde{P}_C \\ =& \begin{pmatrix} I_{n_1} + \frac{n_2-n_1}{n_1^2}\mathbf{1}_C\mathbf{1}_C^T & -\frac{1}{n_1}\mathbf{1}_C\mathbf{1}_I^T \\ -\frac{1}{n_1}\mathbf{1}_I\mathbf{1}_C^T & I_{n_2} \end{pmatrix}.\end{aligned}$$

Suppose that $\mathbf{x} \in \mathcal{N}(\widetilde{P}_T)$, and let $\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T]^T \in \mathbb{R}^n$, where $\mathbf{x}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{n_2}$, we have

$$\begin{pmatrix} I_{n_1} + \frac{n_2-n_1}{n_1^2}\mathbf{1}_C\mathbf{1}_C^T & -\frac{1}{n_1}\mathbf{1}_C\mathbf{1}_I^T \\ -\frac{1}{n_1}\mathbf{1}_I\mathbf{1}_C^T & I_{n_2} \end{pmatrix}\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix},$$

which is equivalent to

$$\begin{cases} (I_{n_1} - \dfrac{1}{n_1}\mathbf{1}_C\mathbf{1}_C^T)\mathbf{x}_1 = \mathbf{0}, \\ \dfrac{1}{n_1}\mathbf{1}_I\mathbf{1}_C^T\mathbf{x}_1 = \mathbf{x}_2. \end{cases} \tag{40}$$

Therefore, the solution is $\mathbf{x} = [\mathbf{1}_C^T \ \mathbf{1}_I^T]^T = \mathbf{1}_n \in \mathbb{R}^n$. So we obtain from Theorem 3.2 that

$$\widehat{T}_{tr} = (I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T) \cdot \widehat{T}_{rt} \tag{41}$$

is a solution to trace-ratio problem (27). We present Algorithm 4 for solving the eigenanalysis step. It is seen that the solutions to the trace-ratio problem (27) and the ratio-trace problem (28) are related, but are different from each other in essence.

Algorithm 4 Eigenanalysis Step: Calculation of T based on the *trace-ratio* model (3.9)

Input: Training labels $\mathbf{g}$, the discriminant space dimensionality d, the number of training samples n and the number of client class n_1;
Output: The target matrix T.

1. Compute T in (3.20) using the first two steps of Algorithm 3;
2. $\mathbf{e} = ones(n, 1)$;
3. $\mathbf{e} = orth(\mathbf{e})$;
4. $T = T - \mathbf{e} * (\mathbf{e}^T * T)$;
5. $[T, \sim] = qr(T, 0)$;
6. $[\sim, \mathbf{c}] = sort(\mathbf{g}, \text{'descend'})$;
7. $[\sim, \mathbf{c}] = sort(\mathbf{c})$;
8. $T = T(\mathbf{c}, :)$.

4 A modified Nyström method based on low-rank approximation for the kernel regression step

In this section, we focus on the kernel regression step. In conventional methods, one has to compute the kernel matrix $K \in \mathbb{R}^{n\times n}$ in this step, and the computational complexities and the storage requirements are $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$, respectively. This will be very time-consuming and even be infeasible when n is extremely large.

Fortunately, kernel matrix is often approximately low-rank, based on the observation that the spectrum of the Gaussian kernel decays rapidly (Hofmann et al., 2008; Pan et al., 2011; Wathen & Zhu, 2015; Wang et al., 2018). Hence, devising scalable algorithms for kernel methods has long been an active research topic, and the key is to construct low-rank approximations to the kernel matrix (Iosifidis et al., 2015; Wang et al., 2018; Wathen & Zhu, 2015). For example, an interpolation method was used to construct the approximation of kernel matrix (Cambier & Darve, 2019; Xing & Chow, 2020), and a low-rank approximation was constructed in Iske et al., (2017) with the help of hierarchical low-rank property of kernel matrix. However, to the best of our knowledge, most of the existing results are purely theoretical and are difficult to use in practice.

In this section, we first show the numerically low-rank property of the popular used Gaussian kernel matrix from a theoretical point of view. Based on the proposed results, we shed light on how to determine an appropriate target rank for randomized algorithms. We then provide a modified Nyström method with fixed-rank, and establish a probabilistic error bound on the low-rank approximation.

4.1 On the approximately low-rank property of kernel matrix

Low-rank approximations are popular techniques to reduce the high computational cost of algorithms for large-scale kernel matrices (Halko et al., 2011; Hofmann et al., 2008; Wang et al., 2018; Wathen & Zhu, 2015). In essence, the success of these low-rank algorithms hinges on a large spectrum gap or a fast decay of the spectrum of the kernel matrix (Halko et al., 2011; Hofmann et al., 2008; Wang et al., 2018; Wathen & Zhu, 2015). This motivates the analysis on the numerical rank of kernel matrix; see Bach (2013), Wathen and Zhu (2015), Wang et al. (2018) and the references therein. However, it seems there are few theoretical results that can provide both simple and feasible strategies for the target rank used in randomized algorithms hitherto.

To fill in this gap, we investigate the numerical rank of the kernel matrix, and provide a suitable target rank for practical use. The popular used Radial Basis Function (RBF) or Gaussian kernel function is considered

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\Big), \tag{42}$$

where the value of the Gaussian scale σ is set to be the mean Euclidean distance between the training vectors, corresponding to the natural scaling value of each data set (Iosifidis & Gabbouj, 2016). We need the following definition for numerical rank of a matrix.

Definition 4.1 Higham and Mary (2019) Let $A \in \mathbb{R}^{n\times n}$ be nonzero. For $k \le n$, the rank-k accuracy of A is

$$\varepsilon_k(A) = \min_{W_k \in \mathbb{R}^{n\times n}} \left\{ \frac{\|A - W_k\|_2}{\|A\|_2} \,:\, rank(W_k) \le k \right\}. \tag{43}$$

We call W_k an optimal rank-k approximation to A if W_k achieves the minimum in (43). The numerical rank of A at accuracy ε, denoted by $k_\varepsilon(A)$, is

$$k_\varepsilon(A) = \min\{k \,:\, \varepsilon_k(A) \le \varepsilon\}.$$

The matrix A is of low numerical rank if $\varepsilon_k(A) \ll 1$ for some $k \ll n$.

Let $U\Sigma V^T$ be the singular value decomposition (SVD) of A, with singular values $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n$. Denote by U_j, V_j be the matrices composed of the first j columns of U, V, respectively, and by Σ_j the j-by-j principle submatrix of Σ. In terms of Definition 4.1, $W_j = U_j\Sigma_j V_j^T$ is an optimal rank-j approximation to A, and if

$$\varepsilon_j(A) = \frac{\sigma_{j+1}(A)}{\sigma_1(A)} \ll 1, \tag{44}$$

then the matrix A is of low numerical rank, where $\sigma_{j+1}(A)$ is the $(j+1)$-th largest singular value of A.

The main aim of this subsection is to show that kernel matrix K has low numerical rank which depends on the number of clusters s. Let $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \in \mathbb{R}^{m\times n}$ be the set of training samples, with $\mathbf{x}_i \in \mathbb{R}^m$ and $\|\mathbf{x}_i\|_2 = 1, i = 1, \dots, n$. Assume that the data matrix is partitioned into s classes as $X = [X_1, X_2, \dots, X_s]$, where X_j is the j-th set with n_j being the number of samples. In supervised methods, the number of s is known, otherwise, one can use, say, the K-means method (Wu et al., 2008) for choosing an appropriate s in advance. Additionally, denote by $\mathbf{c}_j$ the centroid vector of X_j and by $\Delta_j = X_j - \mathbf{c}_j\mathbf{1}_{n_j}^T, j = 1, 2, \dots, s$, then

$$X = [X_1, X_2, \dots, X_s] = [\mathbf{c}_1\mathbf{1}_{n_1}^T, \mathbf{c}_2\mathbf{1}_{n_2}^T, \dots, \mathbf{c}_s\mathbf{1}_{n_s}^T] + [\Delta_1, \Delta_2, \dots, \Delta_s] = \widehat{X} + \Delta, \tag{45}$$

where $\mathbf{1}_{n_j}$ is the vector of all ones with dimension n_j, and

$$\Delta = X - [\mathbf{c}_1\mathbf{1}_{n_1}^T, \mathbf{c}_2\mathbf{1}_{n_2}^T, \dots, \mathbf{c}_s\mathbf{1}_{n_s}^T] = [\Delta_1, \Delta_2, \dots, \Delta_s]. \tag{46}$$

Thanks to the structure of the kernel matrix K, we can decompose it into s blocks corresponding to the classification indexes in X_j, i.e.,

$$K = [K_1, K_2, \dots, K_s] = [\widehat{\mathbf{c}}_1\mathbf{1}_{n_1}^T, \widehat{\mathbf{c}}_2\mathbf{1}_{n_2}^T, \dots, \widehat{\mathbf{c}}_s\mathbf{1}_{n_s}^T] + [\widehat{\Delta}_1, \widehat{\Delta}_2, \dots, \widehat{\Delta}_s] = \widehat{K} + \widehat{\Delta}, \tag{47}$$

where $\widehat{\mathbf{c}}_j$ is the centroid vector of $K_j, j = 1, 2, \dots, s$, and

$$\widehat{K} = [\widehat{\mathbf{c}}_1\mathbf{1}_{n_1}^T, \widehat{\mathbf{c}}_2\mathbf{1}_{n_2}^T, \dots, \widehat{\mathbf{c}}_s\mathbf{1}_{n_s}^T], \quad \widehat{\Delta} = [\widehat{\Delta}_1, \widehat{\Delta}_2, \dots, \widehat{\Delta}_s]. \tag{48}$$

We are ready to present the main theorem of this subsection on numerically low-rank property of Gaussian kernel matrix.

Theorem 4.2 *Under the above notations, we have*

$$\frac{\sigma_{s+1}(K)}{\sigma_1(K)} \leq 4\sqrt{n \max_{1\leq i\leq s} n_i}\frac{e^{\zeta}}{\sigma^2} \cdot \frac{\|\Delta\|_F}{\|K\|_2}, \tag{49}$$

where σ is Gaussian scale value in the radial basis function (RBF), and $-\frac{2}{\sigma^2} < \zeta < 0$.

Proof Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be in the q-th class, $1 \leq i,j \leq n$, $1 \leq q \leq s$. First, we establish the relationship between the i-th column $\mathbf{k}_i$ and the j-th column $\mathbf{k}_j$ of the RBF kernel matrix defined in (42). Notice that

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 = \|\mathbf{x}_i - \mathbf{c}_q + \mathbf{c}_q - \mathbf{x}_j\|_2 \leq \|\mathbf{x}_i - \mathbf{c}_q\|_2 + \|\mathbf{x}_j - \mathbf{c}_q\|_2 \leq 2\|X_q - \mathbf{c}_q \mathbf{1}_{n_q}^T\|_2 = 2\|\Delta_q\|_2. \tag{50}$$

Denote by

$$t_{z,i} = -\frac{\|\mathbf{x}_z - \mathbf{x}_i\|_2^2}{2\sigma^2} \quad \text{and} \quad t_{z,j} = -\frac{\|\mathbf{x}_z - \mathbf{x}_j\|_2^2}{2\sigma^2}, \qquad 1 \leq z \leq n,$$

and without loss of generality, we suppose that $t_{z,i} < t_{z,j}$. Since the exponential function is continuous and derivable in the interval $[t_{z,i}, t_{z,j}]$, it follows from the Lagrange mean value theorem (Zoric, 2008) that there exists a point $\zeta_{i,j,z} \in (t_{z,i}, t_{z,j})$, such that

$$\begin{aligned} |e^{t_{z,j}} - e^{t_{z,i}}| =& e^{\zeta_{i,j,z}} \cdot |(t_{z,j} - t_{z,i})| \\ =& e^{\zeta_{i,j,z}} \cdot \frac{\left|\|\mathbf{x}_z - \mathbf{x}_i\|_2^2 - \|\mathbf{x}_z - \mathbf{x}_j\|_2^2\right|}{2\sigma^2}, \qquad 1 \leq z \leq n, \end{aligned} \tag{51}$$

where we have $-\frac{2}{\sigma^2} < \zeta_{i,j,z} < 0$, as $\|\mathbf{x}_i\|_2 = \|\mathbf{x}_j\|_2 = \|\mathbf{x}_z\| = 1$. Moreover, we obtain from (50) that

$$\begin{aligned} \left|\|\mathbf{x}_z - \mathbf{x}_i\|_2^2 - \|\mathbf{x}_z - \mathbf{x}_j\|_2^2\right| =& \left|(\|\mathbf{x}_z - \mathbf{x}_i\|_2 + \|\mathbf{x}_z - \mathbf{x}_j\|_2) \cdot (\|\mathbf{x}_z - \mathbf{x}_i\|_2 - \|\mathbf{x}_z - \mathbf{x}_j\|_2)\right| \\ \leq& 4\left|\|\mathbf{x}_z - \mathbf{x}_i\|_2 - \|\mathbf{x}_z - \mathbf{x}_j\|_2\right| \\ \leq& 4\|\mathbf{x}_j - \mathbf{x}_i\|_2 \leq 8\|\Delta_q\|_2. \end{aligned} \tag{52}$$

A combination of (51) and (52) yields

$$|e^{t_{z,j}} - e^{t_{z,i}}| \leq e^{\zeta_{i,j,z}} \cdot \frac{4\|\Delta_q\|_2}{\sigma^2} \leq e^{\zeta} \cdot \frac{4\|\Delta_q\|_2}{\sigma^2}, \qquad 1 \leq z \leq n, \tag{53}$$

where $\zeta = \max_{i,j,z} \zeta_{i,j,z}$ and $-\frac{2}{\sigma^2} < \zeta < 0$. As a result,

$$\|\mathbf{k}_i - \mathbf{k}_j\|_2 = \left(\sum_{z=1}^{n} |e^{t_{z,j}} - e^{t_{z,i}}|^2\right)^{\frac{1}{2}} \leq 4\sqrt{n}e^{\zeta} \cdot \frac{\|\Delta_q\|_2}{\sigma^2}. \tag{54}$$

Second, we consider the relation between $\|\Delta_q\|_2$ and $\|\widehat{\Delta}_q\|_F$, $1 \leq q \leq s$. It follows from (54) that

$$\begin{aligned}\|\widehat{\Delta}_q\|_F =& \|K_q - \widehat{\mathbf{c}}_q \mathbf{1}_{n_q}^T\|_F \\ =& \sqrt{\sum_{h=1}^{n_q} \|\mathbf{k}_{q,h} - \widehat{\mathbf{c}}_q\|_2^2} \\ \leq& \sqrt{\sum_{h=1}^{n_q} \frac{(\sum_{t=1}^{n_q} \|\mathbf{k}_{q,h} - \mathbf{k}_{q,t}\|_2)^2}{n_q^2}} \\ \leq& 4\sqrt{n \cdot n_q} e^{\zeta} \cdot \frac{\|\Delta_q\|_2}{\sigma^2},\end{aligned} \tag{55}$$

where $\mathbf{k}_{q,t}$, $t = 1, 2, \ldots, n_q$, are the t-th column of the matrix K_q.

Third, we focus on the relationship between $\|\widehat{\Delta}\|_F$ and $\|\Delta\|_F$. We have from (55) that

$$\|\widehat{\Delta}\|_F = \sqrt{\sum_{i=1}^{s} \|\widehat{\Delta}_i\|_F^2} \leq \sqrt{\sum_{i=1}^{s} \left(4\sqrt{n \cdot n_i} e^{\zeta} \cdot \frac{\|\Delta_i\|_2}{\sigma^2}\right)^2} \leq 4\sqrt{n \cdot \max_{1 \leq i \leq s} n_i} \frac{e^{\zeta}}{\sigma^2} \|\Delta\|_F. \tag{56}$$

Finally, it follows from (48) that $rank(\widehat{K}) \leq s$, and $\sigma_{s+1}(\widehat{K}) = 0$. Thus, we have from the perturbation theory of singular values (Golub & Van Loan, 2014), Corollary 8.6.2 and (56) that

$$\frac{\sigma_{s+1}(K)}{\sigma_1(K)} = \frac{|\sigma_{s+1}(K) - \sigma_{s+1}(\widehat{K})|}{\sigma_1(K)} \leq \frac{\|\widehat{\Delta}\|_F}{\|K\|_2} \leq 4\sqrt{n \cdot \max_{1 \leq i \leq s} n_i} \frac{e^{\zeta}}{\sigma^2} \frac{\|\Delta\|_F}{\|K\|_2},$$

which completes the proof. □

Remark 4 We show that the kernel matrix K has low numerical rank that depends on the number of clusters s. Let $\|\overline{\Delta}\|_2 = \frac{\sum_{i=1}^{s} \|\Delta_i\|_2}{s}$, which reflects the clustering effect of the original data X. Then Theorem 4.2 indicates that

$$\frac{\sigma_{s+1}(K)}{\sigma_1(K)} = \mathscr{O}\left(\frac{\|\overline{\Delta}\|_2}{\|K\|_2}\right). \tag{57}$$

In other words, if $\frac{\|\overline{\Delta}\|_2}{\|K\|_2}$ is sufficiently small, then the kernel matrix K is numerically low-rank, and the number of clusters s can be viewed as a numerical rank of K. This provides a target rank for solving the kernel regression problem, with applications to some randomized algorithms; see Sect. 4.2. Moreover, the proof also applies to other kernel functions such as the Laplacian kernel (Hofmann et al., 2008).

4.2 A modified Nyström method with fixed-rank

In this subsection, we consider how to solve the kernel regression problem (17) efficiently. As was mentioned in Remark 1, an explicit computation of the matrix $\widetilde{K}\widetilde{K}^T$ can be prohibitive, and an alternative is to use some low-rank approximations to $\widetilde{K}\widetilde{K}^T$ without forming it explicitly. We have from Sect. 4.1 that the kernel matrix K is numerically low-rank and the number of clusters s can be used as a numerical rank of K. Hence, s can also be used as

a target rank of the reduced kernel matrix $\widetilde{K} \in \mathbb{R}^{r\times n}$ which is a sub-matrix of the original kernel matrix $K \in \mathbb{R}^{n\times n}$. Thus, the idea is to choose s as the numerical rank of $\widetilde{K}\widetilde{K}^T$, and compute

$$\widetilde{A} = (\widetilde{K}\widetilde{K}^T)^\dagger \widetilde{K}T, \tag{58}$$

instead of (17) for the kernel regression step.

Given the target rank k, the *standard* Nyström method (Williams & Seeger, 2001; Drineas & Mahoney, 2005) constructs a rank-k approximation to an arbitrary symmetric positive semidefinite (SPSD) kernel matrix $H \in \mathbb{R}^{r\times r}$ by using only a few columns (or rows) of the matrix. More precisely, let l be the number of sampling, we denote by $C \in \mathbb{R}^{r\times l}(r > l > k)$ the matrix consists of l columns sampled from the kernel matrix H, and by $W \in \mathbb{R}^{l\times l}$ the intersection matrix formed by the intersection of these l columns and the corresponding l rows. The rank-l and rank-k Nyström approximation are

$$\widetilde{H}_l^{nys} = CW^\dagger C^T \ \text{ and } \ \widetilde{H}_k^{nys} = C[\![W]\!]_k^\dagger C^T, \tag{59}$$

respectively, where $[\![W]\!]_k$ represents the best rank-k approximation to W. Although this method can avoid accessing the entire kernel matrix, and thus greatly reduces the amount of calculation cost and storage requirement, it may suffer from losing of accuracy. Indeed, it was shown that no matter what sampling technique is employed, the incurred error in the Nyström approximation must grow with the matrix size r at least linearly (Wang & Zhang, 2013). As a result, the approximation obtained from the standard Nyström method may be unsatisfactory when r is large, unless a considerable number of columns are selected.

In Cortes et al. (2010), Cortes et al., pointed out that a tighter kernel approximation may lead to a better learning accuracy, so it is necessary to find kernel approximation models with better accuracies than the standard Nyström method. For instance, a modified Nyström method (Wang & Zhang, 2013; Sun et al., 2015) was proposed by borrowing the techniques in CUR matrix decomposition. With the selected columns $C \in \mathbb{R}^{r\times l}$ at hand, the rank-l *modified* Nyström approximation uses

$$\widetilde{H}^{mod} = C(C^\dagger H(C^\dagger)^T)C^T = CU^{mod}C^T \tag{60}$$

as an approximation to the kernel matrix H, where

$$U^{mod} = \underset{U\in\mathbb{R}^{l\times l}}{\operatorname{argmin}} \|H - CUC^T\|_F = C^\dagger H(C^\dagger)^T.$$

It is seen from (60) and (59) that the modified Nyström approximation $\widetilde{H}^{mod}$ is *no worse than* the standard rank-l Nyström approximation $\widetilde{H}_l^{nys}$.

Although Nyström method aims to compute a rank-k approximation, it is often preferred to choose $l > k$ landmark points and then restrict the resultant approximation to have rank at most k. Recently, a new alternative called the fixed-rank Nyström approximation was proposed (Anaraki & Becker, 2019), in which

$$\widetilde{H}^{opt} = [\![CW^\dagger C^T]\!]_k \tag{61}$$

is utilized as an approximation to H. Theoretical analysis and numerical experiments show that the fixed-rank Nyström approximation $\widetilde{H}^{opt}$ is superior to the standard rank-k Nyström method $\widetilde{H}_k^{nys}$ with respect to the nuclear norm (Anaraki & Becker, 2019).

Table 2 A comparison of computational complexities and memory requirements of three different Nyström methods

Algorithms	Computational complexities	Memory requirements
Fixed-rank Nyström	$\mathcal{O}(nrl + rl^2 + rlk + l^3)$	$\mathcal{O}(nr)$
Modified Nyström	$\mathcal{O}(nrk + nk^2 + rk^2)$	$\mathcal{O}(nr)$
Algorithm 5	$\mathcal{O}(nrl + nl^2 + rl^2)$	$\mathcal{O}(nr)$

Inspired by the fixed-rank Nyström method (Anaraki & Becker, 2019) and the modified Nyström method (Wang & Zhang, 2013), we knit the two methods together and propose a *modified Nyström method with fixed-rank*. More precisely, we first perform the economized QR decomposition $C = QR$, then the rank-l approximation (60) can be rewritten as

$$\widetilde{H}^{mod} = C(C^{\dagger}H(C^{\dagger})^T)C^T = CC^{\dagger}HCC^{\dagger} = QQ^THQQ^T. \tag{62}$$

Afterward, we make use of

$$\widetilde{H}^{mod}_{opt} = [\![QQ^THQQ^T]\!]_k, \tag{63}$$

i.e., the best rank-k approximation to QQ^THQQ^T, as approximation to the kernel matrix H. In summary, we present in Algorithm 5 our modified Nyström method with fixed-rank for the computation of the reconstruction weights matrix A arising in (58).

Algorithm 5 A modified Nyström method with fixed-rank for low-rank approximation of matrices

Input: The training labels $\mathbf{g}$, the discriminant space dimensionality d, the number of training samples n, the number of client class n_1, reduced kernel matrix $\widetilde{K} \in \mathbb{R}^{r\times n}$, a target rank k and an integer l, with $r > l > k$;

Output: The reconstruction weights matrix A.

1. Computing the target matrix T using Algorithm 3 or Algorithm 4;

2. Let $S \in \mathbb{R}^{r\times l}$ be a random matrix which has only one entry equals to one and the rest are zero in each column, and at most one nonzero element in each row. Computing the matrix $C = \widetilde{K} \cdot (\widetilde{K}^T S) \in \mathbb{R}^{r\times l}$, where the permutation matrix S is stored as a vector ;

3. Computing the economized QR decomposition of $C = QR$, where $Q \in \mathbb{R}^{r\times l}$ is orthonormal;

4. Let $W = Q^T\widetilde{K}$, and compute the singular value decomposition of the l-by-l matrix WW^T: $[V, D] = svd(WW^T)$;

5. Let $U = QV$, $U_k = U(:, 1:k)$, the first k columns of U, and let $D_k = D(1:k, 1:k)$, the k-by-k principle submatrix of D;

6. Let $A = U_k(D_k^{-1}(U_k^T(\widetilde{K}T)))$.

Remark 5 Compared with the fixed-rank Nyström approximation (61), for the same selected columns $C \in \mathbb{R}^{r\times l}$, the intersection matrix in our method reaches the solution of the optimization problem as in (60), and our approximation (63) is more accurate than the one obtained from the fixed-rank Nyström method. On the other hand, unlike many Nyström methods (Sun et al., 2015), an advantage of (63) is that it is free of computing the Moore–Penrose inverse $C^{\dagger}$. Finally, for clarity, we list the time and space complexities of

the fixed-rank Nyström method, the modified Nyström method, as well as Algorithm 5; see Table 2.

Next, we will establish a probabilistic error bound for our modified Nyström method with fixed-rank. We first need the following Lemma.

Lemma 4.3 *Tropp (2012) Given ℓ independent random $p \times p$ symmetric positive semidefinite (SPSD) matrix $G_1, G_2, \ldots, G_\ell$, with the property*

$$\lambda_1(G_i) \le \gamma, \qquad i = 1, 2, \ldots, \ell,$$

where $\lambda_1(G_i)$ is the largest eigenvalue of G_i and $\gamma > 0$ is a uniform upper bound of $\lambda_1(G_i), i = 1, 2, \ldots, \ell$. Defining $Y = \sum_{i=1}^{\ell} G_i$ and $\beta_{\min} = \lambda_{\min}(\mathbb{E}(Y))$, then for any $\theta \in (0, 1]$, the following probability inequality holds

$$\mathbb{P}\left\{\lambda_{\min}(Y) \le \theta\beta_{\min}\right\} \le p \cdot \left(\frac{e^{\theta-1}}{\theta^\theta}\right)^{\frac{\beta_{\min}}{\gamma}},$$

where $\mathbb{E}(Y)$ denotes expectation with respect to the random matrix Y.

Notice that $\widetilde{K}$ is an $r \times n$ matrix, let $\widetilde{K} = \widetilde{U}\widetilde{\Sigma}\widetilde{V}^T$ be the economized singular value decomposition of $\widetilde{K}$, where $\widetilde{U} \in \mathbb{R}^{r\times n}$, $\widetilde{\Sigma} \in \mathbb{R}^{n\times n}$ and $\widetilde{V} \in \mathbb{R}^{n\times n}$. Then $\widetilde{K}\widetilde{K}^T = \widetilde{U}\widetilde{\Sigma}^2\widetilde{U}^T$, and we rewrite the singular value decomposition as

$$\widetilde{K}\widetilde{K}^T = \widetilde{U}\widetilde{\Sigma}^2\widetilde{U}^T = \widetilde{U}\begin{pmatrix} \widetilde{\Sigma}_1^2 & \mathbf{0} \\ \mathbf{0} & \widetilde{\Sigma}_2^2 \end{pmatrix}\begin{pmatrix} \widetilde{U}_1^T \\ \widetilde{U}_2^T \end{pmatrix}, \tag{64}$$

where $\widetilde{\Sigma}_1^2 \in \mathbb{R}^{k\times k}$ and $\widetilde{U}_1 \in \mathbb{R}^{r\times k}$. Let $S \in \mathbb{R}^{r\times l}$ be a random matrix that has only one entry equals to one and the rest are zero in each column, and at most one nonzero element in each row. Denote by

$$S_1 = \widetilde{U}_1^T S \in \mathbb{R}^{k\times l} \quad \text{and} \quad S_2 = \widetilde{U}_2^T S \in \mathbb{R}^{(n-k)\times l}. \tag{65}$$

As $k < l$, one can assume that S_1 is of full row rank. Now we are ready to present the following theorem for the probabilistic error bound on the low-rank approximation from Algorithm 5.

Theorem 4.4 *Let $\sigma_1 \ge \sigma_2 \ge \sigma_3 \ge \cdots \ge \sigma_r \ge 0$ be the singular values of $\widetilde{K} \in \mathbb{R}^{r\times n}(r < n)$, and let $U_k D_k U_k^T$ be the low-rank approximation from Algorithm 5. If S_1 is of full row rank, then we have that*

$$\frac{\|\widetilde{K}\widetilde{K}^T - U_k D_k U_k^T\|_2}{\|\widetilde{K}\widetilde{K}^T\|_2} \le 2\left(\frac{3}{2} + \frac{1}{\delta\sqrt{\theta}} \cdot \left(1 + \sqrt{\frac{r-1}{r}} + \sqrt{\frac{(r-1)(n-k)}{rl}}\right)\right)\frac{\sigma_{k+1}^2(\widetilde{K})}{\sigma_1^2(\widetilde{K})}$$

with probability at least $1 - 2\delta$, where

$$\delta = k\left(\frac{e^\theta - 1}{\theta^\theta}\right)^{\frac{l}{k\mu_0}}, \quad 0 < \theta \le 1,$$

and

$$\mu_0 = \frac{r}{k} \max_{1 \le i \le r} \|(\widetilde{U}_1^T)_i\|_2^2 = \frac{r}{k} \max_{1 \le i \le r} (\widetilde{U}_1 \widetilde{U}_1^T)_{ii}$$

is the matrix coherence of $\widetilde{U}_1$*, with* $(\widetilde{U}_1^T)_i$ *and* $(\widetilde{U}_1 \widetilde{U}_1^T)_{ii}$ *being the i-th column of* $\widetilde{U}_1^T$ *and the i-th diagonal element of* $\widetilde{U}_1 \widetilde{U}_1^T$*, respectively.*

Proof We have from Algorithm 5 that

$$\|\widetilde{K}\widetilde{K}^T - U_k D_k U_k^T\|_2 = \|\widetilde{K}\widetilde{K}^T - [\![UDU^T]\!]_k\|_2,$$

where $[\![UDU^T]\!]_k$ denotes the best rank-k approximation of the matrix UDU^T. By using the notations in Algorithm 5, we have

$$[\![UDU^T]\!]_k = [\![QVDV^TQ^T]\!]_k = [\![QWW^TQ^T]\!]_k = [\![QQ^T\widetilde{K}\widetilde{K}^TQQ^T]\!]_k,$$

and

$$\|\widetilde{K}\widetilde{K}^T - U_k D_k U_k^T\|_2 = \|\widetilde{K}\widetilde{K}^T - [\![QQ^T\widetilde{K}\widetilde{K}^TQQ^T]\!]_k\|_2. \tag{66}$$

First, we analyze the probabilistic error bound on (66). Notice that

$$\begin{aligned} \|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^TQQ^T\|_2 \le & \|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2 + \|QQ^T(\widetilde{K}\widetilde{K}^T - \widetilde{K}\widetilde{K}^TQQ^T)\|_2 \\ \le & 2\|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2. \end{aligned} \tag{67}$$

Based on (67) and the singular value interlacing theorem (Golub & Van Loan, 2014, p. 443), we have from (66) that

$$\begin{aligned} & \|\widetilde{K}\widetilde{K}^T - [\![QQ^T\widetilde{K}\widetilde{K}^TQQ^T]\!]_k\|_2 \\ & \le \|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^TQQ^T\|_2 + \|QQ^T\widetilde{K}\widetilde{K}^TQQ^T - [\![QQ^T\widetilde{K}\widetilde{K}^TQQ^T]\!]_k\|_2 \\ & \le 2\|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2 + \sigma_{k+1}(QQ^T\widetilde{K}\widetilde{K}^TQQ^T) \\ & \le 2\|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2 + \sigma_{k+1}(\widetilde{K}\widetilde{K}^T), \end{aligned} \tag{68}$$

where $\sigma_{k+1}(QQ^T\widetilde{K}\widetilde{K}^TQQ^T)$ and $\sigma_{k+1}(\widetilde{K}\widetilde{K}^T)$ are the $(k+1)$-th largest singular value of the matrices $QQ^T\widetilde{K}\widetilde{K}^TQQ^T$ and $\widetilde{K}\widetilde{K}^T$, respectively.

Second, we consider the term $\|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2$. As S_1 is of full row rank, we obtain from (Halko et al., 2011), Theorem 9.1) that

$$\|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2^2 \le \|\widetilde{\Sigma}_2^2\|_2^2 + \|\widetilde{\Sigma}_2^2 S_2 S_1^\dagger\|_2^2, \tag{69}$$

and thus

$$\begin{aligned} & \|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2 \\ & \le (\|\widetilde{\Sigma}_2^2\|_2^2 + \|\widetilde{\Sigma}_2^2 S_2 S_1^\dagger\|_2^2)^{\frac{1}{2}} \\ & \le \|\widetilde{\Sigma}_2^2\|_2 + \|\widetilde{\Sigma}_2^2 S_2 S_1^\dagger\|_2 \\ & \le \|\widetilde{\Sigma}_2^2\|_2 + \|\widetilde{\Sigma}_2^2 S_2\|_2 \cdot \|S_1^\dagger\|_2. \end{aligned} \tag{70}$$

Third, we consider the upper bound of $\|S_1^\dagger\|_2$, whose proof is along the line of (Gittens, 2011), Lemma 1. Notice that

$$\|S_1^{\dagger}\|_2^2 = \|(\widetilde{U}_1^T S)^{\dagger}\|_2^2 = \frac{1}{\sigma_k^2(\widetilde{U}_1^T S)} = \frac{1}{\lambda_k(\widetilde{U}_1^T S S^T \widetilde{U}_1)}, \tag{71}$$

where $\lambda_k(\widetilde{U}_1^T S S^T \widetilde{U}_1)$ is the k-th largest eigenvalue of $\widetilde{U}_1^T S S^T \widetilde{U}_1$. Denote by $(\widetilde{U}_1^T)_i$ the i-th column of $\widetilde{U}_1^T$, then we have that $\widetilde{U}_1^T \widetilde{U}_1 = \sum_{i=1}^r (\widetilde{U}_1^T)_i \cdot [(\widetilde{U}_1^T)_i]^T$. Thanks to the property of S, let $G_i \in \mathbb{R}^{k\times k}, i = 1, 2, \ldots, l$, be matrices chosen randomly from the set $\{(\widetilde{U}_1^T)_i \cdot [(\widetilde{U}_1^T)_i]^T\}_{i=1}^r$, then we have from (71) that

$$\|S_1^{\dagger}\|_2^2 = \frac{1}{\lambda_k(\widetilde{U}_1^T S S^T \widetilde{U}_1)} = \frac{1}{\lambda_k(\sum_{i=1}^l G_i)}. \tag{72}$$

Define $\gamma = \max\limits_{1\le i\le l} \lambda_1(G_i)$, then

$$\gamma = \max_{1\le i\le l} \lambda_1(G_i) = \max_{1\le i\le r} \|(\widetilde{U}_1^T)_i\|_2^2 = \frac{k}{r}\mu_0,$$

where $\mu_0 = \frac{r}{k} \max\limits_{1\le i\le r} \|(\widetilde{U}_1^T)_i\|_2^2 = \frac{r}{k} \max\limits_{1\le i\le r} (\widetilde{U}_1 \widetilde{U}_1^T)_{ii}$ is the matrix coherence of $\widetilde{U}_1$ (Gittens, 2011), and $(\widetilde{U}_1 \widetilde{U}_1^T)_{ii}$ stands for the i-th diagonal element of $\widetilde{U}_1 \widetilde{U}_1^T$.

Denote by $\beta_{\min} = \lambda_{\min}(\mathbb{E}(\sum_{i=1}^l G_i))$, then

$$\beta_{\min} = \lambda_{\min}\Big(\mathbb{E}\big(\sum_{i=1}^{l} G_i\big)\Big) = l\lambda_{\min}\Big(\frac{1}{r}\widetilde{U}_1^T \widetilde{U}_1\Big) = \frac{l}{r},$$

where we use the orthogonality of the matrix $\widetilde{U}_1$. From Lemma 4.3, we obtain

$$\mathbb{P}\left\{\lambda_{\min}(\sum_{i=1}^{l} G_i) \le \frac{\theta l}{r}\right\} \le k\left(\frac{e^{\theta-1}}{\theta^{\theta}}\right)^{\frac{l}{k\mu_0}}, \tag{73}$$

where $\theta \in (0, 1]$. Thus, a combination of (72) and (73) yields

$$\mathbb{P}\left\{\|S_1^{\dagger}\|_2 \ge \sqrt{\frac{r}{\theta l}}\right\} \le \delta, \tag{74}$$

where $\delta = k\Big(\frac{e^{\theta-1}}{\theta^{\theta}}\Big)^{\frac{l}{k\mu_0}}$.

Fourth, we establish a probabilistic bound on $\|\widetilde{\Sigma}_2^2 S_2\|_2$ in (70). Recall that S is a Gaussian matrix whose entries are independent normal variables with mean μ and variance ζ^2, i.e., $S \sim N(\mu, \zeta^2)$. Denote by $\Omega = \frac{S-\mu \mathbf{1}_r \mathbf{1}_l^T}{\zeta}$, where $\mathbf{1}_r$ is the vector of all ones with dimension r, then Ω is a standard Gaussian matrix and $S = \zeta\Omega + \mu \mathbf{1}_r \mathbf{1}_l^T$. It follows from (65) that

$$\begin{aligned}
\|\widetilde{\Sigma}_2^2 S_2\|_2 =& \|\widetilde{\Sigma}_2^2 \widetilde{U}_2^T (\zeta\Omega + \mu \mathbf{1}_r \mathbf{1}_l^T)\|_2 \\
\le& \zeta \|\widetilde{\Sigma}_2^2 \widetilde{U}_2^T \Omega\|_2 + \mu \|\widetilde{\Sigma}_2^2 \widetilde{U}_2^T \mathbf{1}_r \mathbf{1}_l^T\|_2 \\
\le& \Big(\zeta \|\widetilde{U}_2^T \Omega\|_2 + \mu \|\widetilde{U}_2^T \mathbf{1}_r \mathbf{1}_l^T\|_2\Big) \|\widetilde{\Sigma}_2^2\|_2 \\
\le& \Big(\zeta \|\widetilde{U}_2^T \Omega\|_2 + \mu\sqrt{rl}\Big) \|\widetilde{\Sigma}_2^2\|_2,
\end{aligned} \tag{75}$$

where we use $\|\mathbf{1}_r \mathbf{1}_l^T\|_2 = \sqrt{rl}$ in the last inequality. Taking expectation with respect to (75) gives

$$\mathbb{E}(\|\widetilde{\Sigma}_2^2 S_2\|_2) \le \|\widetilde{\Sigma}_2^2\|_2 \cdot \left(\zeta \mathbb{E}(\|\widetilde{U}_2^T \Omega\|_2) + \mu\sqrt{rl}\right). \tag{76}$$

Notice that $\widetilde{U}_2$ is a column orthogonal matrix and the distribution of the standard Gaussian matrix Ω is rotationally invariant, and $\widetilde{U}_2^T \Omega \in \mathbb{R}^{(n-k)\times l}$ is also a standard Gaussian matrix. So we have (Halko et al., 2011)

$$\mathbb{E}(\|\widetilde{U}_2^T \Omega\|_2) \le \sqrt{l} + \sqrt{n-k}. \tag{77}$$

Thus, a combination of (76) and (77) gives

$$\mathbb{E}(\|\widetilde{\Sigma}_2^2 S_2\|_2) \le \|\widetilde{\Sigma}_2^2\|_2 \cdot \left(\zeta(\sqrt{l} + \sqrt{n-k}) + \mu\sqrt{rl}\right).$$

In light of the Markov's inequality (Grimmett & Stirzaker, 2001), we get

$$\mathbb{P}\left\{ \|\widetilde{\Sigma}_2^2 S_2\|_2 \ge \frac{\zeta(\sqrt{l} + \sqrt{n-k}) + \mu\sqrt{rl}}{\delta} \|\widetilde{\Sigma}_2^2\|_2 \right\} \le \frac{\mathbb{E}(\|\widetilde{\Sigma}_2^2 S_2\|_2)}{\frac{\zeta(\sqrt{l}+\sqrt{n-k})+\mu\sqrt{rl}}{\delta} \|\widetilde{\Sigma}_2^2\|_2} \le \delta. \tag{78}$$

Combining (74) and (78), and applying the union bound, we have

$$\mathbb{P}\left\{ \|S_1^\dagger\|_2 \ge \sqrt{\frac{r}{\theta l}} \text{ or } \|\widetilde{\Sigma}_2^2 S_2\|_2 \ge \frac{\zeta(\sqrt{l} + \sqrt{n-k}) + \mu\sqrt{rl}}{\delta} \|\widetilde{\Sigma}_2^2\|_2 \right\} \le 2\delta. \tag{79}$$

Hence, we have the following probabilistic error bound for $\|\widetilde{\Sigma}_2^2 S_2\|_2 \cdot \|S_1^\dagger\|_2$ in (70), i.e.,

$$\|\widetilde{\Sigma}_2^2 S_2\|_2 \cdot \|S_1^\dagger\|_2 \le \sqrt{\frac{r}{\theta l}} \cdot \frac{\zeta(\sqrt{l} + \sqrt{n-k}) + \mu\sqrt{rl}}{\delta} \|\widetilde{\Sigma}_2^2\|_2, \tag{80}$$

with probability at least $1 - 2\delta$, where

$$\delta = k\left(\frac{e^\theta - 1}{\theta^\theta}\right)^{\frac{l}{k\mu_0}}, \qquad 0 < \theta \le 1,$$

and

$$\mu_0 = \frac{r}{k} \max_{1\le i\le r} \|(\widetilde{U}_1^T)_i\|_2^2 = \frac{r}{k} \max_{1\le i\le r} (\widetilde{U}_1 \widetilde{U}_1^T)_{ii}$$

is the matrix coherence of $\widetilde{U}_1$.

Finally, based on (68), (70) and (80), a probabilistic error bound on the low-rank approximation $[\![QQ^T\widetilde{K}\widetilde{K}^T QQ^T]\!]_k$ to $\widetilde{K}\widetilde{K}^T$ is given as follows.

$$\begin{aligned}
&\|\widetilde{K}\widetilde{K}^T - [\![QQ^T\widetilde{K}\widetilde{K}^T QQ^T]\!]_k\|_2 \\
&\quad \le 2\|\widetilde{K}\widetilde{K}^T - QQ^T\widetilde{K}\widetilde{K}^T\|_2 + \sigma_{k+1}(\widetilde{K}\widetilde{K}^T) \\
&\quad \le 2\left(\|\widetilde{\Sigma}_2^2\|_2 + \|\widetilde{\Sigma}_2^2 S_2\|_2 \cdot \|S_1^\dagger\|_2\right) + \sigma_{k+1}(\widetilde{K}\widetilde{K}^T) \\
&\quad \le 2\left(1 + \sqrt{\frac{r}{\theta l}} \cdot \frac{\zeta(\sqrt{l} + \sqrt{n-k}) + \mu\sqrt{rl}}{\delta}\right)\|\widetilde{\Sigma}_2^2\|_2 + \sigma_{k+1}(\widetilde{K}\widetilde{K}^T)
\end{aligned}$$

holds with probability at least $1-2\delta$. Recall that $S \in \mathbb{R}^{r\times l}$ is a Gaussian distribution matrix with mean μ and variance ζ^2, and it is easy to check that $\mu = \frac{1}{r}$ and $\zeta^2 = \frac{r-1}{r^2}$. So we have

$$\begin{aligned}
&\|\widetilde{K}\widetilde{K}^T - [\![QQ^T\widetilde{K}\widetilde{K}^TQQ^T]\!]_k\|_2 \\
&\leq 2\left(1+\sqrt{\frac{r}{\theta l}}\cdot\frac{1}{\delta}\cdot\frac{\sqrt{r-1}(\sqrt{l}+\sqrt{n-k})+\sqrt{rl}}{r}\right)\sigma_{k+1}(\widetilde{K}\widetilde{K}^T)+\sigma_{k+1}(\widetilde{K}\widetilde{K}^T) \\
&= 2\left(\frac{3}{2}+\frac{1}{\delta\sqrt{\theta}}\cdot\frac{\sqrt{r-1}\left(1+\sqrt{\frac{n-k}{l}}\right)+\sqrt{r}}{\sqrt{r}}\right)\sigma_{k+1}(\widetilde{K}\widetilde{K}^T) \\
&= 2\left(\frac{3}{2}+\frac{1}{\delta\sqrt{\theta}}\cdot\left(1+\sqrt{\frac{r-1}{r}}+\sqrt{\frac{(r-1)(n-k)}{rl}}\right)\right)\sigma_{k+1}^2(\widetilde{K}),
\end{aligned} \tag{81}$$

holds with probability at least $1-2\delta$. In summary, we have from (66) and (81) that

$$\frac{\|\widetilde{K}\widetilde{K}^T - U_kD_kU_k^T\|_2}{\|\widetilde{K}\widetilde{K}^T\|_2} \leq 2\left(\frac{3}{2}+\frac{1}{\delta\sqrt{\theta}}\cdot\left(1+\sqrt{\frac{r-1}{r}}+\sqrt{\frac{(r-1)(n-k)}{rl}}\right)\right)\frac{\sigma_{k+1}^2(\widetilde{K})}{\sigma_1^2(\widetilde{K})},$$

with probability at least $1-2\delta$. □

Remark 6 Theorem 4.4 gives a relative error bound on approximating the matrix $\widetilde{K}\widetilde{K}^T$. As was mentioned in Remark 4, if the clustering result of the original data X is ideal, then the reduced kernel matrix $\widetilde{K}$ will have low numerical rank that is the number of clusters s. In other words, one can choose s as the target rank in randomized algorithms to solve the kernel regression step. Thus, from the probabilistic error bound established in Theorem 4.4, if we adopt the number of clusters s as the target rank k, then $\sigma_{k+1}^2(\widetilde{K})/\sigma_1^2(\widetilde{K})$ is sufficiently small, and our proposed algorithm will be effective.

5 A randomized block Kaczmarz method for kernel regression problem with multiple right-hand sides

In Sect. 4, we propose a low-rank approximation to $\widetilde{K}\widetilde{K}^T$ for solving (17). Although the proposed Nyström method is not required to form the matrix $\widetilde{K}\widetilde{K}^T$ explicitly, this method needs to form and store the reduced kernel matrix $\widetilde{K}$ explicitly. In the era of big data, the reduced kernel matrix may be so huge that it can not be stored in main memory. To deal with this problem, in this section, we propose a randomized block Kaczmarz method for kernel regression problem. For notation simplicity, we write (17) as

$$X^* = \underset{X\in\mathbb{R}^{r\times d}}{\operatorname{argmin}} \|BX-T\|_F^2, \quad \text{where} \quad B \equiv \widetilde{K}^T \in \mathbb{R}^{n\times r}. \tag{82}$$

As $T \in \mathbb{R}^{n\times d}$ with $d > 1$, we call it a kernel regression problem with *multiple right-hand sides*.

The randomized Kaczmarz method is a popular solver for large-scale and dense linear systems (Strohmer & Vershynin, 2009). An advantage of this type of method is that there is no need to access the full data matrix into main memory, and only a small portion of the data matrix is utilized to update the solution in each iteration. In Zouzias and Freris (2013), Zouzias and Freris introduced a randomized extended Kaczmarz (REK) method for solving the least squares problem $\min_{\mathbf{x}\in\mathbb{R}^r} \|B\mathbf{x}-\mathbf{t}\|_2^2$. In essence, it is a specific combination of the randomized orthogonal projection method together with the randomized Kaczmarz method. It was shown that, the solution of the randomized extended Kaczmarz method approaches the l_2 norm least squares solution up to an additive error that depends on the distance between the right-hand side vector $\mathbf{t}$ and the column space of the matrix B (Zouzias & Freris, 2013). More precisely, the randomized extended Kaczmarz method exploits the randomized orthogonal projection method to efficiently reduce the norm of the "noisy" part $\mathbf{t}_{\mathscr{R}(B)^\perp}$ of $\mathbf{t}$, where $\mathbf{t}_{\mathscr{R}(B)^\perp} = (I_n - BB^\dagger)\mathbf{t}$. As the least squares solution is

$$\mathbf{x}_{LS} = B^\dagger\mathbf{t} = B^\dagger BB^\dagger\mathbf{t} \equiv B^\dagger\mathbf{t}_{\mathscr{R}(B)},$$

the randomized Kaczmarz method is applied to a new linear system whose right-hand side is $\mathbf{t}_{\mathscr{R}(B)} = BB^\dagger\mathbf{t}$.

The block Kaczmarz method is generally considered to be more efficient than the classical Kaczmarz method, because of subtle computational issues involved in data transfer and basic linear algebra subroutines (Necoara, 2019; Needell & Tropp, 2014; Needell et al., 2015). In Needell et al. (2015), Needell et al., put forward a randomized double block Kaczmarz method that is an extension to the block Kaczmarz method and the randomized extended Kaczmarz (REK) method. The randomized double block Kaczmarz method exploits column partition for the projection step and row partition for the Kaczmarz step. Consequently, the computational cost will be very high for large-scale data. As an alternative, a randomized block coordinate descent (RBCD) method was also proposed in Needell et al. (2015), which utilizes only column paving for the projection step and Kaczmarz step for the resulting linear system. To reduce the amount of calculation, this method acquires a suitable partition to the columns based on random selections; for more details, refer to Needell et al. (2015).

Algorithm 6 A randomized block Kaczmarz method for kernel regression problem with multiple right-hand sides

Input: The training labels $\mathbf{g}$, the reference vector set cardinality r, the discriminant space dimensionality d, the number of training samples n, the number of client class n_1, the convergence threshold tol, the maximal iteration number $iter$, and the block cardinality p;
Output: The approximate reconstruction weights matrix $X \in \mathbb{R}^{r\times d}$.

1. Computing the target matrix T using Algorithm 3 or Algorithm 4;
2. Initialize $X^{(0)} = zeros(r,d)$, $X^{(1)} = X^{(0)}$, $Z_0 = T$, $\ell = 1$, and $\mathrm{err} = 1$;
3. **while** $\mathrm{err} > tol$ and $\ell < iter$ **do**
4. Computing a random partition $\widetilde{\mathscr{T}} = \{\widetilde{\tau}_1, \widetilde{\tau}_2, \ldots, \widetilde{\tau}_p\}$ with p blocks of equal size $\frac{r}{p}$ on the column index set $[r]$, where $[r] = \{1, 2, \ldots, r\}$;
5. For each index set $\widetilde{\tau}_i, i = 1, 2, \ldots, p$, selecting the desired index set τ_i from $\widetilde{\tau}_i$, which is corresponding to the first $\lceil \frac{|\widetilde{\tau}_i|}{2} \rceil = \lceil \frac{r}{2p} \rceil$ columns with the largest column norm of $B(:, \widetilde{\tau}_i)$, where $\lceil \frac{r}{2p} \rceil$ stands for the closet integer with respect to $\frac{r}{2p}$. Here $B(:, \widetilde{\tau}_i)$ consists of the columns of B corresponding to the index set $\widetilde{\tau}_i$;
6. Denote by $\mathscr{T} = \{\tau_1, \tau_2, \ldots, \tau_p\}$, and choose an index set $\tau_\ell \in \mathscr{T}$ uniformly at random with the cardinality $|\tau_\ell| = \lceil \frac{r}{2p} \rceil$;
7. Computing $W_\ell = B(:, \tau_\ell)^\dagger Z_{\ell-1}$, where $B(:, \tau_\ell)$ consists of the columns of B corresponding to the index set τ_ℓ;
8. Updating the Kaczmarz step: $X^{(\ell)}(\tau_\ell, :) = X^{(\ell-1)}(\tau_\ell, :) + W_\ell$, where $X^{(\ell)}(\tau_\ell, :)$ consists of the rows of $X^{(\ell)}$ corresponding to the index set τ_ℓ;
9. Updating the projection step: $Z_\ell = Z_{\ell-1} - B(:, \tau_\ell) W_\ell$;
10. Checking the value of $\mathrm{err} = \frac{\|W_\ell\|_F}{\|T\|_F}$ for convergence;
11. **end**;
12. Let $X = X^{(\ell)}$.

To the best of our knowledge, however, Kaczmarz-type methods have not yet been used to solve linear systems and least squares problems *with multiple right-hand sides*. Hence, on the basis of the randomized block coordinate descent method proposed in Needell et al. (2015), we present in Algorithm 7 a randomized block Kaczmarz method for regression problem with multiple right-hand sides.

Remark 7 Some remarks are given to Algorithm 7. First, compared with some existing methods for the optimization problem (17), an advantage of Algorithm 7 is that there is no need to explicitly form and store all the elements of the reduced kernel matrix $\widetilde{K}$. Second, the randomized block Kaczmarz methods proposed in Necoara (2019), Needell and Tropp (2014), Needell et al., (2015) are just for solving least squares problems or linear systems with only one right-hand side. As a comparison, Algorithm 7 can solve the d problems once for all, where d is the discriminant space dimensionality. So the proposed method can accelerate some existing randomized block Kaczmarz methods significantly, especially when d is large. Third, we propose a new sampling scheme which is different from the standard block Kaczmarz method, see Step 5 in Algorithm 7. More precisely, we first perform a random partition to the column index set with p blocks, and select one block arbitrarily from the partition. Then, we choose half of the columns corresponding to the largest norm from the selected block.

The stopping criteria utilized in the existing randomized block Kaczmarz methods often relies on the full coefficient matrix B more or less (Necoara, 2019; Needell & Tropp, 2014; Needell et al., 2015), which contradicts the purpose of not storing the entire coefficient matrix in main memory. To circumvent this difficulty, in Algorithm 7, we propose a practical stopping criterion (83) in which there is no need to access the full coefficient matrix. Indeed, compared with the $(\ell-1)$-th iterative solution $X^{(\ell-1)}$, the ℓ-th iterative solution $X^{(\ell)}$ only updates those rows corresponding to the index set $\boldsymbol{\tau}_\ell$, while the other rows remain unchanged. More precisely, in Step 10 of Algorithm 7, we make use of

$$\text{err} = \frac{\|X^{(\ell)} - X^{(\ell-1)}\|_F}{\|T\|_F} = \frac{\|X^{(\ell)}(\boldsymbol{\tau}_\ell, :) - X^{(\ell-1)}(\boldsymbol{\tau}_\ell, :)\|_F}{\|T\|_F} = \frac{\|W_\ell\|_F}{\|T\|_F} \tag{83}$$

as a stopping criterion, and there is no need to access the entire coefficient matrix B. Now we show the rationality of this scheme. Denote by $X_{LS} = B^\dagger T$ the least square solution with least F-norm of the optimization problem (82). It is easy to see that

$$\frac{\left|\|X^{(\ell)} - X_{LS}\|_F - \|X^{(\ell-1)} - X_{LS}\|_F\right|}{\|T\|_F} \le \frac{\|X^{(\ell)} - X^{(\ell-1)}\|_F}{\|T\|_F} \le \frac{\|X^{(\ell)} - X_{LS}\|_F + \|X^{(\ell-1)} - X_{LS}\|_F}{\|T\|_F}.$$

Thus, if the iterative sequence $\{X^{(\ell)}\}_{\ell=0}^{\infty}$ converges to X_{LS}, then $\|X^{(\ell)} - X^{(\ell-1)}\|_F / \|T\|_F$ converges to 0, and (83) can be used as a stopping criterion for solving (82).

Denote by $B(:, \widetilde{\boldsymbol{\tau}}_\ell) = B_{\widetilde{\boldsymbol{\tau}}_\ell}$, in Algorithm 7, we notice that the conditioning of the blocks $B_{\widetilde{\boldsymbol{\tau}}_\ell}$ plays a crucial role in the behavior of the block Kaczmarz methods. In Needell and Tropp (2014), Needell et al. (2015), Needell and Tropp give a definition of a "paving" for the matrix B.

Definition 5.1 Needell and Tropp (2014) A column paving (p, α, β) of an $n \times r$ matrix B is a partition $\widetilde{\mathcal{T}} = \{\widetilde{\boldsymbol{\tau}}_1, \widetilde{\boldsymbol{\tau}}_2, \ldots, \widetilde{\boldsymbol{\tau}}_p\}$ of the column indices such that

$$\alpha \le \lambda_{\min}(B_{\widetilde{\boldsymbol{\tau}}_i} B_{\widetilde{\boldsymbol{\tau}}_i}^T) \quad \text{and} \quad \lambda_{\max}(B_{\widetilde{\boldsymbol{\tau}}_i} B_{\widetilde{\boldsymbol{\tau}}_i}^T) \le \beta, \quad i = 1, 2, \ldots, p,$$

where $B_{\widetilde{\boldsymbol{\tau}}_i} = B(:, \widetilde{\boldsymbol{\tau}}_i)$ is composed of the columns in matrix B corresponding to the index $\widetilde{\boldsymbol{\tau}}_i$.

Inspired by Needell et al. (2015), Theorem 7), we give the following theoretical analysis on the convergence of the iterative sequence $\{X^{(\ell)}\}_{\ell=0}^{\infty}$ generated by Algorithm 7.

Theorem 5.2 *Denote by $\{X^{(\ell)}\}_{\ell=0}^{\infty}$ the iterative sequence generated by Algorithm 7, and by (p, α, β) a column paving of B. Assume that B is of full column rank, then there exists a scalar $\delta \in (0, 1)$ such that*

$$\mathbb{E}\left(\frac{\|X_{LS} - X^{(\ell)}\|_F^2}{\|X_{LS}\|_F^2}\right) \le \left(1 - \frac{\delta \cdot (\sigma_{\min}^{nz}(B))^2}{p\beta}\right)^{\ell} \kappa^2(B),$$

where $\sigma_{\min}^{nz}(B)$ is the smallest non-zero singular value of B, and $\kappa(B) = \frac{\sigma_{\max}(B)}{\sigma_{\min}^{nz}(B)}$ is the 2-norm condition number of B.

Proof We note that

$$B(X_{LS} - X^{(\ell)}) = BB^\dagger T - BX^{(\ell)} = T - BX^{(\ell)} - (I_n - BB^\dagger)T = T - BX^{(\ell)} - T_{\mathscr{R}(B)^\perp}, \tag{84}$$

where $X_{LS} = B^{\dagger}T$ and $T_{\mathscr{R}(B)^{\perp}} = (I_n - BB^{\dagger})T$.

First, we prove that $Z_{\ell} = T - BX^{(\ell)}$ by induction on ℓ. For $\ell = 0$, we have from Algorithm 7 that $Z_0 = T - BX^{(0)}$. Denote by $B(:, \boldsymbol{\tau}_{\ell}) = B_{\tau_{\ell}}$, we note that $X^{(1)}(\boldsymbol{\tau}_1, :) = W_1$ and $X^{(1)}([r] \setminus \boldsymbol{\tau}_1, :) = \mathbf{0}$, where $[r]$ is the set $\{1, 2, \ldots, r\}$, "\" is the set subtraction operation, and $[r] \setminus \boldsymbol{\tau}_1$ is the set whose elements belong to $[r]$ but not to $\boldsymbol{\tau}_1$. Thus, we have $Z_1 = Z_0 - B_{\boldsymbol{\tau}_1} W_1 = T - BX^{(1)}$. Now we assume that $Z_{\ell-1} = T - BX^{(\ell-1)}$ with $\ell \geq 1$, then

$$Z_{\ell} = Z_{\ell-1} - B_{\tau_{\ell}} W_{\ell} = T - BX^{(\ell-1)} - B_{\tau_{\ell}} W_{\ell}. \tag{85}$$

It follows from Algorithm 7 that $BX^{(\ell)} = BX^{(\ell-1)} + B_{\tau_{\ell}} W_{\ell}$. For the sake of simplicity, we denote by $X^{(\ell)}(\tau_{\ell}, :) = X^{(\ell)}_{\tau_{\ell}}$. From $X^{(\ell)}_{\tau_{\ell}} = X^{(\ell-1)}_{\tau_{\ell}} + W_{\ell}$ and $X^{(\ell)}_{[r]\setminus\tau_{\ell}} = X^{(\ell-1)}_{[r]\setminus\tau_{\ell}}$, we obtain

$$\begin{aligned} BX^{(\ell)} =& B_{\tau_{\ell}} X^{(\ell)}_{\tau_{\ell}} + B_{[r]\setminus\tau_{\ell}} X^{(\ell)}_{[r]\setminus\tau_{\ell}} \\ =& B_{\tau_{\ell}} X^{(\ell-1)}_{\tau_{\ell}} + B_{\tau_{\ell}} W_{\ell} + B_{[r]\setminus\tau_{\ell}} X^{(\ell)}_{[r]\setminus\tau_{\ell}} \\ =& B_{\tau_{\ell}} X^{(\ell-1)}_{\tau_{\ell}} + B_{\tau_{\ell}} W_{\ell} + B_{[r]\setminus\tau_{\ell}} X^{(\ell-1)}_{[r]\setminus\tau_{\ell}} \\ =& BX^{(\ell-1)} + B_{\tau_{\ell}} W_{\ell}. \end{aligned} \tag{86}$$

Combining (85) and (86), we arrive at

$$Z_{\ell} = T - (BX^{(\ell)} - B_{\tau_{\ell}} W_{\ell}) - B_{\tau_{\ell}} W_{\ell} = T - BX^{(\ell)}. \tag{87}$$

As a result, it follows from (84) and (87) that

$$B(X_{LS} - X^{(\ell)}) = Z_{\ell} - T_{\mathscr{R}(B)^{\perp}}. \tag{88}$$

Second, let $F_{\ell} = Z_{\ell} - T_{\mathscr{R}(B)^{\perp}}$, we take conditional expectation on F_{ℓ} over τ_{ℓ}, then

$$\begin{aligned} \mathbb{E}\|F_{\ell}\|_F^2 =& \mathbb{E}\|Z_{\ell} - T_{\mathscr{R}(B)^{\perp}}\|_F^2 \\ =& \mathbb{E}\|Z_{\ell-1} - B_{\tau_{\ell}} W_{\ell} - T_{\mathscr{R}(B)^{\perp}}\|_F^2 \\ =& \mathbb{E}\|Z_{\ell-1} - B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}} Z_{\ell-1} - (I_n - B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}}) T_{\mathscr{R}(B)^{\perp}}\|_F^2 \\ =& \mathbb{E}\|(I_n - B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}})(Z_{\ell-1} - T_{\mathscr{R}(B)^{\perp}})\|_F^2 \\ =& \mathbb{E}\|(I_n - B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}}) F_{\ell-1}\|_F^2, \end{aligned} \tag{89}$$

where the second equality follows from the definition of Z_{ℓ}, and the third one is from the definition of W_{ℓ} and the fact that the two subspaces $\mathrm{span}\{B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}}\}$ and $\mathrm{span}\{T_{\mathscr{R}(B)^{\perp}}\}$ are orthogonal.

Notice that

$$\|(I_n - B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}}) F_{\ell-1}\|_F^2 = \|F_{\ell-1}\|_F^2 - \|B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}} F_{\ell-1}\|_F^2. \tag{90}$$

Next we consider the lower bound on $\mathbb{E}\|B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}} F_{\ell-1}\|_F^2$. Let $B_{\tau_{\ell}} = U_{\tau_{\ell}} \Sigma_{\tau_{\ell}} V^T_{\tau_{\ell}}$ be the economized SVD decomposition of $B_{\tau_{\ell}}$, where $U_{\tau_{\ell}}, V_{\tau_{\ell}}$ are orthonormal and $\Sigma_{\tau_{\ell}}$ is a diagonal matrix containing the non-zero singular values of $B_{\tau_{\ell}}$. Therefore,

$$\mathbb{E}\|B_{\tau_{\ell}} B^{\dagger}_{\tau_{\ell}} F_{\ell-1}\|_F^2 = \mathbb{E}\|U_{\tau_{\ell}} U^T_{\tau_{\ell}} F_{\ell-1}\|_F^2 = \mathbb{E}\|\Sigma^{-1}_{\tau_{\ell}} V^T_{\tau_{\ell}} B^T_{\tau_{\ell}} F_{\ell-1}\|_F^2,$$

and we have from Definition 5.1 that

$$\begin{aligned}
&\mathbb{E}\|B_{\tau_\ell}B_{\tau_\ell}^{\dagger}F_{\ell-1}\|_F^2 \\
&\quad\geq \mathbb{E}\left(\sigma_{\min}^2(\Sigma_{\tau_\ell}^{-1}V_{\tau_\ell}^T)\cdot\|B_{\tau_\ell}^T F_{\ell-1}\|_F^2\right) \\
&\quad= \mathbb{E}\left(\frac{\|B_{\tau_\ell}^T F_{\ell-1}\|_F^2}{\sigma_{\max}^2(B_{\tau_\ell})}\right) \\
&\quad\geq \frac{1}{\beta}\cdot\mathbb{E}\|B_{\tau_\ell}^T F_{\ell-1}\|_F^2.
\end{aligned} \tag{91}$$

Further, we have from (88) that $\text{span}\{F_{\ell-1}\}\subseteq\mathscr{R}(B)$, so it is known from the Courant-Fischer Theorem (Golub & Van Loan, 2014, p. 441) that $\|B^T F_{\ell-1}\|_F^2\geq(\sigma_{\min}^{nz}(B))^2\|F_{\ell-1}\|_F^2$, where $\sigma_{\min}^{nz}(B)$ is the smallest non-zero singular value of B. Thus, it follows from Step 5 in Algorithm 7 that there exists a scalar $\delta\in(0,1)$, such that

$$\begin{aligned}
&\mathbb{E}\|B_{\tau_\ell}B_{\tau_\ell}^{\dagger}F_{\ell-1}\|_F^2 \\
&\quad\geq \frac{1}{\beta}\mathbb{E}\|B_{\tau_\ell}^T F_{\ell-1}\|_F^2 \\
&\quad= \frac{1}{\beta}\sum_{\omega\in\mathscr{T}}\|B_{\omega}^T F_{\ell-1}\|_F^2\cdot\frac{1}{p} \\
&\quad= \frac{\delta}{p\beta}\|B^T F_{\ell-1}\|_F^2 \\
&\quad\geq \frac{\delta\cdot(\sigma_{\min}^{nz}(B))^2}{p\beta}\|F_{\ell-1}\|_F^2.
\end{aligned} \tag{92}$$

Hence, combining (89), (90) and (58), we get

$$\begin{aligned}
\mathbb{E}\|F_\ell\|_F^2 =&\|F_{\ell-1}\|_F^2-\mathbb{E}\|B_{\tau_\ell}B_{\tau_\ell}^{\dagger}F_{\ell-1}\|_F^2 \\
\leq&\left(1-\frac{\delta\cdot(\sigma_{\min}^{nz}(B))^2}{p\beta}\right)\|F_{\ell-1}\|_F^2 \\
\leq&\left(1-\frac{\delta\cdot(\sigma_{\min}^{nz}(B))^2}{p\beta}\right)^{\ell}\|F_0\|_F^2.
\end{aligned} \tag{93}$$

Notice that $\|F_0\|_F^2=\|Z_0-T_{\mathscr{R}(B)^\perp}\|_F^2=\|BB^{\dagger}T\|_F^2$, and a combination of (88) and (93) yields

$$\mathbb{E}\|B(X_{LS}-X^{(\ell)})\|_F^2\leq\left(1-\frac{\delta\cdot(\sigma_{\min}^{nz}(B))^2}{p\beta}\right)^{\ell}\|BB^{\dagger}T\|_F^2. \tag{94}$$

Finally, when B is of full column rank, we obtain

$$\mathbb{E}\left(\frac{\|X_{LS}-X^{(\ell)}\|_F^2}{\|X_{LS}\|_F^2}\right)\leq\left(1-\frac{\delta\cdot(\sigma_{\min}^{nz}(B))^2}{p\beta}\right)^{\ell}\kappa^2(B),$$

where $\kappa(B)=\frac{\sigma_{\max}(B)}{\sigma_{\min}^{nz}(B)}$ denotes condition number of matrix B, and the proof is completed.

□

6 Numerical experiments

In this section, we perform numerical experiments on some real-world databases to illustrate the numerical behavior of our proposed algorithms. In all the experiments, the data vectors are normalized so that $\|\mathbf{x}_i\|_2 = 1, i = 1, 2, \ldots, n$. Moreover, we consider three popular used kernels (Hofmann et al., 2008), including the Gaussian kernel function

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\Big),$$

the Laplacian kernel function

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2}{\sigma}\Big),$$

and the Polynomial kernel function

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^{\ell},$$

where $\ell = 2$, and the value of the Gaussian scale $\sigma > 0$ is set to be the mean Euclidean distance between the training vectors $\mathbf{x}_i$ (Iosifidis & Gabbouj, 2017). All the numerical experiments are carried on a Hp workstation with 16 cores double Intel(R)Xeon(R) E5-2620 v4 processors, and with CPU 2.10 GHz and RAM 64 GB. The operation system is 64-bit Windows 10. The numerical results are obtained from running the MATLAB R2016b software.

There are seven databases used in our experiments, including five *facial image databases* `AR`, `CMU-PIE`, `Extended YaleB`, `Facescrub`, and `YouTube Faces`, a *handwritten digits database* `MNIST`, and a *tiny images database* `CIFAR-100`. Table 3 lists the details of these databases.

- The `AR database`[1] consists of over 4000 facial images (70 male and 56 female) having a frontal facial pose, exhibiting several facial expressions (e.g. anger, smiling and screaming), in different illumination source directions (left and/or right) and with some occlusions (e.g. sun glasses and scarf). A subset of $s = 100$ persons (50 males and 50 females) with 26 images of per people, i.e., 2600 images are utilized in our experiments. We re-scaled the original facial images to 40×30-pixel images, which are subsequently vectorized to $m = 1200$ dimensional facial vectors.
- The `CMU-PIE`[2] (Sim et al., 2003) database consists of more than 40,000 images for $s = 68$ subjects with more than 500 images in each class. These face images are captured by 13 synchronized cameras and 21 flashes under varying pose, illumination, expression and lights. In our experiments, we choose 170 images under different illuminations, lights, expressions and poses for each subject. Thus, the total number of images chosen from `CMU-PIE` database is 11, 560. We crop the images to 32×32 pixels and get $m = 1024$ dimensional facial vector representations.
- The `Extended YaleB`[3] database contains 5760 single light source images of 10 subjects, each seen under 576 viewing conditions (9 different poses and 64 illumination

[1] http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html.
[2] http://www.cs.cmu.edu/afs/cs/project/PIE/web/.
[3] http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html.

conditions of each person). The images have normal, sleepy, sad and surprising expressions. In this experiment, we make use of a subset of $s = 38$ persons with 64 images to per people, i.e., 2432 images, which are cropped and scaled to 64×64 pixels.

- The `Facescrub`[4] (Ng and Winkler, 2014) database contains 106,863 photos of 530 celebrities (265 male and 265 female). The initial images that make up this dataset are procured using Google Image Search. Subsequently, they are processed using the Haarcascade-based face detector from OpenCV 2.4.7 on the images to obtain a set of faces for each celebrity name, with the requirement that a face must be at least 96×96 pixels. In our experiment, we use 22631 photos from 256 male, and scale the images to 9216 pixels.
- The `Youtube Faces`[5] (Wolf et al., 2011) consists of 621126 facial images depicting 1595 persons. In our experiments, we choose the facial images of people with at least 500 images, resulting to a dataset of 370319 images and $s = 340$ classes. Subsequently, each facial image is vectorized to a facial image representation of $m = 1024$ dimensions.
- The `MNIST`[6] database of handwritten digits has 70,000 examples. It was derived from a much larger dataset known as the NIST Special Database 19 (Grother, 1995) which contains digits, uppercase and lowercase handwritten letters. Moreover, the `MNIST` database contains a total of $s = 10$ numbers from 0 to 9. For simplicity, each digit image is flattened and converted into a one-dimensional array of $m = 28 \times 28 = 784$ features.
- The dataset `CIFAR-100`[7] (Krizhevsky, 2009), named after the Canadian Institute for Advanced Research, is labeled subset of the 80 million tiny images dataset. Furthermore, it comes in 20 superclasses of five classes each. For example, the superclass reptile consists of the five classes crocodile, dinosaur, lizard, turtle and snake. The idea is that classes within the same superclass are similar. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). In our experiment, the "fine" labels are utilized, which results in 600 examples of each of $s = 100$ non-overlapping classes. In other words, in our experiment, 60,000 images with each $m = 3072$-pixel are used.

One refers to Figs. 1, 2 and 3 for some samples of the five face databases, the handwritten digits database `MNIST` and the tiny images dataset `CIFAR-100`, respectively. In all the experiments, we randomly split each ID class into two sets, 70 percent for training and 30 percent for testing. To measure the effectiveness of the compared algorithms, we calculate the value of Area Under the Receiver Operating Characteristic Curve (AUC) (Fawcett, 2006; Zhang et al., 2015; Ling et al., 2003) and the value of the Equal Error Rate (EER) (Goudelis et al., 2007; Friedman et al., yyy) for each face verification problem. More precisely, to calculate the AUC and EER metrics, we project the test samples to the corresponding discriminant subspace and depict the similarity between the reduced vector $\mathbf{y}_i$ and the client class mean vector $\mathbf{m}$, by using $s_i = \|\mathbf{y}_i - \mathbf{m}\|_2^{-1}$. The similarity values of all test samples are then sorted in a descending order, and the AUC and EER metrics are calculated. Notice that the smaller the EER values, the larger the AUC values, and the less

[4] http://vintage.winklerbros.net/facescrub.html.
[5] https://www.cs.tau.ac.il/~wolf/ytfaces/.
[6] http://yann.lecun.com/exdb/mnist/.
[7] http://www.cs.toronto.edu/~kriz/cifar.html.

Fig. 1 Some samples of the five face databases, including AR (the first line), the CMU-PIE (the second line), the Extended YaleB (the third line), the Facescrub (the fourth line), and the YouTube Face (the fifth line)

the CPU time, the better an algorithm. In order to eliminate randomness caused by the training-test partition, we apply the above process five times and list the mean values of AUC, EER, the mean CPU time in seconds, as well as the mean standard deviation (Std-Dev) in the tables below.

In this section, the target rank used in all the Nyström-type methods including the standard Nyström, the modified Nyström, the fixed-rank Nyström methods and Algorithm 5, is based on our proposed strategy. That is, the number of clusters s is used as the target rank unless otherwise stated.

Example 1 In this example, we show the efficiency of our proposed trace-ratio model (27) and ratio-trace model (28) in the eigenanalysis step for the calculation of T. To this aim, we compare Algorithm 3 and Algorithm 4 with Algorithm 1 proposed in Iosifidis and Gabbouj (2016a). Three test sets including AR, CMU-PIE and Extended YaleB are used in this example. We choose the reference vector set cardinality $r = 1000$ and the discriminant space dimensionality $d = 10, 20, 30$ for the AR data set, $r = 3000$ and the discriminant

Fig. 2 Some samples of the handwritten digits database MNIST

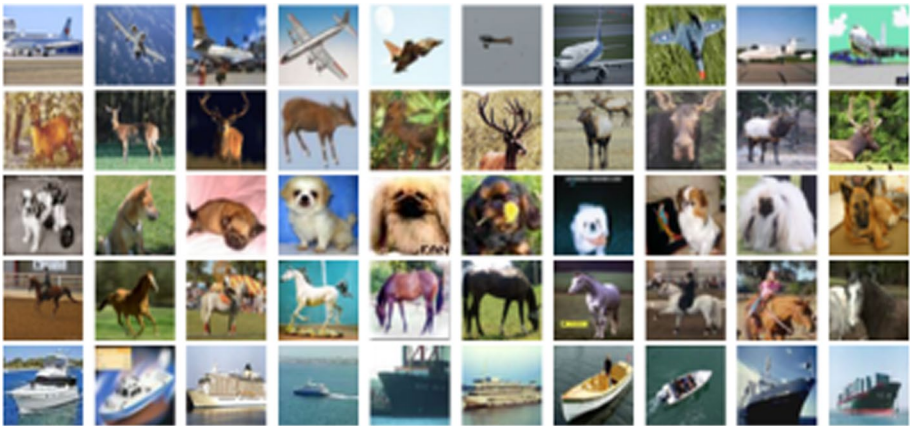

Fig. 3 Some samples (airplane, deer, dog, horse and ship) of the tiny images dataset `CIFAR-100`

space dimensionality $d = 50, 60, 70$ for the `CMU-PIE` data set, and $r = 1000$ and the discriminant space dimensionality $d = 10, 20, 30$ for the `Extended YaleB` data set. Tables 4, 5 and 6 list the experimental results, where we explore the Cholesky factorization for solving (18) in the kernel regression step.

It is obvious to see from Tables 4, 5 and 6 that the AUC values obtained and the CPU time used are comparable for the three algorithms, while the EER values from Algorithm 3 and Algorithm 4 are better than those from Algorithm 1. These show the advantages and illustrate the superiority of the two proposed algorithms over Algorithm 1.

Example 2 The aim of this example is twofold. First, we try to show the effectiveness of Theorem 4.2. Second, we illustrate the rationality of using the number of clusters s as a target rank for randomized algorithms. In Theorem 4.2, it is pointed out that the kernel matrix K is numerically low-rank, and the numerical rank is closely related to the clustering effect of the original data X. Moreover, the better the clustering effect, the closer the numerical rank is to the number of cluster s. We make use of some *semi-artificial* data based on the two data sets `CMU-PIE` and `Extended YaleB` to illustrate this. For the two databases, we first compute the centroid vector $\mathbf{c}_j$ of each class X_j to get $\overline{X} = [\mathbf{c}_1\mathbf{1}_{n_1}^T, \mathbf{c}_2\mathbf{1}_{n_2}^T, \ldots, \mathbf{c}_s\mathbf{1}_{n_s}^T]$, where $\mathbf{1}_{n_i} \in \mathbb{R}^{n_i}$ is the vector of all ones. We then use the MATLAB built-in function `rand.m` to generate a random matrix $\widetilde{\Delta}$ with uniform distribution, and construct the *semi-artificial* data $\widetilde{X} = \overline{X} + \mu \cdot \widetilde{\Delta}$ with $0 < \mu \ll 1$.

Table 7 presents the values of $\frac{\sigma_{s+1}(K)}{\sigma_1(K)}$ and $\|\overline{\Delta}\|_2/\|K\|_2$ in (57). Although the theoretical upper bound given in Theorem 4.2 may not be sharp in practice, it is seen that the values of $\sigma_{s+1}(K)/\sigma_1(K)$ and $\|\overline{\Delta}\|_2/\|K\|_2$ are close to each other. Thus, one can use $\|\overline{\Delta}\|_2/\|K\|_2$ as an estimation to $\sigma_{s+1}(K)/\sigma_1(K)$ in practice.

In Fig. 4, we plot the ratio $\left\{\sigma_i(\widetilde{X})/\sigma_1(\widetilde{X})\right\}_{i=1}^{\min(m,n)}$ of the semi-artificial data $\widetilde{X}$, and the ratio $\left\{\sigma_i(K)/\sigma_1(K)\right\}_{i=1}^{n}$ of the corresponding kernel matrix K for the `CMU-PIE` database and the `Extended YaleB` database with $\mu = 5 \times 10^{-3}$. Here σ_1, σ_i are the largest and the i-th largest singular values, respectively, and the values in brackets are $s + 1$ and the ratio

Table 3 Databases used in the experiments

Datasets	Dimensionality (m)	♯ of classes (s)	♯ of data	♯ of training data (n)	♯ of test data	Background
AR	1200	100	2600	1900	700	Face recognition
CMU-PIE	1024	68	11560	8092	3468	Face recognition
Extended YaleB	4096	38	2432	1710	722	Face recognition
Facescrub	9216	256	22,631	15,956	6675	Face recognition
YouTube Faces	1024	340	370,319	259,223	111,096	Face recognition
MNIST	784	10	70000	49005	20995	Handwritten digits recognition
CIFAR-100	3072	100	60,000	42,000	18,000	Object recognition

Table 4 Example 1: A comparison of three algorithms for eigenanalysis step on AR database with $d = 10, 20, 30$ and $r = 1000$, where the kernel regression step is solved by using the Cholesky factorization

Calculation of T	EER ± Std-Dev%	AUC ± Std-Dev	CPU time(s) ± Std-Dev
AR database (r = 1000,d = 10)			
Algorithm 1	1.03 ± 0.56%	0.993 ± 0.0043	0.00053 ± 0.00016
Algorithm 3	0.29 ± 0.094%	0.999 ± 0.00039	0.0011 ± 0.00013
Algorithm 4	0.38 ± 0.067%	0.999 ± 0.00054	0.0016 ± 0.00018
AR database (r = 1000,d = 20)			
Algorithm 1	1.42 ± 0.46%	0.991 ± 0.0028	0.0011 ± 0.00015
Algorithm 3	0.25 ± 0.088%	0.998 ± 0.00058	0.0024 ± 0.00012
Algorithm 4	0.25 ± 0.081%	0.999 ± 0.00063	0.0035 ± 0.00019
AR database (r = 1000,d = 30)			
Algorithm 1	1.83 ± 0.57%	0.986 ± 0.0047	0.0022 ± 0.00016
Algorithm 3	0.19 ± 0.10%	0.999 ± 0.00053	0.0041 ± 0.00015
Algorithm 4	0.21 ± 0.11%	0.999 ± 0.00059	0.0059 ± 0.00030

$\frac{\sigma_{s+1}(\widetilde{X})}{\sigma_1(\widetilde{X})}$ or $\frac{\sigma_{s+1}(K)}{\sigma_1(K)}$. It is seen from Fig. 4 that when the clustering effect is good (i.e., μ is relatively small), both the singular values of $\widetilde{X}$ and those of the kernel matrix K decay quickly. More precisely, there is a gap between $\frac{\sigma_s}{\sigma_1}$ and $\frac{\sigma_{s+1}}{\sigma_1}$, which validates our theory. Hence, s can be utilized as a numerical rank to K, provided that the clustering effect to X is satisfactory; refer to (45).

Next, we further explain the rationality of using the number of clusters s as a target rank for randomized algorithms. Notice that the performance of a randomized algorithm strongly relies on the chosen target rank, which is difficult to determine in advance, if there is no information available a prior. Indeed, if the chosen parameter is too large, the computational cost and storage requirement will be high. However, if it is too small, the recognition results such as the value of EER will be unsatisfactory. Thus, the idea is to strike a balance between CPU time and EER. More precisely, we aim to find a reasonable target rank that makes both the computation time and the value of EER be relatively small in some degree.

Recall that the kernel spectral regression method is composed of two steps, i.e., the eigenanalysis step and the kernel regression step. Given a range for possible target ranks, we first compute the CPU time (used in seconds) and the value of EER obtained from running "Algorithm 4 + Algorithm 5" for each scatter point, and plot the Time-EER curve. Here "Algorithm 4 + Algorithm 5" means Algorithm 4 for the eigenanalysis step and Algorithm 5 for the kernel regression step. We then use the MATLAB built-in curve fitting toolbox `cftool` to get the analytic expression of the fitted curve. Here the "ideal" target rank is corresponding to the point which has the shortest distance to the origin in the fitted curve. More precisely, this process can be divided into the following four steps.

(1) We first set the target ranks to be $R = \{r_1, r_2, r_3, \ldots, r_t\} = \{5, 15, 25, \ldots, 195\}$, i.e., from 5 to 200 with an interval of 10. Using $r_i (i = 1, 2, \ldots, t)$ as the target rank in "Algorithm 4 + Algorithm 5", we can get the CPU time used in seconds $C = \{c_1, c_2, c_3, \ldots, c_t\}$ and the values of EER, i.e., $F = \{f_1, f_2, f_3, \ldots, f_t\}$.

Table 5 Example 1: A comparison of three algorithms for eigenanalysis step on `CMU-PIE` database with $d = 50, 60, 70$ and $r = 3000$, where the kernel regression step is solved by using the Cholesky factorization

Calculation of T	EER ± Std-Dev%	AUC ± Std-Dev	CPU time(s) ± Std-Dev
CMU-PIE database (r = 3000,d = 50)			
Algorithm 1	4.56 ± 0.42%	0.962 ± 0.0035	0.013 ± 0.00057
Algorithm 3	1.02 ± 0.12%	0.995 ± 0.00050	0.027 ± 0.0014
Algorithm 4	1.07 ± 0.073%	0.995 ± 0.00054	0.035 ± 0.0019
CMU-PIE database (r = 3000,d = 60)			
Algorithm 1	5.84 ± 0.34%	0.952 ± 0.0028	0.016 ± 0.00059
Algorithm 3	1.21 ± 0.24%	0.994 ± 0.00058	0.034 ± 0.0015
Algorithm 4	1.27 ± 0.17%	0.994 ± 0.00063	0.046 ± 0.0023
CMU-PIE database (r = 3000,d = 70)			
Algorithm 1	6.84 ± 0.83%	0.944 ± 0.0070	0.018 ± 0.00078
Algorithm 3	1.52 ± 0.18%	0.993 ± 0.00095	0.039 ± 0.0017
Algorithm 4	1.53 ± 0.26%	0.993 ± 0.0013	0.054 ± 0.0024

Table 6 Example 1: A comparison of three algorithms for eigenanalysis step on `Extended YaleB` database with $d = 10, 20, 30$ and $r = 1000$, where the kernel regression step is solved by using the Cholesky factorization

Calculation of T	EER ± Std-Dev%	AUC ± Std-Dev	CPU time(s) ± Std-Dev
Extended YaleB database (r = 1000,d = 10)			
Algorithm 1	3.60 ± 0.86%	0.971 ± 0.0084	0.00055 ± 0.00022
Algorithm 3	0.31 ± 0.14%	0.999 ± 0.0010	0.0011 ± 0.00019
Algorithm 4	0.36 ± 0.15%	0.999 ± 0.00073	0.0015 ± 0.00030
Extended YaleB database (r = 1000,d = 20)			
Algorithm 1	6.87 ± 0.60%	0.948 ± 0.0038	0.0011 ± 0.00022
Algorithm 3	1.57 ± 0.48%	0.992 ± 0.0032	0.0023 ± 0.00020
Algorithm 4	2.12 ± 0.63%	0.989 ± 0.0032	0.0033 ± 0.00035
Extended YaleB database (r = 1000,d = 30)			
Algorithm 1	11.60 ± 1.90%	0.913 ± 0.012	0.0021 ± 0.00024
Algorithm 3	3.52 ± 0.75%	0.983 ± 0.0038	0.0040 ± 0.00017
Algorithm 4	3.65 ± 0.85%	0.982 ± 0.0047	0.0054 ± 0.00036

(2) To make the roles of CPU time and EER be of equal importance, we scale the data C and F to get $\widetilde{C}$ and $\widetilde{F}$, such that they are in about the same order. We then exploit the MATLAB built-in curve fitting toolbox `cftool` to fit the points in $\widetilde{C} = [\widetilde{c}_1, \widetilde{c}_2, \widetilde{c}_3, \ldots, \widetilde{c}_t]$ and $\widetilde{F} = [\widetilde{f}_1, \widetilde{f}_2, \widetilde{f}_3, \ldots, \widetilde{f}_t]$, in which the type of fit is chosen as "Custom Equation". The analytic expression of the fitted curve is $y = a \cdot e^{-bx} + c, x \in [x_{min}, x_{max}]$, where a, b, c are parameters depending on the data source used.

Table 7 Example 2: The values of $\frac{\sigma_{s+1}(K)}{\sigma_1(K)}$ and $\frac{\|\overline{\Delta}\|_2}{\|K\|_2}$ (in brackets) appeared in (57) for different values μ on the semi-artificial data sets $\widetilde{X}$ based on `CMU-PIE` and `Extended YaleB` databases

Dataset	$\mu = 5 \times 10^{-3}$	$\mu = 5 \times 10^{-4}$	$\mu = 5 \times 10^{-5}$	$\mu = 5 \times 10^{-6}$
$\widetilde{X}$ (`CMU-PIE`)	5.13×10^{-3} (3.92×10^{-4})	4.57×10^{-5} (2.22×10^{-5})	4.49×10^{-7} (2.18×10^{-6})	4.41×10^{-9} (2.18×10^{-7})
$\widetilde{X}$ (`Extended YaleB`)	2.93×10^{-2} (3.54×10^{-3})	2.64×10^{-4} (1.23×10^{-4})	2.51×10^{-6} (1.19×10^{-5})	2.44×10^{-8} (1.18×10^{-6})

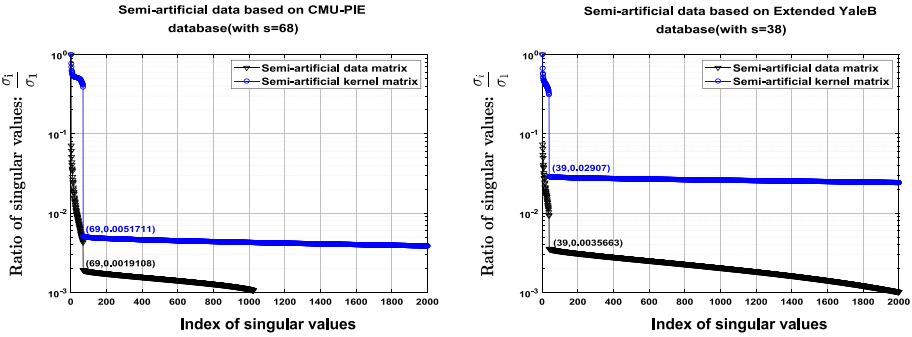

Fig. 4 Example 2: The ratio $\left\{\frac{\sigma_i(\widetilde{X})}{\sigma_1(\widetilde{X})}\right\}_{i=1}^{\min(m,n)}$ of the semi-artificial data $\widetilde{X}$, and the ratio $\left\{\frac{\sigma_i(K)}{\sigma_1(K)}\right\}_{i=1}^{n}$ of the corresponding kernel matrix K for the `CMU-PIE` database (left) and `Extended YaleB` database (right), $\mu = 5 \times 10^{-3}$. Here σ_1, σ_i are the largest and the i-th largest singular values, respectively, and the values in brackets are $s+1$ and the ratio $\frac{\sigma_{s+1}(\widetilde{X})}{\sigma_1(\widetilde{X})}$ or $\frac{\sigma_{s+1}(K)}{\sigma_1(K)}$

(3) By using the MATLAB built-in optimization function `fmincon`, we look for a target point in the fitted curve, which corresponds to the shortest distance from the origin to the curve. Here the target point should be associated with both good recognition effect (small EER value) and less CPU time. In light of the analytic expression of the fitted curve, we define the constraint function as

$$h(x,y) = y - \left(a \cdot e^{-bx} + c\right), \qquad x \in [x_{min}, x_{max}] = \mathbb{X}, \;\; y \in [y_{min}, y_{max}] = \mathbb{Y}. \tag{95}$$

Then, the MATLAB built-in optimization function `fmincon` is exploited to solve the problem

$$(\hat{c},\hat{f}) = \begin{array}{c} \operatorname{argmin} \\ x \in \mathbb{X}, y \in \mathbb{Y} \\ h(x,y) = 0 \end{array} \sqrt{x^2 + y^2}, \tag{96}$$

and $(\hat{c},\hat{f})$ is the desired target point.

(4) Seeking the smallest domain that contains the point $(\hat{c},\hat{f})$. More precisely, we find the index j $(1 \le j \le t-1)$ such that $(\hat{c},\hat{f}) \in [\widetilde{c}_j, \widetilde{c}_{j+1}] \times [\widetilde{f}_j, \widetilde{f}_{j+1}]$. To judge the validity of our theorem, for the index j, we determine the range $[r_j, r_{j+1}]$ $(1 \le j \le t-1)$ in R, and check whether the target rank s defined in Theorem 4.2 is in this interval or not.

We run Algorithm 4 + Algorithm 5 on the `CMU-PIE` database with $r = 3000$, $d = 50$, and the `Extended YaleB` database with $r = 1000$, $d = 10$. The numerical results are the mean of five runs. In Fig. 5, we depict the curves, where the root mean squared errors are $RMSE = 0.0837$ and $RMSE = 0.0858$ for `CMU-PIE` and `Extended YaleB`, respectively. Notice that a smaller RMSE value implies a better fit.

We observe from Fig. 5 that, for the `CMU-PIE` and the `Extended YaleB` databases, the desired balance points p_1 and p_2 corresponding to the shortest distance from the origin to the fitted curve are $(\hat{c},\hat{f}) = (0.63, 0.89) \in [\widetilde{c}_8, \widetilde{c}_9] \times [\widetilde{f}_8, \widetilde{f}_9]$ and $(\hat{c},\hat{f}) = (0.76, 1.14) \in [\widetilde{c}_4, \widetilde{c}_5] \times [\widetilde{f}_4, \widetilde{f}_5]$, respectively. Moreover, for the `CMU-PIE` database with $s = 68$ and the `Extended YaleB` database with $s = 38$, the ranges for target ranks corresponding to the balance point p_1 and p_2 are $[r_8, r_9] = [75, 85]$ and

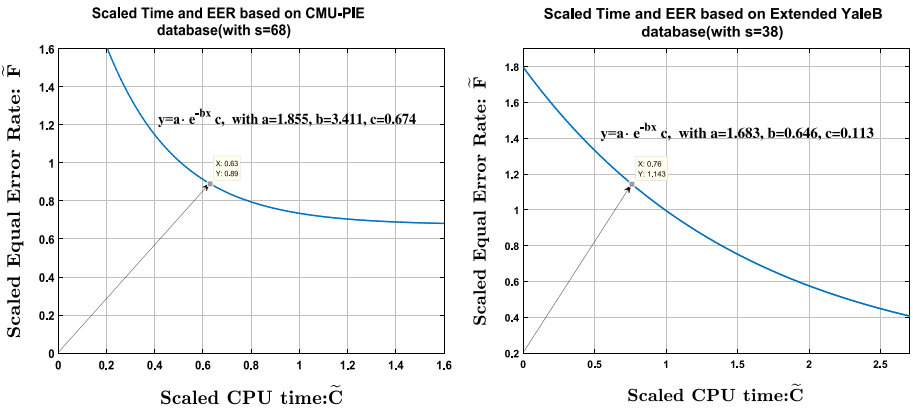

Fig. 5 Example 2: Scaled CPU time and values of EER on `CMU-PIE` (left) with $r = 3000$, $d = 50$, and `Extended YaleB` (right) with $r = 1000$, $d = 10$

$[r_4, r_5] = [35, 45]$, respectively. These demonstrate that Theorem 4.2 is very effective in practice, and our proposed target rank s can be utilized as a numerical rank to the kernel matrix K.

Example 3 In this experiment, we show the numerical behavior of our modified Nyström method with fixed-rank for low-rank approximation of matrices. We run Algorithm 5 and three popular Nyström methods including the standard Nyström (Anaraki & Becker, 2019; Drineas & Mahoney, 2005; Wang & Zhang, 2013), the modified Nyström (Wang & Zhang, 2013) and the fixed-rank Nyström (Anaraki & Becker, 2019) on the `Facescrub` database of size 9216×22631. As the matrix $\widetilde{K}\widetilde{K}^T$ can be very large in practical applications, it is desirable to seek approximations with no need to form it explicitly. Indeed, in the Nyström-type methods, one usually selects a subset of the columns of the matrix in question to build an approximation (Sun et al., 2015). We denote by $S \in \mathbb{R}^{r\times l} (l \ll r)$ a permutation matrix, i.e., a random matrix which has only one entry is one and the rest are zero in each column, and at most one nonzero element in each row. Then we sample the matrix $\widetilde{K}\widetilde{K}^T$ efficiently by computing $\widetilde{K} \cdot (\widetilde{K}^T S) \in \mathbb{R}^{r\times l}$, where the permutation matrix S is stored as a vector, and there is no need to form and store the matrix explicitly. As was stressed in Remark 4, we choose the number of clusters s as the target rank. Moreover, we set the range of the target rank used in the four randomized algorithms from $s - 25$ to $s + 25$ with an interval of 5, and the oversampling parameter is set to be 10.

The numerical performances of four Nyström algorithms on the `Facescrub` database are depicted in Fig. 6, with the reference vector set cardinality $r = 6000$. Denote by $H = \widetilde{K}\widetilde{K}^T$, we define the relative error as $\|H - \widetilde{H}\|_F / \|H\|_F$, where $\widetilde{H}$ is low-rank approximation obtained from different Nyström methods. All the four algorithms are run for 10 times, and the relative error and CPU time are the mean from the 10 runs.

We observe from Fig. 6 that the relative errors of our approximation is much better than those from the standard Nyström and the fixed-rank Nyström methods. In addition, it is a little better than the one from the modified Nyström method. On the other hand, we see that our algorithm is comparable in CPU time to the modified Nyström algorithm, but it

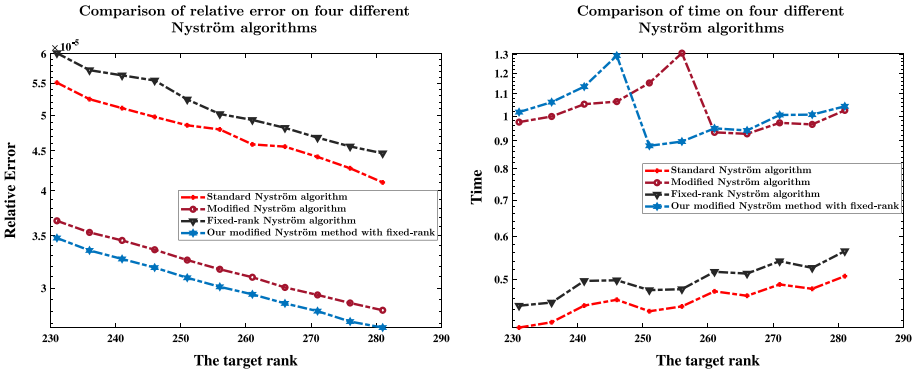

Fig. 6 Example 3: Relative error (left) and CPU time in seconds (right) obtained from four Nyström methods on `Facescrub` with $r = 6000$

is slightly slower than the standard Nyström and the fixed-rank Nyström methods. More precisely, Algorithm 5 is better than the standard Nyström, the modified Nyström and the fixed-rank Nyström methods according to accuracy, and it is comparable to the modified Nyström method in terms of CPU time. Thus, Algorithm 5 is a good choice for overall consideration, and it is a competitive candidate for providing low-rank approximations to large-scale kernel matrices.

Example 4 In this example, we show the efficiencies of Algorithms 5 and 7 for large-scale face verification problem. Three popular used kernels, the Gaussian kernel, the Laplacian kernel and Polynomial kernel are utilized. The test sets are the face databases `YouTube Face` of size $1024 \times 370,319$ and `Facescrub` of size $9216 \times 22,631$. In this example, we set the reference vector set cardinality $r = 6000$ and the discriminant space dimensionality $d = 50, 100$ for the `YouTube Face` database, and $r = 5000$, $d = 10, 50$ for the `Facescrub` database. Recall that the kernel spectral regression methods are composed of the eigenanalysis step and the kernel regression step. For the sake of justification, we choose Algorithm 4 for the eigenanalysis step in all the algorithms.

In the kernel regression step, we run six algorithms including the two proposed Algorithms 5 and 7, the modified Nyström method (Wang & Zhang, 2013), the fixed-rank Nyström method (Anaraki & Becker, 2019), the original ACS-KSR method (Algorithm 2), as well as the randomized block coordinate descent (RBCD) method due to Needell et al., (Needell et al., 2015, Algorithm 2). In view of Theorem 4.2 and Remark 4, the target rank and the oversampling parameter are selected to be the number of clusters s and 10, respectively, in the modified Nyström method, the fixed-rank Nyström method and Algorithm 5. Notice that the kernel regression step in the ACS-KSR-based algorithm is solved "exactly", while it is solved "inexactly" in the other algorithms. Tables 8 and 9 list the numerical results of six algorithms, including the values of EER, AUC, standard deviations, as well as CPU time in seconds. In these tables, the total CPU time includes two parts: the time for generating the reduced kernel matrix $\widetilde{K}$ (the common time), and that for the eigenanalysis step and the kernel regression step (the computational time).

Some remarks are in order. First, regardless of kernel function used, we observe from Tables 8 and 9 that for the YouTube Face dataset, the AUC values of all the algorithms are comparable, while the EER values from ACS-KSR, Needell's RBCD and Algorithm 7

are better than those from Modified Nyström method, Fixed-rank Nyström method and Algorithm 5. For the Facescrub dataset, we see that the AUC and EER values of Algorithm 7 are better than those from the other five algorithms. Moreover, it is seen from the two tables that Algorithm 5 and Modified Nyström method are comparable in view of EER values, and both of them are better than the Fixed-rank Nyström method. It is important to mention that the selection of the target rank in the standard Nyström, the modified Nyström and the fixed-rank Nyström methods, all relies on our proposed strategy, i.e., the number of clusters s is used as the target rank. Without this strategy, the numerical performances of the standard Nyström, the modified Nyström as well as the fixed-rank Nyström methods, may not be so satisfactory.

Second, as is shown in Tables 8 and 9, the computation of the reduced kernel matrix is the main overhead for the first four algorithms. The total CPU timings are about the same for the three randomized algorithms, which are much fewer than those for ACS-KSR. Taking the large-scale dataset `YouTube Face` as an example, we see from Table 8 that Algorithm 5 is about three times faster than the ACS-KSR method. However, when the number of samples is large, explicitly forming or storing the reduced kernel matrix is very costly, and it is interesting to investigate new algorithms that are free of computing and storing the reduced kernel matrix $\widetilde{K}$ directly.

Third, we show the efficiency of our randomized block Kaczmarz method for kernel regression problem with multiple right-hand sides. In Algorithm 7, the stopping criterion *tol* and the maximum number of iteration *iter* are set to be 10^{-2} and 20, respectively. Moreover, we make use of the MATLAB built-in function `svd.m` to compute the Moore-Penrose inverse appeared in Step 7. As a comparison, we apply the randomized block coordinate descent (RBCD) method (Needell et al., 2015, Algorithm 2) proposed by Needell et al., to solve (82), in which the stopping criterion (83) is also used in this algorithm.

For the large-scale dataset `YouTube Face`, it is obvious to see from Table 8 that Algorithm 7 runs much faster than the other algorithms. Indeed, unlike the methods ACS-KSR, modified Nyström, fixed-rank Nyström methods and Algorithm 5, there is no need to explicitly form and store the full reduced kernel matrix $\widetilde{K}$ in Algorithm 7. With $d = 50$ and 100, we see that Algorithm 7 is about 60 and 12 times faster than the ACS-KSR-based algorithm, and it is about 13 and 4 times faster than the modified and fixed-rank Nyström methods, and Algorithm 5. Although there is no need to explicitly form and store the full reduced kernel matrix $\widetilde{K}$ in Needell's RBCD algorithm, one has to solve the kernel regression problem with multiple right-hand sides one by one. Consequently, the RBCD-based algorithm often runs much slower than Algorithm 7. This demonstrate the efficiency of our randomized block Kaczmarz method.

For the relatively small database `Facescrub`, it is seen from Table 9 that the CPU time of Algorithm 7 is comparable to the three Nyström-like methods and Algorithm 5. This is because Algorithm 7 is more suitable to solve large-scale systems with tall (i.e., the number of rows is much larger than the columns) coefficient matrices (Zouzias & Freris, 2013). On the other hand, the AUC and EER values obtained from Algorithm 7 are better than those from the Nyström-like methods and Algorithm 5. Considering the AUC, EER values and CPU time as a whole, Algorithm 7 is a competitive algorithm among all the algorithms.

Table 8 Example 4: Numerical results of the algorithms using the Gaussian kernel, Laplacian kernel and Polynomial kernel on the face database `YouTube Face` with $d = 50, 100$ and $r = 6000$, where the eigenanalysis step is solved by using Algorithm 4 in all the algorithms

Algorithms	Face database: `YouTube Face` (r = 6000,d = 50)			Face database: `YouTube Face` (r = 6000,d = 100)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
	Gaussian Kernel			Gaussian Kernel		
ACS-KSR	0.16 ± 0.021%	0.99 ± 0.00016	185.18 (28.26 + 156.92)	0.17 ± 0.022%	0.99 ± 0.00016	149.14 (28.93 + 120.21)
Modified Nyström method Wang and Zhang (2013)	4.66 ± 0.063%	0.97 ± 0.00087	40.80 (28.26 + 12.54)	4.56 ± 0.16%	0.98 ± 0.0016	50.26(28.93 + 21.33)
Fixed-rank Nyström method Anaraki and Becker (2019)	5.37 ± 0.26%	0.97 ± 0.0022	36.54 (28.26 + 8.28)	5.33 ± 0.051%	0.97 ± 0.00071	46.97(28.93 + 18.04)
Algorithm 5	4.59 ± 0.092%	0.98 ± 0.00052	40.34 (28.26 + 12.08)	4.53 ± 0.035%	0.98 ± 0.00074	50.86(28.93 + 21.93)
Needell's RBCD Needell et al. (2015)	0.71 ± 0.38%	0.99 ± 0.00021	72.71 (0 + 72.71)	0.79 ± 0.37%	0.99 ± 0.00025	158.11(0 + 158.11)
Algorithm 7	1.51 ± 0.79%	0.99 ± 0.0010	2.89 (0 + 2.89)	0.68 ± 0.38%	0.99 ± 0.00042	11.79(0 + 11.79)
	Laplacian Kernel			Laplacian Kernel		
ACS-KSR	1.48 ± 0.049%	0.98 ± 0.00053	186.71 (28.52 + 158.19)	1.30 ± 0.065%	0.98 ± 0.00067	144.44(29.89 + 114.55)
Modified Nyström method Wang and Zhang (2013)	6.50 ± 0.17%	0.96 ± 0.0018	40.10 (28.52 + 11.58)	6.47 ± 0.20%	0.96 ± 0.0016	51.32(29.89 + 21.43)
Fixed-rank Nyström method Anaraki and Becker (2019)	8.48 ± 0.29%	0.95 ± 0.0030	35.65 (28.52 + 7.13)	8.37 ± 0.23%	0.95 ± 0.0020	47.73(29.89 + 17.84)
Algorithm 5	6.70 ± 0.13%	0.96 ± 0.0014	39.35 (28.52 + 10.83)	6.67 ± 0.15%	0.96 ± 0.00082	51.65(29.89 + 21.76)
Needell's RBCD Needell et al., (2015)	0.52 ± 0.31%	0.99 ± 0.0032	83.11 (0 + 83.11)	0.84 ± 0.14%	0.99 ± 0.00020	162.60(0 + 162.60)
Algorithm 7	1.52 ± 1.8%	0.99 ± 0.0027	2.95 (0 + 2.95)	0.92 ± 0.60%	0.99 ± 0.00057	12.15(0 + 12.15)
	Polynomial Kernel			Polynomial Kernel		
ACS-KSR	0.31 ± 0.024%	0.99 ± 0.00029	123.82 (17.59 + 106.23)	0.30 ± 0.017%	0.99 ± 0.00015	134.48(17.86 + 116.62)
Modified Nyström method Wang and Zhang (2013)	4.52 ± 0.066%	0.98 ± 0.00037	29.67 (17.59 + 12.08)	4.60 ± 0.11%	0.98 ± 0.00083	39.41(17.86 + 21.55)
Fixed-rank Nyström method Anaraki and Becker (2019)	5.00 ± 0.075%	0.97 ± 0.00089	25.85 (17.59 + 8.26)	5.05 ± 0.053%	0.97 ± 0.00026	35.54(17.86 + 17.68)
Algorithm 5	4.52 ± 0.095%	0.98 ± 0.0010	29.48 (17.59 + 11.89)	4.53 ± 0.081%	0.98 ± 0.00066	39.67(17.86 + 21.81)
Needell's RBCD Needell et al. (2015)	1.25 ± 0.26%	0.99 ± 0.00018	46.00 (0 + 46.00)	0.97 ± 0.34%	0.99 ± 0.00032	89.15(0 + 89.15)

Table 8 (continued)

Algorithms	Face database: YouTube Face (r = 6000,d = 50)			Face database: YouTube Face (r = 6000,d = 100)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
Algorithm 7	1.55 ± 0.57%	0.99 ± 0.00046	2.41 (0 + 2.41)	1.43 ± 0.59%	0.99 ± 0.00065	12.26(0 + 12.26)

The CPU time Z ($Z_1 + Z_2$) means the total CPU time Z is composed of Z_1 (the common time) for generating the reduced kernel matrix $\widetilde{K}$, and Z_2 (the computation time) for the eigenanalysis step and the kernel regression step

Table 9 Example 4: Numerical results of the algorithms using the Gaussian kernel, Laplacian kernel and Polynomial kernel on the face database `Facescrub` with $d = 10, 50$ and $r = 5000$, where the eigenanalysis step is solved by using Algorithm 4 in all the algorithms

Algorithms	Face database: `Facescrub` (r = 5000,d = 10)			Face database: `Facescrub` (r = 5000,d = 50)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
	Gaussian Kernel			Gaussian Kernel		
ACS-KSR	19.25 ± 0.29%	0.88 ± 0.0018	12.95 (7.06 + 5.89)	19.57 ± 0.24%	0.87 ± 0.0014	12.68(6.99 + 5.69)
Modified Nyström method Wang and Zhang (2013)	22.76 ± 0.20%	0.84 ± 0.0012	7.72 (7.06 + 0.66)	26.53 ± 0.14%	0.80 ± 0.0022	7.69(6.99 + 0.70)
Fixed-rank Nyström method Anaraki and Becker (2019)	25.01 ± 0.30%	0.82 ± 0.0032	7.49 (7.06 + 0.43)	29.91 ± 0.37%	0.76 ± 0.0035	7.46(6.99 + 0.47)
Algorithm 5	22.61 ± 0.27%	0.85 ± 0.0017	7.67 (7.06 + 0.61)	26.39 ± 0.085%	0.80 ± 0.0012	7.64(6.99 + 0.65)
Needell's RBCD Needell et al., (2015)	14.25 ± 5.57%	0.91 ± 0.052	91.67 (0 + 91.67)	30.19 ± 9.51%	0.77 ± 0.0975	431.82(0 + 431.82)
Algorithm 7	13.97 ± 5.45%	0.92 ± 0.045	10.60 (0 + 10.60)	16.47 ± 8.66%	0.89 ± 0.0754	10.30(0 + 10.30)
	Laplacian Kernel			Laplacian Kernel		
ACS-KSR	26.54 ± 0.13%	0.80 ± 0.0016	12.99 (7.11 + 5.88)	17.92 ± 0.25%	0.89 ± 0.00072	12.47(7.15 + 5.32)
Modified Nyström method Wang and Zhang (2013)	25.18 ± 0.52%	0.82 ± 0.0062	7.78 (7.11 + 0.67)	24.44 ± 0.20%	0.82 ± 0.0024	7.85(7.15 + 0.70)
Fixed-rank Nyström method Anaraki and Becker (2019)	28.65 ± 0.44%	0.77 ± 0.0056	7.54 (7.11 + 0.43)	27.50 ± 0.28%	0.79 ± 0.0027	7.63(7.15 + 0.48)
Algorithm 5	24.84 ± 0.37%	0.82 ± 0.0031	7.73 (7.11 + 0.62)	24.37 ± 0.46%	0.82 ± 0.0028	7.80(7.15 + 0.65)
Needell's RBCD Needell et al. (2015)	15.71 ± 5.90%	0.90 ± 0.053	91.77 (0 + 91.77)	30.16 ± 11.31%	0.77 ± 0.18	433.76(0 + 433.76)
Algorithm 7	15.25 ± 5.66%	0.94 ± 0.034	10.68 (0 + 10.68)	16.36 ± 5.92%	0.92 ± 0.056	10.27(0 + 10.27)
	Polynomial Kernel			Polynomial Kernel		
ACS-KSR	22.96 ± 0.32%	0.83 ± 0.0017	11.52 (5.99 + 5.53)	27.93 ± 0.38%	0.78 ± 0.0017	11.81(6.22 + 5.59)
Modified Nyström method Wang and Zhang (2013)	21.90 ± 0.24%	0.85 ± 0.0014	6.64 (5.99 + 0.65)	31.21 ± 0.27%	0.74 ± 0.0028	6.91(6.22 + 0.69)
Fixed-rank Nyström method Anaraki and Becker (2019)	23.86 ± 0.16%	0.82 ± 0.0021	6.41 (5.99 + 0.42)	33.95 ± 0.35%	0.71 ± 0.0021	6.68(6.22 + 0.46)
Algorithm 5	22.00 ± 0.14%	0.85 ± 0.00033	6.59 (5.99 + 0.60)	30.99 ± 0.27%	0.74 ± 0.0016	6.87(6.22 + 0.65)
Needell's RBCD Needell et al. (2015)	17.77 ± 4.37%	0.91 ± 0.038	27.75 (0 + 27.75)	25.54 ± 9.37%	0.80 ± 0.099	146.30(0 + 146.30)
Algorithm 7	12.12 ± 4.54%	0.95 ± 0.016	4.34 (0 + 4.34)	18.80 ± 3.98%	0.89 ± 0.032	4.77(0 + 4.77)

The CPU time Z $(Z_1 + Z_2)$ means the total CPU time Z is composed of Z_1 (the common time) for generating the reduced kernel matrix $\widetilde{K}$, and Z_2 (the computation time) for the eigenanalysis step and the kernel regression step

Example 5 In this example, we show that our strategies proposed in Algorithm 5 and Algorithm 7 also apply to other types of data sets, such as the *handwritten digits database* and the *tiny images database*. The Gaussian kernel, the Laplacian kernel and Polynomial kernel are utilized in this example. The tiny images database `CIFAR-100` is of size $3072 \times 60,000$ and the handwritten digits database `MNIST` is of size $784 \times 70,000$. In this experiment, we set the reference vector set cardinality $r = 6000$ and the discriminant space dimensionality $d = 50, 100$ for the `CIFAR-100` and the `MNIST` databases. Similar to Example 4, we apply Algorithm 4 for the eigenanalysis step in all the algorithms. In the kernel regression step, we also run the proposed Algorithm 5 and Algorithm 7, the modified Nyström method (Wang & Zhang, 2013), the fixed-rank Nyström method (Anaraki & Becker, 2019), the original ACS-KSR method (Algorithm 2), as well as the randomized block coordinate descent (RBCD) method due to (Needell et al., 2015, Algorithm 2). The target rank and the oversampling parameter are selected to be the number of clusters s and 10, respectively.

Tables 10 and 11 list EER, AUC, standard deviations, as well as CPU time in seconds of the six algorithms. For the handwritten digits database MNIST, we see from Table 10 that the EER values obtained from Algorithm 7 are often smaller than the three Nyström-type methods, and it runs much faster than the other algorithms in many cases. For the `CIFAR-100` database in Table 11, it is seen that the EER and AUC values obtained from the six algorithms are with ups and downs, and we cannot tell which one is the best. Indeed, the results are closely related to the data sets and the selection of the kernel functions. Moreover, we observe from Table 11 that the EER and AUC values from the first four algorithms are comparable. This illustrates the rationality of using s as the target rank in the randomized algorithms for solving the kernel regression problem.

Again, we observe from Tables 10 and 11 that the total CPU timings of the modified Nyström method (Wang & Zhang, 2013), the fixed-rank Nyström method (Anaraki & Becker, 2019) and Algorithm 5 are about the same, and they are much faster than the ACS-KSR method. As a comparison, the randomized block coordinate descent (RBCD) method is the slowest one, while the proposed Algorithm 7 performs the best in terms of CPU time. More precisely, Algorithm 5 and Algorithm 7 are about two to three times faster than the ACS-KSR method, respectively. Compared with the RBCD-based algorithm which does not require explicitly form and store the reduced kernel matrix $\widetilde{K}$, either, Algorithm 7 is nearly 10 times faster. All these illustrate the superiority of Algorithm 5 and Algorithm 7 for high-dimensional and large-sample kernel regression problems.

7 Concluding remarks

Face verification is a crucial problem in many applications such as human-computer interaction and human behaviour analysis for assisted living. The approximate class-specific kernel spectral regression (ACS-KSR) method is an improvement to the class-specific kernel discriminant analysis (CS-KDA) and the class-specific kernel spectral regression (CS-KSR) methods, and it is an effective method for face verification problem. However, when the scale of data sets is very large, ACS-KSR may suffer from heavily computational overhead or even be infeasible in practice.

Table 10 Example 5: Numerical results of the algorithms using the Gaussian kernel, Laplacian kernel and Polynomial kernel on the handwritten digits database `MNIST` with $d = 50, 100$ and $r = 6000$, where the eigenanalysis step is solved by using Algorithm 4 in all the algorithms

Algorithms	Handwritten digits database: `MNIST` (r = 6000, d = 50)			Handwritten digits database: `MNIST` (r = 6000, d = 100)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
	Gaussian Kernel			Gaussian Kernel		
ACS-KSR	2.63 ± 0.099%	0.99 ± 0.00033	26.42(4.53 + 21.89)	6.77 ± 0.10%	0.97 ± 0.00088	26.34 (4.60 + 21.74)
Modified Nyström method Wang and Zhang (2013)	16.39 ± 0.80%	0.89 ± 0.0078	5.18(4.53 + 0.65)	15.90 ± 0.56%	0.90 ± 0.0057	6.81 (4.60 + 2.21)
Fixed-rank Nyström method Anaraki and Becker (2019)	16.13 ± 0.20%	0.90 ± 0.0017	5.06(4.53 + 0.53)	15.91 ± 0.090%	0.90 ± 0.0014	6.63 (4.60 + 2.03)
Algorithm 5	15.77 ± 0.13%	0.90 ± 0.0010	5.19(4.53 + 0.66)	15.60 ± 0.12%	0.90 ± 0.00090	6.79 (4.60 + 2.19)
Needell's RBCD Needell et al., (2015)	7.86 ± 0.55%	0.96 ± 0.0048	46.42(0 + 46.42)	8.69 ± 0.74%	0.97 ± 0.0053	92.60 (0 + 92.60)
Algorithm 7	7.85 ± 0.74%	0.97 ± 0.0049	5.12(0 + 5.12)	8.42 ± 0.47%	0.97 ± 0.0028	5.66 (0 + 5.66)
	Laplacian Kernel			Laplacian Kernel		
ACS-KSR	1.21 ± 0.068%	0.99 ± 0.00071	26.20(4.96 + 21.24)	1.19 ± 0.00045%	0.99 ± 0.00017	26.73 (4.95 + 21.78)
Modified Nyström method Wang and Zhang (2013)	16.94 ± 0.39%	0.89 ± 0.0049	5.62(4.96 + 0.66)	16.69 ± 0.30%	0.89 ± 0.0018	7.20 (4.95 + 2.25)
Fixed-rank Nyström method Anaraki and Becker (2019)	16.64 ± 0.55%	0.89 ± 0.0050	5.48(4.96 + 0.52)	16.56 ± 0.55%	0.90 ± 0.0045	7.02 (4.95 + 2.07)
Algorithm 5	16.07 ± 0.20%	0.90 ± 0.0016	5.65(4.96 + 0.69)	15.89 ± 0.13%	0.91 ± 0.0010	7.17 (4.95 + 2.22)
Needell's RBCD Needell et al. (2015)	5.61 ± 0.68%	0.98 ± 0.0035	55.98(0 + 55.98)	6.32 ± 0.88%	0.98 ± 0.0060	118.05 (0 + 118.05)
Algorithm 7	5.37 ± 0.64%	0.98 ± 0.0032	6.55(0 + 6.55)	6.67 ± 0.51%	0.98 ± 0.0041	6.69 (0 + 6.69)
	Polynomial Kernel			Polynomial Kernel		
ACS-KSR	5.56 ± 0.11%	0.97 ± 0.00055	24.66(2.80 + 21.86)	12.96 ± 0.19%	0.93 ± 0.00032	24.29 (2.75 + 21.54)
Modified Nyström method Wang and Zhang (2013)	17.24 ± 0.97%	0.89 ± 0.0088	3.44(2.80 + 0.64)	16.26 ± 0.24%	0.90 ± 0.0027	4.89 (2.75 + 2.14)
Fixed-rank Nyström method Anaraki and Becker (2019)	16.56 ± 0.39%	0.90 ± 0.0038	3.32(2.80 + 0.52)	16.47 ± 0.43%	0.90 ± 0.0038	4.77 (2.75 + 2.02)
Algorithm 5	16.00 ± 0.17%	0.90 ± 0.0013	3.46(2.80 + 0.66)	15.98 ± 0.18%	0.91 ± 0.0011	4.90 (2.75 + 2.15)
Needell's RBCD Needell et al., (2015)	4.95 ± 0.32%	0.98 ± 0.0015	7.89(0 + 7.89)	5.56 ± 1.21%	0.98 ± 0.0047	15.35 (0 + 15.35)

Table 10 (continued)

Algorithms	Handwritten digits database: MNIST (r = 6000, d = 50)			Handwritten digits database: MNIST (r = 6000, d = 100)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
Algorithm 7	10.92 ± 1.61%	0.95 ± 0.012	0.76(0 + 0.76)	9.94 ± 1.18%	0.96 ± 0.0097	2.02 (0 + 2.02)

The CPU time Z ($Z_1 + Z_2$) means the total CPU time Z is composed of Z_1 (the common time) for generating the reduced kernel matrix $\widetilde{K}$, and Z_2 (the computation time) for the eigenanalysis step and the kernel regression step

Table 11 Example 5: Numerical results of the algorithms using the Gaussian kernel, Laplacian kernel and Polynomial kernel on the tiny images dataset `CIFAR-100` with $d = 50, 100$ and $r = 6000$, where the eigenanalysis step is solved by using Algorithm 4 in all the algorithms

Algorithms	Tiny images dataset: `CIFAR-100` (r = 6000,d = 50)			Tiny images dataset: `CIFAR-100` (r = 6000,d = 100)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
	Gaussian Kernel			Gaussian Kernel		
ACS-KSR	32.10 ± 0.19%	0.73 ± 0.0020	26.94 (8.64 + 18.30)	36.85 ± 0.056%	0.67 ± 0.00056	26.53(8.20 + 18.33)
Modified Nyström method Wang and Zhang (2013)	33.84 ± 0.24%	0.70 ± 0.0020	9.68 (8.64 + 1.04)	35.12 ± 0.22%	0.68 ± 0.0020	10.47(8.20 + 2.27)
Fixed-rank Nyström method Anaraki and Becker (2019)	36.57 ± 0.22%	0.67 ± 0.0032	9.37 (8.64 + 0.73)	38.43 ± 0.23%	0.64 ± 0.0027	10.16(8.20 + 1.96)
Algorithm 5	33.80 ± 0.21%	0.70 ± 0.0020	9.68 (8.64 + 1.04)	35.16 ± 0.30%	0.68 ± 0.0027	10.47(8.20 + 2.27)
Needell's RBCD Needell et al. (2015)	42.24 ± 3.26%	0.61 ± 0.039	79.53 (0 + 79.53)	44.24 ± 4.29%	0.57 ± 0.051	146.98(0 + 146.98)
Algorithm 7	39.99 ± 1.33%	0.63 ± 0.011	4.30 (0 + 4.30)	41.34 ± 1.70%	0.61 ± 0.015	5.29(0 + 5.29)
	Laplacian Kernel			Laplacian Kernel		
ACS-KSR	22.67 ± 0.17%	0.84 ± 0.0018	27.34 (9.09 + 18.25)	23.18 ± 0.19%	0.83 ± 0.0018	28.55(8.78 + 19.77)
Modified Nyström method Wang and Zhang (2013)	32.29 ± 0.22%	0.72 ± 0.0015	10.13 (9.09 + 1.04)	32.93 ± 0.28%	0.71 ± 0.0019	11.00(8.78 + 2.22)
Fixed-rank Nyström method Anaraki and Becker (2019)	37.04 ± 0.34%	0.67 ± 0.0035	9.83 (9.09 + 0.74)	38.19 ± 0.25%	0.65 ± 0.0024	10.76(8.78 + 1.98)
Algorithm 5	32.26 ± 0.19%	0.72 ± 0.0016	10.10 (9.09 + 1.01)	32.93 ± 0.13%	0.71 ± 0.0015	11.04(8.78 + 2.26)
Needell's RBCD Needell et al., (2015)	43.43 ± 3.23%	0.61 ± 0.034	414.43 (0 + 414.43)	46.32 ± 1.19%	0.56 ± 0.017	800.49(0 + 800.49)
Algorithm 7	36.47 ± 1.26%	0.68 ± 0.017	11.80 (0 + 11.80)	39.09 ± 1.84%	0.66 ± 0.019	12.70(0 + 12.70)
	Polynomial Kernel			Polynomial Kernel		
ACS-KSR	42.13 ± 0.14%	0.61 ± 0.0017	24.23 (6.68 + 17.55)	45.57 ± 0.12%	0.56 ± 0.00085	25.08(6.52 + 18.56)
Modified Nyström method Wang and Zhang (2013)	38.14 ± 0.16%	0.65 ± 0.0022	7.70 (6.68 + 1.02)	40.29 ± 0.12%	0.62 ± 0.0013	8.76(6.52 + 2.24)
Fixed-rank Nyström method Anaraki and Becker (2019)	39.87 ± 0.19%	0.62 ± 0.0018	7.42 (6.68 + 0.74)	42.14 ± 0.21%	0.60 ± 0.0033	8.50(6.52 + 1.98)
Algorithm 5	38.07 ± 0.075%	0.65 ± 0.0012	7.70 (6.68 + 1.02)	40.30 ± 0.13%	0.62 ± 0.0016	8.78(6.52 + 2.26)
Needell's RBCD Needell et al., (2015)	44.07 ± 2.44%	0.58 ± 0.026	108.81 (0 + 108.81)	45.96 ± 0.65%	0.56 ± 0.015	213.31(0 + 213.31)

Table 11 (continued)

Algorithms	Tiny images dataset: CIFAR-100 (r = 6000,d = 50)			Tiny images dataset: CIFAR-100 (r = 6000,d = 100)		
(Methods)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)	EER ± Std-Dev%	AUC ± Std-Dev	CPU time (s)
Algorithm 7	44.65 ± 1.25%	0.58 ± 0.019	5.67 (0 + 5.67)	48.11 ± 0.85%	0.54 ± 0.0087	7.11(0 + 7.11)

The CPU time Z ($Z_1 + Z_2$) means the total CPU time Z is composed of Z_1 (the common time) for generating the reduced kernel matrix $\widetilde{K}$, and Z_2 (the computation time) for the eigenanalysis step and the kernel regression step

In this paper, we propose new algorithms based on ACS-KSR method to speed up the computation of large-scale face verification problem. By exploiting the special structure of the scatter matrices, we give a correction to the eigenanalysis step in the ACS-KSR method. The first main contribution of this work is to show why low-rank matrix approximation works well for the kernel methods from a theoretical point of view, and we propose a practical strategy for determining target rank for the randomized Nyström method. Based on this strategy, a modified Nyström method with fixed-rank for low-rank approximation of matrices is proposed.

In the big data era, however, the size of kernel matrix is so huge that it is impractical to form or store the matrix in main memory. Therefore, the second main contribution is to propose a randomized block Kaczmarz algorithm for kernel regression with *multiple right-hand sides*, in which one only needs to compute a very small portion of the reduced kernel matrix. The convergence analysis of the new method is established. Numerical experiments on some real-world data sets demonstrate that the proposed approaches achieve satisfactory performance, especially for huge-scale data problems.

We would like to provide more specific information for helping the interested readers to identify pros and cons associated with the two proposed methods. On one hand, if the reduced kernel matrix is not very large and one can store some portions of it, we recommend using Algorithm 5 for its simplicity. On the other hand, if the reduced kernel matrix is so huge that even a small portion is hard to store in main memory, we highly recommend using Algorithm 7 for its high efficiency and low workload. Finally, we would like to stress that the strategies proposed in this paper also apply to many conventional kernel methods (Hofmann et al., 2008), the multiple kernel methods (Bucak et al., 2014; Gönen & Alpayin, 2011), and deep learning (Wang et al., 2021). These are very interesting topics and deserve further investigation.

Acknowledgements We would like to express our sincere thanks to our editor and the anonymous referees for insightful comments and suggestions that greatly improved the representation of this paper.

Funding This work is supported by the Fundamental Research Funds for the Central Universities of China under Grant No. 2019XKQYMS89.

References

Anaraki, F., & Becker, S. (2019). Improved fixed-rank Nyström approximation via QR decomposition: Practical and theoretical aspects. *Neurocomputing, 363,* 261–272.

Arashloo, S., & Kittler, J. (2014). Class-specific kernel fusion of multiple descriptors for face verification using multiscale binarised statistical image features. *IEEE Transactions on Information Forensics and Security, 9,* 2100–2109.

Bach, F. (2013). Sharp analysis of low-rank kernel matrix approximations, JMLR: Workshop and Conference Proceedings, 30: 1–25.

Barr, P., Noble, J., & Biddle, R. (2007). Video game values: Human-computer interaction and games. *Interacting with Computers, 19,* 180–195.

Baudat, G., & Anouar, F. (2000). Generalized discriminant analysis using a kernel approach. *Neural Computation, 12,* 2385–2404.

Bucak, S., Jin, R., & Jain, A. (2014). Multiple kernel learning for visual object recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36,* 1354–1369.

Cambier, L., & Darve, E. (2019). Fast low-rank kernel matrix factorization using skeletonized interpolation. *SIAM Journal on Scientific Computing, 41,* A1652–A1680.

Cao, G., Iosifidis, A., & Gabbouj, M. (2018). Neural class-specific regression for face verification. *IET Biometrics, 7,* 63–70.

Cortes, C., Mohri, M., & Talwalkar, A. (2010). On the impact of kernel approximation on learning accuracy, Journal of Machine Learning Research, 113–120.
Drineas, P., & Mahoney, M. (2005). On the Nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research, 6,* 2153–2175.
Duda, R., Hart, P., & Stork, D. (2000). Pattern Classification, 2nd edition, Wiley.
Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27,* 861–874.
Friedman, L., Stern, H., Prokopenko, V., & Komogortsev, O.. Relationship between number of subjects and biometric authentication equal error rates, arXiv:1906.06272v1.
Gittens, A. (2011). The spectral norm error of the naive Nyström extension, arXiv: 1110.5305.
Golub, G. H., & Van Loan, C. F. (2014). *Matrix Computations* (4th ed.). Baltimore: Johns Hopkins University Press.
Gönen, M., & Alpayin, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research, 12,* 2211–2268.
Goudelis, G., Zafeiriou, S., Tefas, A., & Pitas, I. (2007). Class-specific kernel discriminant analysis for face verification. *IEEE Transactions on Information Forensics and Security, 2,* 570–587.
Grimmett, G., & Stirzaker, D. (2001). Probability and random processes, 3rd edition, Oxford University Press.
Grother, P. (1995). *NIST special database 19 handprinted forms and characters database*. National Institute of Standards and Technology: Tech. Rep.
Halko, N., Martinsson, P., & Troop, J. (2011). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review, 53,* 217–288.
Higham, N. J., & Mary, T. (2019). A new preconditioner that exploits low-rank approximations to factorization error. *SIAM Journal on Scientific Computing, 41,* A59–A82.
Hofmann, T., Schölkopf, B., & Smola, A. (2008). Kernel methods in machine learning. *The Annals of Statistics, 36,* 1171–1220.
Iosifidis, A., & Gabbouj, M. (2016). Hierarchical class-specific kernel discriminant analysis for face verification, Visual Communications and Image Processing (VCIP), pp. 1–4.
Iosifidis, A., & Gabbouj, M. (2016). Scaling up class-specific kernel discriminant analysis for large-scale face verification. *IEEE Transactions on Information Forensics and Security, 11,* 2453–2465.
Iosifidis, A., & Gabbouj, M. (2017). Class-specific kernel discriminant analysis revisited: further analysis and extensions. *IEEE Transactions on Cybernetics, 47,* 4485–4496.
Iosifidis, A., Tefas, A., & Pitas, I. (2015). Class-specific reference discriminant analysis with application in human behavior analysis. *IEEE Transactions on Human-Machine Systems, 45,* 315–326.
Iske, A., Borne, S., & Wende, M. (2017). Hierarchical matrix approximation for kernel-based scattered data interpolation. *SIAM Journal on Scientific Computing, 39,* A2287–A2316.
Jia, Y., Nie, F., & Zhang, C. (2009). Trace ratio problem revisited. *IEEE Transactions on Neural Networks, 20,* 729–735.
Kittler, Y., Li, J., & Matas, J. (2000). Face verification using client specific Fisher faces, The Statistics of Directions, 63–66.
Krizhevsky, A. (2009). Learning multiple layers of features from tiny images .
Lan, L., Zhang, K., Ge, H., et al., (2017). Low-rank decomposition meets kernel learning: A generalized Nyström method. *Artificial Intelligence, 250,* 1–15.
Lei, Z., Liao, S., Jain, A. K., & Li, S. Z. (2012). Coupled discriminant analysis for heterogeneous face recognition. *IEEE Transactions on Information Forensics and Security, 7,* 1707–1716.
Ling, C., Huang, J., & Zhang, H. (2003). AUC: a better measure than accuracy in comparing learning algorithms, Advances in Artificial Intelligence, pp. 329–341.
Li, Z., Park, U., & Jain, A. K. (2011). A discriminative model for age invariant face recognition. *IEEE Transactions on Information Forensics and Security, 6,* 1028–1037.
Lu, J., Plataniotis, K., & Venetsanopoulos, A. (2003). Face recognition using kernel direct discriminant analysis algorithms. *IEEE Transactions on Neural Networks, 14,* 117–126.
Müller, K., Mika, S., Rätsch, G., Tsuda, K., & Schölkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks, 12,* 181–201.
Necoara, I. (2019). Faster randomized block Kaczmarz algorithms. *SIAM Journal on Matrix Analysis and Applications, 40,* 1425–1452.
Needell, D., & Tropp, J. A. (2014). Paved with good intentions: Analysis of a randomized block Kaczmarz method. *Linear Algebra and its Applications, 441,* 199–221.
Needell, D., Zhao, R., & Zouzias, A. (2015). Randomized block Kaczmarz method with projection for solving least squares. *Linear Algebra and its Applications, 484,* 322–343.
Ng, H., & Winkler, S. (2014).A data-driven approach to cleaning large face datasets, IEEE International Conference on Image Processing (ICIP), pp. 343–347.

Ngo, T., Bellalij, M., & Saad, Y. (2012). The trace-ratio optimization problem. *SIAM Review, 54,* 545–569.
Pan, B., Lai, J., & Yuen, P. (2011). Learning low-rank Mercer kernels with fast-decaying spectrum. *Neurocomputing, 74,* 3028–3035.
Park, C., & Park, H. (2008). A comparision of generalized linear discriminant analysis algorithms. *Pattern Recognition, 41,* 1083–1097.
Shi, W., & Wu, G. (2021). New algorithms for trace-ratio problem with application to high-dimension and large-sample data dimensionality reduction. *Machine Learning, Machine Learning, Special Issue on Feature Engineering, Article, 4,* 1–28.
Sim, T., Baker, S., & Bsat, M. (2003). The CMU pose, illumination, and expression database. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 25,* 1615–1618.
Strohmer, T., & Vershynin, R. (2009). A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications, 15,* 262–278.
Sun, S., Zhao, J., & Zhu, J. (2015). A review of Nyström methods for large-scale machine learning. *Information Fusion, 26,* 36–48.
Tavernier, J., Simm, J., Meerbergen, K., Wegner, J. K., Ceulemans, H., & Moreau, Y. (2019). Fast semi-supervised discriminant analysis for binary classification of large data sets. *Pattern Recognition, 91,* 86–99.
Tefas, A., & Pitas, I. (2011). Human centered interfaces for assisted living. *Man-Machine Interactions, 2,* 3–10.
Tran, T., Douzal-Chouakria, A., Yazdi, S., et al., (2020). Interpretable time series kernel analytics by preimage estimation. *Artificial Intelligence, 286,* 103342.
Tropp, J. (2012). User-friendly tools for random matrices: An introdction.
Wang, H., Yan, S., Xu, D., & Huang, X. (2007). Trace-ratio vs. ratio-trace for dimensionality reduction, IEEE Conference on Compute Vision and Pattern Recognition, pp. 1–8.
Wang, R., Li, Y., & Darve, E. (2018). On the numerical rank of radial basis function kernels in high dimensions. *SIAM Journal on Matrix Analysis and Applications, 39,* 1810–1835.
Wang, S., & Zhang, Z. (2013). Improving CUR matrix decomposition and Nyström approximation via adaptive sampling. *Journal of Machine Learning Research, 14,* 2729–2769.
Wang, T., Zhang, L., & Hu, W. (2021). Bridging deep and multiple kernel learning: A review. *Information Fusion, 67,* 3–13.
Wathen, A., & Zhu, S. (2015). On spectral distribution of kernel matrices related to radial basis functions. *Numerical Algorithms, 70,* 709–726.
Williams, C., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems, 13,* 682–688.
Wolf, L., Hassner, T., & Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity, IEEE Conference on Computer Vision and Pattern Recognition, pp. 529–534.
Wu, X., Kumar, V., Quinlan, J., et al., (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14,* 1–37.
Wu, X., Xu, J., Wang, J., Li, Y., Li, W., & Guo, Y. (2019). Identity authentication on mobile devices using face verification and ID image recognition. *Procedia Computer Science, 162,* 932–939.
Xing, X., & Chow, E. (2020). Interpolative decomposition via proxy points for kernel matrices. *SIAM Journal on Matrix Analysis and Applications, 41,* 221–243.
Zafeiriou, S., Tzimiropoulos, G., Petrou, M., & Stathaki, T. (2012). Regularized kernel discriminant analysis with a robust kernel for face recognition and verification. *IEEE Transactions on Neural Networks and Learning Systems, 23,* 526–534.
Zhang, X., Li, X., Feng, Y., & Liu, Z. (2015). The use of ROC and AUC in the validation of objective image fusione valuation metrics. *Signal Processing, 115,* 38–48.
Zhao, M., Chan, R., Tang, P., Chow, T., & Wong, S. (2013). Trace-ratio linear discriminant analysis for medical diagnosis: a case study of dementia. *IEEE Singal Processing Letters, 20,* 431–434.
Zheng, W., Lin, Z., & Wang, H. (2013). L1-norm kernel discriminant analysis via Bayes error bound optimization for robust feature extraction. *IEEE Transactions on Neural Networks and Learning Systems, 25,* 793–805.
Zhou, C., Wang, L., Zhang, Q., & Wei, X. (2013). Face recognition based on PCA image reconstruction and LDA. *Optik, 124,* 5599–5603.
Zoric, V. (2008). *Mathematical Analysis I*. Berlin: Springer.
Zouzias, A., & Freris, N. M. (2013). Randomized extended Kaczmarz for solving least-squares. *SIAM Journal on Matrix Analysis and Applications, 34,* 773–793.

Authors and Affiliations

Ke Li[1] · Gang Wu[1]

Ke Li
likeer1002@163.com

[1] School of Mathematics, China University of Mining and Technology, Xuzhou 221116, Jiangsu, People's Republic of China